Peer-reviewed version available at Geosciences 2018, 8, 139; doi:10.3390/geosciences8040139

Article

A Javascript GIS platform based on invocable geospatial Web services

Konstantinos Evangelidis[1,*] and Theofilos Papadopoulos[2]

[1] Technological Educational Institute of Central Macedonia; kevan@teicm.gr
[2] Technological Educational Institute of Central Macedonia; priestont@gmail.com
* Correspondence: kevan70@gmail.com; Tel.: +30-694-727-8769

Abstract: Semantic Web technologies are being increasingly adopted by the geospatial community during last decade through the utilization of open standards for expressing and serving geospatial data. This was also dramatically assisted by an ever increasing access and usage of geographic mapping and location-based services via smart devices in people's daily activities. In this paper we explore the developmental framework of a pure Javascript client-side GIS platform exclusively based on invocable geospatial Web services. We also extend Javascript utilization on the server side by deploying a node server acting as a bridge between open source WPS libraries and popular geoprocessing engines. The vehicle for such an exploration is a cross platform Web browser capable of interpreting Javascript commands to achieve interaction with geospatial providers. The tool is a generic Web interface providing capabilities of acquiring spatial datasets, composing layouts and applying geospatial processes. In an ideal form the end-user will have to identify those services, which satisfy a geo-related need and put them in the appropriate row. The final output may act as a potential collector of freely available geospatial web services. Its server-side components may exploit geospatial processing suppliers composing that way a light-weight fully transparent open Web GIS platform.

Keywords: Open GIS; geospatial Web services; geospatial Web semantics; Web GIS; Node.js; Javascript

1. Introduction

Geospatial functions range from a simple image map acquisition to a complex geoprocess over a Spatial Data Infrastructure (SDI). Nowadays, a wide range of users exploit geospatial functions in their routine activities. Such users are practitioners, scientists and researchers involved in geosciences and engineering disciplines, as well as individuals employing Geographic Information Systems (GIS) [1-2]. In addition, today we face an ever increasing access and usage of geographic mapping and location-based services via smart devices in people's daily activities [3]. For this reason, emerging computing paradigms show high penetration rates in geospatial developments, with the latest and yet most significant one the Cloud computing [4-5]. As a result, existing systems are transformed from proprietary desktop GIS software applications of the early 80's to free and open source interoperable Cloud GIS solutions built upon geospatial Web services (GWS) [6].

GWSs and service-oriented architecture (SOA) are the key components to achieve interoperability in Web GIS applications. GWSs allow self-contained geospatial functions to operate over the Web while SOA facilitates interoperability between these GWSs by establishing communication and data exchange for requesters and providers in a uniform way [7-8]. The dominant GWS standards adopted by the geospatial community are those introduced by the Open Geospatial Consortium (OGC) including the Web map service (WMS) to visualize [9], the Web feature service (WFS) and the Web coverage service (WCS) to acquire [10-11], the catalog service for the Web (CSW) to discover [12], and also the emerging Web processing service (WPS) to process, spatial data [13].

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 January 2018 doi:10.20944/preprints201801.0268.v1

Peer-reviewed version available at Geosciences 2018, 8, 139; doi:10.3390/geosciences8040139

In this respect, numerous research projects and business solutions rely on the above standards to achieve geospatial data interoperability between custom applications and to satisfy project-specific needs [14-15]. Furthermore, in European Union (EU) level, project actions have to be aligned with regulation No 1312/2014 [16], implementing INSPIRE directive [17] as regards interoperability of spatial data services. According to this, all geospatial data have to be served under *invocable* spatial data services. As a result most applications are nowadays based on Web services, use data provided over the Web or generated by users [18], and are executed on cross-platform browser-based interfaces. In the geospatial community, GWSs and XML-based open geospatial data formats, such as Geography MarkUp Language (GML), have become basic components of desktop and Web GIS software solutions. For example the ESRI's ArcGIS commercial product supports WMS connections through its popular 'Add data' interface [19]. On the other side QGIS open solution also supports connection to GWSs through appropriate plug-ins [20]. For individual Web-based applications it is possible to develop a custom GIS capability through open Javascript libraries such as for example Openlayers (http://openlayers.org/) and GeoExt (https://geoext.github.io/geoext2), and have it executed on the client-side without the need of installing anything but an updated Web browser.

The development of research and commercial projects that utilize open or proprietary Web services and spatial application frameworks is rapidly growing. [21-28]. Several other Cloud GIS solutions are served as software, as platforms, as infrastructure under the popular service models, SaaS, PaaS and IaaS respectively [4]. However an exclusively service-based application composed of open interoperable Web services could be the ideal case. The developer would have to identify the appropriate GWSs and bind them between each-other in the correct order, same way as it happens in the well-known “ArcGIS model builder” [29]. The final outcome would be a transparent to the user Web interface consisting of an interconnected set of Web services. This case may be extended to a Web GIS platform that gathers available GWSs and acts as a platform for building GIS projects.

In this paper we explore the developmental framework for exploiting invocable GWSs, that satisfy routine geospatial needs. A comprehensive and sophisticated implementation might include a Web interface allowing the end user to select between task descriptions composing a GIS project. We demonstrate (212.111.41.209/res/gws) such an implementation which is exclusively based on open standards and services, a light-weight client-side pure JavaScript platform that performs: a) data discovery from public data providers, b) layer-based data view, c) data selection by attributes, d) feature data acquisition and preview, and e) simple geoprocessing tasks. For the last ones, we also explore the applicability of JavaScript, for implementing geoprocesses. Prior to this, the paper explores the effects of semantic Web technologies on fundamental geospatial elements, and discusses critical architectural and development issues.

2. The influence of geospatial Web semantics on GIS

The major components and principal operations and characteristics of an interface implemented according to geospatial Web semantics technologies, are identified and reviewed throughout GIS timeline from desktop and proprietary Web applications to open service-based GIS systems in the Cloud. The historical point that generally represents the geospatial evolution is when Web semantics technology standards were adopted by the geospatial community. The three major areas briefly discussed in the following are a) Formats, b) Interoperability and c) Automations

2.1. Geospatial Data Formats

2.1.1. Vector Data

Vector data are considered the dominant component of a GIS System, holding the critical properties of the spatial entities that they represent such as their shape and spatial representation and topology. Traditionally, vector data were handled by geographers and GIS experts as the valuable form of spatial data, beyond others, for two reasons: their independence from scale and the capability of associating on them, unlimited amount of descriptive information. In addition vector

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 January 2018 doi:10.20944/preprints201801.0268.v1

Peer-reviewed version available at Geosciences 2018, 8, 139; doi:10.3390/geosciences8040139

data production is expensive and time consuming since they are obtained by digitizing map images or as a result of GPS field data collection.

