

**OGC Compatible  
Geographical Information Systems  
Web Services**

*Ahmet Sayar*

*Indiana University - 2005*

# CONTENT

<b>1</b>	<b>INTRODUCTION .....</b>	<b>5</b>
<b>2</b>	<b>GIS TECHNOLOGY AND RELATED WORK .....</b>	<b>6</b>
2.1	Technology and Terms and Definitions .....	6
2.2	Related Works .....	7
2.2.1	GIS software implementation efforts .....	7
2.2.1.1	Server Efforts .....	7
2.2.1.2	Client Efforts .....	10
2.2.2	Academic GIS application efforts and centers .....	11
2.2.3	Other non-OGC efforts .....	12
<b>3</b>	<b>OGC OPENGIS SPECIFICATIONS .....</b>	<b>14</b>
3.1	OGC OpenGIS Specifications we have been using in our GIS project .....	16
3.1.1	WMS Specifications .....	16
3.1.2	WMS Client .....	17
3.1.3	Other OGC Specifications .....	20
3.1.3.1	Geographic Markup Language (GML) Specification .....	20
3.1.3.2	Web Feature Service (WFS) Specifications .....	20
3.1.3.3	Catalog Services (CAT) Specifications .....	21
<b>4</b>	<b>APPLICABILITY OF WEB SERVICE ARCHITECTURE ON TO OGC GIS SERVICES .....</b>	<b>22</b>
4.1	OGC Compatibility with the Web Services .....	23
4.2	How to create valid requests to WMS in case of using Web Services .....	24
<b>5</b>	<b>DEVELOPING WEB SERVICE COMPATIBLE VERSIONS OF OGC SERVICES .....</b>	<b>27</b>
5.1	Implementation of OGC Specs .....	27
5.1.1	WMS Implementation .....	28
5.1.1.1	Cascading WMS .....	31
5.1.2	WMS Client Implementation .....	33
5.1.3	WMS Interactions in a Comprehensive GIS System .....	35
5.1.3.1	WFS .....	35
5.1.3.2	IS (Information-Discovery Services) corresponds to OGC Catalog Services (CAT) .....	36

<b>6</b>	<b>CURRENT STATUS AND FUTURE WORK .....</b>	<b>37</b>
6.1	Current Status .....	37
6.2	Near Term Work .....	37
6.3	Longer Term Research and Development.....	39
<b>7</b>	<b>CONCLUSION.....</b>	<b>41</b>
	<b>APPENDIXES.....</b>	<b>42</b>
APPENDIX -1	Web Service Description File (WMSServices.wsdl).....	42
APPENDIX-2	Sld:StyledLayerDescriptor Element used in getMap request. ....	43
APPENDIX-3	Sample WMS requests created according to valid schema files .....	44
APPENDIX-4	Project page deployed on Portal as a portlet .....	47
APPENDIX-5	Sample WMS Capabilities file used in project.....	48
APPENDIX-6	geometry.xsd, Geometry schema for GML encoding of feature data..	51
APPENDIX-7	Summary of the CrisisGrid related OGC Specifications.....	52
APPENDIX-8	Open Source / Free GIS related software projects.....	56
APPENDIX-9	Workload So far .....	65
	<b>REFERENCES .....</b>	<b>66</b>

## FIGURES:

<b>Figure 1 : Approved OpenGIS Specifications. Figure is created from the specification definitions at <a href="http://www.opengeospatial.org/specs/?page=specs">http://www.opengeospatial.org/specs/?page=specs</a>.....</b>	<b>15</b>
<b><i>Figure 2 : GetCapabilities Request Schema (GetCapabilities.xsd).....</i></b>	<b>24</b>
<b><i>Figure 3 : GetMap Request Schema (GetMap.xsd).....</i></b>	<b>25</b>
<b><i>Figure 4 : GetFeatureInfo Request Schema (GetFeatureInfo.xsd) .....</i></b>	<b>26</b>
<b><i>Figure 5 : OGC Compatible GIS Web Services Architecture. Some of the WFSs are service oriented some are not. Instead of OGC's CAT, we have used service oriented IS-Discovery implemented by CGL. ....</i></b>	<b>27</b>
<b><i>Figure 6 : The concept of Cascading Web Map Server.....</i></b>	<b>32</b>
<b><i>Figure 7 : Sample GetFeature request from WMS to WFS. ....</i></b>	<b>36</b>
<b><i>Figure 8 : Progress picture for OGC compatible data retrieval operations. Architecture details are given in Figure 5. ....</i></b>	<b>38</b>

## TABLES:

<b>Table-1: The Parameters of the GetCapabilities Request .....</b>	<b>18</b>
<b>Table-2: The Parameters of the GetMap Request .....</b>	<b>18</b>
<b>Table-3: The Parameters of the GetFeatureInfo Request .....</b>	<b>19</b>
<b>Table-4: Project Research Stack .....</b>	<b>40</b>

# 1 Introduction

Geographical Information Systems (GIS) introduce methods and environments to visualize, manipulate, analyze and display geographic data. These methods and environments have some interoperability problems. Different organizations and commercial vendors develop their own data models and storage structures. If GIS services are not interoperable, GIS services can not interact with each other even though they are in the same organization or they belong to same commercial vendor. The nature of the geographical applications requires seamless integration and sharing of spatial data from a variety of providers. Interoperability is a main goal for GIS. [12]

To solve the interoperability problems, the Open Geospatial Consortium (OGC) has introduced some 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]. OGC formed Open Geographic Information Systems (OpenGIS) to lead the development of geoprocessing interoperability.

This document gives the details about the design and architecture of our Web Service oriented OGC compatible GIS project [2]. Originally OGC specifications are not designed to be Web Service compatible. In this document we will first give brief explanation about the GIS technology and related work on this area. Under this title, we will give the definitions of some commonly used terms in the GIS. Related work will be grouped into three categories. These are GIS software implementation efforts, academic GIS application efforts and centers, and other non-OGC efforts. In the next chapter we will describe the compatibility requirements of the OGC's OpenGIS Specifications for the Web Map Services (WMS) and other GIS services (Web Feature Services and Catalog Services) that WMS depend upon to fulfill its requirements. In the next chapter we will mention about the Web Services and how to apply it to the OGC compatible WMS. We will be mentioning about the conflicts encountered during the implementation between OGC compatibility and Web Services. We will also mention how Web Service technologies can be applied to the area of geospatial information systems (GIS) while maintaining the OGC compatibility. We will also mention about how to create valid requests to WMS in case of using Web Services. The next chapter will give the architecture and design details. In this chapter we will explain the implementations of services according to specifications defined in Chapter 3.1 with the Web services approach. After that the document is going to give information about the future work. Future work is grouped into three subsections. These are the current status, near future and future sections. Tasks in the near future section have higher priority than those in the future section. The last chapter will be conclusion

Since I have been implementing OGC compatible WMS and WMS client, this document will be focusing on them. WMS has some connections with some other GIS services such as the Web Feature Service (WFS) and the Catalog Services (CAT). These OGC compatible services also have specifications and we also will be mentioning from their specifications and implementation details in separate sections from the WMS point of view.

## **2 GIS Technology and Related Work**

### **2.1 Technology and Terms and Definitions**

Geographic Information System (GIS) is a collection of methods to visualize, manipulate, analyze, and display geographically referenced data or geospatial data. The sources of geospatial data are digitized maps, aerial photographs, satellite images, statistical tables and other related documents. GIS visualization services enable maps, these maps links databases to the maps. These databases keep geospatial data in the predefined form such as binary, xml, string etc. Explaining geographic data by pictures (maps) is much more powerful than explaining same thing by numbers.

GIS relates different information represented in spatial context. For example state boundary lines data can be analyzed and produce a map. By the same way, fault data can be analyzed and produce a map. GIS relates these two data sets by overlaying these two maps produced from the corresponding data and reach a conclusion about this relationship.

Below we list and give the definitions of some commonly used terms in the GIS. We will be using these terms often in the following chapters.

#### **SPATIAL DATA:**

Spatial data are a kind of data that pertains to the space occupied by objects. Example spatial data from the real world are cities, rivers, roads, states, crop coverage, mountain ranges etc. In the implementation these are represented by points, lines, rectangles, surfaces, volumes and etc. Spatial data have some common characteristics. These type of data are geometric data and in high dimensions. These data can be either discrete (vector) or continuous (raster). GIS applications are applied on these types of data.

#### **GEOSPATIAL DATA:**

Geospatial data are spatial data associated with a location relative to the Earth.

#### **FEATURE:**

A feature is an abstraction of a real world phenomenon. A digital representation of the real world can be thought of as a set of features.

#### **GEOGRAPHIC FEATURE:**

A geographic feature is a feature associated with a location relative to the Earth. Geographic features are those that may have at least one property that is geometry-valued. [7]

#### **VECTOR DATA:**

Vector data deals with discrete phenomena, each of which is conceived of as a feature. The spatial characteristics of a discrete real world phenomenon are represented by a set of one or more geometric primitives (points, curves, surfaces, or solids). Other characteristics of the phenomenon are recorded as feature attributes. [27] Usually, a single feature is associated with a single set of attribute values.

#### **RASTER DATA:**

Raster data deals with real world phenomena that vary continuously over space. It contains a set of values, each associated with one of the elements in a regular array of points or cells. It

is usually associated with a method for interpolating values at spatial positions between the points or within the cells. [27]

#### COVERAGE – COVERAGE DATA:

OGC uses the term “coverage” to refer to any data representation that assigns values directly to spatial position. A coverage is a feature that associates positions within a bounded space (its spatiotemporal domain) to feature attribute values (its range). Examples include a raster image, a polygon overlay, or a digital elevation matrix. [27]

The spatio-temporal domain of a coverage is a set of geometric objects described in terms of direct positions. Commonly used spatio-temporal domains include point sets, grids, collections of closed rectangles, and other collections of geometric objects.

#### SPATIAL REFERENCE SYSTEM:

A spatial reference system is a function which associates locations in space to geometries of coordinate tuples in a mathematical space, usually a real valued coordinate vector space, and conversely associates coordinate values and geometries to locations in the real world. [28]

#### TEMPORAL REFERENCE SYSTEM:

A temporal reference system is a function that associates time to a coordinate (usually one dimensional points and intervals) and conversely associates coordinate geometries to real world time. [28]

#### SPATIAL-TEMPORAL REFERENCE SYSTEM:

A spatial temporal reference system is an aggregation of a spatial system and a temporal system that it uses to associate coordinate geometries to locations in space and time. Normally, the aggregation uses orthogonal coordinates to represent space and time, but this is not necessarily the case in more complex, relativistic environments. [28]

## 2.2 Related Works

### 2.2.1 GIS software implementation efforts

#### 2.2.1.1 Server Efforts

(Service providers for the OGC compatible GIS services)

This chapter lists some OGC Compatible servers that they implemented OGC Approved Specifications [37]. Some of the specifications are abstract some of them are just discussion papers. We have not mentioned from them in this chapter.

Here you will see some application efforts implementing OGC approved GIS specifications shown in Figure 1 as red lined boxes.

**AskTheSpider:** AskTheSpider is the OGC Catalog, where you can submit new server URL and discover OGC Services by box, keywords, taxonomies and more. More than 1500 layers are already discoverable. Available at <http://www.askthespider.com>

**Client:** AskTheSpider integrates a client that allows dynamic search in an OGC WRS Catalog, a client to register new services and a client to navigate and access to remote

compliant OGC resources (WMS, WFS, Context). AskTheSpider is build with IONIC RedSpider Studio on top of IONIC RedSpider Catalog. Available at <http://www.askthespider.com/>

**Cadcorp GeognoSIS.NET:** MassGIS data ([www.state.ma.us/mgis](http://www.state.ma.us/mgis)) served using Cadcorp GeognoSIS.NET Web Feature Service. MassGIS data ([www.state.ma.us/mgis](http://www.state.ma.us/mgis)) served using Cadcorp GeognoSIS.NET Web Map Service. Available at <http://www.cadcorp.com>

**CCRS GeoGratis Warehouse:** The Canada Centre for Remote Sensing (CCRS) Web Map Service (WMS) provides an OpenGIS Consortium (OGC) compliant Web Map Service interface as an online web mapping service of various projects, such as GeoGratis, GeoBase, and other CCRS information holdings. The CCRS WMS also allows for custom cartographic styling through the OGC Styled Layer Descriptor (SLD) Specification. Available at <http://demo.cubewerx.com/demo/cubeserv/cubeserv.cgi>

**CubeWerx CubeSERV WMS:** CubeWerx Cascading Map Server with world-wide VMap Level 0 and other data stores. CubeWerx Cascading Map Server (CubeSERV) supports all versions of OGC WMS specifications (1.0.0, 1.0.8, 1.1.0, 1.1.1, 1.1.2 and 1.1.3) and can chain map requests to WMSs from other vendors using any or all of these specifications. CubeSERV also supports the SLD specification version 1.0.0 and 1.0.20. Available at <http://demo.cubewerx.com/demo/cubeserv/cubeserv.cgi>

**Client:** CubeWerx CubeXPLOR WMS viewer client accessing CubeWerx cascading map server. Available at <http://demo.cubewerx.com/demo/cubexplor/cubexplor.cgi>

**Deegree:** Deegree is a Java framework for geospatially-enabled solutions. It is based on common GI standards and allows building applications with spatially referenced content. Deegree components can be used to either develop a standalone desktop mapping solution to be locally installed on a user's machine, or to set up a highly distributed and service-based infrastructure. As the whole architecture of deegree is based on OGC specifications and concepts, there are no problems to integrate standardized products of other vendors (e.g. ArcIMS by ESRI(c)). Available at <http://deegree.sourceforge.net/>

**Client:** Available at <http://demo.deegree.org/>

**FEMA Q3 Flood Data:** This WMS service provides the nation-wide FEMA Q3 Flood Map. The Q3 Flood Data are derived from the Flood Insurance Rate Maps (FIRMs) published by the Federal Emergency Management Agency (FEMA). The file is georeferenced to the earth surface using the EPSG:4326 (latitude/longitude coordinate system). Specifications for the horizontal control of Q3 Flood Data files are consistent with those required for mapping at a scale of 1:24000. The map service is for advisory purposes only. Available at <http://www.hazardmaps.gov/wmsRequest.php>

**GeoServer:** The GeoServer project is a Java implementation of the OpenGIS Consortium's Web Feature Server specification. It is free software. Available at <http://geoserver.sourceforge.net/html/index.php>

**IGeoPortal:** This is the first release of the deegree iGeoPortal. The new client/portal component of deegree is a modular client which configuration is based on OGC Web Map Context specification/document. Different modules can offer web map client functionality as



well as functions for gazetteer clients, catalog clients or WFS clients. Available at <http://deegree.sourceforge.net/src/demos.html#client>

**JUMP:** The Java Unified Mapping Platform (JUMP) is a GUI-based application for viewing and processing spatial data. It includes many common spatial and GIS functions. It is also designed to be a highly extensible framework for developing and running custom spatial data processing applications. JUMP supports important industry standards such as GML and the OpenGIS Consortium spatial object model. Available at <http://www.vividsolutions.com/jump/>

**MassGIS Shared Services Geocoder:** MassGIS Shared Services Geocoder provides statewide address geolocation through a WFS 1.0.0 GetFeature interface. Available at <http://www.mass.gov/mgis/>

**Client:** Oliver - Java Web Start application for browsing maps and metadata from the MassGIS holdings. Also provides access to gazetteer of place names, geocoding services, and custom extraction of geodata. Available at <http://www.mass.gov/mgis/mapping.htm>

**MySQL Spatial:** "MySQL implements spatial extensions following the specification of the Open GIS Consortium (OGC). Available at [http://dev.mysql.com/doc/mysql/en/Spatial\\_extensions\\_in\\_MySQL.html](http://dev.mysql.com/doc/mysql/en/Spatial_extensions_in_MySQL.html)

**National Atlas of The United States – WMS:** Over 400 data layers covering geological, hydrological, biological, geographical, demographic, and agricultural geospatial information. Available at <http://nationalatlas.gov/natlas/wmsprocess.asp>

**Client:** Available at <http://nationalatlas.gov/natlas/natlasstart.asp>

**OpenMap:** BBN Technologies' OpenMap package is a JavaBeans based programmer's toolkit. Using OpenMap, you can quickly build applications and applets that access data from legacy databases and applications. OpenMap provides the means to allow users to see and manipulate geospatial information. This is OGC WMS compatible project. Available at <http://openmap.bbn.com/>

**PyOGCLib:** PyOGCLib aims to develop and distribute a Python based library for the implementation of the OpenGIS® specifications, notably Web Map Server (WMS) and Web Feature Server (WFS). By basing the project on Sourceforge, we hope to attract user and developers who are interested in improving and expanding implementation of the OpenGIS® specifications in a lightweight, purely Python fashion, prioritizing conformance, simplicity, portability and ease of use and maintenance. Available at <http://pyogclib.sourceforge.net/>

**USGS\_WMS\_XXX:** (XXX: WMS services listed below)

BTS\_Roads() : Bureau of Transportation Statistics, transportation dataset. Served by the USGS EROS Data Center.

