

Analysis of Server-side and Client-side Web-GIS data processing methods on the example of JTS and JSTS using open data from OSM and Geoportal

Marcin Kulawiak^{a,*}, Agnieszka Dawidowicz^b, Marek Emanuel Pacholczyk^{b,1}

^a Department of Geoinformatics, Faculty of Electronics, Telecommunication and Informatics, Gdansk University of Technology, Poland

^b Department of Real Estate Resources, Faculty of Geodesy, Geospatial and Civil Engineering, University of Warmia and Mazury in Olsztyn, Poland

Abstract

The last decade has seen a rapid evolution of processing, analysis and visualization of freely available geographic data using Open Source Web-GIS. In the beginning, Web-based Geographic Information Systems employed a thick-client approach which required installation of platform-specific browser plugins. Later on, research focus shifted to platform-independent thin client solutions in which data processing and analysis was performed by the server machine. More recently, however, the rapid development of computer hardware as well as software technologies such as HTML5 has enabled the creation of platform-independent thick clients which offer advanced GIS functionalities such as geoprocessing. This article aims to analyse the current state of Open Source technologies and publicly available geographic data sources in the context of creating cost-effective Web-GIS applications for integration and processing of spatial data. For this purpose the article discusses the availability and potential of Web-GIS architectures, software libraries and data sources. The analysis of freely available data sources includes a discussion of the quality and accuracy of crowd-sourced as well as public sector data, while the investigation of software libraries and architectures involves a comparison of server-side and client-side data processing performance under a set of real-world scenarios. The article concludes with a discussion of the choice of cost-effective Web-GIS architectures, software libraries and data sources in the context of the institution and environment of system deployment.

Keywords: OpenStreetMap, geoprocessing, Web-GIS, performance, JTS, architecture

1. Introduction

Over the last decade, processing, analysis and visualization of freely available geographic data using Open Source Web-GIS has come a long way. Initially such systems primarily used a thick-client approach, which required installation of platform-specific browser plugins. One of the technologies commonly used for this purpose was Adobe Flex, which enabled the integration of various geographic functionalities (Vanmeulebrouk et al., 2008). Other browser plugin technologies included the Cortona VRML client, which enabled 3D modelling of boreholes and their geological context (Masumoto et al., 2008) as well as Java Web Applets, which were used inter alia for

* Corresponding Author, e-mail: Marcin.Kulawiak@eti.pg.edu.pl,

1 Authorship statement

Marcin Kulawiak performed client-side and server-side data processing tests, and wrote the introduction, discussion as well as conclusions sections. Marek Emanuel Pacholczyk performed analysis of freely available data sources and conducted OSM data accuracy tests. Agnieszka Dawidowicz was the initiator of the research, supervised it and wrote parts of the free geographic data analysis section.

constructing online services for 3D city navigation using Java3D and OSM data (Schilling et al., 2009a),(Schilling et al., 2009b).

Later research focused on platform-independent thin client solutions, which moved critical system functionality server-side. Map services like those offered by OpenStreetMap (OSM) and Google Maps popularised the modern web mapping solution which provides simple user-friendly tools for easy zooming and panning (often referred to as “Slippy Map”) (Haklay & Weber, 2008). Popular JavaScript libraries which deliver Application Programmin Interfaces (API) for constructing Slippy Maps include Google Maps and OpenLayers. Slippy Maps have been used e.g. to deliver map-based environmental information with OpenLayers, Mapnik, PostGIS and a modified version of the OSM dataset (Ciepluch et al., 2009), serving as background for integration of multi-source data regarding marine ecosystem components (Chybicki et al., 2008; Dabrowski et al., 2009), creating map mashups by overlaying custom-made kml or shapefile data over Google Maps or OSM background tiles (Batty et al., 2010), analysis and prediction of traffic flows using Bing Maps API (Tostes et al., 2013), mapping Dengue patient location overlaid on government-provided maps using GeoServer and OpenLayers (Tiwari & Jain, 2013), building a multidimensional mapping system with Java, GeoServer and OpenLayers using Google Maps data (Zavala-Romero et al., 2014) as well as online mapping of floods, landslides and other hazards overlaid on Google Maps and OSM data using OpenLayers (Lagmay et al., 2017). The functionality of thin clients was often considerably reduced in comparison to thick clients, however they offered the advantage of being platform-independent.

Latest research has been focused on taking advantage of the rapid development of computer hardware as well as software technology in order to create platform-independent thick clients which offer advanced GIS functionalities. This has been made possible by the introduction of HTML5 and optimized web browser JavaScript interpreters, which have enabled e.g. client-side implementation of several spatial analysis algorithms for the purpose of Marine Cadastre data analysis (Dawidowicz & Kulawiak, 2017), or semi-real-time spatial data interpolation on a low-power mobile device (Kulawiak & Wycinka, 2017).