Various forms of vector data were adopted throughout GIS timeline from coverage and shapefile to proprietary and open geographic database formats. Today spatial coordinates of the vertices composing a vector graphic may be easily modeled through XML-based open formats (KML, GML, SVG) and transferred through OGC-WFS service requests.

2.1.2. Raster Data

Traditionally, raster data in the form of scanned maps (gif, jpeg, tiff etc.) were used as the base for producing vector data through digitization tasks. Therefore, the more detailed and of high resolution, a raster was the more analytical and precise was the digitization process. As a result, raster data were usually heavy-sized and their management in a desktop GIS environment required high efficiency computer hardware resources. Servicing maps and satellite images through static Web pages or through raster data repositories were also tasks dependent to hardware efficiency including internet infrastructures.

When the first map servers appeared, raster data were being served over the Web as textures of the ground surface, mainly satisfying navigation experience in earth browsers. Today image compression and tiled rendering techniques along with extremely high wireless internet connections make it possible to employ high quality raster data as the background for location-based services provided to smart device users. Raster data used as cartographic background are transferred through OGC-WMS service requests. Other raster formats like GeoTIFF that are used for coverage purposes (e.g. elevation or results from geoprocessing) are served via OGC WCS standard.

2.1.3. Descriptive Data

A fundamental structural characteristic of a GIS is the capability of associating the spatial features with descriptive data related to them. That way it is possible to perform sophisticated cartographic representations for decision and policy makers as well as to execute complex processes over descriptive data and produce valuable geoinformation. Descriptive data were normally easy to manage throughout GIS timeline because of the simultaneous emergence of database technologies. The external data sources to be associated with spatial features included a wide range of alternatives from simple comma separated values and single database files to relational geographic databases installed in remote servers.

Today the Web of Data and associated semantic technologies, support interoperability and standard formats to model and transfer descriptive data. ISO 191xx series and RDF are XML encoded data standards employed in the geospatial web [30].

2.2. Geospatial Interoperability

Geospatial interoperability became an issue, when the need for data communication and exchange between diverse geospatial stakeholders became a necessity. Till early '90s, GIS vendors used their own proprietary formats, however they agreed to common standards and formats and they established connections to commonly shared repositories. As the technologies that developed by World Wide Web Consortium (W3C) matured, OGC introduced appropriate spatial related technologies to achieve syntactical and semantic interoperability.

2.2.1. Syntactical Interoperability

Syntactic interoperability assures data transfer between connected systems through Web services. In the geospatial community it is currently achieved through OGC Web Services. For example WFS/GetFeatures request, provides the standard interface and message types for Web services transferring features through XML. In the past, syntactical interoperability could be considered as the result of applying SQL commands through ODBC connectivity.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 January 2018 doi:10.20944/preprints201801.0268.v1

Peer-reviewed version available at Geosciences 2018, 8, 139; doi:10.3390/geosciences8040139

2.2.2. Semantic Interoperability

Semantic interoperability is the ideal situation where the exchanged content is machine understandable. To be such it has to be conceptualized formally and explicitly through appropriate specifications, such as GML, the standard for the exchange of service-based spatial data. Traditionally, semantic interoperability could be only achieved via pre-constructed data formats resulting from predefined domain specific data models (e.g. ArcFM [31], UML data models).

2.3. Geospatial Automations

A GIS project is usually a composition of single geospatial activities which normally begin with the acquisition of thematic layers, and other data involved and the application of geospatial processes, depending on the exact domain of the geoscientific field of expertise. Automating these activities under a workflow of sequentially executed processes may be achieved by creating specialized batch files, or scripts. Traditionally, geospatial automations are implemented through sophisticated modules of the popular desktop GIS environments offering tools to manage geospatial processes, like for example ModelBuilder [31], or Processing Modeler [32].

Now that all types of geospatial activities may be served through geospatial Web services, automation is achieved by 'orchestrating' these Web services. Orchestration *"describes collaboration of the Web services in predefined patterns based on local decision about their interactions with one another at the message/execution level"* [33]. OGC WPS can be designed to call a sequence of web services [13].

Table 1 collects all related terminology in the above specified sections before and after Geospatial Web Semantics influence.

Table 1. Impact of Web semantics on geospatial technologies

	Past	*Today*
	Geospatial Data Structures	
Vector data	Binary files (Shapefiles, coverages etc.), proprietary database formats (e.g. ESRI geodatabase)	Text files in XML-based formats (GML, SVG, KML)
Raster data	Image files (Raster)	Image files (Raster)
Descriptive data	Text files, proprietary database formats	Text files in XML-based formats (ISO 191xx, RDF etc.)
	Geospatial Interoperability	
Syntactic	Common data formats, ODBC connections to spatial databases	OGC Web Services
Semantic	Common data models (e.g. UML data models)	OWL, GML, RDF
	Geospatial Automations	
Workflows	Batch files and scripts Special model builders and process modelers	Web service orchestration (OGC WPS)

3. Software Prototype Design & Development

3.1. Functional Architecture

The successful operation of an application based on GWSs prerequisites the existence of available open geospatial Web services for data acquisition and data processing purposes. The end user interface should support access to the services via a Web browser, without the need of installing

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 January 2018 doi:10.20944/preprints201801.0268.v1

Peer-reviewed version available at Geosciences 2018, 8, 139; doi:10.3390/geosciences8040139

additional software. Figure 1 represents graphically the functional architecture of such an implementation, which includes:

- Free WMS and WFS geospatial services provided either by open-source (e.g. Boundless) or commercial (e.g. ESRI) GIS product leaders, satisfy the need of obtaining features and images
- Accessible processing platforms like 52º North initiative, or Javascript node servers developed to support custom WPS implementations.
- an HTML browser-based interface developed in Javascript, undertakes to serve user needs over a functional GIS-based environment as described below

