Course title:	EOTIST Standard course
Course subject:	Computer Science
Teacher:	Joan Masó
Co-authors	Ester Prat, Lluís Pesquer

LESSON SC3
INTRODUCTION TO WEB SERVICES (OGC STANDARDS)

EOTIST project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 952111

[H2020 WIDESPREAD-05-2020 (Twinning)]

TABLE OF CONTENTS

INTRODUCTION TO GEOSPATIAL WEB SERVICES 2
BACKGROUND 2
THE CLIENT-SERVER ARCHITECTURE 3
COMMUNICATION PROTOCOLS 4
HTTP 5
WEB SERVER 6
WEB CLIENT 6
TECHNOLOGICAL EVOLUTION OF DISTRIBUTED GIS 6
PUBLICATION OF STATIC MAPS 7
STATIC WEB MAPS 9
DYNAMIC WEB MAPS 10
DISTRIBUTED GEOSERVICES 11
INTRODUCTION TO OGC MAIN WEB SERVICES 13
COMMON ARCHITECTURE FOR GEOSPATIALWEB SERVICES 14
BIBLIOGRAPHY 15

INTRODUCTION TO GEOSPATIAL WEB SERVICES

Geospatial web services are based on the communication networks that connect computers on the Internet. The communication network protocols are structured in layers. Top layers hide the incremental complexity of the heterogeneous hardware. Geospatial web services can be implemented using Hypertext Transfer Protocol (HTTP) to distribute geospatial resources (resource-oriented architecture style) or can be implemented as a new layer of protocol on top of HTTP (service-oriented architecture style). Geospatial web services separate the application into a client side that requests information and a server side that responds to that request, using a common communication protocol (that specifies the communication language and request and respond formats). These two sides might be developed using different programming languages and might be implemented by several vendors; however they are able to interoperate because they use a set of well-defined internationally agreed open standards.

BACKGROUND

Most of us use a mobile phone on a daily basis for tasks other than making phone calls. We know that most of the apps that we have installed in our phone require internet connectivity (so called "mobile data" or WiFi). These applications store a minimum set of data in our phone but also store other data in the cloud (in the network, somewhere). We have become accustomed to applications requiring external and distributed computers to work. Possibly, the oldest popular example of this kind of application is the email client. We know that each time we write and send an email, other computers will help in the task of routing it to the final email box and, at the same time, there is a remote computer that will later give us access to the incoming responses. The most popular internet application is still the web browser. In web browsers, a white page is filled with content coming from the outside (from remote web servers). When the web began, we got used to waiting a bit for the page to react to our actions and repopulate the white screen with content, and that was because we knew it was a "long-distance" communication. We also got used to the fragility of the remote computing that sometimes responded with mysterious numbers like 404 or 500, indicating some kind of failure. Web interfaces matured, and the web interaction was more fluid and the transitions between actions were smoother until a new kind of web page emerged: the web applications. They also run in web browsers but look and feel similar to desktop programs, except that they rarely save content onto our machine but into the cloud.

The idea behind the web services is getting the maximum simplicity and compatibility. Web browsers have undergone a major evolution in recent times (*Edge, Firefox, Opera, Safari, Chrome*...). We have moved from the browser war to the standardization of HTML5 which is improving interoperability, although it's also true that most browsers are now based on Chromium (the source code for the Chrome browser). Mobile devices are on a par with desktop computers, web browsers are becoming an omnipresent tool in mobile platforms, smart devices, and PCs as they diversify into operating systems. Therefore, there is a concern about the **simplicity** and use of **cross-cutting** and well-established technologies that can work on as many devices as possible.

Nevertheless, they have also clear limitations. Customer bandwidth will severely limit what can be done. Moderation in data transfer volume makes useful to split data into fragments and use data at different resolutions (scales). And for security reasons, the browser software (web client) CANNOT:

- save anything locally, it can only keep data in memory (exception: *cookies* and *localStorage*)
- communicate with other open applications on the same machine, it can only launch a file that cannot be read to a format *reader* (e.g. MiraMon Map Reader or Adobe Acrobat Reader)
- read from local disk (allowed in HTML5 only if user has selected the file)

- simultaneously interpret data from different servers unless explicitly authorized by the servers (CORS: *Cross-Origin Resource Sharing)*
- access the history of previous pages (can only be done by the user)

Focusing on the geospatial domain, *Google Maps* popularized the use of geospatial information by removing the difficulties in dealing with large geospatial files and simplifying location-based queries to remote geospatial databases. This transformation from monolithic applications to distributed computing was made possible by separating the different parts of the applications and distributing them in separated (far apart) computers in a computer architecture called the web services architecture [1].

THE CLIENT-SERVER ARCHITECTURE

A typical application usually includes three essential elements: presentation, logic, and data. The graphical user interface is the responsible element for presentation, the processing code is the responsible element for logic, and the database or the file management system is the responsible element for the data. The three elements are interconnected and, when needed, one element makes a request to another one, which fulfils the request and responds with the necessary information to the first.

The element making the request is called *the client*, and the element fulfilling the request is called *the server*. The user typically interacts with the graphical user interface (the client), and the client software is the responsible for communicating with the logic and the data (the server). Which physical machine contains the three elements depends on the architecture of the application.

- In a typical stand-alone GIS application, the three elements reside in the same computer, and you need to be a programmer to actually realize that this separation exists.
- In a typical GIS application accessing the data over a local area network, the presentation and the logic are on one machine, but the data is on another.
- In a web application, the presentation is on one machine, while the logic and the data are on another.

The last alternative is the one selected by the web services. In a client-server architecture, there is an exchange of messages between 2 different computers, which use 2 different software, and information is exchanged using an agreed protocol. The user only talks to the client and the client software has a graphical user interface (Figure 1).

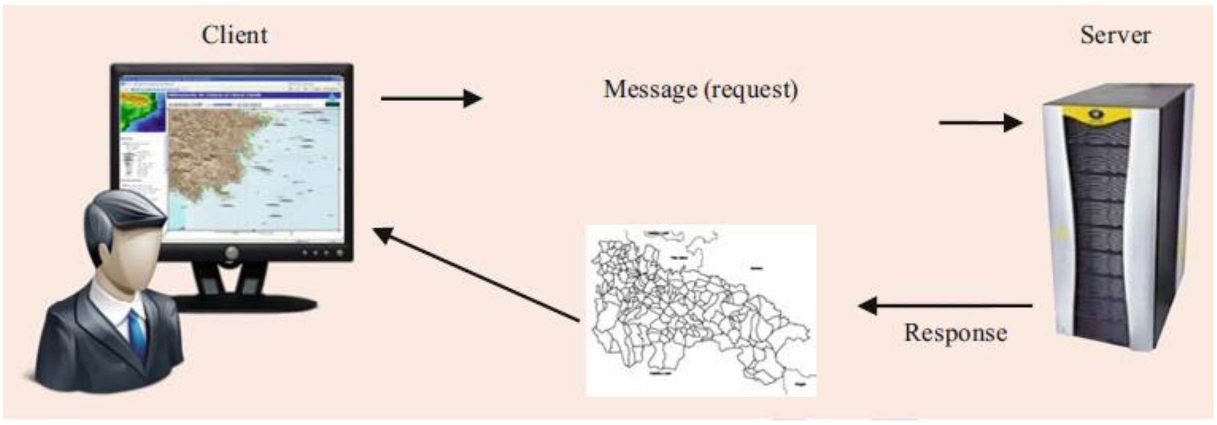


Figure 1. Client–server architecture and the protocol used.