[http://gisdata.usgs.net/servlet/com.esri.wms.Esrimap?servicename=USGS\\_WMS\\_BTS\\_Roads&](http://gisdata.usgs.net/servlet/com.esri.wms.Esrimap?servicename=USGS_WMS_BTS_Roads&)

LANDSAT7: USGS LandSAT7 imagery

[http://gisdata.usgs.net/servlet/com.esri.wms.Esrimap?WMTVER=1.1.0&ServiceName=USGS\\_WMS\\_LANDSAT7](http://gisdata.usgs.net/servlet/com.esri.wms.Esrimap?WMTVER=1.1.0&ServiceName=USGS_WMS_LANDSAT7)

NED: USGS National Elevation Dataset - Shaded Relief

[http://gisdata.usgs.net/servlet/com.esri.wms.Esrimap?servicename=USGS\\_WMS\\_NED&](http://gisdata.usgs.net/servlet/com.esri.wms.Esrimap?servicename=USGS_WMS_NED&)

NHD : USGS National Hydrography Dataset (NHD) including NHD Waterbodies, NHD Waterbody Reaches, NHD Transport Reaches and NHD Networks

[http://gisdata.usgs.net/servlet/com.esri.wms.Esrimap?servicename=USGS\\_WMS\\_NHD&](http://gisdata.usgs.net/servlet/com.esri.wms.Esrimap?servicename=USGS_WMS_NHD&)

NLCD : USGS NATIONAL LandCover Database

[http://gisdata.usgs.net/servlet/com.esri.wms.Esrimap?servicename=USGS\\_WMS\\_NLCD&](http://gisdata.usgs.net/servlet/com.esri.wms.Esrimap?servicename=USGS_WMS_NLCD&)

REF : Series of layers from USGS that can be used as basemaps

[http://gisdata.usgs.net/servlet/com.esri.wms.Esrimap?servicename=USGS\\_WMS\\_REF&](http://gisdata.usgs.net/servlet/com.esri.wms.Esrimap?servicename=USGS_WMS_REF&)

### **2.2.1.2 Client Efforts**

(Service requestor for the OGC compatible GIS services)

By the Clients we mean Catalog Client, Map Client, and Data Client. Please see the Figure 1.

**Chameleon:** Chameleon is a distributed, highly configurable, environment for developing Web Mapping applications. It is built on OGC standards for Web Mapping Services (WMS) and WMT Viewer Contexts. Available at <http://www.maptools.org/chameleon/>

**Fulcrum:** Fulcrum is a free Java library that includes user interface components, data models, and utilities useful to Java developers building distributed mapping applications. While the Fulcrum libraries may be useful for other purposes, it is currently targeted at creating applications that need to consume map data over a network. The source of the data can come from stand-alone open source or commercial map servers or from something more complex such as the OpenGIS Consortium (OGC) Web Services. Available at <http://fulcrum.traversetechnologies.com/>

**GIServer:** The GIServer is an initiative from the inovaGIS project that gives free access to GIS functions through the Internet. GIServer is compliant with the OGC WMS specs 1.0 to 1.1. Available at <http://www.inovagis.org/giserver/index.asp>

**Intergraph WMS Viewer:** Geospatial Intelligence portal with WRS search and Context capability. Can use Intergraph WRS or OGC Geospatial portal WRS implemented by Compusult. Available at <http://ogc.intergraph.com/webmapviewer/main.asp>

**J2ME OGC WMS Client:** J2ME OGC WMS Client is a program for accessing OGC Web map services from Java enabled mobile phone or PDAs. It accesses the OGC WMS according to WMS 1.1.0 and 1.1.1 specifications and supports zooming and panning in the Map, WMS sublayers, and a bookmark management system for quick access. Available at [http://www.boege.net/wmsclient\\_en.html](http://www.boege.net/wmsclient_en.html)

**Mapbender:** The Mapbender Client Suite software package provides user interfaces for displaying, navigating and querying OGC WMS compliant map services. The Mapbender Client Suite software furthermore contains interfaces for user and group administration and provides management functionality for accessing maps rendered by Web Map Services. Available at <http://www.mapbender.org/>

**NASA Web Map Viewer:** Basic HTML-only WMS client. Available at <http://viewer.digitalearth.gov/>

**Owsview Viewer Client Generator** : Owsview is a web-based thin client which supports discovery, access and visualization of supported specifications of the OpenGIS Consortium (OGC) and Canadian Geospatial Data Infrastructure (CGDI) endorsed specifications, such as Web Map Service (WMS), Web Feature Service (WFS), Web Coverage Service (WCS), Styled Layer Descriptor (SLD), Web Registry Service (WRS), Web Map Context Documents, Catalog Service, Gazetteer Service, Sensor Collection Service (SCS) and the GeoConnections Discovery Portal API (searching for products and services). owsview also exemplifies the benefits of a standards based services for chaining between services for discovery, access and visualization of information holdings. Available at <http://cgdi-dev.geoconnections.org/prototypes/owsview/index.html>

**QuickWMS**: JavaScript classes for creating Web Map clients and interfacing WMS servers according to OpenGIS Web Mapping Specification (versions 0.7 to 1.1). The goal of this project is to enable the fast creation of web mapping clients using javascript. The target browsers are Internet Explorer (version 5.5 and up) and Netscape (7.00 and up) both for Windows, Mac and Linux. Available at <http://www.inovagis.org/quickwms/>

## 2.2.2 Academic GIS application efforts and centers

**Indiana University - GIS at Indiana University**: Aerial photos and data for counties, cities, and localities. Miscellaneous vector data are provided. Interactive state maps to download data and topographic maps. Available at <http://www.indiana.edu/~gis/> and <http://www.indiana.edu/%7Egisdata/>

**MIT - MITOrthoServer**: The MIT OrthoServer is a set of components that serve large collections of geo-image libraries online, in a seamless, multi-resolution fashion. OGC OpenGIS compatible. Available at <http://tull.mit.edu/orthoserver/>

**Ohio State University - Center for Mapping**: The Center for Mapping's mission includes research in the following topic areas: theoretical frameworks for GIS and spatial information, automated cartography, GIS representation, multi-media visualization, quality control, sensor integration/orientation, GPS/IMSU-based navigation, mobile mapping, photogrammetry, data fusion, experimental instrumentation, image processing, and image understanding. Available at <http://www.cfm.ohio-state.edu/>

**Penn State University**: The geospatial Information System Councils. GIS council facilitate GIS-related educational and research activities, provides state's spatial data. Available at <http://www.gis.psu.edu/>

**Purdue University - CAAGIS**: CAAGIS (Center for Advanced Applications in Geographic Information Systems). Provides Indiana state's topographic maps, aerial photos, digital elevation maps (DEMs) and provides interactive maps for the States spatial data. Available at <http://danpatch.ecn.purdue.edu/~caagis/>

**University of Alabama in Huntsville – SST**: Space Time Toolkit (STT) is a Java-based toolkit that provides advanced capabilities for integrating spatially and temporally-disparate data within a highly interactive 3D display environment. This is basically Web Mapping project. Available at <http://vast.uah.edu/SpaceTimeToolkit/index.html>

**University of California at Berkeley - GISViewer:** GIS Viewer is a web-based Java tool for displaying and manipulating layers of geographical points and vectors, and raster data such as maps and images. Available at <http://elib.cs.berkeley.edu/gis/>

**University of California at Santa Barbara – Alexandria Digital Library:** The Alexandria Digital Library (ADL) is a distributed digital library with collections of georeferenced materials. ADL includes the operational library, with various nodes and collections, and the research program through which digital library architectures, gazetteer applications, educational applications, and software components are modeled, prototyped, and evaluated. ADL provides HTML clients to access its collections and gazetteer, and provides specific information management tools, such as the Feature Type Thesaurus for classing types of geographic features, as well as downloadable software code.

**University of Minnesota - Mapserver:** MapServer is a CGI-based application for delivering dynamic GIS and image processing content via the World-Wide Web (WWW). The package also contains a number of stand alone applications for building maps, scale bars and legends offline. Access to the development environment of MapServer is possible with a number of different programming languages. MapServer supports several Open Geospatial Consortium web specifications: WMS (client/server), non-transactional WFS (client/server), WCS (server only), WMC, SLD, GML and Filter Encoding. OGC OpenGIS compatible. Available at <http://mapserver.gis.umn.edu/>

**University of Wisconsin-Madison - Paradise:** A Parallel Database System for GIS Applications. The objective of the Paradise project is to design, implement, and evaluate a scalable, parallel geographic information system that is capable of storing and manipulating massive data sets. Available at <http://www.cs.wisc.edu/paradise/>

**York University - OGC 3D Client:** Developed by GeoICT Lab at York University, OGC 3D Client has been successfully applied in OGC CIPI-1 initiative. As an server- and platform- independent client, its key features include: Pure Java Client, Compliance with OGC WMS and WCS, Accessing and operating DEM, Image and vector data through OGC WMS, WCS and XML, 2D zoom, pan, query, overlap, 3D zoom, rotation, query, profile, flood, etc. Available at [http://www.geoict.net/GSN\\_main\\_ogc.htm](http://www.geoict.net/GSN_main_ogc.htm)

### 2.2.3 Other non-OGC efforts

**ESRI:** ESRI produces tools and software packages that enable you to build intelligent geographic information systems. Some of them are for GIS Clients such as ArcReader, ArcView, ArcEditor, ArcInfo, and some of them for GIS servers such as ArcIMS, ArcGIS, ArcSDE, GIS Portal Toolkit. Available at <http://www.esri.com/>

**GDAL:** GDAL is a translator library for raster geospatial data formats that is released under an Open Source license. As a library, it presents a single abstract data model to the calling application for all supported formats. This service is related to file format conversion services. Available at <http://www.remotesensing.org/gdal/>

**Geographic information / Geomatics:** This is standardization in the field of digital geographic information. This work aims to establish a structured set of standards for information concerning objects or phenomena that are directly or indirectly associated with a location relative to the Earth. These standards may specify, for geographic information, methods, tools and services for data management (including definition and description), acquiring, processing, analyzing, accessing, presenting and transferring such data in digital/electronic form between different users, systems and locations.

Available at <http://www.isotc211.org/>

**GRASS:** GRASS GIS (Geographic Resources Analysis Support System) is an Open Source Geographical Information System (GIS) with raster, topological vector, image processing, and graphics production functionality that operates on various platforms through a graphical user interface and shell in X-Windows. This is related to Base GIS Visualization, Remote Sensing, Flights, File-format conversion, Projection conversion, Customizable with Add-ons. Available at <http://grass.itc.it/index.php>

**GeoTools:** Geo Tools is a free Java based mapping toolkit that allows maps to be viewed interactively on web browsers without the need for dedicated server side support. Available at <http://www.geotools.org/>

**GISToolKit:** The GISToolkit software is a java toolkit for building spatially enabled applications. It has some ability to read data from a variety of data sources, and to display that data. It can directly edit geographic features stored in databases to which it has access. Available at <http://gistoolkit.sourceforge.net/>

**GML4J:** GML4J is a Java API for facilitating work with the Geography Markup Language (<http://www.gmlcentral.com>). GML is an XML-based framework for encoding geography information adopted as a recommendation paper by OGC. Currently only support read access. This service is related to file format conversion services. Available at <http://gml4j.sourceforge.net/>

**KDEM:** kdem is a program for displaying United States Geological Survey (USGS) Digital Elevation Models (DEMs). This is related to Visualization, Interactive Viewing. Available at <http://www.mindspring.com/%7Ejamovers/kdem/>

**Mapyrus:** Mapyrus is software for creating plots of points, lines, polygons and labels to PostScript, PDF and web image output formats. The software combines the following three components: A Logo or turtle graphics language, reading of GIS datasets and RDBMS tables, running as a stand-alone program or as a web-server. Available at <http://mapyrus.sourceforge.net/>

**NetMaps:** NetMaps is a Java applet that allows one to view vectorial maps in any Java enabled browser. NetMaps can load and display ArcInfo shapefiles (SHP/DBF) and MapInfo MIF/MID files. This is basically web mapping project. Available at <http://www.sitex.ro/netmaps/>

**OpenSVG Mapserver:** An open source solution for publishing ArcView shapefiles with attributes to the web based on html, SVG, javascript, php and mysql database. It supports interactivity and filtering. Available at [http://www.carto.net/projects/open\\_svg\\_mapserver/](http://www.carto.net/projects/open_svg_mapserver/)

**Thuban:** Thuban is an Interactive Geographic Data Viewer with the following features: 1) Navigation Zoom In/Out, Pan 2) Identify Attributes by object selection, objects by record selection. 3) Layer Management Layer types: Line, Polygon, Point, Georeferenced Image 4) Legend Editor Visual appearance of objects can be controlled. 5) Table Management Query and join tables. 6) Printing Print and export maps for further processing. Available at <http://thuban.intevation.org/>

**Terralib:** TerraLib is a GIS classes and functions library, allowing a collaborative environment and its use for the development of multiple GIS tools. TerraLib aims to provide a large set of data structures and algorithms for GIS developers. Available at <http://www.terralib.org/>

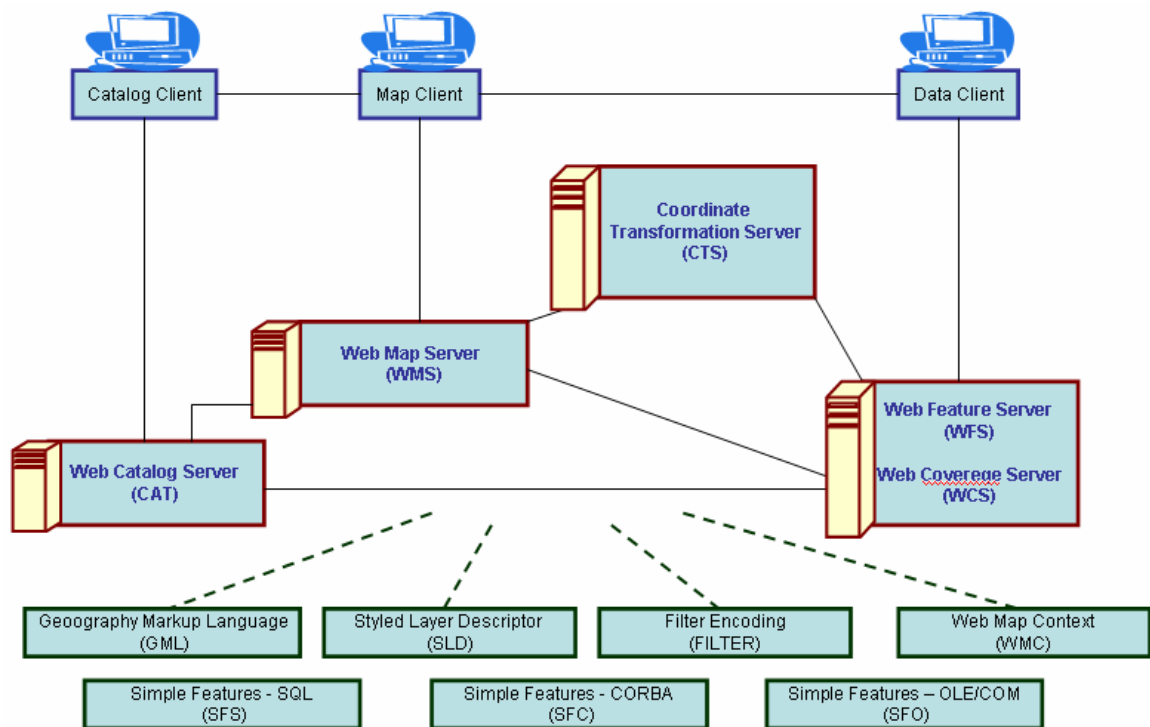
**USGS (US GEOLOGICAL SURVEY):** is responsible for building, maintaining, and applying *The National Map*. USGS provides scientific information to describe and interpret America's landscape by mapping the terrain, monitoring changes over time, and analyzing how and why these changes have occurred. USGS behaves as a geographic data provider for the other GIS projects and applications. Available at <http://store.usgs.gov/>

**WinDisp:** Windisp is software package for the display and analysis of satellite images, maps and associated databases, with an emphasis on early warning for food security. WinDisp was originally developed for the FAO Global Information and Early Warning System. Available at <http://www.fao.org/giews/english/windisp/windisp.htm>

### 3 OGC OpenGIS Specifications

OpenGIS is the activity pursued by the OGC to form bases of the interoperability between GIS services such as mapping services, data services, and portrayal services. OpenGIS tries to achieve its interoperability aims by providing a rich suite of open interface and implementation specifications. Some of these specifications are used in our GIS project and explained in Chapter 3.1 in detail but the other approved specifications will be mentioned in this chapter roughly. These interface specifications will enable GIS developers to create interoperable components.

OGC OpenGIS Specifications enables you to get, mix and match your GIS services from multiple sources over the web. [29] These sources might be from different vendors and different geographic areas but they must be implemented according to approved OGC OpenGIS specifications. The approved OGC specifications are displayed in the Figure 1.



**Figure 1 :** Approved OpenGIS Specifications. Figure is created from the specification definitions at <http://www.opengeospatial.org/specs/?page=specs>

**Web Catalog Server:** Defines a common interface that enables diverse but conformant applications to perform discovery, browse and query operations against distributed and potentially heterogeneous catalog servers. Please see the Chapter 3.1.3.3 for more information [16].

**Coordinate Transformation Server:** Provides interfaces for general positioning, coordinate systems, and coordinate transformations. [34]

**Geography Markup Language:** The Geography Markup Language (GML) is an XML encoding for the transport and storage of geographic information, including both the geometry and properties of geographic features. Please see the Chapter 3.1.3.1 for more information. [10]

**Simple Features - CORBA :** The Simple Feature Specification application programming interfaces (APIs) provide for publishing, storage, access, and simple operations on Simple Features (point, line, polygon, multi-point, etc). [32]

**Simple Features – SQL:** The Simple Feature Specification application programming interfaces (APIs) provide for publishing, storage, access, and simple operations on Simple Features (point, line, polygon, multi-point, etc). [30]

Simple Features – OLE/COM: The Simple Feature Specification application programming interfaces (APIs) provide for publishing, storage, access, and simple operations on Simple Features (point, line, polygon, multi-point, etc). [31]

Styled Layer Descriptor : The SLD is an encoding for how the Web Map Server (WMS 1.0 & 1.1) specification can be extended to allow user-defined symbolization of feature data. [6]

Web Coverage Server: Extends the Web Map Server (WMS) interface to allow access to geospatial "coverages" that represent values or properties of geographic locations, rather than WMS generated maps (pictures). [13]

Web Feature Server: The purpose of the Web Feature Server Interface Specification (WFS) is to describe data manipulation operations on OpenGIS® Simple Features (feature instances) such that servers and clients can “communicate” at the feature level. Please see the Chapter 3.1.3.2 for more information. [7]

Web Map Context: Create, store, and use "state" information from a WMS based client application. [33]

Web Map Server: Provides three operations protocols (GetCapabilities, GetMap, and GetFeatureInfo) in support of the creation and display of registered and superimposed map-like views of information that come simultaneously from multiple sources that are both remote and heterogeneous. Please see the Chapter 3.1.1 for more information. [4]

Filter Encoding: A filter is a construct used to describe constraints on properties of a feature class for the purpose of identifying a subset of feature instances to be operated upon in some way. [35]

In addition to these Approved Specifications [37] there are other Abstract Specifications [38], Recommendation Papers [39], and Discussions Papers [40].