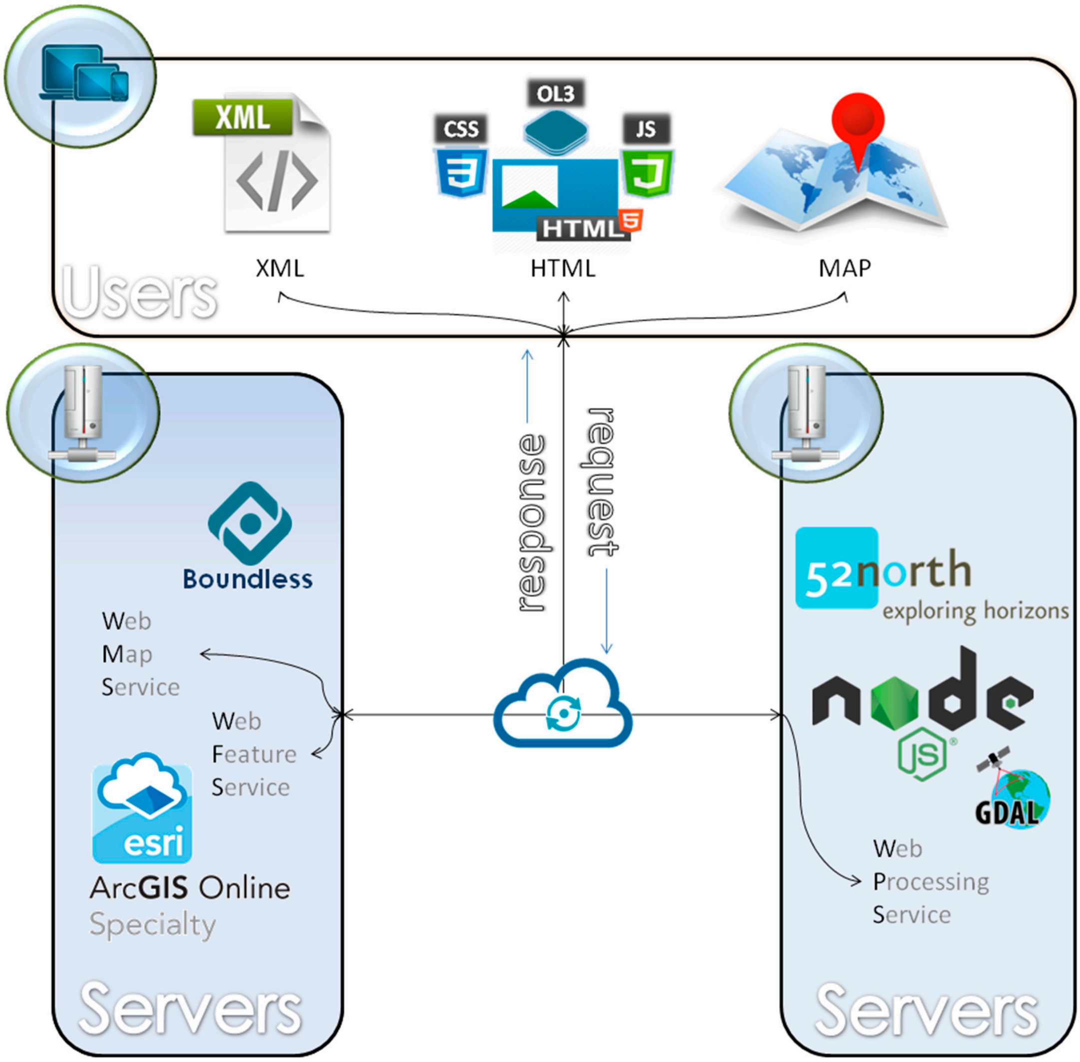

Figure 1: Functional architecture of a system exploiting GWSs

3.2. Development Issues

3.2.1. Raster and Vector Layer Views

To get raster and vector layers, WMS and WFS services, respectively, are employed. The user interacts with the following ways:

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 January 2018 doi:10.20944/preprints201801.0268.v1

Peer-reviewed version available at *Geosciences* **2018**, *8*, 139; doi:10.3390/geosciences8040139

- Requesting for available maps in the form of raster or image views of vectors through a WMS/GetCapabilities request and receiving a list with the offered layers along with further metadata descriptions in XML format
- Requesting for available features through a WFS/GetCapabilities request and receiving a list with the offered feature layers along with further metadata descriptions in XML format
- Requesting for a specific raster (or image views of a vector) layer through a WMS/GetMap request and receiving an image file
- Requesting for a specific vector layer through a WFS/GetFeatures request and receiving an XML file

Figure 2, illustrates an example of a WMS/GetCapabilities request coded in Javascript along with the server XML response:

- the client makes an AJAX (Asynchronous JavaScript and XML) request using the XMLHttpRequest, either WMS or WFS with a URI parameter 'request=GetCapabilities'.
- the server responds with XML data that will thereafter be parsed to JSON object and finally be viewed by the user as paged table data.

Practically, the above interaction takes place, whenever the user declares a potential service provider and checks geospatial data provision.

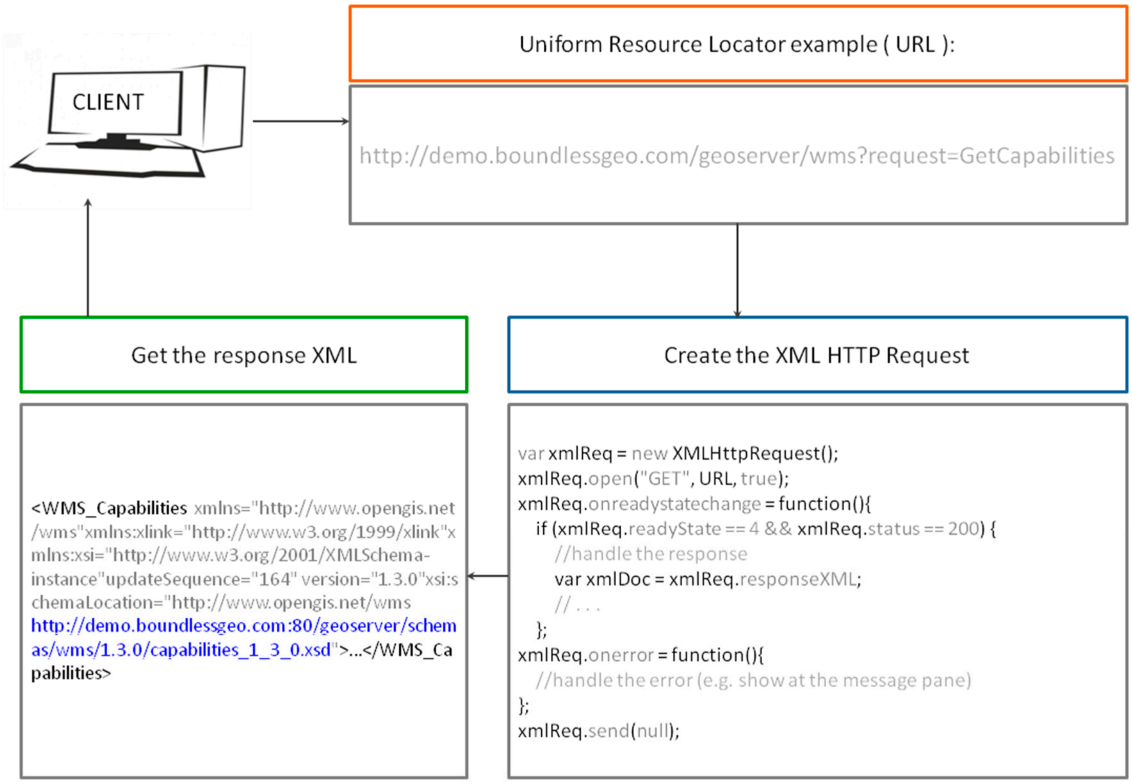

Figure 2: Requesting a WMS/GetCapabilities request and receiving the XML response

3.2.2. Geospatial Processes

Geospatial processes were implemented by employing the 52° North WPS HTML interface freely provided through the wps-js Javascript library. This way an HTML form was generated through which it is possible to encode and parse XML-based WPS requests (GetCapabilities, DescribeProcess, Execute) for the geospatial processes offered by 52° North initiative WPS interface implementation, as well as some other OGC WPS compatible geoprocessing servers (e.g. GeoViQua) [34].