But be careful not to get confused! A computer can be a client, a server, or both at the same time. A computer can act as a mediator between two: this is both a client of one and a server of another. This makes possible a **cascade** architecture: a server (also acting as a client) can query information that it does not have at that time, from other servers. An example in HTTP is the ***proxy***, *a* computer which all communications go through, and which acts as a smart router, storing the information and avoiding asking a remote computer for the same information twice (Figure 2).

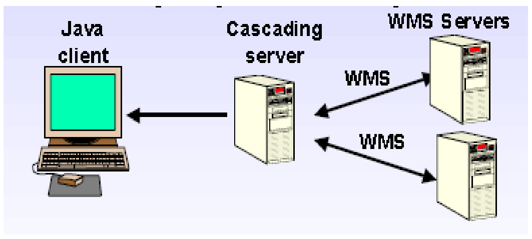
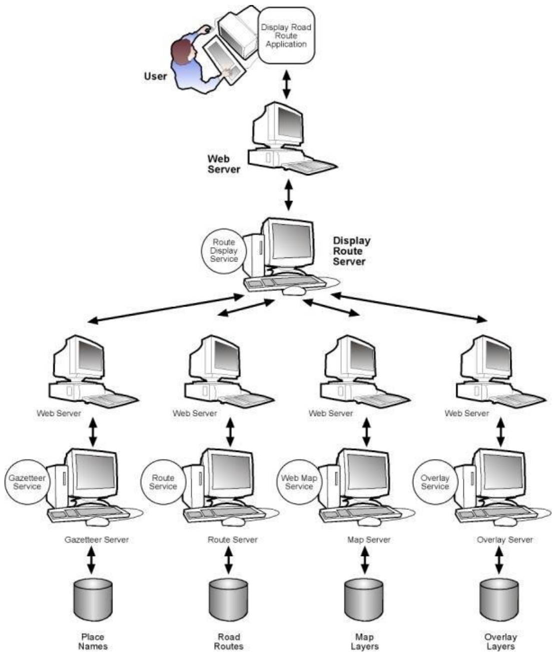


Figure 2. Example of a cascade architecture.



We have seen that the physical role the three elements depends on the architecture, but the logical roles can depend on the architecture as well, so the logical role of client and server is not fixed. For example, the logic element is the server from the presentation point of view, but it is the client from the data access point of view that acts as a server. Later we are going to see that the web client–server architecture is a special case of the general client–server architecture. In the web client–server architecture, the client and server communicate using a web protocol.

COMMUNICATION PROTOCOLS

Client and server are two pieces of software that need to communicate in a common agreed upon language and using a common set of rules (protocol) in such a way that they can understand each other. The role of the protocol is to connect the client and the server in a way that is transparent to users and that hides the complexity of the system.

Web services are developed over a stack of protocols organized in different levels, one on top of the other. To understand the need for a stack of protocols, an analogy can be used as follows. The bank industry had set up a communication protocol with their clients based on sending them some standardized letters where different expenditures and incomes are detailed. Clients regularly receive these messages, review them, and react accordingly. This protocol is part of a bank mail notifications protocol where commercial advertisements and other communications are performed by letters all having the bank brand logo very visible on the envelope. This protocol level uses yet another lower-level protocol of the postal correspondence. In the postal protocol, letters need to be in an envelope, with a destination postal address and a stamp. Envelopes are deposited in a post office, and they will be classified and travel a sequence of post offices until they are close enough to be delivered to the final address. This communication protocol relies on a heterogeneous very lower-level protocol: the communications network formed by cars, boats, and planes that follow a set of rules through their own media to move around people, goods and, of course, mail (including the bank mail). The bank industry is using the whole stack of protocols without even realizing the complexity of the system. However, dependencies exist, and if something goes wrong in a low-level protocol (e. g., a truck strike), the whole stack cannot function properly.

The web protocol is a high level protocol and, below that, other layers of protocol are defined one on top of the other (Figure 3). At the very bottom of the stack, the physical communication layer uses a mixture of protocols orchestrating communications among the physical devices and cables of the network, such as mobile phones, Ethernet cables, fiber optic cables, wireless connections, etc., all making internet communication possible. On top of the physical layer the transport protocol uses transmission control protocol/internet protocol (TCP/IP) as a network protocol. It is based on fragments of messages called packets that are transmitted through a sequence of computers that route each packet from the source to the target computer. Each computer has an IP logical address that univocally identifies it, and the IP address is used to route the packet from one computer to the next. On top of TCP/IP, the actual web protocol is defined as hypertext transfer protocol (HTTP). HTTP is an application protocol for hypermedia information systems. Hypertext is structured text that uses logical links (hyperlinks) between documents containing text. HTTP is the protocol designed to exchange or transfer hypertext.

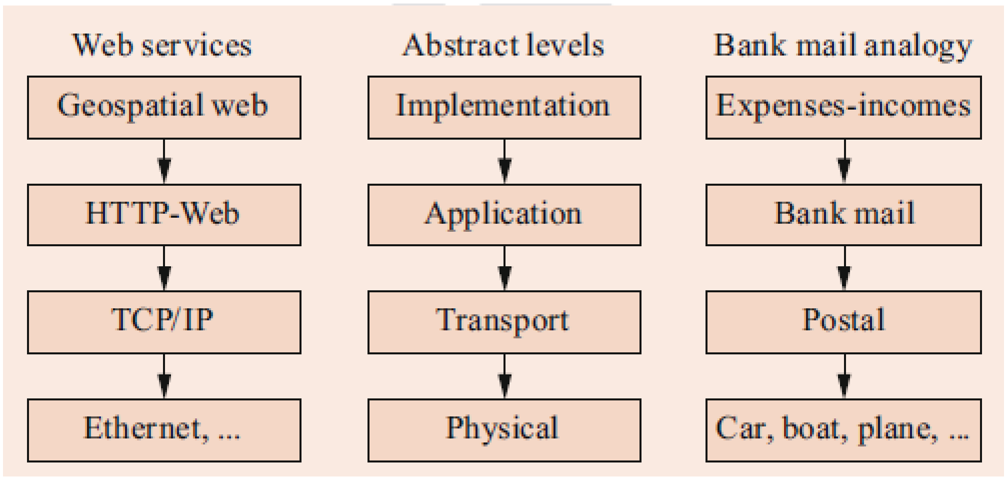


Figure 3. Levels of protocol

Upper levels of protocol depend on the lower levels; sometimes the separation between levels of protocol is not strict. We shall see *implementation protocols* that use (or misuse) some of the HTTP *application protocol* characteristics.

