World Applied Sciences Journal 3 (1): 140-153, 2008
ISSN 1818-4952


# A Service Oriented Framework for Disseminating Geospatial Data to Mobile, Desktop and Web Clients

*Pouria Amirian and Ali A. Alesheikh*

Faculty of Geodesy and Geomatics Engineering,
K.N. Toosi University of Technology, Vali-e-Asr St., Mirdamad Cross, Tehran, Iran

**Abstract:** Geospatial data is vitally important to the ways thousands of government agencies, private companies and non-profit organizations do their businesses. The widespread access and sharing of geospatial data on the Internet yield many benefits. But, there are two important barriers in front of sharing and accessing geospatial data; non-interoperability and insufficient message exchange patterns of geospatial processing systems which are widely used in various organizations. Using standard geospatial Web services together with efficient technologies provide different kinds of message exchange patterns for various kinds of clients facilitate to share and access to geospatial data. This paper proposes a service oriented framework for disseminating geospatial data over the Web. Based on our practical tests, the proposed framework proved to be an efficient solution for sharing and accessing to geospatial data.

**Key words:** Service oriented . geospatial web service . web services . message exchange patterns

## INTRODUCTION

Nowadays geospatial data have become central to the practice of decision makers in private sectors as well as state authorities. The ever-increasing access to geospatial data on the Web enhanced system performance through cost/time reduction.. Moreover, such access helps decision-makers to manage their assets better, enables faster responses for time-sensitive decisions and improves the communication process across diverse agencies. If geospatial data are accessed and shared across diverse partners, different organizations can save time and money by avoiding data reproduction. also Information exchanges can solve problems relating to inconsistencies and quality differences in the geospatial data from various sources. In this case there are two important barriers in front of share and access of geospatial data; non-interoperability of geospatial processing systems which prevents sharing geospatial data and insufficient message exchange patterns which restricts access to geospatial data by subset of all potential users.

Based on OGC Reference Model [1], spatial interoperability refers to capability to communicate, execute programs, or transfer geospatial data among various functional units with a little or no knowledge of the unique characteristics of those units. Therefore, non-interoperability of geospatial processing systems prevents sharing of geospatial data and services among software applications. Spatial interoperability faces two main challenges; syntactic heterogeneity and semantic heterogeneity [2]. Syntactic heterogeneity which is the result of differences in storage formats and software incompatibility is a technical issue and can be addressed by technical means. Semantic heterogeneity arises as a result of incompatibility in meanings of data.

Syntactic heterogeneity of geospatial information systems can be categorized in data and access heterogeneity. In data heterogeneity geospatial processing systems use various internal proprietary data formats. To share geospatial data, converters and/or transfer formats must be developed, which is a resource and time consuming task. In addition, there are several standards for geospatial data that converting various data formats. This can itself become a barrier to interoperability.

Access heterogeneity restricts inter-process communication among various geospatial processing systems, since different vendors' geospatial processing systems use proprietary software access methods with proprietary software interfaces. In other words, interface definition languages, communication protocols, communication ports and even object transfer mechanisms, varies in each software development platform. So the software platform which has been used to develop the geospatial processing system imposes the use of specific and proprietary communication methods among various parts of such

**Corresponding Author:** Dr. Pouria Amirian, Faculty of Geodesy and Geomatics Engineering, K.N. Toosi University of Technology, Vali-e-Asr St., Mirdamad Cross, Tehran, Iran

system. For this reason, different geospatial processing systems which have been developed by different software development platforms, cannot communicate and share services automatically and in an interoperable manner.

From GIS point of view, Open Geospatial Consortium (OGC) has introduced specific kind of online services, to overcome spatial non-interoperability problems. These services which are called OGC geospatial Web services (or Geospatial Web Services for short) have been developed with the goal of sharing geospatial data and services among heterogeneous geospatial processing systems. Web Feature Service (WFS) and Web Map Service (WMS) are the most fundamental geospatial Web services that are introduced by OGC. At the same time, in IT (Information Technology) world, the best solution for providing interoperability among heterogeneous software systems in distributed and decentralized environments are Web services technologies [3].

Geospatial Web services and Web services differ in a way that Web services are composed of particular set of technologies and protocols but Geospatial Web services are comprised of defined set of interface implementation specifications which can be implemented with diverse technologies [4].

Although Geospatial Web services provide interoperability among heterogeneous geospatial processing systems (sharing geospatial data), in most cases they just provide geospatial data in one message exchange pattern. More accurately, geospatial data are to be accessed by various kinds of clients such as desktop, Web and mobile clients. Desktop clients are almost always access to geospatial data which are resided in a central data server through internal networks. They access to geospatial data in its lowest level of details (feature level access) but in most cases they cannot make use of other geospatial data resources over the Web. At the other hand, Web clients in most cases just retrieve an image of geospatial data (high level access to geospatial data) via Web-GIS applications. Web clients can take advantage of many resources of geospatial data but they are always restricted to high level access to geospatial data. Mobile users can request geospatial data from some specific services as long as they are connected to a wireless network.

In order to make geospatial data accessible to various kinds of clients, geospatial processing systems should implement various message exchange patterns.

This paper proposes a service oriented framework for disseminating geospatial data to desktop, Web and mobile clients. Services in this framework provide three different message exchange patterns for three different kinds of clients. Two distributed object technologies are used to implement the services in this framework.

The paper first assesses distributed applications (particularly distributed object technologies), .NET Remoting technology, Web services and various message exchange patterns. Then Asynchronous JavaScript and XML (AJAX) technology as a modern implementation of one of the message exchange patterns discussed briefly. Afterwards, Geospatial Web services described and finally the proposed framework will be dissected. In order to evaluate the proposed framework, three different client applications were utilized to retrieve geospatial data from the developed geospatial Web services.

**Distributed architecture and distributed component technologies:** The architecture of a software application is the structure and configuration of its components. One of the most important characteristics of a software application is its ability to share data, information and processing services. In this context distributed architectures are often used to achieve high level interoperability in many types of systems, including GIS, databases and location-based services [2]. Generally, by dividing an application into multiple layers (or tiers), it becomes possible for several computers to contribute in the processing of a single request, thus making application a distributed one. This distribution of logic, typically slows down individual client requests (because of the additional overhead required for network communication), but it improves the scalability for the entire system. Distributed applications are implemented based on distributed architecture which enables them to make use of remote computers processing resources. More accurately, a distributed architecture is a collection of independent computers that appears to its users as a single coherent system [6].

There are several technologies and concepts for implementing distributed applications. Distributed component technologies provide an efficient software pattern and platform for implementing distributed applications [7].