To contribute over the above, a Node.js server was developed in the present work, in order to interface user generated WPS requests with GDAL/OGR library functionalities. These OGC

 doi:10.20944/preprints201801.0268.v1

Peer-reviewed version available at Geosciences 2018, 8, 139; doi:10.3390/geosciences8040139

compliant WPS requests are transmitted through 52° North WPS client interface where the Node.js server was also declared in it.

Below is a step by step representation of how interaction between client (WPS Client) – server(Node.js) – Cloud servers (WPS Servers) is taking place to complete a WPS request with wps-js and Node.js server.

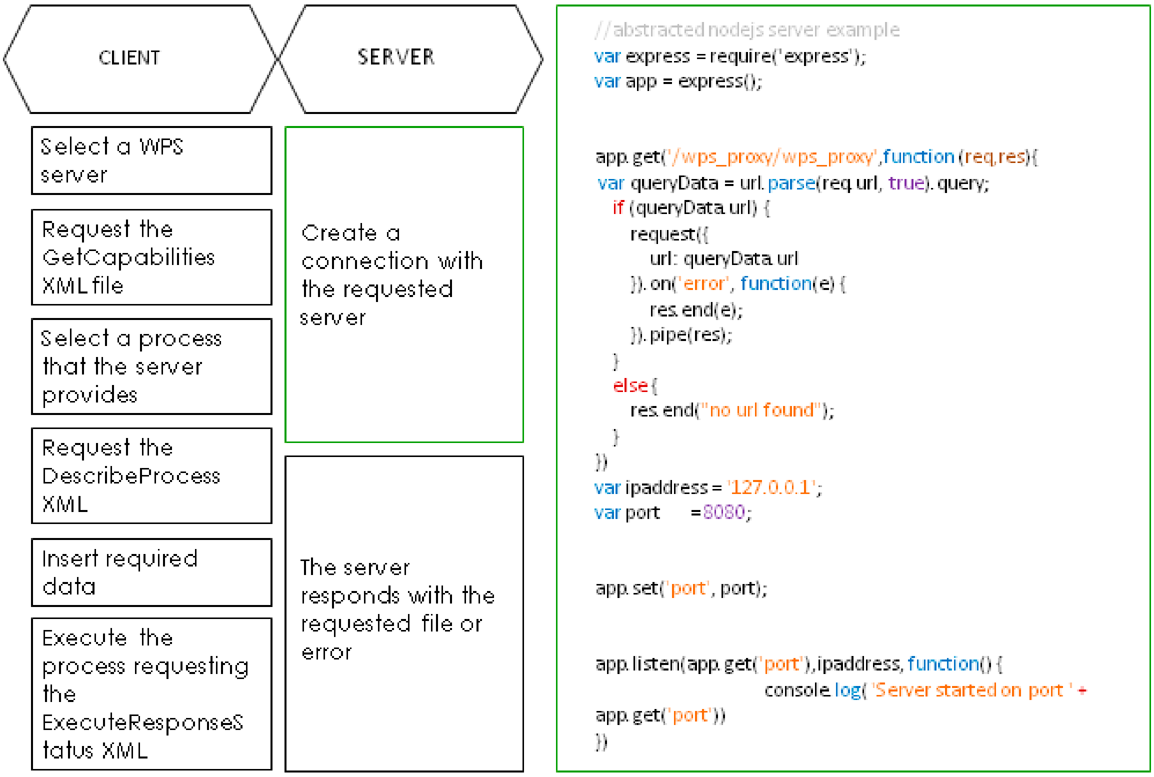

Figure 3: Utilizing Node.js as a Proxy server to achieve cross-origin connections with OGC implementations

3.2.3. Descriptive data management

Descriptive data involved in OGC Web services are an essential part of the development process because they specify the parameters of any type of request. These parameters are composed/expressed/edited in many ways and four of them are mentioned below. (1) and (2) concern requests submitted to geospatial servers, while (3) and (4) concern handling of the requests on the client-side:

(1) HTTP GET Requests

HTTP is the simplest way to submit a request to an OGC service implementation through the browser's URL bar and may also be incorporated in a Javascript interface using AJAX requests. The URL expression below represents a WFS request for getting features from a geospatial server

```
http://nsidc.org/cgi-bin/atlas_north?
service=WFS&
version=1.1.0&
request=GetFeature&
typename=greenland_elevation_contours
```

(2) HTTP POST XML requests

OGC Web services may support the "POST" method of the HTTP protocol and the request message is formulated as an XML document. XML tags, host the values of the parameters composing a request in a tree structure. In addition, they host the features and attributes of a vector layer. In any case, XML files establish OGC based interoperability acting as the medium for data and processes exchange between machines. The XML code represented below provides a WPS request

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 January 2018 doi:10.20944/preprints201801.0268.v1

Peer-reviewed version available at Geosciences 2018, 8, 139; doi:10.3390/geosciences8040139

which returns to the requester the description of all the geospatial processes offered by a WPS server.

```
<?xml version="1.0" encoding="UTF-8"?>
<wps:DescribeProcess service="WPS" version="1.0.0"
     xmlns:wps="http://www.opengis.net/wps/1.0.0" xmlns:ows="http://www.opengis.net/ows/1.1"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:schemaLocation="http://www.opengis.net/wps/1.0.0
http://schemas.opengis.net/wps/1.0.0/wpsDescibeProcess_request.xsd">
     <ows:Identifier>all</ows:Identifier>
</wps:DescribeProcess>
```

Another example has been presented in Figure 2.