HTTP

HTTP is a request–response application protocol that is also based on the client–server architecture. The most used client application is the web browser, and a web server application running on a computer hosting a website is the most common server. The client submits an HTTP request message to the server and waits. The server returns a response message to the client, generally containing a single document, the most common being a hypertext markup language (HTML) file.

In HTTP, resources are identified and located on the network by uniform resource locators (URLs), using the uniform resource identifiers (URI's). URIs are the basis for requesting documents and are also the basis of the hyperlinks in HTML, acting as a way to link a fragment of an HTML page to another HTML file. A URL can represent pre-stored data or data that is generated dynamically; depending on the implementation on the server side. Commonly, resources correspond to pre-existing files but they can be the dynamic output of an executable application residing on the server side.

The HTTP protocol was designed to transport information in plain text mode. Due to the nature of TCP/IP, the computers in the local area network of the client and the server and all intermediate computers that route the communication have access to plain text and can easily read it or even store it. Confidential information transmitted in this mode can fall into the hands of unwanted people, including passwords, credit card numbers, etc. To prevent this a secured version of HTTP was introduced: HTTPS. In HTTPS, the URL and the message are encrypted and cannot be read by any computer but the receiver.

WEB SERVER

A web server is a software (and a hardware that executes it) that implements the server-side HTTP application protocol by processing incoming network request and respond documents. The primary function of a web server is to store, process, and deliver web pages to clients. Most common web pages are HTML documents and all related content, which may include images, style sheets, scripts (e. g., JavaScript), etc. Common examples of web server software are internet information services (*MicroSoft*), Apache server (*Apache*), Nginx (*Nginx Inc*.), etc. [2].

A web client, commonly a web browser, initiates a communication by making a request for a specific resource URL, and the server responds with the requested document or an error message (e. g., if the URL resource does not correspond to any content). The resource might be a real file on the server storage system, but it could also be generated on the fly based on the content of the server database. This is particularly useful for serving geospatial information that is usually stored in databases that can be dynamically converted into documents (e. g., a GML or a GeoJSON document) on the fly.

WEB CLIENT

A web client is a piece of software that implements the client-side HTTP application protocol by generating network requests and handling and displaying response documents. The primary function of a web client is to visualize web pages for users. The most common web client is the web browser, which is a generic application for accessing and visualizing many kinds of information available on the World Wide Web (WWW) on the user's device. It also has to react to human actions; the most common one being a click on a hyperlink. The web browser is able to visualize HTML pages, images, videos, etc. Modern web browsers are able to handle several kinds of information, apply styles to the HTML pages using style sheets, and execute client-side scripts (e. g., JavaScript). Common examples of web client software are *Internet Explorer, Microsoft Edge, Google Chrome, Mozilla Firefox, Safari, Opera*, etc.

Images embedded in HTML files were the first map applications shown in web browsers. Modern web browsers are complicated applications supporting several formats, but they can accept the inclusion of external supplementary code (e. g., with ActiveX). This method is used, for example, to add JPEG2000 imagery viewers.

In recent years, web browsers have evolved fast, but their performance cannot be compared to the desktop solutions. In addition, they have security restrictions that severely limit access to local resources (in particular to the local storage drive). To overcome those limitations, application programmers can implement or incorporate web clients in desktop solutions specifically tailored for the GIS data requirements that will be capable of combining local files on top of remote data.

TECHNOLOGICAL EVOLUTION OF DISTRIBUTED GIS

HTML was initially defined to support hypermedia. To make the presentation more visually appealing, GIF, JPEG, or PNG images were included. Soon, it became apparent that a server could be configured to automatically create maps by generating GIFs or JPEGs. In June 1993, the Xerox Corp. generated what is considered as the first interactive map viewer. Based on the selection of the user on a web page, the client software requested maps to a service: an application that was able to retrieve geospatial data from a database and render it at screen resolution and convert it into a GIF file that was returned to the client to be included as an illustration on a HTML page. Many other map browser solutions started to appear, such as the open source *MapServer*, and many proprietary web pages started to operate, such as *MapQuest*, all using the same principle. However, none of them could communicate with any of the other ones.

Technological evolution of distributed GIS has evolved from the publication of static maps, static web maps and interactive web maps to the actual distributed geoservices and cloud geoservices, and from low to high functionality and interactivity (Figure 4).