The main concept behind all the existing distributed component technologies, is to ask objects which are running on remote machines to perform computational tasks on behalf of the objects residing in a local machine. Objects which are resides in local and remote machines are in different memory spaces and communicate via messages over a network through predefined communicational channels and protocols.

Out of the existing alternatives, Java-RMI (Remote Method Invocation) from Sun Microsystems, CORBA (Common Object Request Broker

Architecture) from Object Management Group, DCOM (Distributed Component Object Model) and .NET Remoting from Microsoft and the Web services are the most popular distributed-component technologies among researchers and practitioners [7].

Web services are unique distributed component technology; since they are a defined set of open and non-proprietary XML (eXtensible Markup Language) vocabularies, they provide true interoperability among heterogeneous software platforms. But the unprecedented level of interoperability provided by Web services comes at a price; Web services are quite slower than the other distributed component technologies.

Interoperability is especially important to users of GIS, as geospatial analysis often relies on the integration of data from multiple resources. Considering unique characteristics of geospatial data and geoprocessing needs, high performance and communication speed is as important as interoperability.

In this research Microsoft's .NET platform was used for implementing such services. .NET Remoting is an infrastructure which enables inter-process communication.

**.NET remoting:** .NET Remoting was introduced as part of Microsoft's .NET platform. From the developer's perspective, .NET platform is a huge set of types (Classes, Structures, Namespaces, Interfaces and Delegates), which substituted the older Windows programming API's (such as the Windows 32-bit API and the MFC (Microsoft Foundation Classes)), Web APIs (like ASP (Active Server Page)). With respect to the distributed component technologies, it replaced DCOM. From component oriented view point, ordinary .NET components are used instead of COM (Component Object Model) and DCOM Components. .NET Web Service components are intended to be cross-platform and cross-language Web-based components [8].

.NET Remoting allows client applications to instantiate components on remote computers and uses them like local components [5]. .NET Remoting is the ideal choice for intranet scenarios. It typically provides the greatest performance and flexibility, especially if remote objects have to be maintained their state or there is a need for creating peer-to-peer applications [9].

Using .NET Remoting in an enterprise application means hosting software components on separate computers and making them available through .NET Remoting infrastructure. Software components which are hosted on remote computers are called remote components. Remote components have to be hosted and activated using a component host. Component host typically is a service on Microsoft Windows which is responsible for listening and replying to client requests. In .NET Remoting, a remote component can be hosted in one of several different types of applications. It can be hosted in a typical desktop application, dedicated Windows service or by Microsoft's main Web server software (IIS (Internet Information Services)).

With .NET Remoting, the communication between local and remote components is accomplished through a proxy layer. This approach is conceptually similar to the way other distributed object technologies communicate. With this type of communication, local component communicates directly to proxy layer which mimics the remote component's interface; that is when the local component calls a method on proxy layer, it calls the remote component behind the scene, waits for response and then returns the appropriate information to local component (Fig. 1). In .NET Remoting, when communicating with remote component, proxy layer does not actually perform all the required tasks for remote communication. Instead, it communicates with a formatter object and transport channels to perform the remote communication. The formatter object packages

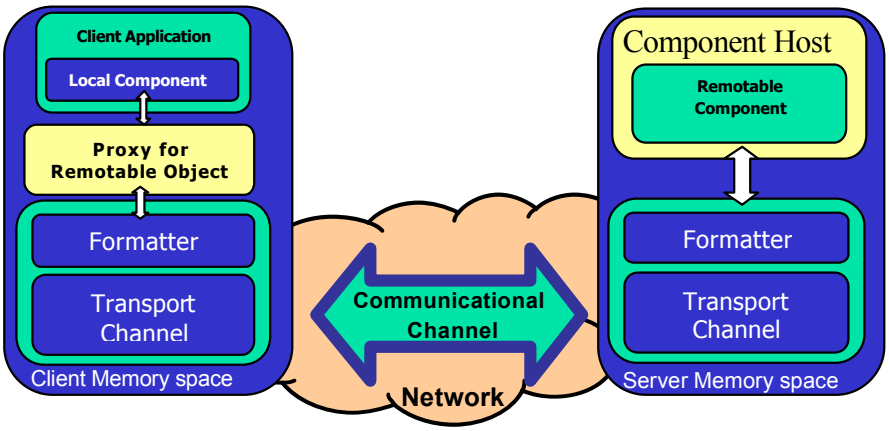


Fig. 1: .NET remoting architecture

the client request or server response in an appropriate format then communicates with a transport channel which transmits the packages using the appropriate protocol. Figure 1, illustrated how a method invocation is packaged on the client side and then unpacked on the server side. In this case, the information flows from the client to the server, but the return value flows back from server to the client along the same communicational path.

By default, .NET Remoting provides two formatters (binary and SOAP (Simple Object Access Protocol) formatters) and two channels (HTTP and TCP (Transmission Control Protocol) channels). Binary formatter packages the request and response to a compact, proprietary .NET format. As a result, this formatter offers the best performance but can be used only by .NET applications. SOAP formatter serializes the request and response to SOAP messages, which is a cross-platform XML-based plain-text format. SOAP format requires larger message size, therefore, it can reduce overall performance but it provides cross-platform communication with a remote .NET component [10].

There are also two predefined transport channels; TCP channel and HTTP channel. As the name implies, TCP channel uses the TCP protocol and it is ideal for internal networks. At the other hand, HTTP channel uses the HTTP protocol which is ideal for scenarios in which there is a need for communicating across Web.

.NET Remoting is not the only way of implementing distributed applications in .NET platform. XML Web services (which are the concrete implementation of Web services technologies) are among the most widely hyped features of Microsoft .NET platform.

Service Oriented Architecture and Web Services Technologies

Service orientation is a trend in software engineering that promotes the construction of applications based on entities called services [11].

Service-Oriented Architecture (SOA), as an architectural platform, is adopted today by many businesses as an efficient means for integrating enterprise applications built of Web services [12]. Services in an SOA are modules of business or technical functionality with exposed interfaces to the functionality [13].

Web services technologies are the implementation of a conceptual architecture, which is called Service Oriented Architecture [14]. SOA is frequently characterized as a style that supports loose coupling, business alignment and Web-based services, which permits extensibility and interoperability independent of the underlying technology [15]. SOA provide a set of guidelines, principles and techniques in which business processes, information and enterprise assets can be effectively (re)organized and (re) deployed to support and enable strategic plans and productivity levels that are required by competitive business environments [16].