(3) GeoJSON

XML files are transformed to GeoJSON using the new specification RFC 4976 in order to be expressed as native Javascript objects and handled appropriately, in terms of parsing and generating the parameters of OGC service requests. Being JSON objects they may be easily visualized as paged tables, may be modified by the end-user and may be reconstructed in XML code. An example of the bounding box property of a layer coded as properties of a JSON Javascript object is shown below:

```
"type": "Feature",
  "geometry": {
    "type": "Point",
    "coordinates": [125.6, 10.1]
  },
  "properties": {
```

(4) Paged Tables

A paged table can contain inner tables in its rows and this way of representation is convenient when dealing with layers and their properties (e.g. bounding box, EPSG etc.). In addition it is possible to provide domain values for every attribute assisting further request manipulation to the end-user, as shown in the figure below:

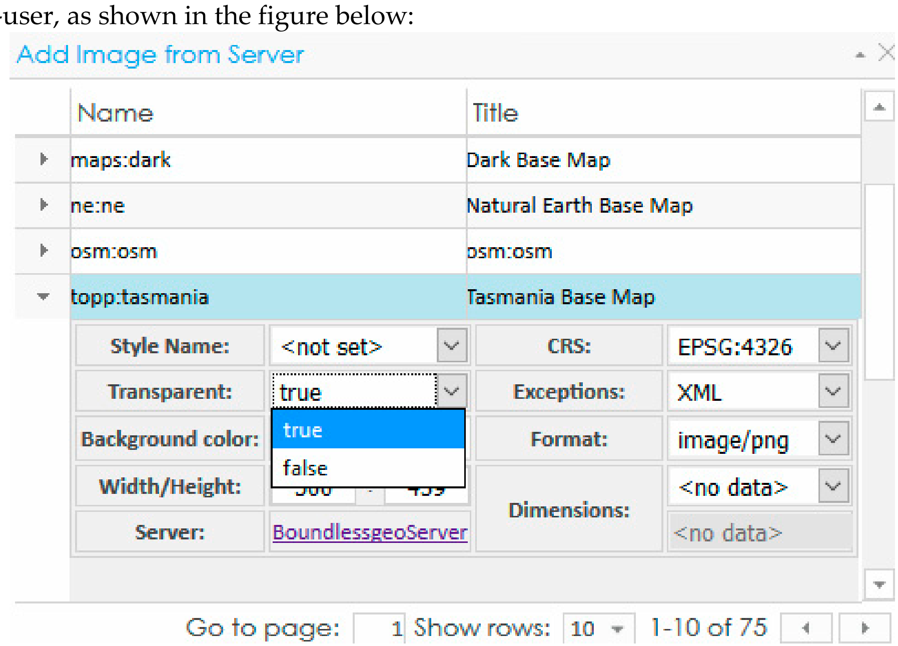

Figure 4: Setting WMS parameters through a paged table

3.3. End-user interface

3.3.1. User Interaction

The end-user interface implements request and response interaction with the available OGC Web services (e.g. WMS, WFS, WPS). As already discussed the results of the above interaction may be XML-based files or images as shown in figure 5 (Papadopoulos & Evangelidis, 2016).

Peer-reviewed version available at Geosciences 2018, 8, 139; doi:10.3390/geosciences8040139

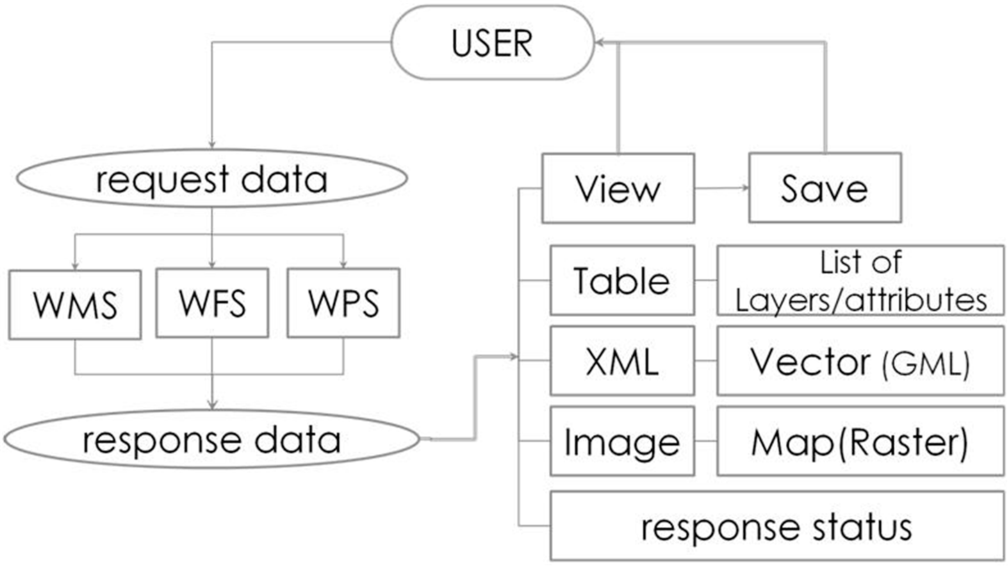

Figure 5: User interaction and data type results (Papadopoulos & Evangelidis, 2016)

Specifically, user interaction results involve:

- Tabular data with a) the available raster or vector layers formed by WMS/WFS GetCapabilities XML-based files and b) attributes of selected layers formed by WFS/GetFeatures XML-based files
- Vector data coded in GML, the prevailing XML-based format
- Raster data in image file formats representing maps

3.3.2. Major Operational Areas

A prototype service-based end-user interface has been proposed (Papadopoulos & Evangelidis, 2016) and is adopted in the present work as the base for the presented implementation. In the presented work this is extended to include geospatial data processing functions. Aim of the final prototype design is to achieve a typical desktop GIS-based 'look and feel' interface, exclusively exploiting geospatial Web services for data retrieval and processing purposes and this is performed with a completely transparent to the simple user way. The following major operational areas for both advanced and simple operations are identified:

- Data Management Area

At this area it is possible to declare the geospatial service providers. As soon as a server is declared WMS-WFS/GetCapabilities requests are submitted to it, resulting to the development of lists with the available raster and vector data. By selecting a layer from the above lists, either raster or vector it is possible to view and select its parameters, preparing that way the exact WMS/GetMap or WFS/GetFeatures respectively, request for submission. Alternatively, the user is capable of uploading layers to be included in the project.

Since, the whole environment is a service-based environment the presented layers are dynamically requested by the servers offering them, whenever the user checks for their visibility. To permanently obtain desired layers, at this area it is possible to clarify which of the requested layers will be cloned to form the GIS project on a local environment.

- Content Area

As already stated, layers selected in the Data Management Area are requested on a real time basis directly from the service provider. Whenever the end-user performs additional requests according to a desired parameterization, the server responds accordingly and the result is temporarily rendered in the front-end. This area contains the spatial content that has been permanently selected to form the GIS project and is therefore stored locally.

- Data Visualization Area

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 January 2018 doi:10.20944/preprints201801.0268.v1

Peer-reviewed version available at Geosciences 2018, 8, 139; doi:10.3390/geosciences8040139

This area is charged with visualizing the desired spatial content. Visualization concerns either the results of the service requests individually, such as for example an image returned or an XML file itself, or various themes overlaid to form a GIS project.