## **3.1 OGC OpenGIS Specifications we have been using in our GIS project**

In Chapter 3 we give general information about the OGC OpenGIS specifications. Here in this chapter there will be more detailed information about the specifications which we have been using in our GIS project.

### **3.1.1 WMS Specifications**

Web Map Service produces maps from geographic data. A map is not the data itself. Maps create information from raw geographic data. Maps are generally rendered in pictorial formats such as jpeg, GIF, png. WMS also produce maps from vector-based graphical elements in Scalable Vector Graphics (SVG) [26].



Basically 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, GetFeatureInfo. For the detailed explanations about the descriptions and compatibility requirements for these requests please see Chapter 3.1.2. 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 [5]. 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. Unless we say SLD-enabled all the word WMS used in this document represent for basic WMSs.

The distributed computing platform supported by the OGC WMSs is HTTP. 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. Detailed parameter lists, parameter types and sample parameters are given in Chapter 3.1.2.

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]

### 3.1.2 WMS Client

There is no official specification defined for the Web Map Client. Clients are thin clients and just invoke WMS operations by submitting requests in the form of Uniform Resource Locators (URLs). Users can set the required parameters by getting the values from the users by using user interface created for them. User interfaces can be created using jsp, JavaScript etc.

There are three operations defined by WMS specifications. GetCapabilities and GetMap are required and GetFeatureInfo is optional. For the detailed information about WMS and its operations please see Chapter 3.1.1 and/or OGC WMS specifications [4]. WMS clients interact with the existing WMSs by using these interfaces which are implemented as Java Servlets. Interfaces, operations and requests for these operations are well defined in the OGC specifications. Requests should have some parameters in defined formats and numbers. Below we will explain the requests for these three operations briefly in tabular fashion.

#### *GetCapabilities Request:*

GetCapabilities request is for obtaining service metadata, which is machine readable description of the information of the servers.

OGC Required Parameters	Description	Example Value
VERSION	Requested Version	1.1.1
SERVICE	Service Type	WMS
REQUEST	Request Name	getCapabilities
FORMAT	Output format of service metadata	text/plain
UPDATESEQUENCE*	Sequence Number or string for cache control	Any string value

**Table-1:** The Parameters of the GetCapabilities Request.

(\*) values are not implemented in our project

Sample GetCapabilities request:

<http://toro.ucs.indiana.edu:8080/deegree/wms/?REQUEST=GetCapabilities&VERSION=1.1.1&SERVICE=WMS>

Please see Chapter 4.2 and Figure 2 to see how to invoke getCapabilities operations when WMSs are implemented as Web Services.

### GetMap Request

It basically returns a map in specified format. Format is specified in getMap request by the WMS Client. Upon receiving getMap request WMS either sends a map or an exception.

OGC Required Parameters	Description	Example Value
VERSION	Request version	1.3.0
REQUEST	Request name	getMap
LAYERS	Comma-separated list of one or more map layers	layer_list (Indiana:rivers)
STYLES	Comma-separated list of one rendering style per requested layer.	Style_list(user-defined) (ex:Blue-broken)
CRS	Coordinate reference system	namespace:identifier (ex:EPSG:4326)
BBOX	Bounding box corners (lower left, upper right) in CRS units	minx,miny,maxx,maxy (ex:3,4,78,45)
WIDTH	Width in pixels of map picture	300
HEIGHT	Height in pixels of map picture	400
FORMAT	Output format of map	Image/jpeg
TRANSPARENT	Background transparency of map (default=FALSE).	true
BGCOLOR	Hexadecimal red-green-blue color value for the background color (default=0xFFFFFF)	0xff8ff
EXCEPTIONS*	The format in which exceptions are to be reported by the WMS (default=XML).	application/vnd.ogc.se_image
TIME*	Time value of layer desired	20030409
ELEVATION*	Elevation of layer desired	1000
Other sample dimension(s)	Value of other dimensions as appropriate	

**Table-2:** The Parameters of the GetMap Request

(\*) values are not implemented in our project

Sample GetMap request:

[http://toro.ucs.indiana.edu:8080/deegree/wms?service=WMS&VERSION=1.1.1&REQUEST=GetMap&LAYERS=Indiana:county83,Indiana:railroad83&STYLES=default&SRS=EPSG:4326&BBOX=528343.087726371,4434873.868501969,564476.4342992618,4478218.803239176&WIDTH=400&HEIGHT=300&FORMAT=image/jpg&BGCOLOR=0xffff8f&TRANSPARENT=true&EXCEPTIONS=application/vnd.ogc.se\\_inimage](http://toro.ucs.indiana.edu:8080/deegree/wms?service=WMS&VERSION=1.1.1&REQUEST=GetMap&LAYERS=Indiana:county83,Indiana:railroad83&STYLES=default&SRS=EPSG:4326&BBOX=528343.087726371,4434873.868501969,564476.4342992618,4478218.803239176&WIDTH=400&HEIGHT=300&FORMAT=image/jpg&BGCOLOR=0xffff8f&TRANSPARENT=true&EXCEPTIONS=application/vnd.ogc.se_inimage)

Please see Chapter 4.2 and Figure 3 to see how to invoke getMap operations when WMSs are implemented as Web Services.

### *GetFeatureInfo Request*

GetFeatureInfo is an optional operation. This operation is supported for only the layers whose attribute 'queryable' set to 1. A client can not make GetFeatureInfo request for the other layers. Clients need this operation to get more information about the feature displayed as a map on the screen. Returned format is defined in WMS capabilities document. Table-3 gives you more information about the getFeatureInfo request and its parameters.

OGC Required Parameters	Description	Example Value
VERSION	Request version	1.3.0
REQUEST	Request name	GetFeatureInfo
map request part	Partial copy of the Map request parameters that generated the map for which information is desired	From getMap request
QUERY_LAYERS	Comma-separated list of one or more layers to be queried	layer_list (ex: Indiana:county83)
INFO_FORMAT	Return format of feature information (MIME type).	text/xml
FEATURE_COUNT	Number of features about which to return information (default=1).	1
I	coordinate in pixels of feature in Map CS	34
J	coordinate in pixels of feature in Map CS	67
EXCEPTIONS*	The format in which exceptions are to be reported by the WMS (default= XML).	application/vnd.ogc.se_inimage

**Table-3:** The Parameters of the GetFeatureInfo Request

(\*) values are not implemented in our project. The others are not completed totally.

Sample GetFeatureInfo request:

[http://toro.ucs.indiana.edu:8080/deegree/wms?REQUEST=GetFeatureInfo&WIDTH=640&HEIGHT=480&BBOX=-110.,40.,-80.,30.&VERSION=1.1.1&SRS=EPSG:4326&QUERY\\_LAYERS=Indiana:county83&X=12&Y=165](http://toro.ucs.indiana.edu:8080/deegree/wms?REQUEST=GetFeatureInfo&WIDTH=640&HEIGHT=480&BBOX=-110.,40.,-80.,30.&VERSION=1.1.1&SRS=EPSG:4326&QUERY_LAYERS=Indiana:county83&X=12&Y=165)

Please see Chapter 4.2 and Figure 4 to see how to invoke getFeatureInfo operations when WMSs are implemented as Web Services.

### 3.1.3 Other OGC Specifications

In this chapter, there will be brief explanations about three different specifications defined by the OGC. These are Geographic Markup Language (GML) Specifications, Web Feature Service (WFS) specifications, and Catalog Services (CAT) specifications. The reason that we are explaining these three specifications in a WMS related document is that they have some interconnections with WMS. In this section we will mention about the OGC requirements for these services. Interactions of these services with the WMS will be explained in the section 5.1.3.

#### 3.1.3.1 Geographic Markup Language (GML) Specification

GML is an XML encoding for the transport and storage of geographic information, including both the spatial (attributes) and non-spatial (geometric) properties of geographic features. XML feature instances which are compliant to this specification shall validate against a conforming application schema. A conforming application schema shall import the Geometry schema (geometry.xsd) the Feature Schema (feature.xsd) and the XLinks schema (xlinks.xsd) as base schemas. [10]

A feature is an abstraction of a real world phenomenon; it is a geographic feature if it is associated with location relatives to the earth such as faults, rivers, roads, lakes. All the feature instances should be created according to these schema files and encoded in GML.

WMS requests feature data from the WFS in the form of GML. WFS store and serve the geospatial data encoded in GML. Since these schema files are too long we did not put them in this document. To give an idea about how to visualize a simple feature we just documented geometry.xsd to some degree of detail in APPENDIX-6

For the schema files please see the GML specification document. [10]

#### 3.1.3.2 Web Feature Service (WFS) Specifications

WFS store and serve the geospatial data encoded in GML. The WFS operations support INSERT, UPDATE, DELETE, QUERY and DISCOVERY operations on geographic features using HTTP as the distributed computing platform. You can create, delete or update a feature instance, get or query features based on spatial and non-spatial constraints.

WFS services are grouped into two categories, Basic WFSs and Transaction WFSs.

Basic WFSs provide GetCapabilities, DescribeFeatureType and GetFeature operations. These operations are explained below.

*GetCapabilities:* A web feature service must be able to describe its capabilities. Specifically, it must indicate which feature types it can service and what operations are supported on each feature type.

*DescribeFeatureType:* A web feature service must be able, upon request, to describe the structure of any feature type it can service.

*GetFeature:* A web feature service must be able to service a request to retrieve feature instances. In addition, the client should be able to specify which feature

properties to fetch and should be able to constrain the query spatially and non-spatially. [7]

Transaction WFSs support all the operations of Basic Web Feature Services and implement transaction operations. Transaction WFSs can also implement LockFeature operation optionally. These additional operations are described below.

*Transaction:* A web feature service may be able to service transaction requests. A transaction request is composed of operations that modify features; that is create, update, and delete operations on geographic features.

*LockFeature:* A web feature service may be able to process a lock request on one or more instances of a feature type for the duration of a transaction. This ensures that serializable transactions are supported. [7]

We have been implementing just Basic WFS. From now on by the WFS we will mean Basic WFSs.

### **3.1.3.3 Catalog Services (CAT) Specifications**

Catalogue services support the ability to publish and search collections of descriptive information (metadata) for data, services, and related information objects. Metadata in catalogues represent resource characteristics that can be queried and presented for evaluation and further processing by both humans and software. Catalogue services are required to support the discovery and binding to registered information resources within an information community. [16]

According to specification WMS and WFS can make metadata update on CAT. WMS and WMS Client can make metadata search on CAT. CAT stores and serves metadata about the services. CAT is too complicated and need to be explained in detail. Since we have not implemented it and used our own catalogue service (IS-Information Discovery Service) [12] we will finish the explanation of the OGC CAT here.

## 4 Applicability of Web Service Architecture on to OGC GIS Services

We have implemented standard three operations of OpenGIS as Web Services. These are getCapabilities service, getMap service and getFeatureInfo service.

Web Services give us a means of interoperability between different software applications, running on a variety of platforms. A Web Services support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format. Web Service interface are described in a file called Web Service Description Language (WSDL) [23] file. Other systems interact with the Web Service in a manner as described in WSDL using Simple Object Access Protocol (SOAP) messages.

SOAP [22] is an access protocol for exchanging the information in distributed environment. It is an XML based protocol and has three parts for message exchange. 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.

WSDL files are written as XML documents. WSDL is used for describing and locating Web Services. Web Services are defined by 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.

We have used *Apache Axis* for creating and publishing the Web Service. *Axis* takes care of the SOAP communication between server and the client. Axis 1.1 has proven itself to be a reliable and stable base on which to implement Java Web Services.

The premise of the SOAP experiment is the belief that porting OGC services to Web Services will offer several key benefits, including:

*Distribution:* It will be easier to distribute geospatial data and applications across platforms, operating systems, computer languages, etc.

*Integration:* It will be easier for application developers to integrate geospatial functionality and data into their custom applications.

*Infrastructure:* The GIS industry could 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. [15]

*Easy to extend:* Web Services have lots of specifications. By using any configuration of these specifications we will make our implementation more secure, more robust and fault tolerant as Web Service specs improve themselves.

## 4.1 OGC Compatibility with the 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 (DCP). Web Map Service 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. But they have not added this capability into their specification documents yet.

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), with convention for Remote Procedural Calls (RPCs) and a serialization mechanism based on XML Schema data types. SOAP is being developed by W3C in cooperation with the Internet Engineering Task Force (IETF). [17]

In the current version of the WMS, requests are done by the user with HTTP GET and POST, and Web Map Server has main java Servlet to respond these requests. Since our distributed computing platform will be SOAP over HTTP we will be implementing map services as Web Services. It seems that it is a conflict but without violating the specification we solved this conflict and explained the architecture details in this document. This is a partial solution and we used cascading WMS concept defined in OGC Specifications.

In addition to fundamental conflict encountered above, we had encountered some minor technical problems.

For the efficiency we made a decision about using the castor source files to manipulate all the XML files in the project. Data binding frameworks such as *Castor* [20] or *XMLBeans* [21] take XML Schemas as input and produce java sources. But one major problem with these frameworks is that sometimes it is not easy to find an object oriented correspondence of the XML Schema constructs. In such cases either the source codes can not be generated or generated source codes may not produce correct XML instances.

Some of the XML Schema types -such as substitutions and abstract types- used in OGC Schemas are currently not supported by Castor. We had to make several changes to make these schemas compatible with *Castor Source Generator*. Modifications are done just for the latest versions of the schema files of the GIS services. These modified schemas are available at <http://complexity.ucs.indiana.edu/~asayar/ogc/modified/>.

In general OGC Schemas use substitution to express a group of elements can be used interchangeably. Since Castor does not support substitution groups we had to find a way around this problem and we have used choices to solve the problem.

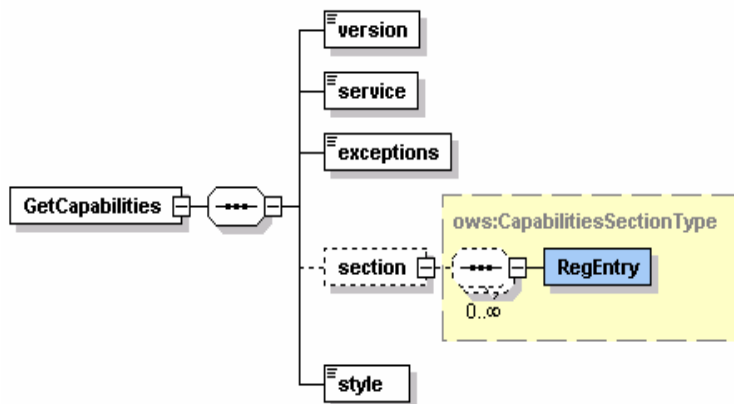
## 4.2 How to create valid requests to WMS in case of using Web Services

Invoking WMS operations should be according to specifications. OGC compatible requests to WMS are well defined in the WMS specifications and we mentioned from them in Chapter 3.1. Requests must have some parameters whose names, numbers, and values assigned to them should obey the rules defined in the specifications to be OGC compatible. In this chapter we have tried to define these requests in the schema files to use them in accordance with the GIS services implemented as Web Services.

These schema files are created to be used during the invocation of operations implemented as Web Services at the WMS side. [15] 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. 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.

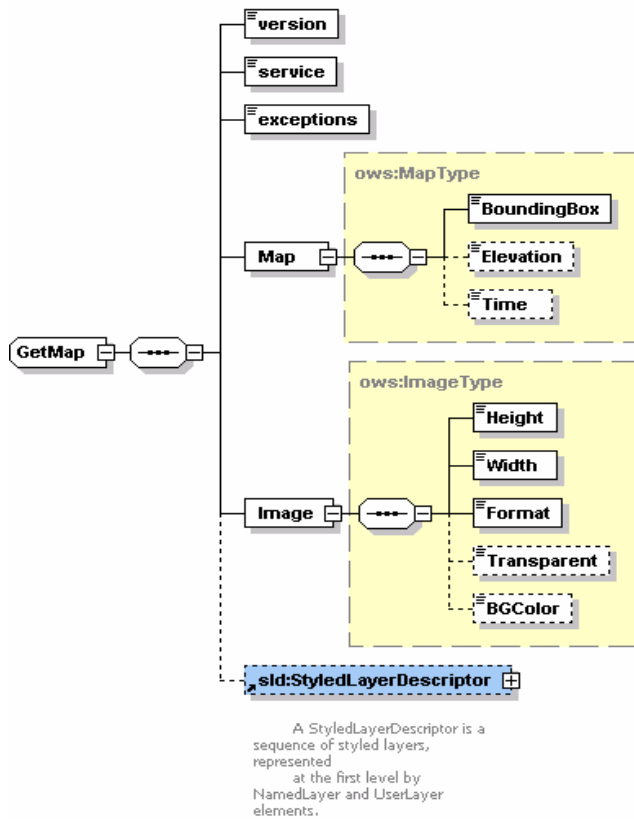
Below schema files are created with the help of Altova XmlSpy.

To see the sample Requests according to Figure 2, Figure 3 and Figure 4 please see APPENDIX-3.