In SOA, the central elements are services. In many respects, a service is the natural evolution of the component, just as the component was the natural evolution of the object. Services are autonomous, platform-independent computational elements that can be described, published, discovered, orchestrated and programmed using standard protocols for the purpose of building networks of collaborating applications distributed within and across organizational boundaries [17].

Service-orientation is the correct way to build maintainable, robust and secure applications [18]. As illustrated in Fig. 2, SOA consists of three primary roles and three primary tasks. Service provider, service requester and service broker are distributed computational nodes in the network environment. Service provider publishes its own service with service broker. Service requester uses the service broker to find desirable services and then binds to a service provider to invoke the service (Fig. 2).

SOA architecture allows obtaining a loose coupling between its processing components (service requester and service provider), because, it uses simple generic and application-independent connectors and messages defined by an XML schema [19].

The actual implementation of SOA using open, standard and widely used protocols and technologies is called Web services [14]. Web Services are the basic components of distributed service-oriented systems. The World Wide Web Consortium (W3C) defines Web Services as a software system designed to support machine-to-machine interaction over the Internet [20, 21, 22].

Each Web service has an interface described in a machine-processable format. Other systems and services interact with the Web service in a manner described by its description using messages. Messages are conveyed typically using HTTP with an XML

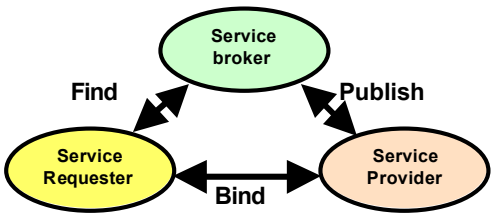


Fig. 2: Major components of service oriented architecture

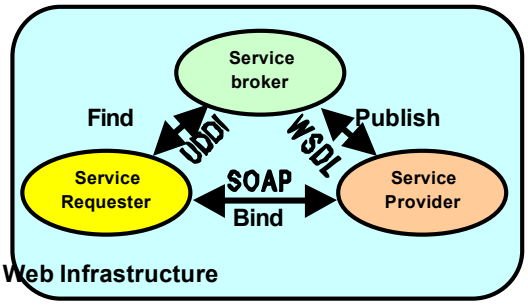


Fig. 3: Mapping between SOA and web services technologies

serialization, in conjunction with other Web-related standards, but any other communication protocol can be used as well [21].

Web services are based on open standards, so they provide interoperability in decentralized and distributed environments like Web. These new technologies can be developed by using any software platform, operating system, programming language and object model.

Web services are implemented using a collection of standards and technologies. SOAP, WSDL (Web Service Description Language) and UDDI (Universal Description, Discovery and integration) form the core technologies for implementing Web services [23].

SOAP is a lightweight, XML-based protocol for exchanging information in decentralized, distributed environments. SOAP is used for messaging among various SOA's components in a Web services platform. SOAP is platform independent and also it can be used with virtually any Network Transport protocols such as FTP (File Transfer Protocol), HTTP, HTTP-S (Secure HTTP) and HTTP-R (Reliable HTTP).

WSDL is XML-based specification for describing the capabilities of a service in a standard and extensible manner. Technically, WSDL defines the software interface of Web service in platform independent approach.

UDDI is a set of specifications and APIs (Application Programming Interfaces) for registering, finding and discovering services.

As illustrated in Fig. 3, these layers establish an explicit mapping between elements of SOA as a conceptual and technology independent architecture and Web services as specific collection of standards, protocols and technologies.

In this case, Web service provider publishes its own service description using WSDL, then Web service requester take advantage of search API's of UDDI to find appropriate Web services and finally, Web service requester binds to the service provider using SOAP.

From a technical standpoint, each Web service has three main parts: Service description, Executable agent and the mapping layer between the two (Fig. 4).

The machine-readable service description (that is a WSDL document) contains network address for the service, the operation it supports and other necessary information for consuming the service. The executable agent is responsible for implementing the functionality of the service The description is separated from the executable agent using a mapping layer The mapping layer is often implemented using proxies and skeleton in service requester and service provider respectively. This layer is responsible for accepting the message, transforming the XML data to and from the native format of executable agent and finally dispatching the data to the executable agent.

On account of separation between executable environment and description of service or separation between semantic and functionality of services in the Web services world, each service can be developed by using any software development platform, operating system, programming language and object model.

From middleware point of view, Web service technologies are one of distributed component technologies. But the goal of Web services goes beyond those of classical distributed component technologies

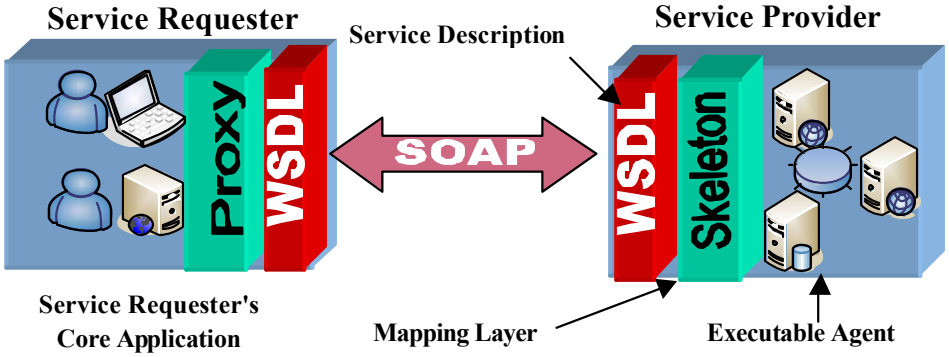


Fig. 4: Major parts of a web service

such as .NET Remoting and CORBA: Web services aim at standardized support for higher level interactions such as service and process flow orchestration, enterprise application integration and provision of middleware of middleware [24]. Instead of building applications that result in collections of objects or components that are firmly integrated and understood just in development time (but fairly hard to configure in deployment time), the service approach of Web services platform is much more dynamic and is able to find, retrieve and invoke a distributed service dynamically. Another key difference is that with Web Services the industry is solving problems using technologies and specifications that are being developed in an open way, via partnerships and consortia such as the W3C and the Organization for the Advancement of Structured Information Standards (OASIS) and using standards and technologies that are the basis of the Internet. Next section will briefly explain the major differences between .NET Remoting and Web services technologies.

**Web services vs. .NET remoting:** Implementing SOA using Web services can be taught as adaptation of distributed component technology with Web infrastructure. Some of the important advantages of using Web services as the technology platform for implementing SOA are derived from the way in which the World Wide Web achieved its tremendous success; in particular, the fact that a simple markup language (XML) can provide a powerful interoperability solution and the fact that a lightweight document transfer protocol (HTTP) can provide an effective, universal data transfer mechanism.