- Messages Area

This area provides feedback to the end-user by presenting messages returned by server responses.

- Data Processing Area

This area provides the necessary capabilities for declaring a geoprocessing server compatible with OGC/WPS specification and parameterizing a data processing request. The WPS implementation of this area is dynamically formed according to the type and the complexity of the requested geoprocessing job.

Figure 6 provides a visualization of the end-user interface operational areas:

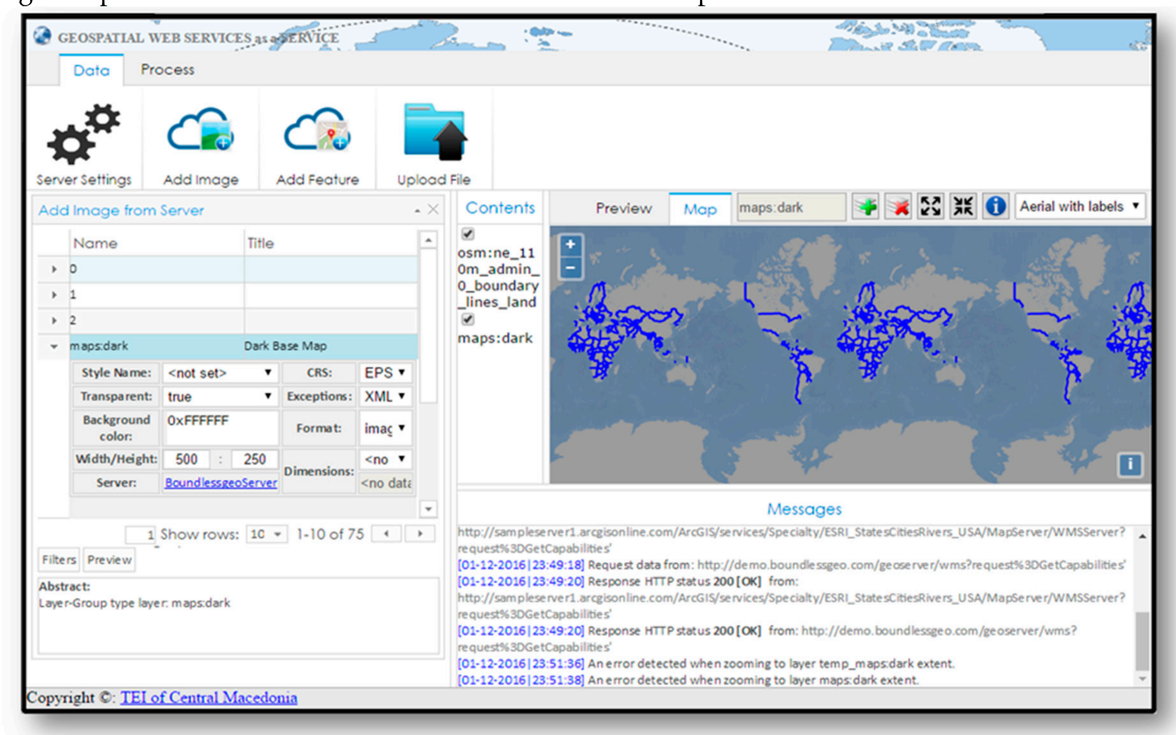

Figure 6: A Web interface implementing geospatial Web services

4. Demo Presentation

A demonstration case containing routine geospatial activities similar to those performed in a desktop GIS environment is presented, implementing the following scenario:

'Create a simple layout of the world overlaid by the country boundaries and export a vector layer of the boundaries in a shapefile format'

The scenario is further analyzed to the following geospatial activities:

- Import a world map
- Import country boundaries
- Export the features of the buffer in shapefile format

Each of the above mentioned geo-activities will be performed by employing respective geospatial services by different servers. In detail:

(1) The ArcGIS online sample server (http://sampleserver1.arcgisonline.com/) will be employed to provide the world map through the appropriate WMS service

(2) The Boundless demo Geoserver (http://demo.boundlessgeo.com/geoserver/web/) will offer features of the country borders through its WFS services

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 January 2018 doi:10.20944/preprints201801.0268.v1

Peer-reviewed version available at Geosciences 2018, 8, 139; doi:10.3390/geosciences8040139

(3) A custom Node.js server was developed for the purposes of the present work and was registered in 52o North WPS HTML interface developed with wps-js Javascript library (https://github.com/52North/wps-js), with the aim to transform the GML file in to shapefile format, by exploiting GDAL/OGR libraries as described in paragraph 3.2.2

Below are presented the end-user (U) actions and the subsequent server (S) reactions, both handled by the JS interface (I).

Table 2: User actions, interface handling and server reactions

Actor	*User Action – Interface – Server Reactions*
U	Declares WMS and WFS servers
I	Submits WMS-WFS/GetCapabilities requests to the declared servers
S	Return XML files with the offered raster and vector layers
I	Transforms XML files to lists of available raster and vector data in the Data Management area
U	Scans the lists with the available raster data and selects a layer of the world map
I	Submits WMS/GetMap request to the WMS Server offering the requested map
S	Returns the requested raster image map
I	Displays raster image map in the Data View area
U	Scans the lists with the available vector data and selects a layer of the world boundaries
I	Submits WFS/GetFeature request to the WFS Server offering the requested features
S	Returns GML file with the requested features
I	Displays raster image in the Data View area
U	Selects layers to form Layout
I	Permanently stores locally the selected layers which are overlaid in Content area
U	Selects Geospatial Processing Tools and declares WPS server
I	Submits WPS/GetCapabilities request to the declarred Server
S	Returns XML file with the offered processes
U	Selects the Convert file process
I	Submits a WPS/DescribeProcess request
S	Returns XML file with a description of the specifications of the requested process
I	Displays the specifications of the requested process and prompts for user action in filling out parameters and, if required, providing data
U	Fills the requested data/parameters and submits a request to execute the process
I	Submits a WPS/ExecuteProcess request
S	Returns the results of the requested process
I	Provides the results

Figure 7 visualizes the above scenario workflow:

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 January 2018 doi:10.20944/preprints201801.0268.v1

Peer-reviewed version available at Geosciences 2018, 8, 139; doi:10.3390/geosciences8040139