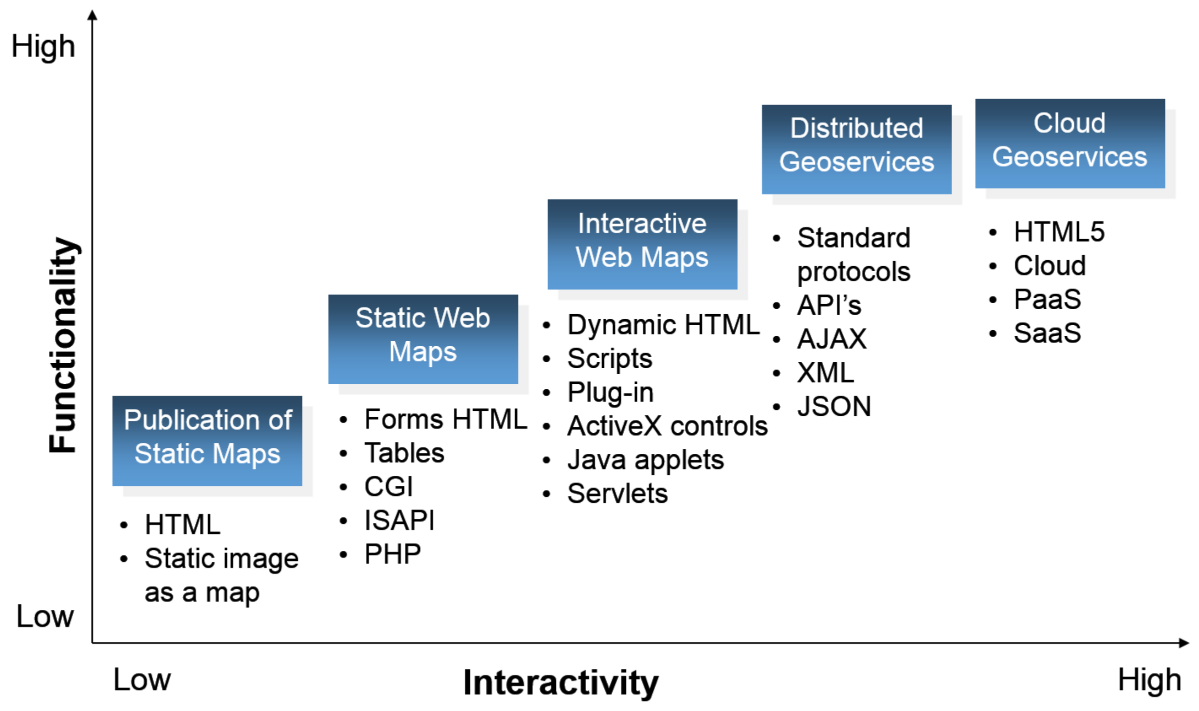


Figure 4. Technological evolution of distributed GIS. Modified from Peng, Z.R. and Tsou, M.H [2].

PUBLICATION OF STATIC MAPS

Consists on the use of GIF, PNG, or JPEG images within HTML pages to include maps. They can also use a link to a file in PDF format, but this cannot be integrated into the rest of the HTML page that links to it and must be viewed separately. The client has no information about the objects and only knows how to "paint" the map. Any web browser is suitable, even the most rudimentary versions. It is only used to present graphic information, as if it were a graphics terminal. Information is accessed only from URLs. The server does not need to do anything about maps, so any web server is adequate, as long as it only offers HTML pages (or pdf files).

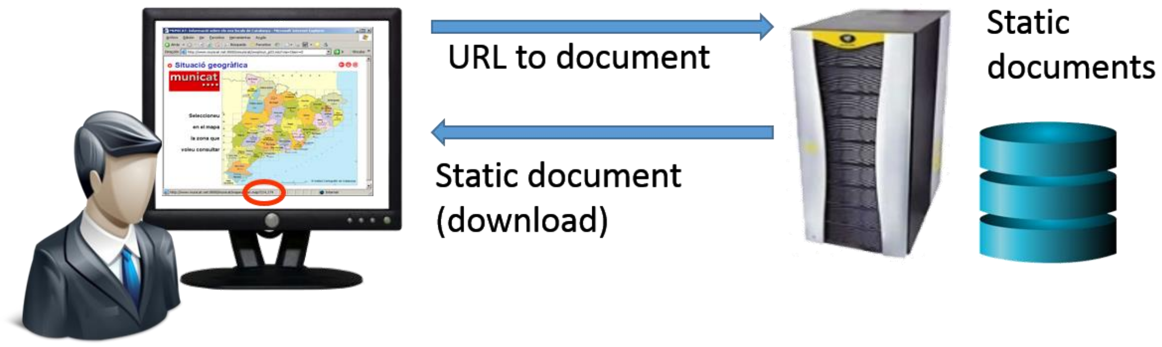
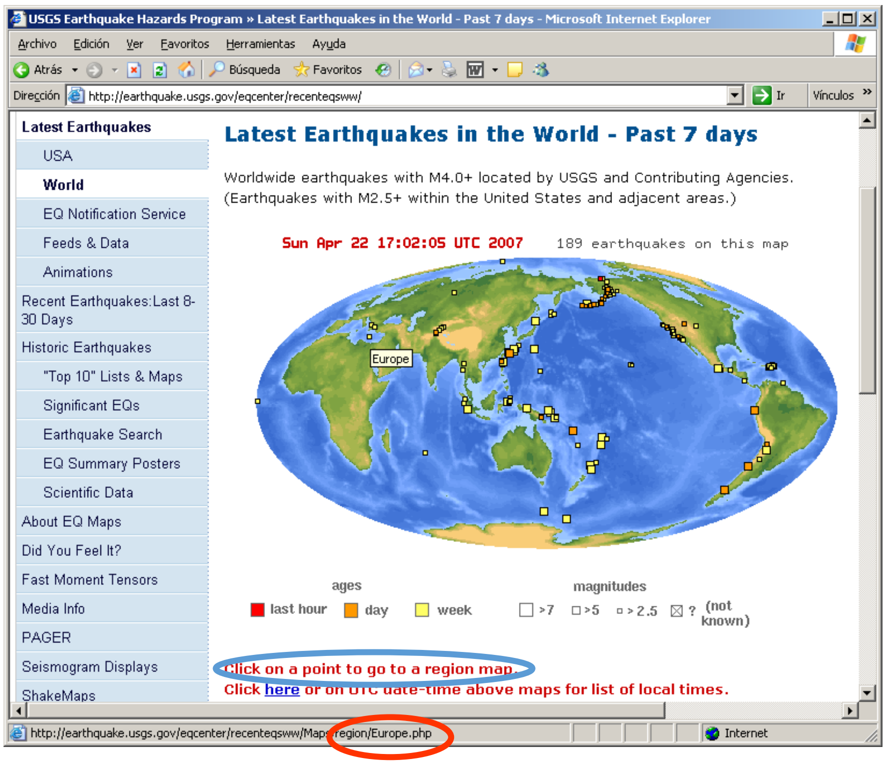


Figure 5. Publication of static maps architecture.

There are two possibilities to offer the map: as a simple link to an image, where the map is just another illustration without being able to click or zoom or anything (in this case the map can be produced from GIS software, saving the map as JPEG), or defining an image as sensitive by zones.



Latest Earthquakes in the World - Past 7 days

Worldwide earthquakes with M4.0+ located by USGS and Contributing Agencies.
(Earthquakes with M2.5+ within the United States and adjacent areas.)

Sun Apr 22 17:02:05 UTC 2007 189 earthquakes on this map

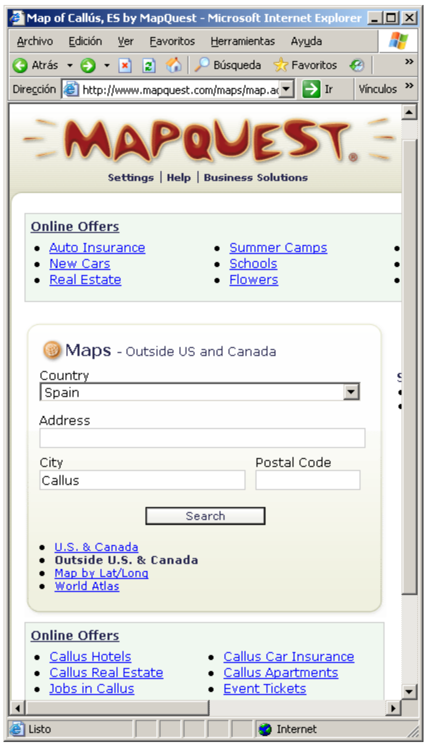
Click on a point to go to a region map.
Click here or on UTC date-time above maps for list of local times.

http://earthquake.usgs.gov/eqcenter/recenteqsww/Maps/region/Europe.php

Figure 6. Example of the publication of a static map.