The following items describe major differences between .NET Remoting and Web services technologies (or more accurately XML Web services in .NET platform):

- Components published via Web services are more restricted than components exposed over .NET Remoting. Web services are designed around the stateless exchange of messages. In .NET Remoting it is possible to expose statefull components as well.
- Since .NET Remoting Infrastructure can use binary formatter and TCP channel, communication with .NET Remoting is faster than Web service communication
- Web services support open standards which target cross-platform use. Any client that can parse XML message and is connected over an HTTP channel may use a Web service. Even if the client is written in Java and hosted on Linux it can consume a Web service which is written in C# and hosted on Windows.
- Web services are designed for use between organizations. They can use a discovery mechanism or a UDDI registry that advertises services to interested parties over the Web. With .NET Remoting there is no clearly defined registry service.
- Web services are firewall-friendly. This means since most Web services communicate through HTTP channel, there is no need for an administrator to open additional ports on firewall. In contrast, with .NET Remoting any port can be utilized for inter-process communication, doing so could leave a major hole in security.
- Unlike Web services .NET Remoting can be used for implementing peer-to-peer applications in which individual clients communicate back and forth and there is no central server.
- .NET Remoting is more suitable as a high-speed solution for binary communication between proprietary .NET components usually over internal networks. Although Web services cannot match the communication speed and stateful communication scheme of .NET Remoting, they can be used in cross-platform scenarios.

**Message exchange patterns in web services and .NET remoting technologies:** Messaging or communication between applications is the main idea of distributed applications. Flexibility and extensibility of Web services protocol stack and .NET Remoting infrastructure allows various message exchange patterns to be implemented for various kinds of clients. In this research three message exchange patterns are used for mobile, Web and Desktop clients:

- Synchronous Request/Response interactions.
- Asynchronous Request/callback interactions.
- Message Queuing Interactions.

**Synchronous request/ response pattern:** This pattern resembles the Remote Procedure Call communication pattern in DCOM and Java RMI. In the request/response pattern, the service requester sends the request and then waits for the reply. In other words, service requester will be blocked until the response is made. This pattern is Synchronous because it requires the sender and the receiver to be online simultaneously for data to be exchanged. Also this pattern is Request/response because it allows for data to be exchanged in both directions.

Although WSDL and SOAP both support a request/response interaction style, it is worth noting that the SOAP and WSDL definitions are not executable and that any business logic needs to be implemented by a run-time environment such as J2EE, .NET Framework, or CORBA [25].

**Request/callback interaction paradigm:** The request/callback interaction paradigm is usually utilized when the service requester cannot be blocked while waiting for a synchronous response, so instead it sets up a callback agent (or process) to handle the response.

In Request/Callback pattern, the typical sequence of actions is:

- The service requester sends a request message to the service provider using a one-way request message. The request message includes a correlation ID and a callback address. After the service requester sends the request message, it continues executing and does not block while waiting for the response (for this reason the service requester sends the one-way message).
- The service provider receives the request message, composes a response and sends a callback message to the callback service by sending a one-way response message to the callback address that was included in the service requester's original request, including the correlation ID.
- The callback service receives the response message and processes it as appropriate (which may include notifying the service requester of the response or dispatching the result of a remotely processed task).

In Web service protocol stack, WSDL and SOAP do not provide formal support for request/callback interactions and it is up to the application layer to manage the various elements of the request/callback interaction, including defining callback addresses and generating correlation IDs [25]. Today, this pattern is heavily used in Web-based applications to create rich and responsive user interfaces through AJAX (Asynchronous Javascript and XML) technology. The AJAX technology will be described later in this paper.

**Message queuing interaction:** In this interaction style, communication between service provider (component host) and service requester (client) is accomplished via the use of persistent queues. This style is well suited for the partially connected and mobile clients. In this case the service on mobile client places a request message in a request queue and the messaging technology reliably delivers the message to another mobile client or the main service provider where the receiving service dequeues it and processes it (when the connection is available).

One of the advantages of this message exchange pattern is that a request queue can be persistent; allowing the application to continue working whether or not a connection to the remote machine is available.

**AJAX technology:** The Ajax technology is the explicit access of server-side code from within the context of client-side scripts [26]. To understand how AJAX works in a web application it is necessary to show the fundamental differences between it and traditional Web application models. In traditional Web application a web browser requests a webpage, normally indicating that the request is being processed by animating a logo and altering the status bar. When the user clicks on a link, an HTTP Get request is sent to the server. The web server deals with the request and sends the web page to the client. If the client is to send information back to the server, another request is made following the same process. Under this synchronous click-and-wait communication method, information is exchanged by requesting and receiving whole web pages [27]. While waiting for the server, the user loses the focus of the application and cannot interact with it. This loss of focus has long been a source of dissatisfaction with traditional web applications and if the wait for a round trip from the server is sufficiently long, users may leave the site.

In AJAX web applications a client requests a webpage. Once this full page is loaded, communication between the client and the server can be conducted in an asynchronous callback manner. This minimizes the client's waiting time, because only partial user interface update requests are made [27]. Only aspects of the client's user interface are updated in an AJAX scenario. Those that are not modified by the user remain static, reducing the communication overhead. This leaves the focus of the application with the user, creating a feeling of seamless interactivity.

**Geospatial web services:** Nowadays, geospatial Web services have been considered as the promising technology to overcome the non-interoperability problem associated with current geospatial processing systems. They are particular kind of online services which deal with geographical information and can provide access to geographical information stored in a database, perform simple and complex geospatial analysis and return messages that contain geographical information [28].

In this context, OGC has defined a comprehensive framework of geospatial Web services which is known as OGC Web services framework (OWS). OWS allows distributed spatial processing systems to interact with the Hypertext Transfer Protocol (HTTP) technique and provides a framework of interoperability for the many web-based services, such as accessing spatial data services, spatial processing services and data locating services [29]. OWS framework consists of interface implementation specification and encodings which are openly available to be implemented by developers. The interface implementation specifications are software technology neutral details about various operations of each geospatial Web service. The encodings provide the standard glue among different parts of geospatial Web services. Each service of this framework can be implemented using various software technologies and systems. The most fundamental services and encodings of the OGC Web service framework are Web Map Service (WMS), Web Feature Service (WFS) and Geography Markup Language (GML) [30]. Next sections briefly introduce WFS, WMS and GML.

**GML:** GML is an XML-based markup language that is used to encode information about real world objects. In GML these real world objects are called features and they have geometry and non-geometry properties.