**Figure 2 :** *GetCapabilities Request Schema (GetCapabilities.xsd)*

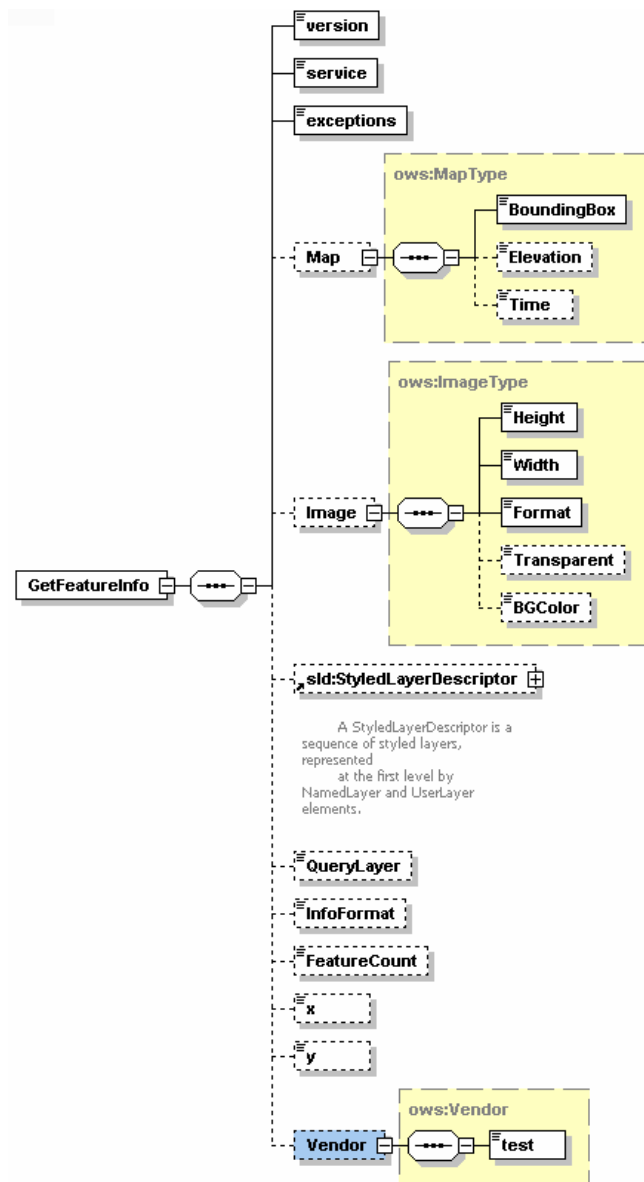




**Figure 3 :** GetMap Request Schema (GetMap.xsd)

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 Chapter 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-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 requests. For defining styling in the getMap request we use StyledLayerDescriptor element. StyledLayerDescriptor has other sub elements and attributes. For more information please see APPENDIX-2.



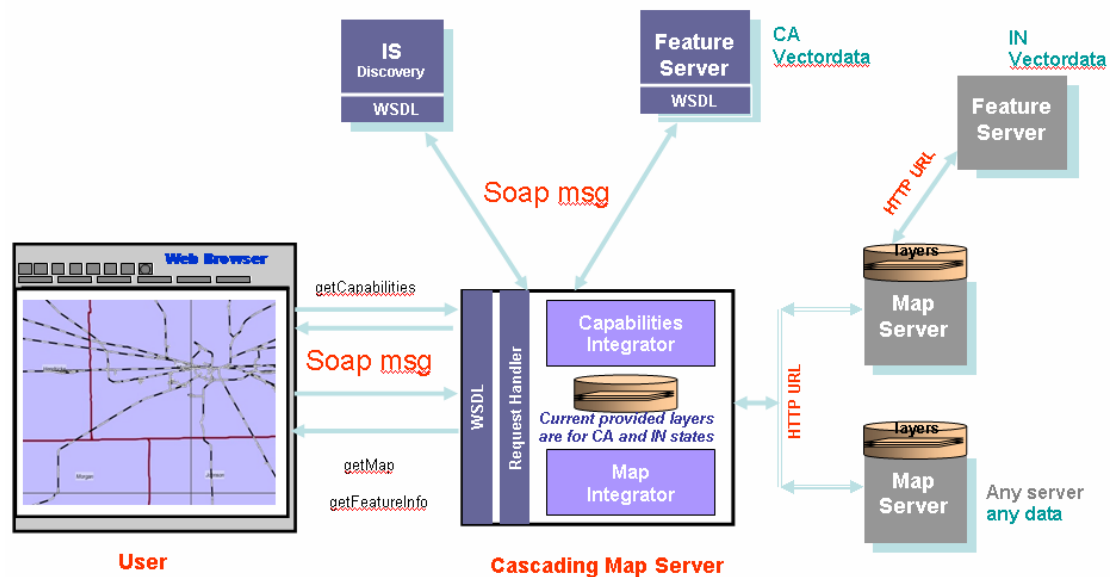
**Figure 4 :** *GetFeatureInfo Request Schema (GetFeatureInfo.xsd)*

## 5 Developing Web Service Compatible Versions of OGC Services

We have tried to combine our Web Services based implementation of GIS systems with the third party GIS systems. Third party systems use HTTP as distributed computing platform. This chapter explains the implementation details of the OGC specifications mentioned in Chapter 3.1.

### 5.1 Implementation of OGC Specs

This section is a kind of explanation of Figure 5. OGC specifications are roughly described in chapter 3.1. In this chapter we will describe all the functionalities of the elements in Figure 5.



**Figure 5 :** OGC Compatible GIS Web Services Architecture. Some of the WFSs are service oriented some are not. Instead of OGC's CAT, we have used service oriented IS-Discovery implemented by CGL.

### 5.1.1 WMS Implementation

WMS gets three different requests. We tried to explain what it does when it gets these requests. We also tried to explain how, when and why it needs operations of other OGC Services such as Web Feature Services and Catalog Services. All the functionalities of our OGC compatible WMS applications are grouped into different categories by the questions below.

*What does WMS do when it receives*

*1- getCapabilities Request?:*

WMS keeps its capability document in local file system. When a getCapability request comes, WMS reads its capability document from a file into a String object and returns it back to WMS client. An example of a WMS capabilities file is given in APPENDIX-5. This part of implementation is totally completed.

*2- getMap request?:*

WMS first parses the parameters and their values from getMap request coming from WMS Client. Depending on these parameters it might need to make some other requests to some other GIS services (see Chapter 5.1.3). WMS first defines what layers are requested in which boundingbox in which form etc. After defining 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 IS Web Service requests. To be able to call IS services we should first create service client stubs. getAccess\_point returns the wsdl address of the WFS that provides the requested feature. WMS makes getFeature request to this WFS and gets the requested feature data in GML format. If the format parameter in getMap request is Scalable Vector Graphics (SVG), then WMS creates SVG from returned feature data by using its geometry elements. After creating SVG file, we can easily convert the SVG file into any other image formats such as png, gif, jpeg etc. Apache Batik provides libraries for this conversion. Batik is a Java(tm) technology based toolkit for applications or applets that want to use images in the SVG format for various purposes, such as viewing, generation or manipulation. Creating an SVG file from geometry elements of the received feature data is a little hard. Schema files for the geometry elements are well defined. By using these schema files we created elements to visualize the feature data. These elements are basically Point, Polygon, LineString, LinearRing, MultiPoint, MultiPolygon, MultiGeometry etc.

There are basically two main ways to create an image from the simple features. First one is to create SVG file and convert it into any image format. Second is using java graphics2D libraries. First create graphics object then overlay another layers created as graphics object. We have been using both ways in different places but we saw that images drawn with graphics2d are with higher quality then the images drawn by SVG conversion. When first way is not necessary we will be using the second

approach mostly. Below you will see a sample code for giving some idea how to overlay two different layers from two different WMSs.

*Note: This is not a real code!:*

```
URL url = new URL( http://www.demis.nl/mapserver/request.asp?REQUEST=map&WMTVER=1.0.0&BBOX=-124.85.32.26.113.56.42.75&SRS=EPSG:4326&HEIGHT=300&WIDTH=400&FORMAT=JPEG&BGCOLOR=0xFFFFFF&LAYERS=Bathymetry.Countries.Topography&STYLES=.&TRANSPARENT=TRUE );
```

```
BufferedImage im = ImageIO.read(url);
Graphics2D g = im.createGraphics();
...
if(istherePoint)
    String[] points = getPointsFromFeatureData();
if(isthereLineString)
    String [] LineStrings = getLineStringFromFeatureData();
if(isthereLineRing)
    String [] LineRings = getLineRingFromFeatureData();
if(istherePolygon)
    String [] polygons = getPolygonsFromFeatureData();
...
...

if(polygons!=NULL){
    for(int i=0; i<polygons.length; i++){
        int [][] xypoints = wm.getXYpoints(polygons[i]);
        g.setColor(Color.darkGray);
        g.drawPolygon(xypoints[0], xypoints[1], xypoints[0].length
    }
}
if(LineRings!=NULL){
    for(int i=0; i<LineStrings.length; i++){
        int [][] xypoints = wm.getLinesInStr(LineStrings[i]);
        g.setColor(Color.darkGray);
        g.drawPolyline(xypoints[0], xypoints[1], xypoints[0].length
    }
}
...
g.dispose();
...
```

Check all the geometry data of the feature, Point, LineString Polygon etc.

If you find any geometry data above such as Points, LineStrings, convert the numbers in the GML file for the feature data into appropriate format to draw shapes for representing these geometry elements and display them by using graphics2D object. If you use the same graphics2D data the layers will be overlaid.

*Sample output:*



This code shows how to put two layers on each other by using cascading WMS approach defined according to OGC WMS specification. Here we are putting state boundaries layer over California Base map.

WMS gets 'state boundaries' data from our implementation of WFS. WFS provides feature data in vector format. This data is encoded in GML. California base map is created from coverage data. Since we have not implemented WCS (Web Coverage Service) so far, we are using third party WMS (from demis.nl)[25] to get maps created from coverage data. Cascaded WMS returns image in specified format as defined in the getMap request above. Since these two different layers from different WMSs have same boundingbox values and same width and height, they can be overlaid. *(Please note there are lots of details not shown in this code.)*

Since we have not implemented SLD-enabled WMS we have been using hard coding whenever we needed styling for the geographic data. For example for the river data we have used blue as color of the river data and for the boundary-lines feature data we have used darkGrey. You can see a simple example in the above code about how to make styling by using graphics2D libraries. When we implemented our WMS as SLD-enabled we will use more complex styling facilities and operations of the graphics class and graphics2D objects.

Most of the implementation is done but as described in WMS specification there are lots of details that should be handled. To make our WMS 100% compatible to OGC we need to handle these implementation details. We have finished almost 70% of the details mentioned in the specification. As we noted at Chapter 3.1.2 and Table2, there are some parameters in the getMap request that are not handled. Since these are not crucial elements (and not mandatory) in the request and most of the WMSs and WMS Clients do not use these parameters we postponed the implementation of these parts.

### 3- *getFeatureInfo request?*

We have not finished implementation yet but we are still working on it. We have finished almost 40% of the implementation. For the time being we are just getting the feature data as it is in the form of GML and display on the screen as a text. We are working on creating XSL (Extensible Stylesheet Language) [19] file to convert geographic data in the form of GML into appropriate good looking output.

When the WMS get getFeatureInfo request it makes a getFeature request to WFS. WFS provides these OGC compatible service interface as Web Services. getFeature operation is one of these operations that WFS provides. Since WMS is stateless it does not keep states of each WMS Client. WMS first find out what features are requested and then find out the WFS address which provides this feature. To figure out the WFS address WMS makes a search request to IS. IS defines its search criteria in its WSDL file. WMS gets the result from IS (service address), makes another request to WFS to get the feature in a given boundingbox. After receiving requested feature as

a string encoded in GML from WFS, WMS sends it to WMS Client in a format that is defined in the getFeatureInfo request.

*When does WMS make*

*1- getFeature request? (TO WFS):*

When WMS Client make a request for a layer which consists of vector data or simple feature data then WMS needs this data before starting to draw an image. To get this data it should make a getFeature request to WFS. WFS addresses are defined in the WMS capabilities document. Or if you use our IS implementation as catalog service, you can find WFS addresses by making find\_service and getAccess\_point requests to IS. In short, WMS makes this request to WFS when it receives getFeatureInfo and getMap request from the WMS Client.

*2- describeFeatureType request? (TO WFS):*

To be able to respond getFeatureInfo requests from WMS Clients, WMS should make a describeFeatureType requests to WFS. WMS makes this request to WFS to get schema of the specific feature data. After getting schema file WMS knows how to parse and get the information from the feature data encoded in GML.

*3- find\_service request? (TO IS):*

When WMS Clients make getFeatureInfo or getMap request to WMS, WMS needs to find out which WFSs provide the requested geographic data. When WMS needs to find out specific WFS service address, WMS makes find\_service request. WMS search the WFS service address by using the geographic data name as search criteria. To do that WMS sends its search request to IS. Search request includes search criteria. Search criteria can be just one property or more than one property. So far we have been using just one property. It is called as “feature\_name”. By giving feature name we search the catalog in IS and IS returns the WFS wsdl URL value providing this feature data. For example if we set the search criteria as feature\_name and give the value as “Indiana” IS is going to return WFS address for the this feature data. If there is more than one WFS providing any geographic data that their feature names include “Indiana” IS is going to return them in the String array.

#### **5.1.1.1 Cascading WMS**

To have a cascading WMS would mean that clients could ask a WMS for map layers, which the WMS does not serve by itself, but is able to receive from other WMS. The client would thereby not need to keep track of several WMS servers; it only has to be aware of one. The client simply asks the WMS for map layers and the WMS delivers the map layers to the client. If the information comes from the WMS server itself, or from a remote WMS, is not important to the client. As you will see in our application Indiana map set and California base map come from third party remote WMS servers. [4]

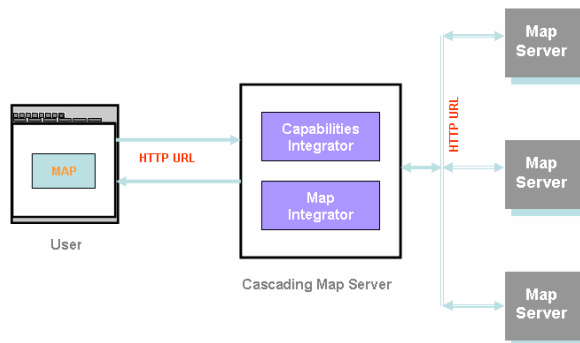
Having a cascading WMS would give an abstraction level towards the client and make it easy to change the system infrastructure behind the WMS server, through which the client has connection. Without the knowledge of the client, WMS and WFS servers can be added or removed. In the same way, WFS servers are possible to cascade. We have not implemented this kind of cascading yet.

A Cascading Web Map Server is a WMS which aggregates the contents of several individual WMSs into one service that can be accessed by clients. Cascading WMS acts like a client to the other WMSs and as a server to the clients.

When two or more maps are produced with the same Bounding Box, Spatial Reference System, and output size, the results can be accurately layered to produce a composite map. The use of image formats that support transparent backgrounds allows the lower Layers to be visible. Furthermore, individual map Layers can be requested from different Servers. The WMS GetMap operation thus enables the creation of a network of distributed map servers from which WMS clients can build customized maps.

A particular WMS provider in a distributed WMS network need only be the steward of its own data collection (In our application California data). This stands in contrast to vertically-integrated web mapping sites that gather all of the data in one place that is to be made accessible by their own private interface.

A cascading map server reports the capabilities of other WMS server(s) as its own and aggregates the contents 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 6 :** The concept of Cascading Web Map Server



## 5.1.2 WMS Client Implementation

There are two different map sets defined in the WMS capabilities document. These are California and Indiana. When client first time opened the client interface page, these two map sets are defined and listed in the "Select Layers for [Dropdown List]". Dropdown lists all this general map sets that cascading WMS provides. Under these general map sets there are some layers to be used creating maps. For the sample Capabilities file for the actual WMS please see APPENDIX-5.

When you select California map from the dropdown list, WMS client makes a new getCapabilities request to WMS. WMS sends all the available layers for these California map set. These layers are listed as select options, by selecting these layers you can overlay on each other and create different maps. Every time you add or remove a layer/ layers you should "Redraw map!".

Since we have been implementing thin clients for the regular users all the heavy tasks are assigned to WMS, especially cascading WMS. Cascading is a kind of proxy; gets requests from the users (WMS clients) interprets and handles the requests. WMS client even do not make rendering of the images. Images are sent by WMS in the ready to display format. I mean png, jpeg, gif or bmp. All these types are supported by the html to be shown on the screen. WMS Client just prepares valid requests according to specifications and wrap them in the SOAP messages (Please see Section 4.2) and waits WMS for its responses to these requests. Return types are defined in WMS capabilities document. All the requests and responses are over the SOAP and all the actions triggered by WMS client causes invoking a Web Service deployed on WMS.

To be able to create requests after getting required parameters from the users, WMS client is implemented by using jsp, JavaScript, cascading style sheets. Creating, parsing and editing capabilities documents, we have been using DOM and SAX. Since WMS clients use SOAP as Distribute Computing Platform, they prepare their requests as parameters for the remote service. Remote services are deployed as Web Services. Each type of request should comply with standards defined in specifications for these requests. To make verifying easy we have implanted requests in xml format. By encoding requests in xml, verifying will be easy. All the getCapabilities, getMap and getFeatureInfo requests should be created according to defined schema. These schemas created according the OGC specifications (Please see Section 4.2). Web Services for OGC compatible operations are deployed at <http://toro.ucs.indiana.edu:8086/deegreewms/test/services/WMSservices>

To be more familiar with the user interface please go to APPENDIX-4.

### *When does WMS Client make*

#### *1- getCapabilities request?:*

Two actions cause getCapabilities request, "Selecting Layers for [Map set dropdown list]" and first time launching the project page. When the project page first time opened Client makes an automatic getCapabilities request to define list of map sets available. Client can make this request more than one WMS or any WMS can be cascading WMS and provide information for more than one WMS. After defining and listing all the available map sets on the screen client makes second getCapabilities request when user selects any map set (here in our project Indiana or California

data). This request is done for defining sub layers under the parent layer which is map set. This sub layers are listed as options on the screen. User can select one or more than one layer to create a new map on the screen. GetMap request is created as an xml instance complies with the getMap request schema defined (Please see Section 4.2). To extract request parameters WMS and WMS client uses DOM libraries.

2- *getMap request?:*

All the actions except "Select Map Size", "Select Layers for ...." and calculating distance between two points on the map cause a getMap request with different parameters. For example, selecting different layers to be overlaid causes change in LAYER parameter, selecting zoom in, zoom out or panning map tools cause change in BBOX parameter in getMap request. If client does not change anything else the other parameters will be same. GetMap request is created as an xml instance complies with the getMap request schema defined (Please see Section 4.2). To extract request parameters WMS and WMS client uses DOM libraries.