STATIC WEB MAPS

It consists on the generation of on-demand maps. The final map is not consultable or interactive. In this case, the server has a web mapping service that coordinates with the web server (CGI, PHP, ASAPI, FastCGI, JSP). Technically, few change are required on the client. Forms appear where the user fills in a number of variables and chooses what the user wants to see (Figure 7).


Map of Callús, ES by MapQuest - Microsoft Internet Explorer

Archivo Edición Ver Favoritos Herramientas Ayuda

Dirección http://www.mapquest.com/maps/map.a

MAPQUEST

Settings | Help | Business Solutions

Online Offers
- Auto Insurance
- New Cars
- Real Estate
- Summer Camps
- Schools
- Flowers

Maps - Outside US and Canada

Country: Spain

Address:

City: Callus

Postal Code:

Search

- U.S. & Canada
- Outside U.S. & Canada
- Map by Lat/Long
- World Atlas

Online Offers
- Callus Hotels
- Callus Real Estate
- Jobs in Callus
- Callus Car Insurance
- Callus Apartments
- Event Tickets

Listo — Internet

Figure 7. Example of a client form.

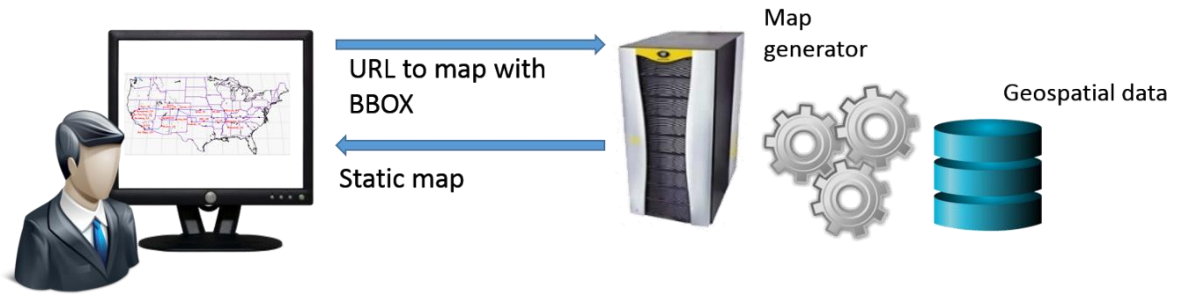


Figure 8. Static Web Maps architecture.

The first one that existed was the mentioned XEROX Map Viewer (June 1993), that was created by Steve Putz at Xerox Corporation's Palo Alto Research Center, as an experiment to serve interactive content via the World Wide Web. It was a CGI written with PERL (a scripting language). It was composed of 2 programs, one that took the vectors and rasterized them and another that converted them to GIF format. It is currently "offline“.

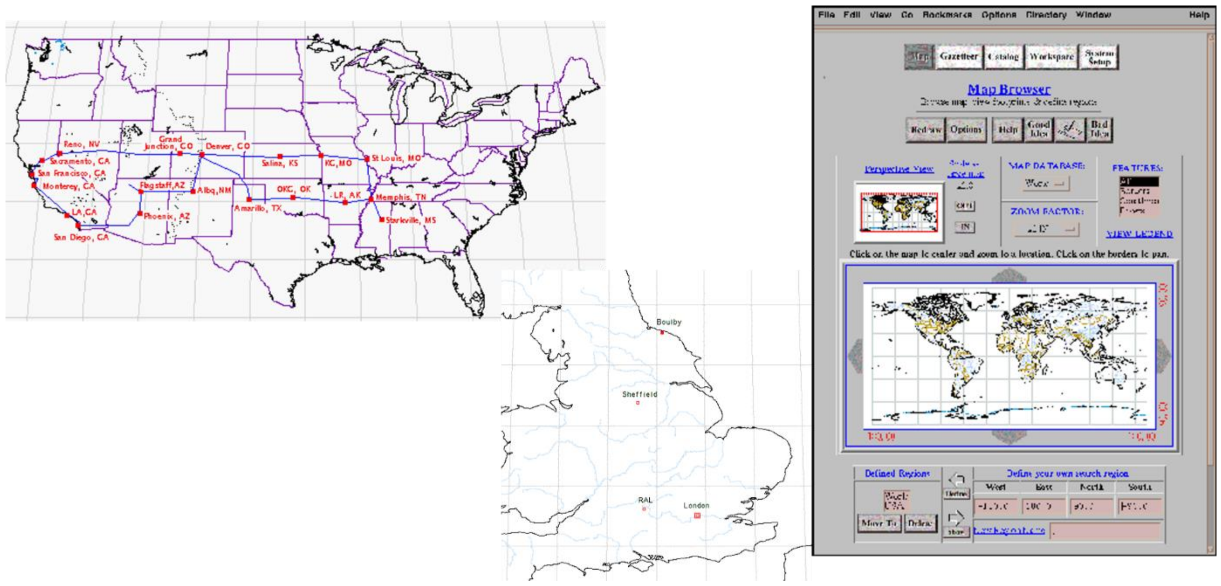


Figure 9. The Xerox Map Viewer.

DYNAMIC WEB MAPS

It also consists on the generation of on-demand maps but the final map is consultable and interactive: the map page is full of tools that allow the user to do more things. In this case the client needs a DHTML: set of techniques to allow user interaction without refreshing the page, such as Javascript or VBScript, DOM (Document Object Model), CSS (Cascading Style Sheets), Plug-ins, ActiveX, Java applets. The use of HTML5 enhances with more functionality and user can change the area (pan), zoom in, consult ... Technically few changes are required in the server side, except improvements to the CGI concept (use of FastCGI, Servlets, ASAPI, ASP, ColdFusion).