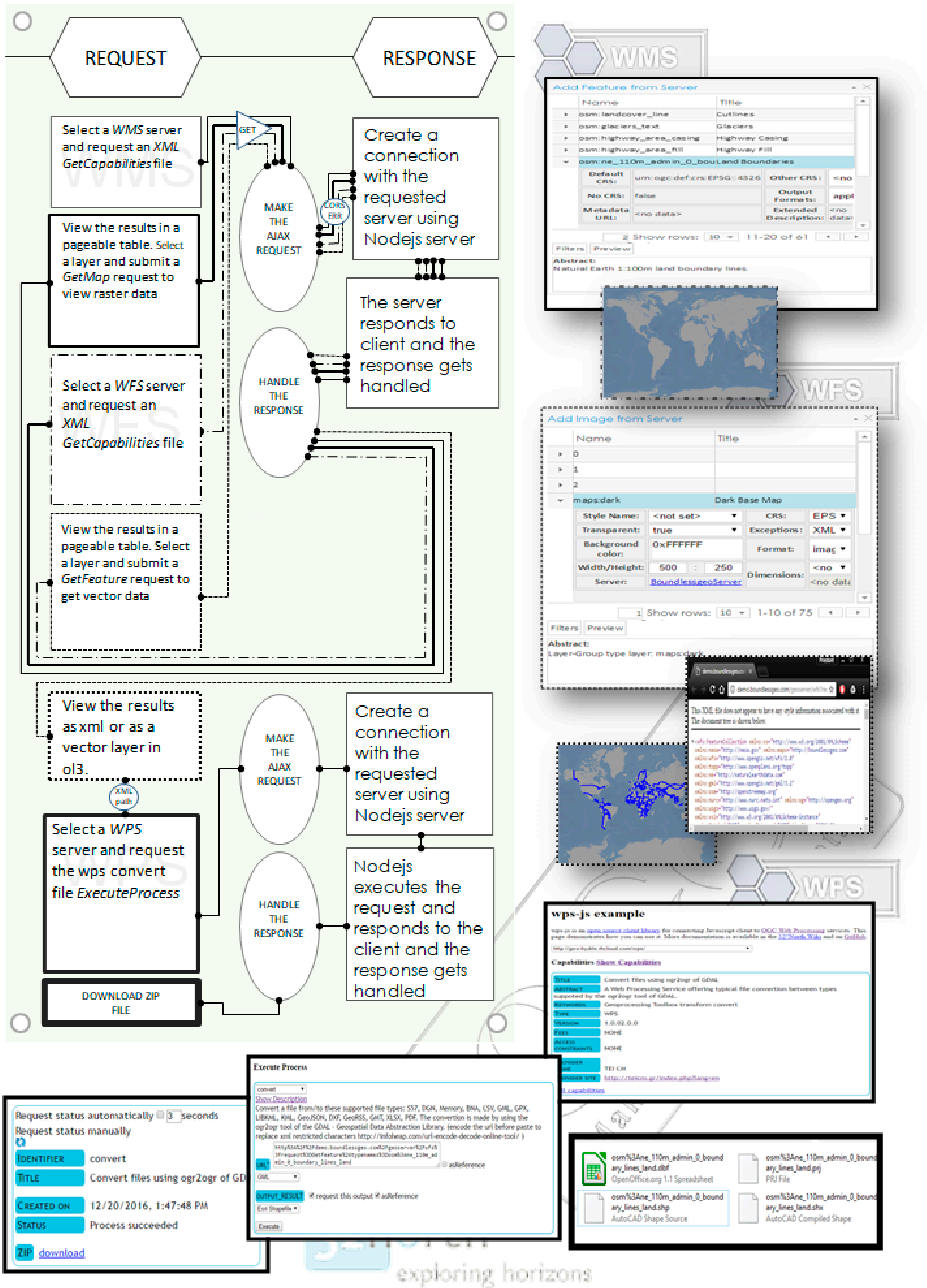

Figure 7: Scenario workflow diagram

Peer-reviewed version available at Geosciences 2018, 8, 139; doi:10.3390/geosciences8040139

5. Conclusions

The presented work deals with invocable geospatial Web services and explores the potentiality of re-serving them under a fully transparent Web-based cross-platform interface in order to satisfy routine GIS functionalities. As such, the presented solution, is based on Javascript, relies on open standards, is independent of additional software components, add-ins or APIs, and all is needed is an updated Web browser. Even in the case of utilizing a server to implement a custom WPS service to satisfy a specific geo-process, the presented solution remains in Javascript. This way both server and client components are light enough to reside on the client side, making the whole venture highly efficient and unique.

An interesting topic worth discussing in the present work is the development of the geospatial processing service provided by Node.js server which is invoked through 52oNorth wps-js interface. This task is subdivided into two discrete subtasks:

- the creation of the appropriate XML content modeling the description and execution of an OGC WPS compatible process and,
- the employment of a GIS engine performing this geospatial process.

The first subtask is a matter of editing the exact parameters of the WPS requests inside the appropriate XML tags. The second subtask requires the existence of GIS engines inside the WPS server and thereafter the establishment of an interaction between the engines and the server. In this respect Node.js was proved to be a convenient solution due to the direct communication with GDAL/OGR libraries command line. Extending this to other GIS APIs is expected to be a quite efficient and easy to implement task due to the capability of calling functionalities in most free and open source projects like those supported by open source geospatial foundation, OSGeo (e.g. GRASS GIS and QGIS). Even more, in the case of ArcGIS the Javascript API may also be employed to facilitate the Node.js communication with its GIS engine. Therefore, building WPS geospatial processes through Node.js may be considered as a great opportunity for further developments and extensions of the presented work.

Three of the most representative projects of the geospatial community, dealing exclusively with WPS standard are briefly cited: a) 52oNorth initiative serves a significant number of WPS implementations, and offers wps-js, a Javascript library that makes possible to register WPS implementations and provide Web access for requesting and executing geospatial processes, b) ZOO-Project , an OSGeo incubating project, offers an integrated WPS suite covering all the way from server to client including a server solution with a huge collection of implemented WPS services, a Javascript API for services creation and a Javascript library for Web interaction and c) PyWPS , also an OSGeo incubating project is a server side Python solution assisting the development and exposure of custom geospatial calculations. The presented work is in its very early stage, however it may potentially be enriched with stuff provided by all of the above mentioned. For the time being it adopts wps-js, registers in it a Node.js server and implements a demo WPS service. Thus, it provides a client interface together with a WPS server, that both of them employ Javascript libraries. In addition, the presented work does not focus only on WPS and extends its vision to satisfy a complete geospatial environment offering routine GIS functions.

Acknowledgments: The authors wish to acknowledge financial support provided by the Research Committee of the Technological Educational Institute of Central Macedonia under grant SAT/GS/170915-169/14.