GML has three main roles with respect to geospatial information. First as an encoding for the transport of geospatial information from one system to another; second as a modeling language for describing geospatial information types and third as storage format for geospatial information [31].

Typically in any management related tasks (like environmental management, natural resource and so on) one needs to examine and explore data from several sources, use simulation models, develop scenarios, assess impacts and provide support for decision makers [32]. In this case, use of XML-based languages for data exchange is an improvement on non XML data formats because the XML format is partially self-documenting and provides common methods for parsing files, obtaining their structure and transforming them to alternative formats [33]. GML (as an XML-based language) is well suited for encoding the geospatial information sent to and from geospatial Web services. GML is used in both the request and response messages of the WFS, which is a standard service for accessing geospatial feature data.

As a modeling language, GML provides a rich variety of objects for describing geospatial information, including geospatial features, coordinate reference systems, topology, time, units of measure and generalized values [34]. In addition, using GML spatial and non-spatial relationships among real world objects can be modeled efficiently.

As storage format GML is a plain textual file format which can be managed using any database management system.

Figure 5, illustrates a simple GML document fragment which consist of two features. City feature has three properties; Name, Position and IsCapitalOf. Name of the city is declared using Name element and Position of the city is expressed using gml:Point element which is defined in GML standard. The gml:Point has a srsName attribute for denoting Spatial Reference

```
<City gml:id="C30">
        <Name>Tehran</Name>
        <gml:position>
                <gml:Point srsName="WGS84">
                        <gml:coordinates>3950000,530000</gml:coordinates>
                </gml:Point>
        </gml:position>
        <IsCapitalOf xlink:href="#T1" />
</City>
<Country gml:id="T1">
        <Name>IRAN</Name>
        <Continent>Asia</Continent>
        <Region>Southern Asia</Region>
        <Capital xlink:href="#C30"/>
</Country>
```

Fig. 5: A simple GML document fragment which describes some properties of Tehran as City feature and Iran as Country feature. Position of City is indicated using gml:Point element which is defined in GML standard. Association between Tehran and Iran is expressed using xlink:href attribute

System (SRS) in which the coordinates are represented. As the name implies IsCapitalOf property states relationship between city and country features. In this case Tehran is the capital of country feature which has "T1" as its gml:id attribute. At the other hand, country feature has four non-geometry properties which state its name, continent, region and its capital city. The Capital property of country feature is used to indicate its capital city. In both features Xlink:href attribute is used to express association between country and city features. Xlink:href is defined in XLink standard. XLink is a W3C standard that specifies the syntax and behavior for hyperlink traversal in a set of XML documents [35]. These links are used in GML to express associations between geospatial features.

**Web Feature Service (WFS):** Web Feature Service is the main geospatial Web service for publishing and requesting vector geospatial data in GML format. When a client sends a request to an OGC WFS, the service sends a response message that provides geographical feature data in GML**.** Three classes of Web Feature Services are defined in the WFS implementation specification: Basic WFS, XLink WFS and Transaction WFS [36].

A Basic WFS service implements three operations: GetCapabilities, DescribeFeatureType and GetFeature.

A client can request an XML-encoded capabilities document (containing the names of feature types that can be accessed via WFS service, the spatial reference system(s), the spatial extent of the data and information about the operations that are supported) by sending the GetCapabilities request to the WFS service.

The purpose of the DescribeFeatureType operation is to retrieve an XML Schema document with a description of the data structure (or schema) of the feature types served by that WFS service.

The GetFeature operation allows for the retrieval of feature instances (with all or part of their attributes) as GML.

An XLink WFS supports all the operations of a basic web feature service and in addition it would implement the GetGmlObject operation for local and/or remote XLinks.

A Transaction WFS supports all the operations of a basic web feature service and in addition it implements the transaction operation. A transaction request is composed of operations that modify features; that is create, update and delete operations on geographic features.

**WMS:** Based on WMS implementation specification [37], The Web Map Service enables maps in graphical form to be delivered in response to queries from HTTP clients (any desktop or Web application which is connected to World Wide Web). In the context of WMS a map is a raster graphic picture of the data rather than the actual data itself.

Clients request maps from a WMS instance in terms of named layers and provide parameters such as the size of the returned image as well as the spatial reference system to be used in drawing the map. In this way, a client application can make requests to different WMS instances and the results can be overlaid to form a rich layering of map information based on the transparency capability of WMS instances.

The WMS specification standardizes two mandatory operations by which maps are requested by clients: GetCapabilities and GetMap.

The purpose of the GetCapabilities operation is to obtain service metadata, which is a machine readable (and also human-readable) description of the server's information content and acceptable request parameter values.

The GetMap operation returns a map whose geographical and dimensional parameters are well defined in the GetMap request. The GetMap request allows the WMS client to specify distinct layers, the spatial reference system, the geographic area and other parameters describing the returned map format.

**Message exchange patterns in geospatial web services:** In geospatial Web services each message exchange pattern can be used in different scenarios and for various kinds of clients.

The synchronous Request/response pattern can be used in situations in which a specific analysis or low volume geospatial data is needed in order to perform another (local) process. In this case, the execution of the local process is blocked until the required action takes place. This pattern is the foundation of next generation of distributed desktop GIS applications in which functionality provided remotely and through the use of the registered (and also trusted) service providers over the decentralized environment (such as WWW). As mentioned before, in the context of geospatial Web service, WMS is intended for publishing geospatial data as image files (which is called maps in WMS specification). Since image files are considered as low volume data, requesting and waiting for response is negligible for desktop users. In addition, multi-threading capability of desktop application, make them responsive while response is being returned to the application. Since responsiveness is implemented in user interface layer, there is no need for desktop applications to make asynchronous calls. As a result Request/response pattern is the natural