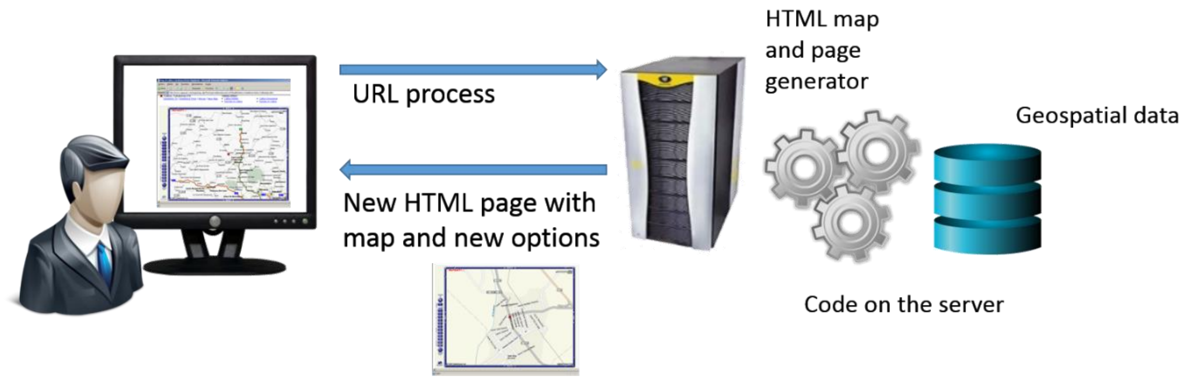

URL process
New HTML page with map and new options
HTML map and page generator
Geospatial data
Code on the server

Figure 10. Dynamic Web Maps architecture.

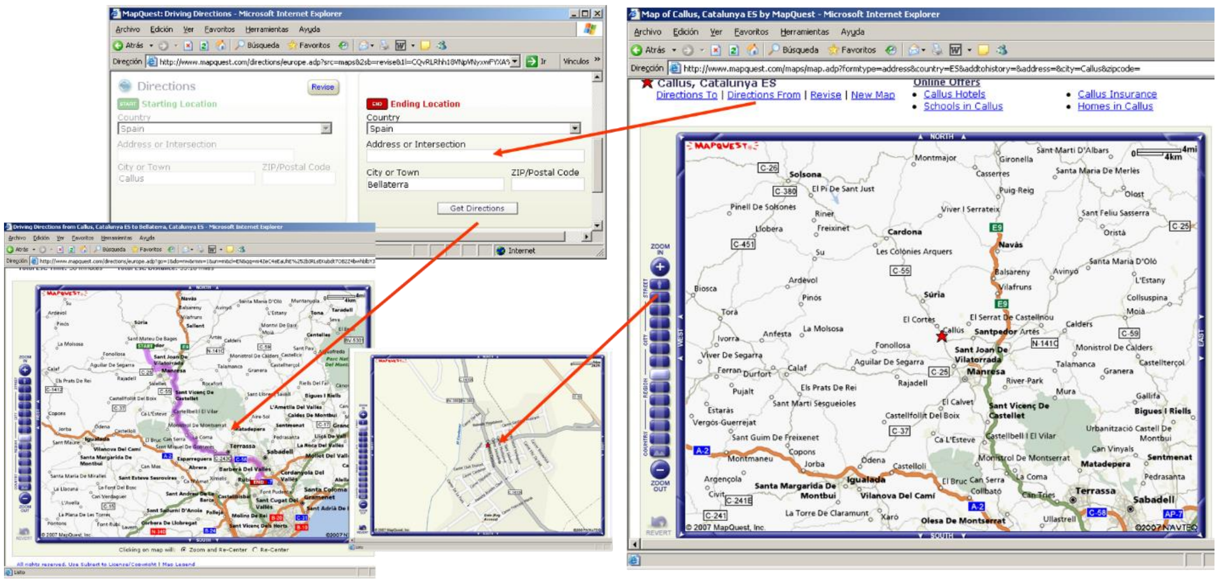

MapQuest: Driving Directions - Microsoft Internet Explorer
Directions
Starting Location
Country
Spain
City or Town
Callus
Ending Location
Spain
City or Town
Bellaterra
Get Directions
Map of Callus, Catalunya ES by MapQuest - Microsoft Internet Explorer
Callus, Catalunya ES
Directions To | Directions From | Revise | New Map

Figure 11. Interactive map page.

DISTRIBUTED GEOSERVICES

In distributed geoservices, data is managed by services or operations that are spread across different servers and that can be exchanged. Elements are individualized, they may reside on separate servers (or nodes) and can be searched. Each operation is independent: "map", "buffer", "point in a polygon" etc. Processes can be chained and they follow interoperable protocols that follow standards. Web browsers include asynchronous requests (AJAX) and HTML5 to make this possible. They are more and more present. There is a great deal of effort in this direction by setting up: technologies and standards (OGC WxS), databases (data catalogs and services), as well as laws and directives (INSPIRE).

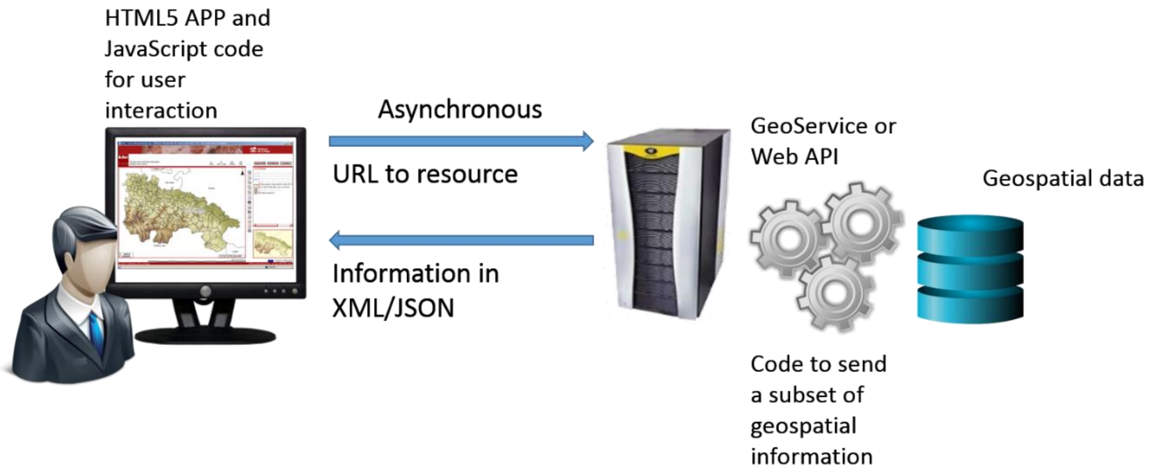

HTML5 APP and JavaScript code for user interaction
Asynchronous
URL to resource
Information in XML/JSON
GeoService or Web API
Geospatial data
Code to send a subset of geospatial information

Figure 12. Distributed geoservices architecture.

The paradigms for distributed geoservices include better targeting and less redundancy of effort, the removal of barriers imposed by manufacturers due to the independence from systems and manufacturers, cheapness, the improving of the quality and performance of products and services, and the room left for small entrepreneurs to focus on a specific problem. On the other hand, they also imply some problems, such as that the architecture relies on external services that may change or disappear and that process speed or data download between processes can be slow and depends on many factors over which we have no control.