References

1. Dragicevic, S. (2004). The potential of Web-based GIS. *Journal of Geographical Systems*, *6*(2), 79-81.
2. Chow, T. E. (2008). The potential of maps APIs for internet GIS applications. *Transactions in GIS*, *12*(2), 179-191.
3. Oxera, 2013. *What is the economic impact of Geoservices? Prepared for Google.* Available from: http://www.oxera.com/Latest-Thinking/Publications/Reports/2013/What-is-the-economic-impact-of-Geo-services.aspx [Accessed 5 December 2016]
4. McKee, L., Reed, C., & Ramage, S. (2011). OGC Standards and Cloud Computing. *OGC White Paper*.
5. Yang, C., Goodchild, M., Huang, Q., Nebert, D., Raskin, R., Xu, Y., Bambacus, M., & Fay, D. (2011). Spatial cloud computing: how can the geospatial sciences use and help shape cloud computing?. *International Journal of Digital Earth*, *4*(4), 305-329.
6. Evangelidis, K., Ntouros, K., Makridis, S., & Papatheodorou, C. (2014). Geospatial services in the Cloud. *Computers & Geosciences*, *63*, 116-122.
7. Aktas, M. S., Aydin, G., Fox, G. C., Gadgil, H., Pierce, M., & Sayar, A. (2005, June). Information Services for Grid/Web Service Oriented Architecture (SOA) Based Geospatial Applications. In *Proceedings of 1st International Conference Beijing China November* (pp. 27-29).
8. Buyya, R., Yeo, C. S., Venugopal, S., Broberg, J., & Brandic, I. (2009). Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation computer systems*, *25*(6), 599-616.
9. de la Beaujardiere, J., 2004. Web map service (WMS). Version 1.3. OGC 04-024. Open Geospatial Consortium, Inc., 85pp.
10. Evans, J., 2003. Web coverage service (WCS). Version 1.0.0. OGC 03-065r6. Open Geospatial Consortium, Inc., 67pp.
11. Vretanos, P., 2005. Web feature service (WFS) implementation specification. Version 1.1.0. OGC 04-094. Open Geospatial Consortium, Inc., 131pp.
12. Martell, R., 2004. OGC™ catalogue services—ebRIM(ISO/TS 15000-3) profile of CSW. Version 0.9.1. OGC 04-017rl. Open Geospatial Consortium, Inc., 87pp.
13. WPS Concepts. Available online: http://geoprocessing.info/wpsdoc/Concepts#chaining (accessed on 26th January 2018)
14. Percivall, G. (2010). The application of open standards to enhance the interoperability of geoscience information. *International Journal of Digital Earth*, *3*(S1), 14-30.
15. Papadopoulos, T., & Evangelidis, K. (2016). An HTML tool for exploiting geospatial web services. In Geospatial World Forum, 23-26 May 2016, Rotterdam. Geospatial World Forum.
16. European Commission, 2014. Commission Regulation (EU) No 1312/2014 of 10 December 2014 amending Regulation (EU) No 1089/2010 implementing Directive 2007/2/EC of the European Parliament and of the Council as regards interoperability of spatial data services Available from: http://data.europa.eu/eli/reg/2014/1312/oj [Accessed 5 December 2016]
17. European Commission, 2007. European Commission Directive 2007/2/EC of the European Parliament and of the Council of 14 March 2007 establishing an Infrastructure for Spatial Information in the European Community (INSPIRE). Off. J. Eur. Union, 50 (2007), pp. 1–14
18. Haklay, M., & Weber, P. (2008). Openstreetmap: User-generated street maps. Pervasive Computing, IEEE, 7(4), 12-18
19. Adding WMS services. Available online: http://desktop.arcgis.com/en/arcmap/10.3/map/web-maps-and-services/adding-wms-services.htm (accessed on 26th January 2018)
20. QGIS Python Plugins Repository. Available online: https://plugins.qgis.org/plugins/wfsclient/ (accessed on 26th January 2018)
21. Granell, C., Díaz, L., & Gould, M. (2010). Service-oriented applications for environmental models: Reusable geospatial services. *Environmental Modelling & Software*, *25*(2), 182-198.
22. Stollberg, B. and Zipf, A., 2007, OGC Web Processing Service Interface for Web Service Orchestration – Aggregating Geo-Processing Services in a Bomb Threat Scenario, pp. 239–251 (Cardiff, UK: Springer).
23. Lapierre, A., & Cote, P. (2007, October). Using Open Web Services for urban data management: A testbed resulting from an OGC initiative for offering standard CAD/GIS/BIM services. In Urban and Regional Data Management. Annual Symposium of the Urban Data Management Society (pp. 381-393).

Peer-reviewed version available at *Geosciences* **2018**, *8*, 139; doi:10.3390/geosciences8040139

24. Meng, X., Xie, Y., & Bian, F. (2010). Distributed Geospatial Analysis through Web Processing Service: A Case Study of Earthquake Disaster Assessment.Journal of Software, 5(6), 671-679.
25. Evangelidis, K., Ntouros, K., & Makridis, S. (2012). Geoprocessing Services over the Web. In Proceedings of the 32nd EARSeL Symposium, Mykonos, Greece (pp. 344-349).
26. Tzotsos, A., Alexakis, M., Athanasiou, S., & Kouvaras, Y. Towards Open Big Geospatial Data for geodata. gov. gr.(2015). *Free and Open Source Software for Geospatial (FOSS4G)* (pp. 247-258).
27. Sayar, A., Pierce, M., & Fox, G. (2005, November). Developing GIS visualization web services for geophysical applications. In *ISPRS 2005 spatial data mining workshop, Ankara, Turkey*.
28. Sayar, A., Pierce, M., & Fox, G. (2006, February). Integrating AJAX approach into GIS visualization web services. In *Telecommunications, 2006. AICT-ICIW'06. International Conference on Internet and Web Applications and Services/Advanced International Conference on* (pp. 169-169). IEEE.
29. ModelBuilder tutorial. Available online: http://pro.arcgis.com/en/pro-app/help/analysis/geoprocessing/modelbuilder/modelbuilder-tutorial.htm (accessed on 26th January 2018)
30. Vockner, B., & Mittlböck, M. (2014). Geo-enrichment and semantic enhancement of metadata sets to augment discovery in geoportals. ISPRS International Journal of Geo-Information, 3(1), 345-367.
31. Utility Market Embraces ArcFM GIS Solution. Available online: http://www.esri.com/news/arcnews/spring99articles/05_utilitymkt.html (accessed on 26th January 2018)
32. Automating Complex Workflows using Processing Modeler. Available online: http://www.qgistutorials.com/en/docs/processing_graphical_modeler.html (accessed on 26th January 2018)
33. Sun, J., Liu, Y., Dong, J. S., Pu, G., & Tan, T. H. (2010, November). Model-based methods for linking web service choreography and orchestration. In *2010 Asia Pacific Software Engineering Conference* (pp. 166-175). IEEE.
34. GEO User Feedback System. Available online: http://geoviqua.stcorp.nl/home.html (accessed on 26th January 2018)