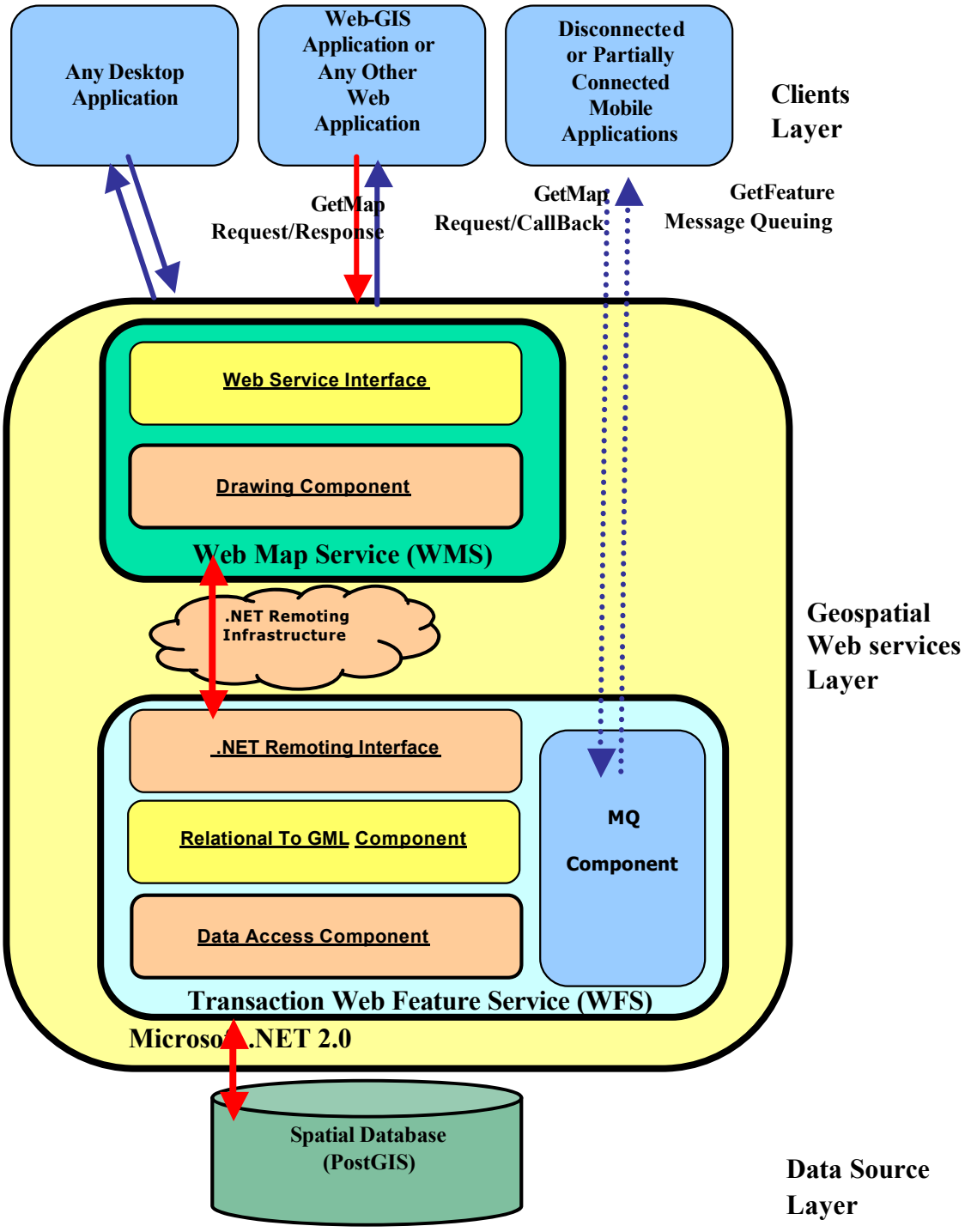


Fig. 6: Architecture of service oriented framework

choice for consuming geospatial Web services by desktop applications.

The asynchronous Request/Callback pattern can be employed when there is a need to perform some (more than one) tasks in sequential order. An example of this situation is the chain of geoprocessing analysis in which the output of the first task will be the input of the second one. Since that is a time consuming task, the requester should be unblocked to be able to perform other tasks. In addition to chain of services, when high volume geospatial data is needed to be downloaded from remote resource, this pattern provides an alternative solution over the use of queuing systems. With respect to the above description, web applications

should be implemented using Request/callback interaction pattern. In most cases Web GIS applications should be implemented using this pattern. More accurately, Web GIS applications which consume geospatial data (in the form of image or features) from geospatial Web services should request geospatial data in asynchronous manner to provide rich user experience.

Message Queuing Pattern is the main candidate for the partially connected or mobile GIS applications. The partially connected clients are not always connected to networks and, therefore, they cannot always interact with server side code in synchronous or asynchronous manner. Particularly, updating and gathering geospatial data can be done by field workers using this pattern (in the case of Transaction WFS). In this case geospatial data modification (Insert, Update and Delete geospatial data) requests are placed in a request queue and the messaging technology reliably delivers the message to main service provider (WFS server) which is in charge of committing changes to the spatial database. Subsequently, main service provider dequeues incoming messages and processes them sequentially. In addition to gathering or updating geospatial data when mobile applications need geospatial data, this pattern can be used to request geospatial data. In this case request is placed in persistent request queue and sent to geospatial Web service when a network connection became available to the mobile device.

**Service oriented framework for disseminating geospatial data:** Figure 6, shows the architectural view of service oriented framework for disseminating geospatial data. This framework has three layers: client layer, geospatial Web services layer and data source layer.

The data source layer consists of geospatial data which are stored in a spatial database. In this research PostGIS is used to store and manage geospatial data. The PostGIS is a spatial extension of open source PostgreSQL relational database. The PostGIS spatially enables the PostgreSQL allowing it to be used as a backend spatial data base for geospatial data.

The geospatial Web services layer include a WMS and a Transaction WFS. Implementing transaction WFS enables users to update data source layer. In addition, WMS retrieves geospatial data from WFS. In other words, access to geospatial data which are resided in spatial database is only permitted to WFS. Designing WFS as a gateway to spatial database provides the flexibility to change data sources or data access technologies without affecting the WMS. This is important because there may be a need to switch from one database vendor to another at some point.

The WFS has three main components (Fig. 6). The data access component interacts with the data source layer to retrieve and update geospatial data. The data access component doesn't actually manage or store the data; it merely provides an interface between the WFS and the spatial database. The Relational-to-GML component turns geospatial data (which are stored as relational tables in PostgreSQL) to GML 3.1. The MQ component is used to deliver geospatial data to partially disconnected clients using message queuing pattern.

The WMS retrieves geospatial data from WFS and converts it to image formats using drawing component. All components of WFS and WMS was developed using Microsoft .NET Framework 2.0.

Since WMS and WFS are in the same layer (and also same machine) interaction between them should be performed in fastest possible way. So interaction between WMS and WFS is performed using .NET Remoting infrastructure. To make geospatial data available to any type of client on any type of software platform, a Web services interface is implemented for WMS.

Clients in this architecture interact with geospatial Web services in three different ways. Any desktop application which is able to connect to Web can request geospatial data from WMS (request/response message exchange pattern). Figure 7, illustrates a ESRI's ArcGIS desktop application which is connected to the implemented WMS.