3- *getFeatureInfo request?:*

This service is not mandatory. We started to implement but not finished yet. Two actions cause getFeatureInfo, 'i' in "Select Map Tool" and 'i' at the beginning of each name of sub layers. To understand what 'i' means please see the APPENDIX-4. The boolean attribute "queryable" indicates if the server supports getFeatureInfo operation on that layer.

GetFeatureInfo request is created as an xml instance complies with the getMap request schema (Please see Section 4.2). To extract request parameters WMS and WMS client uses DOM libraries.

*What does WMS Client do when it receives response to its*

1- *getCapabilities request?:*

It first makes some interoperability checking and determines if there is a version conflict. If everything is ok then tries to get list of available layers and sub layers of each parent layer (map set). When client gets response its capabilities request for the layers it parses the capabilities document and extract the layer information.

2- *getMap request?:*

Since getMap operation is implemented as a Web Service, client gets map as an attachment to SOAP message returned as a response to getMap request. When client get the response it creates a map file from the attachment and move it to the public directory to be shown on the screen. Client just gives the name of the map file in the html page.

3- *GetFeatureInfo request?:*

This service is not mandatory. We started to implement but not finished yet. According to specification the returned value will be in the form of GML, in other words in the form of xml. Clients can use its own XSLT [18] machine and creates html pages to be shown on the screen. XSLT is the most important part of the XSL [19] Standards (The Extensible Stylesheet Language). It is the part of XSL that is used to transform an

XML document into another XML document, or another type of document that is recognized by a browser, like HTML and XHTML. For this, client should create its own XSL file to create html page from returned xml file. If we implement this transformation in WMS transformation might be much easier. To be able to transform any feature into an html we also need to figure out the schema file for this feature type. We need to make one more request to get schema file. This request is done to WFS and it is called “describeFeatureType”. We will thing the implementation details later.

### **5.1.3 WMS Interactions in a Comprehensive GIS System**

WMS depends upon WFS and IS to accomplish its required tasks. These two services are being implemented by Galip Aydin and Mehmet Aktas correspondingly. They are working in CGL (Community Grids Lab.). This chapter will just mention what WMS requires from these two GIS services and how WMS interconnect with them.

#### **5.1.3.1 WFS**

WFSs keep geographic data and serves on request from clients. Clients to WFSs are Web Map Servers and other WFSs (in case of cascading WFS). WFSs provide vector data (not picture). Vector Data are encoded in GML (Geographic Markup Language). [10]. GML is 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 Web Feature Server is transactional than this WFS provides two more services. These are transaction and lockFeature services.

Since we have implemented basic WFS, WMS will be using basic WFS services, getCapabilities, describeFeatureType and getFeature. WMS sends a getCapabilities requests to learn which feature types it can service and what operations are supported on each feature type. Since I have been using IS service, I do not need to make a getCapabilities request. WMS makes its request to IS to get a specific WFS address that provides needed feature. Please see 3.2.3.2 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 to WFS. Address of WFS is found by using IS. All the services are implemented as Web Services. Service calls are made over SOAP. WFS make getFeature requests to get feature data from specific WFS. Sample request is shown in Figure 7. GML file in String will be returned 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 7 : Sample GetFeature request from WMS to WFS.*

### 5.1.3.2 IS (Information-Discovery Services) corresponds to OGC Catalog Services (CAT)

An OGC Catalog is a collection of descriptive information (metadata) regarding the data stored in a geographic database (see Chapter 3.1.3.3). OGC catalog service is too 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. These features of OGC Catalog service is good, however it is only scalable for data services. For instance, a registry should also allow discovery of services based on non-functional requirements of services such as Quality of Service attributes. Also, OGC Catalog Service is not consistent with other existing 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). An Information Service (IS) is a general Service Registry & Discovery model based on UDDI Specifications. 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 of Web Services and (WS-Context) [36] contextual information of GIS Services. Since ISs 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 Information Service in order to be "discoverable". Once the registry is completed, the IS starts interacting

with WFSs to retrieve more information about their capabilities. So, Information Service stores information about the functionality's of each WFS.

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

## 6 Current Status and Future Work

This chapter explains first current status of the project (what we have done) without giving any implementation details, and then gives explanations about what we plan to do in the near term and long term.

### 6.1 Current Status

The current status of the project is displayed in Figure 8. Since our implementation of Information Discovery Service (IS) is not OGC compatible we did not put it into this figure. As we mentioned before in section 5.1.3.2, IS has interactions with WMSs and WFSs. From the point of WMS, IS provides some useful operations such as searching for a specific feature data. IS provides these service interfaces as Web Services. Every element in Figure 8 part B has been implemented except for SLD [6] (shown as blur). For the implementation details please see Chapter 5.1.

Our implementation of WMS and WMS Client has some interactions with other WMSs and WFSs. These are third-party OGC compatible projects. For the test cases we have interacted with WMSs from deegree[24] and demis[25] projects. Their WMS and WFS implementations are OGC compatible. We have made a couple of successful tests and proved that our implementations are OGC compatible. For the demo please see the project home site [2].

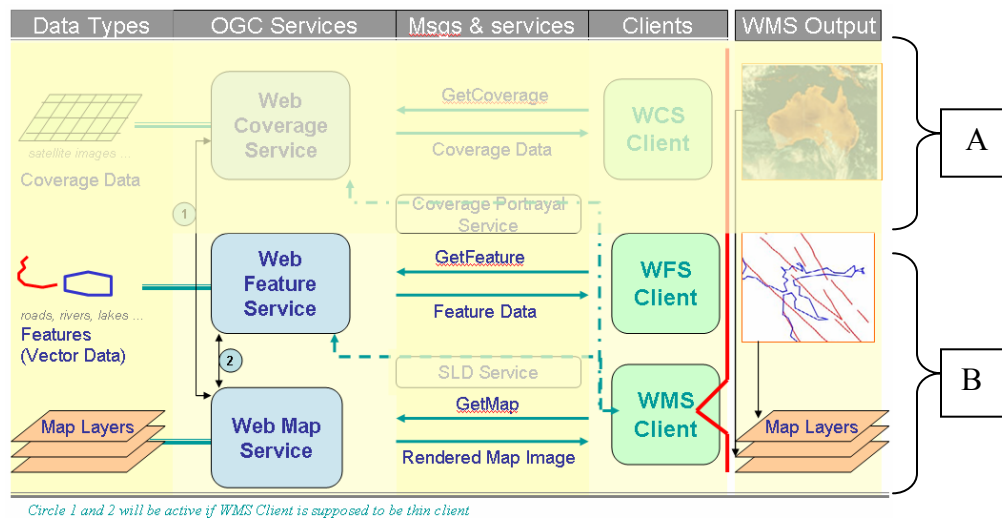
So far we have been implementing Web Map Service, Web Map Service Client, and Web Feature Service (implemented by Galip Aydin) and IS (implemented by Mehmet Aktas).

### 6.2 Near Term Work

The first goal in the near feature is overcoming the shortcomings in the *quality of the maps* and the services. Improving the user interfaces and adding some more functionality. To improve the quality of maps we are planning to implement Web Coverage Service (WCS), Web Coverage Service Client, Coverage Portrayal Service (CPS) and Styled Layer Descriptor (SLD) Service. All these services have corresponding OGC specifications and they should be implemented according to the specifications. After finishing implementation of these services we will deploy them into Figure 5 and *Figure 8*. All should have well defined user

interfaces and all the operations defined in their specifications should be implemented as Web Services. As we said before just distributed computing platform will be changed, all other details will be totally compatible with the OGC specifications. To be able to use operations as Web Services which are deployed on these OGC services, we should create appropriate schema files for the request objects as we did for WMS and WMS Client interconnections in Chapter 4.2.

We have pictured out these near future plans in Figure 8, **blurred** parts will be implemented soon.



**Figure 8 :** Progress picture for OGC compatible data retrieval operations. Architecture details are given in Figure 5.

Below you will see the general descriptions of the OGC compatible GIS services mentioned in Figure 8 and will be implemented in the near future. As we said above there are some challenges to implement these services. 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 (Figure 8 part A).

**WCS:** The *Web Coverage Service* (WCS) supports the networked interchange of geospatial data as "coverages" containing values or properties of geographic locations. Unlike the Web Map Service, which filters and portrays spatial data to return static maps (server-rendered as pictures), the Web Coverage Service provides access to intact (unrendered) geospatial information, as needed for client-side rendering, multi-valued coverages, and input into scientific models and other clients beyond simple viewers. The Web Coverage Service consists of three operations: GetCapabilities, GetCoverage, and DescribeCoverageType. [13]

**CPS:** A *Coverage Portrayal Service* produces a visual picture from data returned by a Web Coverage Service. This service will facilitate wider use of coverage data by making it accessible to thin clients. To the client, the Coverage Portrayal Service appears as a Web Map Service, but with additional parameters to control the retrieval and/or rendering of coverage

data. The Coverage Portrayal Service may require the client to specify the targeted Coverage Service.

SLD: (*Styled Layer Descriptor*) WMS describes the appearance of a map in terms of ‘styled layers’. A styled layer can be considered as a transparent sheet with features symbolized upon it. A map is made up of a number of these styled layers put together in a specified order. The styled layers are said to be Z-ordered. Users can define more complex or simpler maps by adding or removing styled layers. WMS services providing SLD capability said to be SLD-enabled WMSs.

A styled layer itself represents a particular combination of ‘layer’ and a ‘style’ in which that layer can be symbolized. Conceptually, the layer defines a stream of features and the style defines how those features are symbolized. This concept is underlined by the fact that there may be multiple styles in which a layer can be symbolized.

*Deployment of the project into Portal:*

We have already deployed the project into the Portal but we have some JavaScript bugs. It will be done soon. For the web site of the demo of the project see the references. [14]. If the web site is down see the APPENDIX-4 for the snapshots from the project portal page.

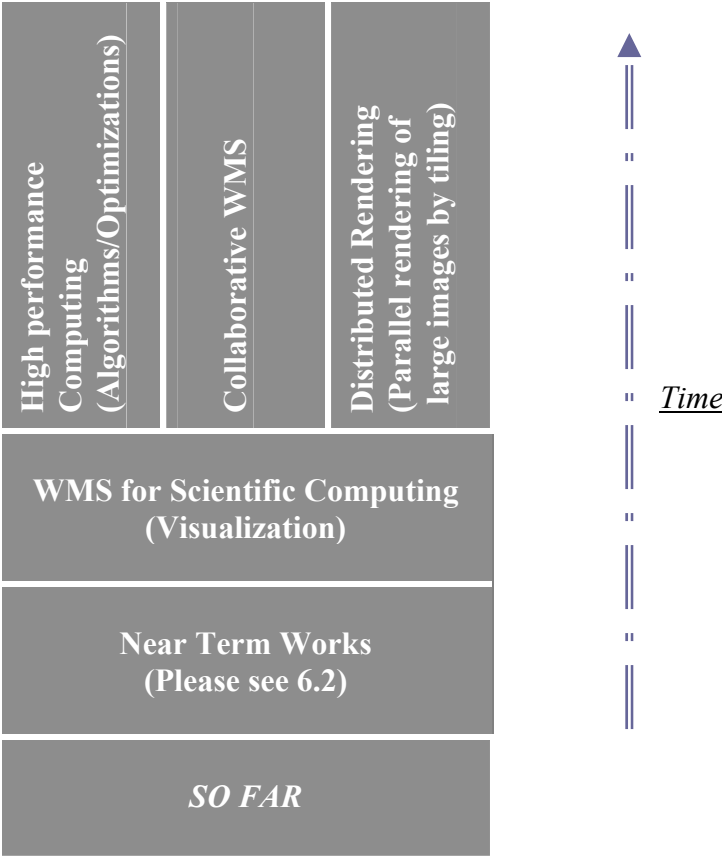
## **6.3 Longer Term Research and Development**

Future plan about the project is based on performance issues. To improve 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 Web Service oriented OGC compatible WMS services for scientific visualization. To be able to adopt 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 this end, we need to handle image pipelining, faster rendering, caching or client rendering. We might need to make clients thick client instead of thin that we have currently. This is a tradeoff between performance and easy to use.

Longer term project research steps are shown in Table-4.



**Table-4:** *Project Research Stack.*



## 7 Conclusion

With the development of spatial information application and the network technique, the spatial data between different districts and different departments need to be shared and to be interoperated. ISO/TC211 and OGC have done lots of work to define Interface specifications and standards to ensure sharing and interoperable capability of the spatial data.

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.

In this document we basically have been trying to explain the efforts spent on building OGC compatible GIS Services by using Web Service technologies and OGC specifications.

We can extend OGC OpenGIS specifications as much as we can but we need to consider the performance issue. This is a big problem in the GIS area. Since images can be too large, capabilities documents can be too large and transferring these data over the internet is a cumbersome, our first priority should be improving performance of GIS systems. Visualization can be slow as overlays or even basic maps large. Nowadays everyone working in this area has same problem. There are some ad-hoc solutions but not generic. To be more specific, when WMS capabilities become very large (it is highly possible), it can be inefficient to request and parse the full capabilities for some parts of an application. There are some characteristics of GIS services that make it difficult to design GIS Web Services with satisfactory performance. The most important ones

- Services provided by a GIS require heavy CPU usage due to complex computation involved in the underlying computational geometry.
- GIS services often transmit large resulting data sets such as images.
- The clients of the GIS Web Services are often some complex software tools.

By introducing Web Services technologies into GIS, we will take advantage of the Web Services. As Web Services technologies evolve our proposed GIS systems evolve.

# APPENDIXES

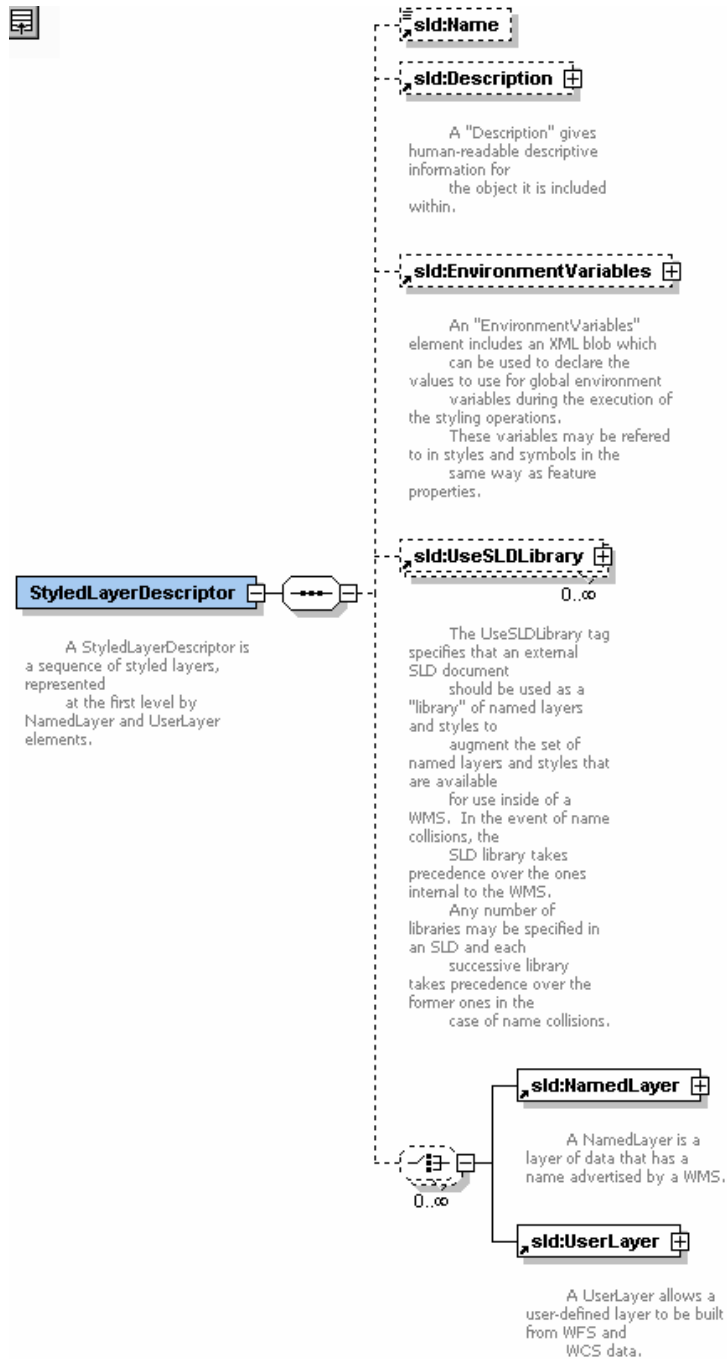
## APPENDIX -1

### Web Service Description File (WSServices.wsdl)

```
<?xml version="1.0" encoding="UTF-8" ?>
- <wsdl:definitions targetNamespace="http://deegreewmstest" xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:apache="http://xml.apache.org/xml-
  soap" xmlns:impl="http://deegreewmstest-impl" xmlns:intf="http://deegreewmstest" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
- <wsdl:message name="getCapabilityResponse">
  <wsdl:part name="getCapabilityReturn" type="xsd:string" />
</wsdl:message>
- <wsdl:message name="getMapResponse">
  <wsdl:part name="getMapReturn" type="xsd:anyType" />
</wsdl:message>
<wsdl:message name="getCapabilityRequest" />
<wsdl:message name="getMapRequest" />
- <wsdl:portType name="WSServices">
  - <wsdl:operation name="getMap">
    <wsdl:input message="intf:getMapRequest" name="getMapRequest" />
    <wsdl:output message="intf:getMapResponse" name="getMapResponse" />
  </wsdl:operation>
  - <wsdl:operation name="getCapability">
    <wsdl:input message="intf:getCapabilityRequest" name="getCapabilityRequest" />
    <wsdl:output message="intf:getCapabilityResponse" name="getCapabilityResponse" />
  </wsdl:operation>
</wsdl:portType>
- <wsdl:binding name="WSServicesSoapBinding" type="intf:WSServices">
  <wsdlsoap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http" />
- <wsdl:operation name="getMap">
  <wsdlsoap:operation soapAction="" />
  - <wsdl:input name="getMapRequest">
    <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://deegreewmstest" use="encoded" />
  </wsdl:input>
  - <wsdl:output name="getMapResponse">
    <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://deegreewmstest" use="encoded" />
  </wsdl:output>
</wsdl:operation>
- <wsdl:operation name="getCapability">
  <wsdlsoap:operation soapAction="" />
  - <wsdl:input name="getCapabilityRequest">
    <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://deegreewmstest" use="encoded" />
  </wsdl:input>
  - <wsdl:output name="getCapabilityResponse">
    <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://deegreewmstest" use="encoded" />
  </wsdl:output>
</wsdl:operation>
</wsdl:binding>
- <wsdl:service name="WSServicesService">
  - <wsdl:port binding="intf:WSServicesSoapBinding" name="WSServices">
    <wsdlsoap:address location="http://localhost:8086/deegreewmstest/services/WSServices" />
  </wsdl:port>
</wsdl:service>
</wsdl:definitions>
```