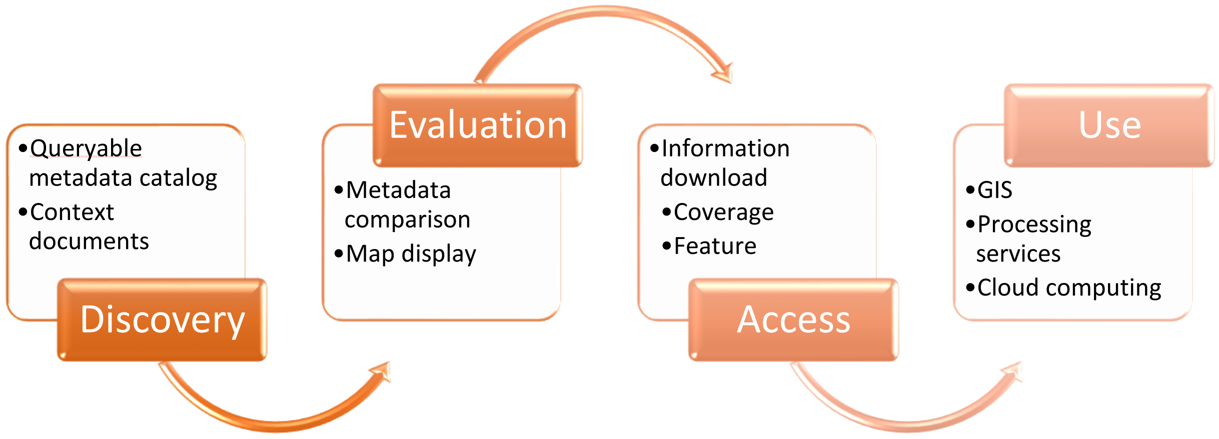


Figure 13. Phases in the use of distributed data.

LIGHT CLIENTS AND HEAVY CLIENTS

Two alternatives exist in the use of distributed data:

- Applications running entirely in web browsers: a client will connect to a single server which will open the door to the entire distributed scheme and channel requests to it. Requires an improved client compared to current but emerging technologies like AJAX and HTML5 can help.
- Applications outside of the browser that must be installed on each machine: fully accepted on mobile phones (apps). Not widely accepted on PCs, and users only do so if a benefit is clearly offered in return. Ex: *Google Earth*, P2P (*Skype, Emule, BitTorrent*).

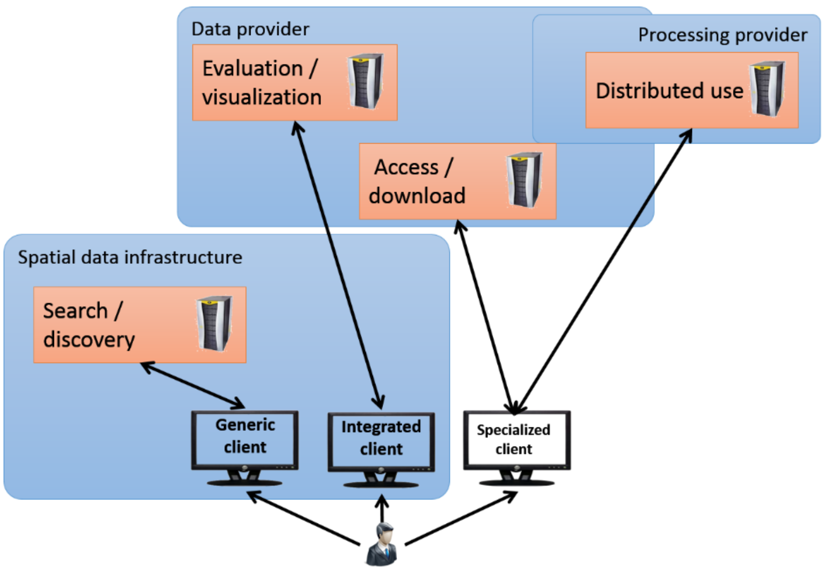


Figure 14. Usual architecture in distributed data.

INTRODUCTION TO OGC MAIN WEB SERVICES

In 1997, the Open GIS Consortium (later renamed as Open Geospatial Consortium; OGC) proposed the OGC WWW mapping framework with the mission to investigate how to make similar mapping approaches implemented by different vendors interoperable. In 2000, the OGC proposed the version 1.0.0 of the web map service (WMS). WMS has been a huge success, and today there are thousands of standard web map services in the world produced by governments, academia, and the commercial sector, and hundreds of web map browsers [3].

The OGC standardized other services that exchange other kinds of geospatial information and adopted XML and XML schema as the lingua franca to encode many media types, for example service metadata, feature-based data (GML), observations-based data (O&M), etc. None of the other OGC services has become so popular or has had the same impact as WMS, which may be due to the fundamental limitations in the early versions of HTML.

Geospatial data is constantly being created by cartographic agencies, automatic in-situ sensors, remote sensing satellites, etc. In many countries, it is in the aim of a transparent administration to share and exchange data with citizens, who can then be informed and formulate opinions. In addition, governments have recognized that by making data public, private companies can discover new usages that go beyond the ones initially foreseen by their producers and generate new business models and profits by selling new products and services. In this chapter, we shall review current geospatial web services that facilitate the dissemination of geospatial data over the web [4].

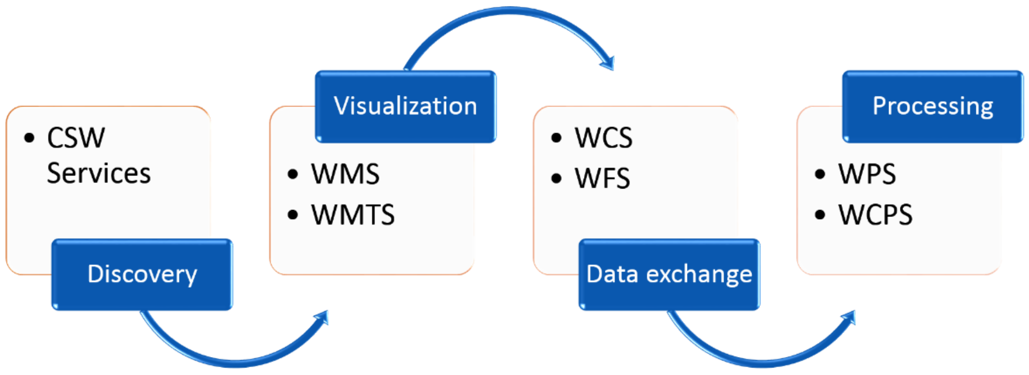
Most of these geospatial services have been defined by the OGC community, based on a consensus process that favours interoperable geoprocessing and provides services to communicate to remote geodatabases. In principle, OGC standards are defined for multiple distributed computing platforms while maintaining common geospatial semantics across the underlying technology.