Using AJAX technology, a basic Web GIS application was developed to provide a basic Web-based user interface for WMS using ASP .NET (Technically ASP .NET is set of Class hierarchies for developing Web-based applications, both Web sites and Web services) platform (Fig. 8). Web GIS application provides basic tools such as: Zoom in, Zoom out, Full Extent and Identify. Functionality of these tools developed using JavaScript language. Map image of Web GIS application created using WMS service (Request/Callback message exchange pattern). In fact, Web GIS application just provides a user friendly gateway to WMS service. Each tool of Web GIS, just send an asynchronous GetMap request to WMS service to request appropriate map image.

In addition to desktop and Web GIS application any mobile client can take advantage of developed services and retrieve geospatial data in its preferred format. For practical test of message queuing exchange pattern, a simple mobile application was developed using .NET Compact Framework 2.0. This mobile application uses Microsoft MSMQ2.0 component to provide message queuing exchange pattern. In this application, user provides information (such as location in the form of latitude and longitude, name, time and other information) about a point object to be inserted

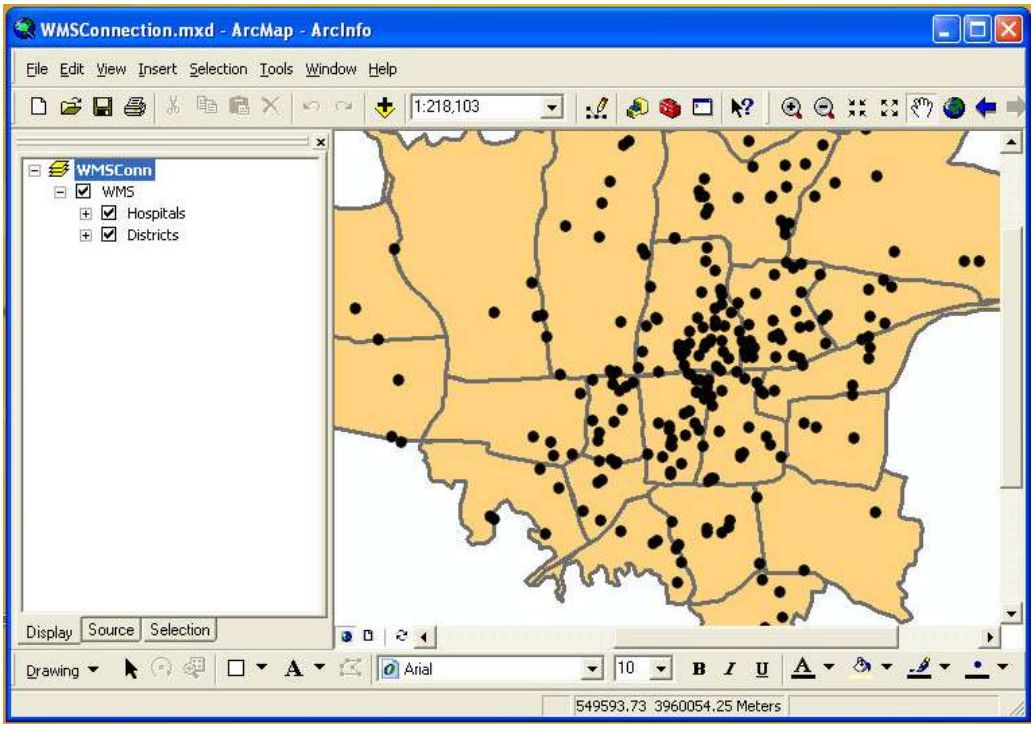


Fig. 7: ESRI ArcGIS desktop Application which is connected to WMS and retrieve geospatial data from that service

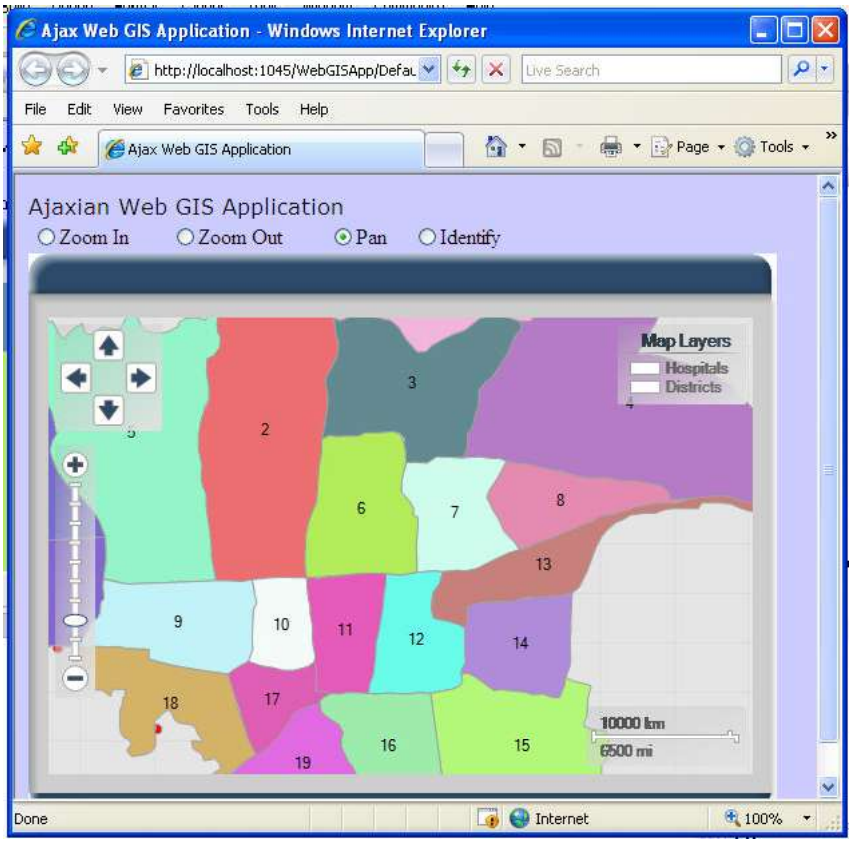


Fig. 8: AJAX Web GIS Application as a gateway to WMS


Fig. 9: Mobile Application to insert new geospatial objects into spatial database through WFS

into spatial database. In this case, information about point object is persisted in a queue and posted to WFS when a network connection became available to mobile device (Fig. 9).

## CONCLUSION

In this paper the design and implementation of a service oriented framework for disseminating geospatial data were demonstarted. The proposed framework provides geospatial data through geospatial Web services using various kinds of message exchange patterns for different kinds of clients.