## APPENDIX-2

### Sld:StyledLayerDescriptor Element used in getMap request.



## APPENDIX-3

### Sample WMS requests created according to valid schema files

Sample GetCapabilities Request  
( sampleGetCapabilities.xml ) :

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Sample XML file generated by XMLSPY v2004 rel. 4 U (http://www.xmlspy.com)-->
<GetCapabilities xmlns="http://www.opengis.net/ows" xmlns:gml="http://www.opengis.net/gml"
xmlns:ogc="http://www.opengis.net/ogc" xmlns:sim="http://www.opengis.net/sim"
xmlns:sld="http://www.opengis.net/sld" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengis.net/ows
C:\w_m_s\wms_schema.xsd">
  <version>1.1.1</version>
  <service>wms</service>
  <exceptions>application_vnd_ogc_se_xml</exceptions>
  <section name="/">
    <RegEntry regEntryId="String" updateSequence="0"/>
    <RegEntry regEntryId="String" updateSequence="0"/>
  </section>
  <style>full</style>
</GetCapabilities>
```

## Sample GetMap Request (GetMap.xml) :

```

<?xml version="1.0" encoding="UTF-8"?>
<!--Sample XML file generated by XMLSPY v2004 rel. 4 U (http://www.xmlspy.com)-->
<GetMap xmlns="http://www.opengis.net/ows"
  xmlns:gml="http://www.opengis.net/gml" xmlns:ogc="http://www.opengis.net/ogc"
  xmlns:sim="http://www.opengis.net/sim" xmlns:sld="http://www.opengis.net/sld"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.opengis.net/ows
  C:\mywms\wms_schema.xsd">
  <version>1.1.1</version>
  <service>wms</service>
  <exceptions>application/vnd.ogc.se_xml</exceptions>
  <Map>
    <BoundingBox decimal="." cs="," ts=" ">-120.001106,34.6297150000000004 -122.803596,38.088043</BoundingBox>
    <Elevation>0</Elevation>
    <Time>String</Time>
  </Map>
  <Image>
    <Height>0</Height>
    <Width>0</Width>
    <Format>String</Format>
    <Transparent>>false</Transparent>
    <BGColor>0xFFFFFF</BGColor>
  </Image>
  <sld:StyledLayerDescriptor version="1.0.20">
    <sld:NamedLayer>
      <sld:Name>California:Rivers_3</sld:Name>
      <sld:Description>
        <sld:Title>California:Rivers_3</sld:Title>
        <sld:Abstract>California:Rivers_3</sld:Abstract>
      </sld:Description>
    </sld:NamedLayer>
    <sld:NamedLayer>
      <sld:Name>California:Country_Boundaries</sld:Name>
      <sld:Description>
        <sld:Title>California:Country_Boundaries</sld:Title>
        <sld:Abstract>California:Country_Boundaries</sld:Abstract>
      </sld:Description>
    </sld:NamedLayer>
  </sld:StyledLayerDescriptor>
</GetMap>

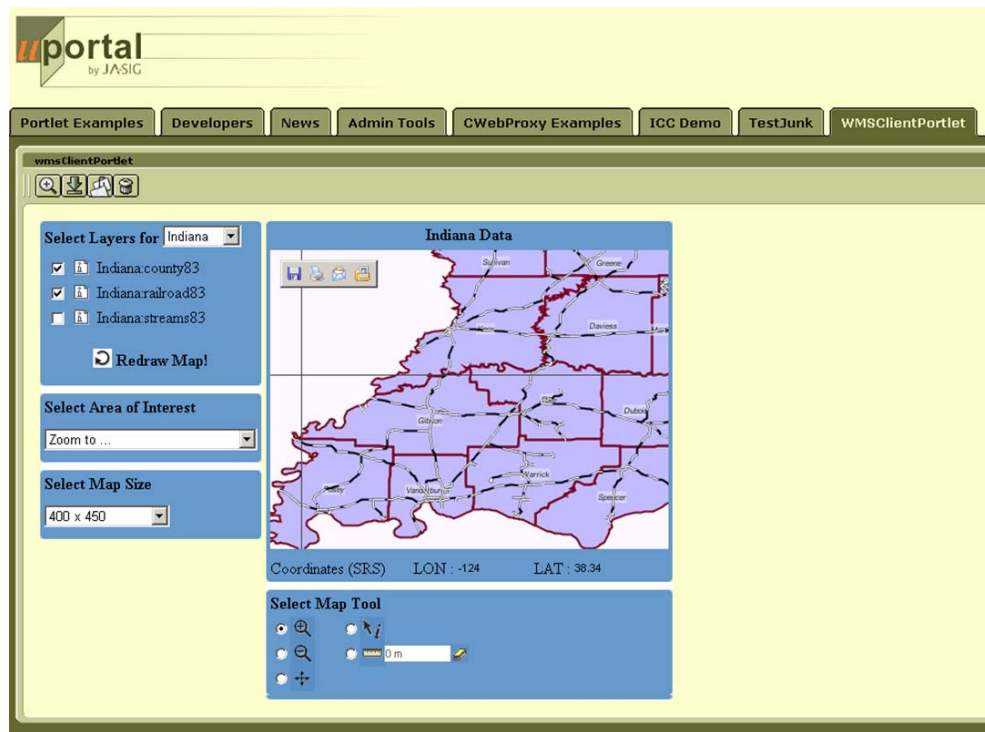
```




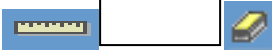


## Sample GetFeatureInfo Request (GetFeatureInfo.xml) :

```
<?xml version="1.0" encoding="UTF-8"?>
<!--Sample XML file generated by XMLSpy v2005 sp1 U (http://www.xmlspy.com)-->
<GetFeatureInfo xmlns="http://www.opengis.net/ows" xmlns:gml="http://www.opengis.net/gml" xmlns:ogc="http://www.opengis.net/ogc" xmlns:sim="http://www.opengis.net/sim" xmlns:sld="http://www.opengis.net/sld" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.opengis.net/ows C:\w_m_slwms_schema.xsd">
  <version>1.1.1</version>
  <service>wms</service>
  <exceptions>application_vnd_ogc_se_inimage</exceptions>
  <Map>
    <BoundingBox ts=" " cs=" " decimal=".">String</BoundingBox>
    <Elevation>0</Elevation>
    <Time>String</Time>
  </Map>
  <Image>
    <Height>0</Height>
    <Width>0</Width>
    <Format>String</Format>
    <Transparent>true</Transparent>
    <BGColor>0x000000</BGColor>
  </Image>
  <sld:StyledLayerDescriptor version="1.0.20">
    <sld:Name>String</sld:Name>
    <sld:Description>
      <sld:Title>String</sld:Title>
      <sld:Abstract>String</sld:Abstract>
    </sld:Description>
    <sld:EnvironmentVariables/>
    <sld:UseSLDLibrary>
      <sld:OnlineResource arcrole="http://www.altova.com" actuate="onLoad" title="String" type="simple" href="http://www.altova.com" show="new" role="http://www.altova.com"/>
    </sld:UseSLDLibrary>
    <sld:NamedLayer>
      <sld:Name>String</sld:Name>
      <sld:Description>
        <sld:Title>String</sld:Title>
        <sld:Abstract>String</sld:Abstract>
      </sld:Description>
      <sld:LayerFeatureConstraints>
        <sld:FeatureTypeConstraint>
          <sld:FeatureTypeName>String</sld:FeatureTypeName>
          <ogc:Filter>
            <ogc:Overlaps>
              <ogc:PropertyName/>
            </ogc:Overlaps>
          </ogc:Filter>
          <sld:Extent>
            <sld:Name>String</sld:Name>
            <sld:Value>String</sld:Value>
          </sld:Extent>
        </sld:FeatureTypeConstraint>
      </sld:LayerFeatureConstraints>
      <sld:NamedStyle>
        <sld:Name>String</sld:Name>
        <sld:Description>
          <sld:Title>String</sld:Title>
          <sld:Abstract>String</sld:Abstract>
        </sld:Description>
      </sld:NamedStyle>
    </sld:NamedLayer>
  </sld:StyledLayerDescriptor>
  <QueryLayer>String</QueryLayer>
  <InfoFormat>String</InfoFormat>
  <FeatureCount>0</FeatureCount>
  <x>0</x>
  <y>0</y>
  <Vendor>
    <test>String</test>
  </Vendor>
</GetFeatureInfo>
```

## APPENDIX-4

### Project page deployed on Portal as a portlet



SELECTED MAP TOOLS			
Tool	Explanation	Tool	Explanation
	Zoom In -Draw a rectangle or click on		GetFeatureInfo Request
	Zoom out -Click on a point		Calculates Differences in km, if you select two different points on the scrn.
	Panning -Drag and drop the mouse on the scrn.	 Redraw Map!	Refresh the layers displayed on the scrn. If you want to change the layer list displayed click on

## APPENDIX-5

### Sample WMS Capabilities file used in project

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Sample XML file generated by XMLSPY v2004 rel. 4 U (http://www.xmlspy.com)-->
<WMS_Capabilities xmlns="http://www.opengis.net/wms" xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.opengis.net/wms
C:\capabilities_1_3_0.xsd" version="1.3.0" updateSequence="String">
  <Service>
    <Name>WMS</Name>
    <Title>Pervasive WMS</Title>
    <Abstract>wms reference implementation</Abstract>
    <KeywordList>
      <Keyword>pervasive</Keyword>
      <Keyword>wms</Keyword>
    </KeywordList>
    <OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink" xlinktype="simple" xlink:href="http://toro.ucs.indiana.edu:8086/WMSservices.wsdll"/>
    <!-- the following service information is optional -->
    <ContactInformation>
      <ContactPersonPrimary>
        <ContactPerson>Ahmet Sayar</ContactPerson>
        <ContactOrganization>Pervasive Tech Lab</ContactOrganization>
      </ContactPersonPrimary>
      <ContactPosition>Research Assistant</ContactPosition>
      <ContactAddress>
        <AddressType>XXXX</AddressType>
        <Address>501 N. Morton St. Rm 222</Address>
        <City>Bloomington</City>
        <StateOrProvince>IN</StateOrProvince>
        <PostCode>47404</PostCode>
        <Country>USA</Country>
      </ContactAddress>
      <ContactVoiceTelephone>1(812)8560752</ContactVoiceTelephone>
      <ContactFacsimileTelephone>1(812)8567972</ContactFacsimileTelephone>
      <ContactElectronicMailAddress>asayar@cs.indiana.edu</ContactElectronicMailAddress>
    </ContactInformation>
  </Service>
  <Capability>
    <Request>
      <GetCapabilities>
        <Format>application/vnd.ogc.wms_xml</Format>
        <DCPType>
          <!-- Currently there is just one DCPT environment supported HTTP.
               In the near future there will be web services support by the Open-GIS.
               Whenever they update their standard schemas, I will update my capabilities document.
               Since I am going to use web map services I do not need these informations -->
          <HTTP><Get><OnlineResource /></Get> <Post><OnlineResource /></Post></HTTP>
        </DCPType>
      </GetCapabilities>
      <GetMap>
        <Format>image/gif</Format>
        <Format>image/png</Format>
        <Format>image/jpg</Format>
        <Format>image/tif</Format>
        <Format>image/bmp</Format>
        <Format>image/svg+xml</Format>
        <DCPType>
```



```

</GetMap>
</Request>
<Exception>
  <Format>application/vnd.ogc.se_xml</Format>
  <Format>application/vnd.ogc.se_inimage</Format>
  <Format>application/vnd.ogc.se_blank</Format>
</Exception>
<Layer queryable="0" cascaded="1" opaque="0" noSubsets="0" fixedWidth="1" fixedHeight="1">
  <Name>pervasive WMS-demo Layers</Name>
  <Title>pervasive WMS-demo Layers</Title>
  <Abstract>pervasive WMS-demo Layers</Abstract>
  <KeywordList>
    <Keyword>pervasive</Keyword>
    <Keyword>WMS</Keyword>
    <Keyword>layer</Keyword>
  </KeywordList>
  <CRS>EPSG:4326</CRS>
  <EX_GeographicBoundingBox>
    <westBoundLongitude>-150</westBoundLongitude>
    <eastBoundLongitude>100</eastBoundLongitude>
    <southBoundLatitude>30</southBoundLatitude>
    <northBoundLatitude>50</northBoundLatitude>
  </EX_GeographicBoundingBox>
  <MinScaleDenominator>0</MinScaleDenominator>
  <MaxScaleDenominator>100000000</MaxScaleDenominator>

  <!-- CALIFORNIA -->
  <Layer queryable="0" cascaded="0" noSubsets="0">
    <Title>California</Title>
    <Abstract>layers for the California WMS implementation</Abstract>
    <CRS>EPSG:4326</CRS>
    <Layer queryable="1" noSubsets="0" fixedWidth="0" fixedHeight="0">
      <Name>California:Faults</Name>
      <Title>California:Faults</Title>
      <EX_GeographicBoundingBox>
        <westBoundLongitude>-150</westBoundLongitude>
        <eastBoundLongitude>-100</eastBoundLongitude>
        <southBoundLatitude>30</southBoundLatitude>
        <northBoundLatitude>50</northBoundLatitude>
      </EX_GeographicBoundingBox>
      <BoundingBox CRS="EPSG:4326" minx="-180" miny="-90" maxx="180" maxy="90" resx="1" resy="1" />
      <MinScaleDenominator>0</MinScaleDenominator>
      <MaxScaleDenominator>100000000</MaxScaleDenominator>
    </Layer>
    <Layer queryable="1" noSubsets="0" fixedWidth="0" fixedHeight="0">
      <Name>California:Rivers</Name>
      <Title>California:Rivers</Title>
      <EX_GeographicBoundingBox>
        <westBoundLongitude>-150</westBoundLongitude>
        <eastBoundLongitude>-100</eastBoundLongitude>
        <southBoundLatitude>30</southBoundLatitude>
        <northBoundLatitude>50</northBoundLatitude>
      </EX_GeographicBoundingBox>
      <BoundingBox CRS="EPSG:4326" minx="-180" miny="-90" maxx="180" maxy="90" resx="1" resy="1" />
      <MinScaleDenominator>0</MinScaleDenominator>
      <MaxScaleDenominator>100000000</MaxScaleDenominator>
    </Layer>

    <Layer queryable="1" noSubsets="0" fixedWidth="0" fixedHeight="0">
      <Name>California:States</Name>
      <Title>California:States</Title>
      <EX_GeographicBoundingBox>
        <westBoundLongitude>-150</westBoundLongitude>
        <eastBoundLongitude>-100</eastBoundLongitude>
        <southBoundLatitude>30</southBoundLatitude>
        <northBoundLatitude>50</northBoundLatitude>
      </EX_GeographicBoundingBox>
      <BoundingBox CRS="EPSG:4326" minx="-180" miny="-90" maxx="180" maxy="90" resx="1" resy="1" />
      <MinScaleDenominator>0</MinScaleDenominator>
      <MaxScaleDenominator>100000000</MaxScaleDenominator>
    </Layer>
  </Layer>