One might expect that a single big web service would be able to deal with all geospatial aspects by incorporating all the necessary operations, in the same way that a desktop GIS in your computer is able to perform most of the necessary operations. There have been some successful attempts to do this in the commercial sector (such as the ESRI ArcGIS REST API), however, this is not a common situation. Standard geospatial web services deal with a single aspect and a single data model. If we want to make sense of this variety of alternative services, we need to classify them. The OGC reference model proposes a way of classifying geospatial web services by purpose and the later by the data model used [5]:

- **Visualization services** have the purpose of presenting the data in an easy to understand way. Clients are able to request formats that can be easily placed on a computer screen by client software, such as JPEG, PNG, or KML. Servers will be able to respond with visualization formats on demand. To be able to do this, visualization services need to combine geospatial data with symbolization rules to create a presentation for the geospatial data.
- **Data exchange services** have the purpose of giving immediate access to data and, in some cases, permitting the upload of new geospatial data to the server. Clients should be able to request geospatial data exchange formats such as GML, GeoJSON, GeoTIFF, or NetCDF. Servers need to respond with the subset of data (as requested), and clients will be able to import the data in their GIS data storage systems. In some cases, clients should also be able to send geospatial features to the server that should store them for further use.
- **Data scheduling and notification services** have the purpose of preparing the future acquisition and notifying when it becomes available in the data exchange services. The client should be able to specify

the use needs in terms of extent, time, data type, etc. The server will schedule the acquisition and issue a notification when new data becomes available.

- **Discovery services** have the purpose of storing metadata about the data and services available on the web. Clients will formulate a request describing the type of data they need in terms of extent, projection, resolution, data quality, theme, etc. Servers should be able to analyse the query and respond with a set of metadata records describing the data or services that match the needs of the client.
- **Processing services** offer geospatial processing routines and hardware. Clients should be able to formulate a request that includes the processing operation, the data to be processed and the parameters to be used to tune the process execution. Servers should be able to perform the requested task and respond with the result of the operation, which will commonly be in the form of a geospatial dataset.



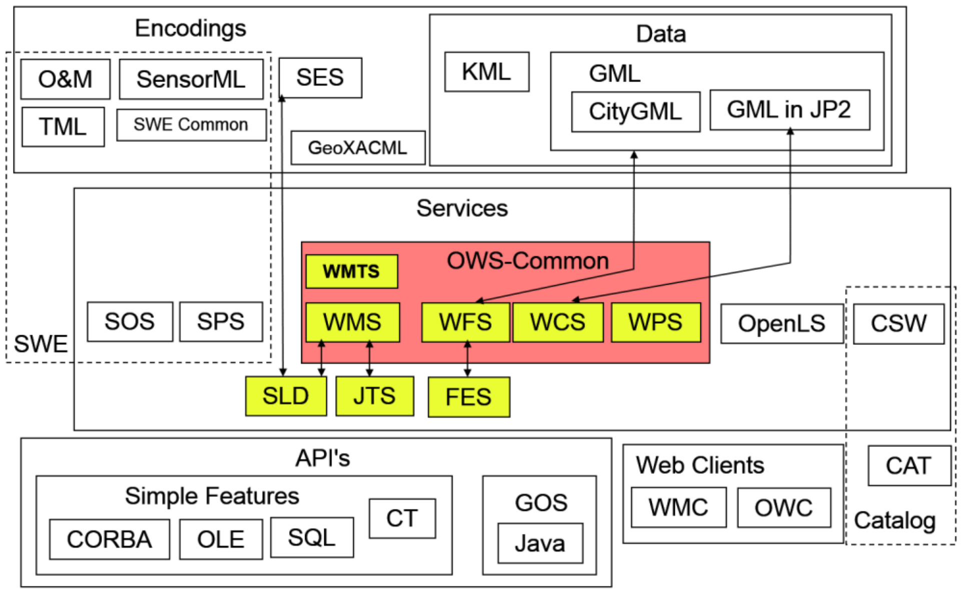


Figure 15. OGC Standards scheme.

COMMON ARCHITECTURE FOR GEOSPATIALWEB SERVICES

As discussed before, the geospatial web services are separated into different types that follow standards developed by the consensus of different groups. Fortunately, there are mainly two common functionalities that all services implement: a common mechanism for formulating requests and a service self-description in the form of service metadata (the *GetCapabilities* operation). These and other minor common characteristics are defined in the OGCWeb Service Common standard [6].

A self-describing service provides enough information for a client to discover the resources available and the capacities of the service without user intervention. In OWS Common, this is done by specifying that all services will provide a service metadata document. The OGC uses a simplified version of the ISO 19115-1 metadata for services. A service metadata document should contain the following information:

- Identification: contains metadata to identify this server in terms of standards and profiles followed, keywords describing the general content, and fees and access constraints.
- Provider: contains information about the organization operating this service.

Recently, a new format to describe a web service API emerged: OpenAPI (formerly known as Swagger). OpenAPI is a format that describes the resource types in web services by providing URL templates, the definition of all variables in a URL template, and the format and schema of the expected resource representation. WFS 3.0 is the first OGC standard to incorporate this metadata format, and it is foreseen that others will follow. OpenAPI documents are written in YAML and will be discussed in the Advanced course.

BIBLIOGRAPHY

[1] Champion, M., Ferris, C., Newcomer, E., Orchard, D. (eds.): Web Services Architecture. W3C, Cambridge (2002). W3C Working Draft 14. November 2002, http://www.w3.org/TR/ws-arch/

[2] Peng, Z.R., Tsou, M.H.: Internet GIS: Distributed Geographic Information Services for the Internet and Wireless Network. Wiley, New York, p 720 (2003)

[3] Döllner, J., Jobst, M., Schmitz, P.: Service Oriented Mapping. Lecture Notes in Geoinformation and Cartography. Springer Nature, Cham (2019)

[4] Deng, M., Di, L.: Facilitating Data-Intensive Research and Education in Earth Science: A Geospatial Web Service Approach. Lambert Academic, Saarbrücken, p 212 (2010)

[5] OGC: OGC Reference Model (ORM). OGC, Wayland (2011). OGC 08-062r7, http://www.opengeospatial.org/standards/orm, http://portal.opengeospatial.org/files/?artifact_id=31112

[6] Whiteside, A., Greenwood, J.: OpenGIS Web Service Common Standard 2.0. OGC, Wayland (2010). http://www.opengeospatial. org/standards/is