Two distributed object technologies are utilized to implement the mentioned framework: .NET Remoting and Web services. .NET Remoting is an efficient and high-speed solution for binary communication between proprietary .NET components over internal networks. By considering high volume of geospatial data and faster communication speed provided by .NET Remoting, modifying geospatial data through the use of transaction WFS can be efficiently performed if .NET Remoting is used rather than HTTP-based solutions. Although Web services cannot match the communication speed and statefull communication scheme of .NET Remoting, they can be used in cross-platform scenarios. So the functionality of the implemented geospatial Web services can be simply added to any geospatial or non-geospatial software systems which are running on heterogeneous platforms.

## REFERENCES

1. OGC, 2003. The OpenGIS Reference Model, http://portal.opengeospatial.org/files, visited on October 2007.
2. Worboys, M. and M. Duckham, 2004. GIS a Computing Perspective, Florida, USA, CRC Press.
3. Volter, M., M. Kricher and U. Zdun, 2005. Remoting Patterns: Fundamental of Enterprise, Internet and Real-time Distributed Object Middleware, New Jersey, USA, John Wiley and Sons, Inc.
4. Amirian, P., 2006. Design and Development of a Distributed Geospatial Web services using XML and .NET technologies. Msc thesis, K.N. Toosi University of Technology, Tehran, Iran.
5. Selly, D., A. Troelsen and T. Barnaby, 2006. Expert ASP .NET 2.0 Advanced Application Design. California, USA, Apress Publishing.
6. Tanenbaum, A. and M. Van Steen, 2005. Distributed Systems: Principles and Paradigms, New Jersey, USA, Prentice Hall, Inc.
7. Hariri, S. and M. Parashar, 2004. Tools and Environments for Parallel and Distributed Computing, New Jersey, USA, John Wiley and Sons, Inc.
8. Wang, A.. and K. Qian, 2005. Component Oriented Programming, New Jersey, USA, John Wiley and Sons, Inc.
9. MacDonald, M., 2003. Peer To Peer with VB .NET . California, USA, Apress Publishing.
10. Rammer, I., 2005. Advanced .NET Remoting, 2nd Edn. California, USA, Apress Publishing.
11. Stojanovic, Z. and A. Dahanayake, 2005. Service Oriented Software system engineering: Challenges and practices, Idea Group Publishing.
12. Vasiliev, Y., 2007. SOA and WS-BPEL. Packt Publishing.
13. Lawler, J. and H. Hawel Barber, 2008. Service Oriented Architecture: SOA strategy. Methodology and Technology, Taylor and Francis Group.
14. Marks, E. and M. Werrell, 2003. Executive's Guide to Web Services, New Jersey, USA, John Wiley and Sons, Inc.
15. Murakami, E., A.M. Saraiva, L.J. Ribeiro, C.E. Cugnasca, A.R. Hirakawa and P.L. Correa, 2007. An infrastructure for the development of distributed service-oriented information systems for precision agriculture. Journal of Computers and Electronics in Agriculture, May 2007, 58: 37-48.

16. Papazoglou, M.P. and W.J. van den Heuvel, 2006. Service-oriented design and development methodology. International Journal of Web Engineering and Technology (IJWET 2006), pp: 412–442.
17. Papazoglou, M.P. and W.J. van den Heuvel, 2005. Web Services Management: A Survey. IEEE Journal of Internet Computing, November/ December 2005, 9 (6): 58-64.
18. Lowy, J., 2005: Programming WFC Services, California, USA, O'Reilly Media, Inc., Orielly.
19. Fallside, D.C. and P. Walmsley, 2004. XML Schema Part 0: Primer Second Edition. W3C Recommendation. Available from http://www.w3.org/TR/xmlschema-0 (accessed 05-05-2007).
20. Stal, M., 2002. Web Services: Beyond Component -based Computing. Journal of Communications of the ACM, 45 (10): 71–76.
21. Booth, D., H. Haas, F. McCabe, E. Newcomer, M. Champion, C. Ferris and D. Orchard, 2004. Web Services Architecture. W3C Working Group. Available from http://www.w3.org/TR/ws-arch (accessed 19-05-2007).
22. W3C, 2006. The World Wide Web Consortium. Web Services Activity Statement. Available from http://www.w3.org/2002/ws/Activity (accessed 05-05-2007).
23. Gailey, J.H., 2004. Understanding Web Services Specifications and the WSE. Washington, USA, Microsoft Press.
24. Vinoski, S., 2003. Integration With Web Services IEEE Journal of Internet Computing. November-December 2003.
25. Newcomer, E. and G. Lomow, 2005. Understanding SOA with Web Services, Maryland, USA, Addison Wesley, Inc.
26. Woolston, D., 2006. Pro Ajax and the .NET 2.0 Platform, California, USA, Apress Publishing.
27. Ritchie, P., 2007. The security risks of AJAX and web 2.0 applications. Network Security Volume 2007, Issue 3, March 2007, pp: 4-8.
28. Lake, R., D. Burggraf, M. Trinic and L. Rae, 2004. Geography Markup Language, Chichester, England, John Wiley and Sons.
29. Zhang, J., J. Gong, H. Lin, G. Wang, J. Huang, J. Zhu, B. Xu and J. Teng, 2007. Design and development of Distributed Virtual Geographic Environment system based on web services. Information Sciences, 1 October 2007, 177 (19): 3968-3980.
30. Amirian, P. and A. Alesheikh, 2008. Publishing Geospatial Data through Geospatial Web Service and XML Database System. American Journal of Applied Sciences, 5 (10): 1358-1368.
31. Lake, R., 2005. The application of Geography Markup Language (GML) to the geological sciences. Computers&Geosciences, November 2005, 31 (9) 1081-1094.
32. Kokkonen, T., A. Jolma and H. Koivusalo, 2003. Interfacing environmental simulation models and databases using XML. Environmental Modelling & Software, 18 (5): 463-471.
33. Sen, M. and T. Duffy, 2005. GeoSciML: Development of a generic GeoScience Markup Language. Computers and Geosciences, 31 (9): 1095-1103.
34. Nativi, S., J. Caron, E. Davis and B. Domenico, 2005. Design and implementation of netCDF markup language (NcML) and its GML-based extension (NcML-GML) Computers and Geosciences, 31 (9): 1104-1118.
35. Open GIS Consortium, 2005. Geography Markup Language Specification 3.1. available at: http://portal.opengeospatial.org/files/?artifact_id=4700 (accessed 19-05-2007).
36. Opern GIS Consortium, 2005. Open GIS Web Feature Service implementation specification 2.1. available at: https://portal.opengeospatial.org/files/ (accessed 05-05-2007).
37. OGC, 2004, Open GIS Web Map Service implementation specification. Available at: http://portal.opengeospatial.org/files/?artifact_id=5316.