```

```

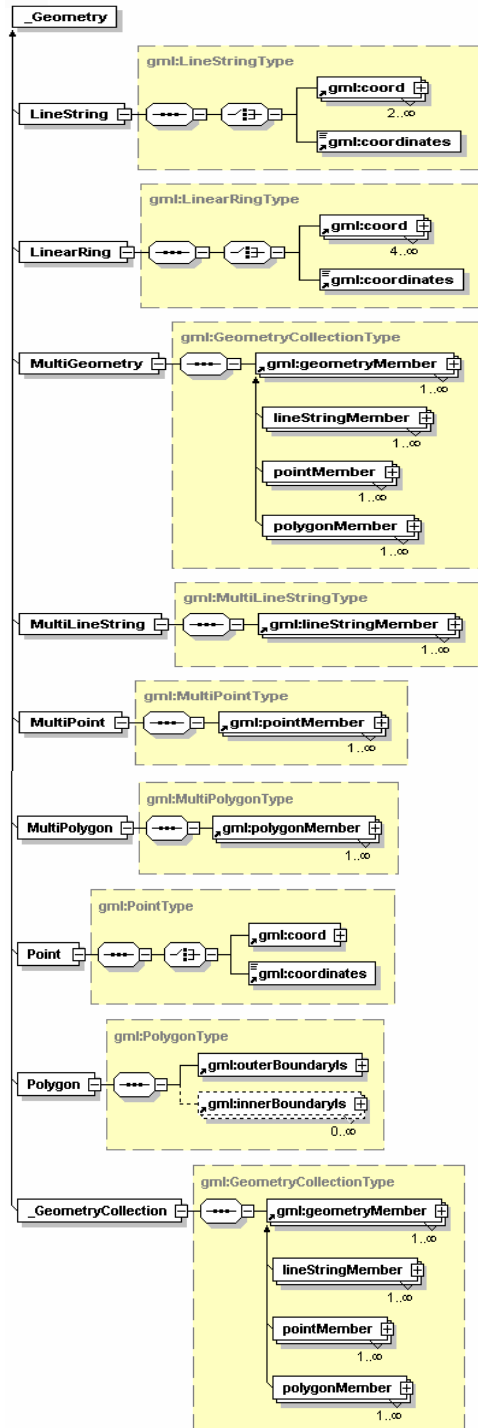
<!-- INDIANA -->

<Layer queryable="0" cascaded="0" noSubsets="0">
  <Title>Indiana</Title>
  <Abstract>layers for the California WMS implementation</Abstract>
  <CRS>EPSG:4326</CRS>
  <Layer queryable="1" noSubsets="0" fixedWidth="0" fixedHeight="0">
    <Name>Indiana:county83</Name>
    <Title>Indiana:county83</Title>
    <EX_GeographicBoundingBox>
      <westBoundLongitude>-150</westBoundLongitude>
      <eastBoundLongitude>-100</eastBoundLongitude>
      <southBoundLatitude>30</southBoundLatitude>
      <northBoundLatitude>50</northBoundLatitude>
    </EX_GeographicBoundingBox>
    <BoundingBox CRS="EPSG:4326" minx="-180" miny="-90" maxx="180" maxy="90" resx="1" resy="1" />
    <MinScaleDenominator>0</MinScaleDenominator>
    <MaxScaleDenominator>100000000</MaxScaleDenominator>
  </Layer>
  <Layer queryable="1" noSubsets="0" fixedWidth="0" fixedHeight="0">
    <Name>Indiana:railroad83</Name>
    <Title>Indiana:railroad83</Title>
    <EX_GeographicBoundingBox>
      <westBoundLongitude>-150</westBoundLongitude>
      <eastBoundLongitude>-100</eastBoundLongitude>
      <southBoundLatitude>30</southBoundLatitude>
      <northBoundLatitude>50</northBoundLatitude>
    </EX_GeographicBoundingBox>
    <BoundingBox CRS="EPSG:4326" minx="-180" miny="-90" maxx="180" maxy="90" resx="1" resy="1" />
    <MinScaleDenominator>0</MinScaleDenominator>
    <MaxScaleDenominator>100000000</MaxScaleDenominator>
  </Layer>
  <Layer queryable="1" noSubsets="0" fixedWidth="0" fixedHeight="0">
    <Name>Indiana:streams83</Name>
    <Title>Indiana:streams83</Title>
    <EX_GeographicBoundingBox>
      <westBoundLongitude>-150</westBoundLongitude>
      <eastBoundLongitude>-100</eastBoundLongitude>
      <southBoundLatitude>30</southBoundLatitude>
      <northBoundLatitude>50</northBoundLatitude>
    </EX_GeographicBoundingBox>
    <BoundingBox CRS="EPSG:4326" minx="-180" miny="-90" maxx="180" maxy="90" resx="1" resy="1" />
    <MinScaleDenominator>0</MinScaleDenominator>
    <MaxScaleDenominator>100000000</MaxScaleDenominator>
  </Layer>
</Layer>
</Layer>
</Capability>
</WMS_Capabilities>

```

## APPENDIX-6

geometry.xsd, Geometry schema for GML encoding of feature data.



## APPENDIX-7

### Summary of the CrisisGrid related OGC Specifications

*(Assembled by Dr. Sunghoon Ko, Community Grids Lab)*

Specification Name (Abbreviation)	Type	Description	Relationship to CrisisGrid Framework	Status of Specification
Web Feature Service (WFS)	Service	The Web Feature Service supports the query and discovery of geographic features. In a typical Web-base scenario, Web Feature Service delivers GML (XML) representations of simple geospatial features in response to queries from HTTP clients. Clients (service requestors) access geographic feature data through a WFS by submitting a request for just those features that are needed for an application. The client generates a request and posts it to a WFS instance (a WFS server on the Web). The WFS instance executes the request, returning the results to the client (service requester) as GML. A GML-enabled client can manipulate or operate on the returned features.	WFS will provide the means to access geographic features including critical infrastructure features, incident locations (points, paths, or areas), and flood-related geographic features including inundation areas, watershed boundaries, demographic feature, and any other feature-based information required for flood or impact analysis and simulation.	Category: OpenGIS Implementation Specification  V1.0 : 2002-09-19 Adopted Specification HTTP post/get GML 2.0  V1.1 : will support GML 3.0
Geography Markup Language (GML)	Encoding	The Geography Markup Language (GML) is an XML encoding for the transport and storage of geographic information, including both the geometry and properties of geographic features. GML utilizes the OpenGIS® Abstract Specification geometry model which has been harmonized with the ISO geospatial geometry model. The GML Specification includes the ability to handle complex geometries and properties. GML is used to represent geographic features conforming to well-defined application schema for purposes of transport across computational interfaces (e.g., WFS)	GML will be an important element of the information modeling effort conducted for CrisisGrid. GML will be the starting place for application ontologies built under the CrisisGrid effort.	Category: OpenGIS Implementation Specification  V3.0 : 2003-01-29 Adopted Specification  V3.1 : 2004-04-19
Web Coverage Service (WCS)	Service	The Web Coverage Service supports the query and discovery of geographic information that conforms to a gridded (regularly spaced or not) data model, termed "coverages" – that is, digital geospatial information representing space-varying phenomena. In a typical Web-base scenario, Web Coverage Service delivers arrays of data representations (in existing image or gridded data formats) of subsets of geographic coverages in response to queries from HTTP clients. Clients (service requestors) access geographic coverage data through a WCS by submitting a request for a subset of a coverage needed for an application. The	WCS will provide the means to access geographic coverages including digital elevation models, imagery, orthophotography, weather coverages (such as predicted rainfall, air pressure, wind speed and direction), and any other space-varying flood-related phenomena.	Category: OpenGIS Implementation Specification  V1.0 : 2003-10-16 Adopted Specification HTTP post/get GML 3.0

		<p>client generates a request and posts it to a WCS instance (a WCS server on the Web). The WCS instance executes the request, returning the results to the client (service requester) in one of the following supported formats:</p> <p>a) GeoTIFF  <a href="http://www.remotesensing.org/geotiff/geotiff.html">&lt;http://www.remotesensing.org/geotiff/geotiff.html&gt;</a>  b) HDF-EOS  <a href="http://heineken.gsfc.nasa.gov">&lt;http://heineken.gsfc.nasa.gov/&gt;</a>  c) DTED  <a href="http://www.nima.mil/publications/specs/printed/89020A/89020A_DTED.pdf">&lt;http://www.nima.mil/publications/specs/printed/89020A/89020A_DTED.pdf&gt;</a>  d) NITF  <a href="http://www.ismc.nima.mil/ntb/baseline/1999.html">&lt;http://www.ismc.nima.mil/ntb/baseline/1999.html&gt;</a>  e) GML  <a href="http://www.opengis.org/techno/documents/02-023r4.pdf">&lt;http://www.opengis.org/techno/documents/02-023r4.pdf&gt;</a></p>		
Web Map Service (WMS)	Service	<p>A Web Map Server (WMS) generates "pictures" of georeferenced data. Independent of whether the underlying data are simple features (such as points, lines and polygons) or coverages (such as gridded fields), the WMS produces an image of the data that can be directly viewed in a graphical web browser or other picture-viewing software. An extension of the basic Web Map Server is the Styled Layer Descriptor (SLD) Web Map Server. The SLD enabled WMS inherits all of the attributes from the Web Map Server and adds support for the use of Styled Layer Descriptor documents to specify styling. Instead of generating maps of particular named layers in one or more predefined styles, an SLD Map Server extracts features from a data provider and renders them using a stylistic description encoded in XML.</p>	WMS will provide a means to portray geographic information independent of the underlying data model (feature or coverage).	<p>Category: OpenGIS Implementation Specification</p> <p>V1.1 : 2001-06-21  HTTP post/get  GML ?</p> <p>V1.2 :  Adopted  Specification</p> <p>V1.3 : 2004-01-20  HTTP post/get  GML ?</p>
Styled Layer Descriptor (SLD)	Encoding	<p>The Styled Layer Descriptor (SLD) encoding specifies the format of a map-styling language for producing georeferenced maps with user-defined styling. This language is used to create XML documents that control the visual portrayal of the data with which they work. The ability for a human or machine client to define the styling rules requires a styling language that the client and server can both understand. The SLD language can be used to portray the output of Web Map Servers, Web Feature Servers and Web Coverage Servers. The SLD is defined using XML Schema.</p>	SLD will enable different communities in the Emergency Response area to develop a set of portrayal rules that best fit their mission requirements.	<p>Category: OpenGIS Implementation Specification</p> <p>V1.0 : 2002-09-19  Adopted  Specification  HTTP post/get  GML ?</p>
Coverage Portrayal Service (CPS)	Service	<p>The Coverage Portrayal Service (CPS) defines a standard interface for producing visual pictures from coverage data. Typically coverage data are</p>	CPS will enable the CrisisGrid-related coverages to be visualized.	<p>Category: Interoperability Program Report</p> <p>V0.02 : 2002-02-28</p>

		<p>retrieved via a WCS instance. CPS extends the WMS interface and uses the Styled Layer Descriptor (SLD) language to support rendering of WCS coverages. CPS facilitates wider use of coverage data by making views of coverages visible within thin-clients (e.g., Web browsers). To a service requestor, the CPS appears as a WMS instance, but with additional parameters to control the retrieval and/or rendering of coverage data. The CPS may require the client to specify the targeted WCS.</p> <p>CPS may be used to support:</p> <ul style="list-style-type: none"> <li>assigning multi-spectral bands in an image to color channels in a picture,</li> <li>creating choropleth maps from coverage data using client-specified color-bins</li> <li>preset rendering mechanisms such as hill-shaded elevation</li> <li>combining multi-spectral pixel values according to client-specified or server-defined formulas (e.g., Normalized Difference Vegetation Index).</li> </ul>		
Web Terrain Service (WTS)	Service	<p>WTS is a companion specification to the OpenGIS Web Map Service Interface Implementation Specification. WMS specifies how individual map servers describe and provide their map content. The present Web Terrain Service specification describes a new operation, GetView, and extended Capabilities which allow a 3D terrain view image to be requested, given a map composition, a terrain model on which to drape the map, and a 3D viewpoint from which to render the terrain view. A simple attempt is also made to reconcile 2D and 3D viewpoints by allowing the requested 3D area of view to be approximated with a WMS bounding box</p>	WTS will provide a means to visualize information in a 3D view to support the needs of Emergency Response professionals.	<p>Category: Request for Comment</p> <p>V0.5 : 2003-11-07 HTTP post/get GML ?</p>
Catalog Service – Web Profile (CS-W)	Service	<p>Catalog-Registry Services provide a common mechanism to classify, register, describe, search, maintain and access information about network resources. Resources are network addressable instances of typed data or services. Registries may be differentiated by their role such as registries for cataloging data types (e.g., types of geographic features, coverages, sensors, symbols), online data instances (e.g., datasets, repositories, symbol libraries), service types (e.g., portrayal, processing, data services) and online service instances.</p> <p>The metadata content published to the registry, while conforming to the same Registry Information Model (RIM), describes different kinds of resources using metadata that may be structurally and semantically different than metadata for resources of other types or for other purposes or organizations. The Web</p>	CS-W will provide a means for cataloging the computer-based resources that are available within the CrisisGrid Framework	Document N/A : OpenGIS Membership required

		<p>Registry Service (soon to become the Web Profile of the OpenGIS Catalog Service) defines a common information model and the service interfaces to access resource offers, regardless of the type of resource and the content of the metadata.</p> <p>Type Registries contain metadata about resource (data and service) types (e.g., types of images, features, feature collections, styles, symbols, and services) as taxonomies that are shared and used within information communities. The ability to publish and share this information is an essential requirement for distributed applications to be able to share and exploit, with a common semantic, these resources. Type Registry Services provide access to these metadata and taxonomies of types. Support for publishing and referencing taxonomies is explicitly supported in the Web Registry Service (WRS) Interface and Registry Information Model (RIM) specification.</p>		
Sensor Collection Service (SCS)	Service	Service to fetch observations from a sensor or group of sensors.	SCS will provide an interface that can be used to gather readings from all kinds of sensors. In Flood CrisisGrid, SCS can be used to access sensor relevant to the flood case (such as, stream gauges and weather related sensors)	<p>Category: Interoperability Program Report</p> <p>V0.5.1 : 2002-04-19 HTTP post/get GML ?</p>
Sensor Planning Service (SPS)	Service	Service to assist in 'collection feasibility plans' and to process collection requests for a sensor or group of sensors.	SPS may provide a means to invoke pre-determined collection plans that have been developed to be executed in the case of an emergency.	Document N/A : OpenGIS Membership required
Web Notification Service (WNS)	Service	Service to manage dialogue between a client and Web Service(s) for long duration asynchronous processes.	WNS will provide a means to notify interested parties that a particular sensor collection has been completed.	<p>Category: Discussion Paper</p> <p>V0.1.0 : 2003-04-21 HTTP post, email, SMS, Instant Message, Fax, Phone</p>
Sensor Markup Language (SensorML)	Encoding	Information model and XML encodings for discovering, querying and controlling Web-resident sensors.	SensorML will provide the means of describing the various sensors that are required for CrisisGrid.	V0.7 : 2002-12-20
Observations & Measurements (O&M)	Encoding	Information model and encodings for observations and measurements.	O&M will be used to defined the observations that are returned from various sensors that are required for CrisisGrid.	V0.86 : 2002-05-31 GML 3

Note: - Much of the current work in OGC involves geoprocessing via the IT industry's Web Services standards framework. The OpenGIS Specifications that make this possible are referred to as "OGC Web Services."

## APPENDIX-8

### Open Source / Free GIS related software projects

(Assembled by Dr. Sunghoon Ko, Community Grids Lab)

Software	Description	Service	OGC comparability	Version/ PL/OS	License	System (Monolithic/ One/ Few services?)
<a href="#">Chameleon</a>	Chameleon is a distributed, highly configurable, environment for developing Web Mapping applications. It is built on OGC standards for Web Mapping Services (WMS) and WMT Viewer Contexts.	Web Mapping	WMS, WMT	V1.0.5 PHP Linux, Unix <b>No-windows</b>	X11-style	
<a href="#">Deegree</a>  University of Bonn	Deegree is a <b>Java</b> framework for geospatially-enabled solutions. It is based on common GI standards and allows building applications with spatially referenced content. Deegree components can be used to either develop a standalone desktop mapping solution to be locally installed on a user's machine, or to set up a highly distributed and service-based infrastructure. As the whole architecture of deegree is <b>based on OGC</b> specifications and concepts, there are no problems to integrate standardized products of other vendors (e.g. ArcIMS by ESRI(c)).	Base GIS, Visualization, Interactive Viewing, Web Mapping	GML2.1.1, Web Map Service (WMS)1.1.1, Web Feature Service (WFS)1.0.0, Web Coverage Service (WCS)1.0.0, Web Catalog Service (WCAS) based on OGC Web Services Stateless Catalog Profile, Web Gazetteer Service (WFS-G), Web Terrain Service (WTS), Web Coordinate Transformation Service (WCTS).	V0.7.7/ Java/ Windows, Linux, Unix	<a href="#">GNU Lesser General Public License (LGPL)</a>	Non-monolithic: Deegree components can be set up as distributed and service-based infrastructure.
<a href="#">Demeter Terrain Engine</a>	Demeter is a cross-platform C++ library that renders 3D terrains using OpenGL. Demeter is designed for fast performance and good visual quality and makes use of advanced techniques	3D Map Rendering		V3.14 C++ Windows, Linux, Unix, MacOS X	<a href="#">GNU Lesser General Public License (LGPL)</a>	One service



	such as dynamic tessellation (adaptive mesh) to render vast landscapes in real-time, without the need for high-end hardware. It is written as a stand-alone component that can be easily integrated into any kind of application.					
<a href="#">DEMViewer</a>	DEMViewer is a <b>Java</b> digital elevation model viewer for ArcGrid ASCII export files. With DEMViewer you can visualize digital elevation models generated by ArcInfo and combine it with data (in the same ArcGrid ASCII export format and/or Jpeg/Gif images).	Visualization		Java Windows, Linux, Unix, MacOS X	<a href="#">GNU General Public License (GPL)</a>	One service
<a href="#">FMaps</a>	FMaps is a Geographic Information System and Remote Sensing application which stores its data in a PostgreSQL database. It uses a special GTK+ widget which was originally called GtkFMaps. However current version only supports C and Linux platforms.	Base GIS, Interactive Viewing	Not yet	V0.0.2 C Linux, Unix <b>No-windows</b>	<a href="#">GNU General Public License (GPL)</a>	A few services
<a href="#">GDAL</a>	GDAL is a translator library for raster geospatial data formats that is released under an Open Source license. As a library, it presents a single abstract data model to the calling application for all supported formats.	File-format conversion, Projection conversion		V1.2.0/ C/C++/ Windows, Linux, Unix, MacOS X	<a href="#">GNU General Public License (GPL)</a> , MIT	One service: C/C++ library
<a href="#">GeoServer</a>	The GeoServer project is a <b>Java</b> implementation of the <b>OpenGIS Consortium's Web Feature Server</b> specification. It is <b>free software</b> .	Web Mapping	GML2.1.2, Web Map Service (WMS)1.1.1, Web Feature Service (WFS)1.0.0	V1.1.0 Java/C++ Windows, Linux, Unix, MacOS X	<a href="#">GNU General Public License (GPL)</a>	A few services
<a href="#">GeoTools</a>	"Geo Tools is a <b>free Java</b> based mapping toolkit that allows	Interactive Viewing, Web Mapping	GML 2.*, <a href="#">Grid Coverage</a> 1.0, <a href="#">Coordinate</a>	V0.8.0 Java Windows,	<a href="#">GNU General Public</a>	A few services

	maps to be viewed interactively on web browsers without the need for dedicated server side support.		<a href="#">Transformation Services 1.0</a> , <a href="#">Styled Layer Descriptor specification 1.0</a> , <a href="#">Filter Encoding specification 1.0</a>	Linux, Unix, MacOS X	<a href="#">License (GPL)</a>	
<a href="#">GeoVRML</a>	GeoVRML is an official Working Group of the Web3D Consortium. It was formed on 27 Feb 1998 with the goal of developing tools and recommended practice for the representation of geographical data using the Virtual Reality Modeling Language (VRML). The desire is to enable geo-referenced data, such as maps and 3-D terrain models, to be viewed over the web by a user with a standard VRML plugin for their web browser.	File-format conversion		V1.1 Java Windows, Linux, Unix, MacOS X	<a href="#">Apache license</a>	One service
<a href="#">GIServer</a>	The GIServer is an initiative from the inovaGIS project that gives free access to GIS functions through the Internet. <a href="#">The SOAP GIServer Web Services available for testing.</a>	Interactive Viewing, Web Mapping	Web Map Service (WMS)1.1.0	V0.9 Windows	?	Non-monolithic: GIServer is capable of registering new spatial data process algorithms and defining them as new services. These services can be locally on the server or in other remote Internet Server.
<a href="#">GISToolKit</a>	The GISToolkit software is a <b>java</b> toolkit for building spatially enabled applications. It has some ability to read data from a variety of data sources, and to display that data. It can directly edit geographic features stored in databases to which it has access.	Visualization, Interactive Viewing	Not compliant but very close to WMS	V2.0(Server) , 2.8.1(Editor) Java Windows, Linux, Unix, MacOS X	<a href="#">GNU Lesser General Public License (LGPL)</a>	A few services
<a href="#">GIS Viewer</a>	GIS Viewer is a web-based <b>Java</b> tool for displaying and manipulating layers of geographical points and vectors, and raster data such	Interactive Viewing		V4.0 Java Windows, Linux, Unix, MacOS X	GIS Viewer license	One service

	as maps and images.					
<a href="#">GML4J</a>	GML4J is a <b>Java API</b> for facilitating work with the Geography Markup Language ( <a href="http://www.gmlcentral.com">http://www.gmlcentral.com</a> ). GML is an XML-based framework for encoding geography information adopted as a recommendation paper by OGC ( <a href="http://www.opengis.org">http://www.opengis.org</a> ). Currently only support read access.	File-format conversion	GML 2.0	V1.0.2beta Java Windows, Linux, Unix, MacOS X	Apache Software License	One service
<a href="#">GRASS</a>	GRASS GIS (Geographic Resources Analysis Support System) is an Open Source Geographical Information System (GIS) with raster, topological vector, image processing, and graphics production functionality that operates on various platforms through a graphical user interface and shell in X-Windows.	Base GIS Visualization, Remote Sensing, Flights, File-format conversion, Projection conversion, Customizable with Add-ons		V5.0.3 C Linux, Unix, MacOS X <b>No-windows(only Win/Cygnus)</b>	<a href="#">GNU General Public License (GPL)</a>	Non-monolithic: GRASS might be run in clustered environments.
<a href="#">JUMP</a>	The <b>Java</b> Unified Mapping Platform (JUMP) is a GUI-based application for viewing and processing spatial data. It includes many common spatial and GIS functions. It is also designed to be a highly extensible framework for developing and running custom spatial data processing applications.	Visualization, Interactive Viewing, Customizable with Add-ons	GML, WMS	V1.1.1/ Java Windows, Linux, Unix	GNU GPL	Non-monolithic: Workbench GUI components can be used independently
<a href="#">kdem</a>	kdem is a program for displaying United States Geological Survey (USGS) Digital Elevation Models (DEMs).	Visualization, Interactive Viewing		v1.1.1 C++ Linux, Unix <b>No-windows</b>	kdem	A few services
<a href="#">MapServer</a>	MapServer is a CGI-based application for delivering dynamic GIS and image processing content via the World-Wide	Web Mapping	GML2.*, Web Map Context Specification v1.0.0, Web Map Service1.1.1, Web Feature Service	V4.0 C Windows, Linux, Unix, MacOS X	<a href="#">Map Server License</a>	A few services

	Web (WWW). The package also contains a number of stand alone applications for building maps, scale bars and legends offline. Access to the development environment of MapServer is possible with a number of different programming languages.					
<a href="#">Mapyrus</a>	Mapyrus is software for creating plots of points, lines, polygons and labels to PostScript, PDF and web image output formats. The software combines the following three components: A Logo or turtle graphics language, reading of GIS datasets and RDBMS tables, running as a stand-alone program or as a web-server.	Visualization		V0.41 Java Windows, Linux, Unix	<a href="#">GNU General Public License (GPL)</a>	One service
<a href="#">MIT OrthoServer</a>	The MIT OrthoServer is a set of components that serve large collections of geo-image libraries online, in a seamless, multi-resolution fashion.	Web Mapping	Web Mapping Testbeds (WMT)1.0	Perl Windows, Linux, Unix	<a href="#">GNU General Public License (GPL)</a>	A few services
<a href="#">MySQL Spatial</a>	MySQL implements spatial extensions following the specification of the <b>Open GIS Consortium (OGC)</b> .	Database(SQL RDBMS to support spatial data)	<a href="#">Simple Features Specifications For SQL</a> 1.1	V4.1 C Windows, Linux, Unix, MacOS X	<a href="#">GNU General Public License (GPL)</a>	One service
<a href="#">NetMaps</a>	NetMaps is a <b>Java applet</b> that allows one to view vectorial maps in any Java enabled browser. NetMaps can load and display ArcInfo shapefiles (SHP/DBF) and MapInfo MIF/MID files.	Web Mapping		V2 Java	?	One service
<a href="#">OpenMap</a>	BBN Technologies' OpenMap package is a <b>JavaBeans</b> based programmer's toolkit. Using OpenMap, you can quickly build applications and	Interactive Viewing	Web Map Service (WMS)	V4.6 Java Windows, Linux, Unix	<a href="#">Open Map License</a>	Non-monolithic: Using OpenMap components, you can access data from

	applets that access data from legacy databases and applications. OpenMap provides the means to allow users to see and manipulate geospatial information.					legacy applications, in-place, in a distributed setting.
<a href="#">OpenSVG Mapserver</a>	An open source solution for publishing arcview shapefiles with attributes to the web based on html, SVG, javascript, php and mysql database. It supports interactivity and filtering.	Web Mapping		V1.01 Windows, Linux, Unix	<a href="#">GNU General Public License (GPL)</a>	A few services
<a href="#">OSSIM</a>	OSRS (Open Source Remote Sensing)'s OSSIM (Open Source Software Image Map) project. The OSSIM project leverages existing open source algorithms, tools, and packages to construct an integrated library for remote sensing, image-processing, and Geographical Information Sciences (GIS) analysis.	Interactive Viewing, Remote Sensing		V1.4.0 C++ Linux, Unix <b>No-windows</b>	<a href="#">GNU General Public License (GPL)</a>	A few services
<a href="#">PostGIS</a>	PostGIS adds support for geographic objects to the PostgreSQL object-relational database. In effect, PostGIS "spatially enables" the PostgreSQL server, allowing it to be used as a backend spatial database for geographic information systems (GIS), much like ESRI's SDE or Oracle's Spatial extension. <a href="#">PostGIS Installer for Windows</a>	Database	<a href="#">Simple Features Specifications For SQL 1.1</a>	V0.8.0 Java, C++ Linux, Unix <b>No-windows</b>	<a href="#">GNU General Public License (GPL)</a>	One service
<a href="#">PyOGCLib</a>	PyOGCLib aims to develop and distribute a <b>Python</b> based library for the implementation of the <b>OpenGIS</b> specifications, notably Web Map Server (WMS) and	Web Mapping	Web Map Service (WMS), Web Feature Service (WFS)	V0.1 Python Linux, Unix <b>No-windows</b>	MIT	One service

	Web Feature Server (WFS).					
<a href="#">QuickWMS</a>	JavaScript classes for creating Web Map clients and interfacing WMS servers according to OpenGIS Web Mapping Specification (versions 0.7 to 1.1). The goal of this project is to enable the fast creation of web mapping clients using javascript. The target browsers are Internet Explorer (version 5.5 and up) and Netscape (7.00 and up) both for Windows, Mac and Linux.	Web Mapping	Web Map Service (WMS) 1.1	V0.2 Windows, Linux, Unix, MacOS X	<a href="#">GNU Lesser General Public License (LGPL)</a>	One service
<a href="#">Rez</a>	Open Source framework and tools for translating terrain data and images to different formats and generating multiresolution versions optimised for online broadband delivery.	Visualization, File-format conversion		Java Windows,	<a href="#">GNU Lesser General Public License (LGPL)</a>	A few services
<a href="#">Space Time Toolkit</a>	Space Time Toolkit (STT) is a Java-based toolkit that provides advanced capabilities for integrating spatially and temporally-disparate data within a highly interactive 3D display environment.	Web Mapping	WCS, WMS, WMT	V1.2 Java Windows, Linux, Unix, MacOS X	U. of Alabama in Huntsville	One service
<a href="#">TerraLib</a>	TerraLib is a GIS classes and functions library, allowing a collaborative environment and its use for the development of multiple GIS tools. TerraLib aims to provide a large set of data structures and algorithms for GIS developers.	Database		V2.0 C++ Windows, Linux, Unix	GNU LGPL	A few services
<a href="#">Thuban</a>	Thuban is an Interactive Geographic Data Viewer with the following features: 1) Navigation Zoom In/Out, Pan 2) Identify Attributes by	Interactive Viewing, Customizable with Add-ons		V1.0.0 Python Windows, Linux, Unix	<a href="#">GNU General Public License (GPL)</a>	Monolithic

	object selection, objects by record selection. 3) Layer Management Layer types: Line, Polygon, Point, Georeferenced Image 4) Legend Editor Visual appearance of objects can be controlled. 5) Table Management Query and join tables. 6) Printing Print and export maps for further processing.					
<a href="#">TOPAZ</a>	TOPAZ (Topographic Parameterization) is an automated digital landscape analysis tool for topographic evaluation, drainage identification, watershed segmentation and subcatchment parameterization. While TOPAZ is designed primarily to assist with topographic evaluation and watershed parameterization in support of hydrologic modeling and analysis, it also has application to a variety of geomorphological, environmental and remote sensing applications.	Visualization, Remote Sensing, Landscape Analysis		V3.0 Fortran77/90 Windows, Linux, Unix	?	One service
<a href="#">WinDisp</a>	Windisp is software package for the display and analysis of satellite images, maps and associated databases, with an emphasis on early warning for food security. WinDisp was originally developed for the FAO Global Information and Early Warning System.	Visualization, Remote Sensing		V5.1 Windows	?	One service
<a href="#">ZMapServer</a>	ZMapServer is a plug-in for Zope that provides a framework for web mapping applications using	Web Mapping	Web Map Service (WMS) 1.1	V0.1 Python Linux, Unix No-windows	MIT	One service

	<p>MapServer. The essential web mapping components are included: maps, legends, scale bars, labels, data layers and layer styles. These components are managed through Zope, helping to maintain separation between GIS content and its presentation.</p>					
--	---	--	--	--	--	--



## APPENDIX-9

### Workload So far

#### GIS Web Map Server

WFS Client stubs generated by Apache-axis	700
IS-Discovery Services Client stubs generated by axis	2,200
Gml handler classes – converts any gml to svg. (Generic coding)	
Interface classes	900
Implementation classes	4,900
Utility classes for xml documents	1,600
	7,400
Image rendering classes . . . . .	3,600
(To provide OGC compatibility	
Configuration and Capability classes) . . . . .	800
Total: 11,800 + 2,900 =	14,700

#### GIS Web Map Client

Castor generated classes	
For capabilities and exceptions schema	
(These OGC compatible schema files are updated	
to be able to create source files by using castor). . .	29,400
WMS Client stubs generated by Apache-axis . . . . .	500
Jsp . . . . .	900
Javascript . . . . .	1,100
Stylesheets . . . . .	250
JavaBeans and remote job invoking classes . . . . .	900
Total: 3,150 + 29,900 =	33,050

**Total: 14,950 + 32,800 = 47,750**

I also have been working on deploying application into Portal as a portlet. This work is not included in the report.

- ◆ Castor and axis generated classes.
- Implemented classes.

## REFERENCES

- [1] OGC (Open Geospatial Consortium) official web site <http://www.opengeospatial.org/>
- [2] Project web site <http://toro.ucs.indiana.edu:8089/newmap.jsp>
- [3] Webservices Technologies <http://www.w3.org/2002/ws/>
- [4] Jeff De La Beaujardiere, OpenGIS Consortium Web Mapping Server Implementation Specification 1.3, OGC Document #04-024, August 2002. Available at <http://www.opengis.org/techno/specs/>
- [5] Kris Kolodziej, OGC OpenGIS consortium, OpenGIS Web Map Server Cookbook 1.0.1, OGC Document #03-050r1, August 2003. Available at <http://www.ogcnetwork.org/docs/03-050r1.pdf>
- [6] Lalonde, W. (ed.), Styled Layer Descriptor (SLD) Implementation Specification 1.0.0, OGC Document #02-070, August 2002. Available at <http://www.opengis.org/techno/specs/>
- [7] Vretanos, P. (ed.), Web Feature Service Implementation Specification (WFS) 1.0.0, OGC Document #02-058, September 2003. Available at <http://www.opengis.org/techno/specs/>
- [8] GIS project - Crisisgrid <http://www.crisisgrid.org>
- [9] David Cruikshank, John Gebhardt, Lofton Henderson, Roy Platon, Dieter Weidenbrueck, WebCGM 1.0, December 2001. Available at <http://www.w3.org/TR/REC-WebCGM/>
- [10] Simon Cox, Paul Daisey, Ron Lake, Clemens Portele, Arliss Whiteside, Geography Markup Language (GML) specification 3.0, Document #02-023r4., January 2003. Available at <http://www.opengeospatial.org/docs/02-023r4.pdf>
- [11] Galip Aydin, Marlon Pierce, Geoffrey Fox, Mehmet Aktas and Ahmet Sayar "Implementing GIS Grid Services for the International Solid Earth Research Virtual Observatory". To appear in a special issue of Pure and Applied Geophysics.
- [12] 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.
- [13] John D. Evans, OGC Web Coverage Service (WCS) Specifications 1.0.0, Document #03-065r6 August 2003. Available at <http://www.opengis.org/techno/specs/>
- [14] WMS application deployed on Portal <http://toro.ucs.indiana.edu:8092/uPortal>

- [15] Jérôme Sonnet, Charles Savage. OGC Web Service Soap Experiment Report 0.8 Document#03-014, Jan 2003. Available at <http://www.opengeospatial.org/specs/?page=baseline>
- [16] Douglas Nebert, Arliss Whiteside, OpenGIS Consortium Catalogue Services Specifications 2.0. OGC Document# 04-021r2, May 2004. Available at <http://www.opengis.org/techno/specs/>
- [17] Fran Berman, Geoffrey C. Fox, Anthony J. G. Hey., Grid Computing: Making the Global Infrastructure a Reality. John Wiley, 2003.
- [18] James Clark, XSL Transformation (XSLT) Version 1.0., W3C Recommendation, November 1999. Available at <http://www.w3.org/TR/xslt>
- [19] Sharon Adler, Anders Berglund, Jeff Caruso, Stephen Deach, Tony Graham. Extensible Stylesheet Language (XSL) Version 1.0., W3C Recommendation, October 2001. Available at <http://www.w3.org/TR/xsl>
- [20] Castor <http://castor.exolab.org>
- [21] *XMLBeans* (<http://xml.apache.org/xmlbeans> )
- [22] Don Box, David Ehnebuske, Gobal Kakivaya, Andrew Layman, Dave Winer., Simple Object Access Protocol (SOAP) Version 1.1, May 2000,. Available at <http://www.w3.org/TR/2000/NOTE-SOAP-20000508/>
- [23] Erik Christensen, Francisco Curbera, Greg Meredith, Sanjiva Weerawarana, Web Service Description Language (WSDL) Version 1.1, March 2001. Available at <http://www.w3.org/TR/wsdl>
- [24] deegree project (OGC Compatible GIS services) <http://deegree.sourceforge.net/>
- [25] demis project (OGC compatible Web Map Services) <http://www.demis.nl/home/pages/home.htm>
- [26] Jon Ferraiolo, Dean Jackson, Scalable Vector Graphics (SVG) Sprcification 1.1., January 2003, Available at <http://www.w3.org/TR/SVG/>
- [27] George Percivall, OpenGIS Consortium Reference Model 0.1.3, OGC Document #04-040, September 2003. Available at <http://www.opengeospatial.org/specs/?page=orm>
- [28] Roel Nicolai, The OpenGIS® Abstract Specification, Topic 2: Spatial referencing by coordinates 2.0.0. Document #03-073r1, October 2003. Available at <http://www.opengeospatial.org/specs/?page=abstract>
- [29] Sam Bacharach, OGC Consortium presentation, “Advancing Your Mission Objectives With Open Standards Frameworks.”, July 2003. Available at <http://www.opengeospatial.org/press/?page=presentations>

- [30] OpenGIS Simple Features Specification For SQL, Revision 1.1. May 1999. Available at <http://www.opengeospatial.org/docs>
- [31] OpenGIS Simple Features Specification For OLE/COM, Revision 1.1. May 1999. Available at <http://www.opengeospatial.org/docs>
- [32] OpenGIS Simple Features Specification For CORBA, Revision 1.8. March 1998. Available at <http://www.opengeospatial.org/docs>
- [33] Jean-Philippe Humblet, , OGC OpenGIS consortium, OpenGIS Web Map Context Documents 1.0.0, OGC Document #03-036r2, June 2003. Available at <http://www.opengeospatial.org/specs/?page=specs>
- [34] OpenGIS Coordinate Transformation Services 1.00. OGC Document #01-009, January 2001. Available at <http://www.opengeospatial.org/specs/?page=specs>
- [35] 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>
- [36] Mark Little, Eric Newcomer, Greg Pavlik., OASIS Web Services Context Specifications (WS-Context) 0.8. November 2004. Available at <http://xml.coverpages.org/WS-ContextCD-9904.pdf>
- [37] OGC OpenGIS Approved Specifications list. Available at <http://www.opengeospatial.org/specs/?page=specs>
- [38] OGC OpenGIS Abstract Specifications list. Available at <http://www.opengeospatial.org/specs/?page=abstract>
- [39] OGC OpenGIS Recommendation Papers list. Available at <http://www.opengeospatial.org/specs/?page=recommendation>
- [40] OGC OpenGIS Discussion Papers list. Available at <http://www.opengeospatial.org/specs/?page=discussion>