The presented examples show that Open Source Web-GIS provides a cost-effective solution to the problem of remote collection, integration, modification, analysis and dissemination of spatial data. The focus on optimizing the economical aspects of Web-GIS development likely also influences the choice of applied data sources, as all of the aforementioned systems work solely on freely available datasets.

This article aims to analyse the current state of Open Source technologies and publicly available geographic data sources in the context of creating cost-effective Web-GIS applications for integration and processing of spatial data.

2. Sources and quality of free geographic data

The establishment of web mapping services like those provided by OpenStreetMap (2004) and Google Maps (2005) in the mid 2000’s has in many ways revolutionized public access to spatial data. Web mapping services have become popular, perhaps even ubiquitous, systems created, used and maintained by governments, major technology companies as well as non-profit organizations. As a result, Web-GIS developers are faced with a diverse choice of freely accessible geospatial data. Thus, the available data sources should be adequately analysed before selecting the the appropriate datasets.

2.1 Data source evaluation: content

Open web map sources that have been chosen for this study represent three economic sectors:

the profit-driven private sector (Google Maps and Bing Maps), the public sector (Geoportal.gov.pl, Atlas Warmii i Mazur, MSIPMO) and the third (non-profit, crowd-sourced) sector (OSM). A short summary of the selected map sources is presented in Table 1.

Name	Google Maps/Earth	Bing Maps	OpenStreetMap	Geoportal.gov.pl	Atlas Warmii i Mazur	MSIPMO
Owner/Maintainer	Google	Microsoft Corporation	OpenStreetMap community	GUGiK (Head Office of Geodesy and Cartography)	Marshal Office of Warmia and Mazury Voivodeship	Miasto Olsztyn
Type of organization:	Subsidiary of Alphabet Inc. (a public company)	Public company	Collaborative mapping project	National mapping agency	Regional government (voivodeship level)	Local government (city level)
Coverage	Global	Global	Global	National (Poland)	Regional (Warmian-Masurian Voivodeship)	Municipal (Olsztyn)
URL	https://www.google.com/maps	http://www.bing.com/maps	http://www.openstreetmap.org	http://www.geoportal.gov.pl	http://atlas.warmia.mazury.pl	https://msipmo.olsztyn.eu
License/Terms of use	Google Maps/Earth Terms of Service	Bing Maps Terms of Use	ODbL (data), CC-BY-SA (cartography and documentation)	Regulated by: PGiK, UoIIP		
Data sources	Various (commercial data providers, public domain data, user-generated content)		Crowdsourced geographical information	State Geodetic and Cartographic Resource, local data		
Available as	Website, Mobile application, Desktop application	Website, Mobile application, Desktop application	Website	Website, Mobile application	Website	Website
Character	Commercial	Commercial	Non-commercial	Governmental	Governmental	Governmental
Started	2005	2005	2004	2005	2012	2011

Table 1. General information about studied sources of free spatial data.

The content of the studied map sources has been evaluated using a modified version of the method previously used to measure their progress towards the goals set by the INSPIRE directive (Dawidowicz, 2015). In its essence, the process involves describing the presence of a given spatial data type or lack of it with a + (value 1) or a - (value 0) respectively. A total score is then calculated as the sum of values for all data types. The comparative analysis has been performed in the context of the data themes defined by Annexes I-III of the INSPIRE Directive (2007). The comparison includes the Polish Geoportal due to it being the national implementation of INSPIRE.

The chosen investigation method has been further modified by addition of an in-between state, denoted by the plus-minus sign (value 0.5) representing partial coverage of a topic.

Annex	No	Theme	Google Maps/Earth	Bing Maps	OSM	Polish Geoportal	Atlas Warmii i Mazur	MSIPMO
Annex I	1	Coordinate reference systems	+	+	+	+	+	+
	2	Geographical grid systems	+	-	-	+	-	-
	3	Geographical names	+	+	+	+	+	+
	4	Administrative units	+	+	+	+	+	+
	5	Addresses	+	+	+	+	+	+
	6	Cadastral parcels	-	-	-	+	+	+
	7	Transport networks	+	+	+	+	+	+
	8	Hydrography (complex)	+/-	+/-	+/-	+	+	+/-

	9	Protected sites(complex)	+	+	+	+	+	+
Annex II	1	Elevation	+	+	-	+	+	-
	2	Land cover	+/-	+/-	+	+	+	+
	3	Orthoimagery	+	+	-	+	+	+
	4	Geology	-	-	-	-	-	-
Annex III	1	Statistical units	-	-	-	+	+	+
	2	Buildings	+	+/-	+	+	+	+
	3	Soil	-	-	-	+		
	4	Land use	+/-	+/-	+/-	+/-	+	+
	5	Human health and safety	-	-	-	-	-	-
	6	Utility and governmental services	-	-	-	+	+	-
	7	Environmental monitoring facilities	-	-	-	+	-	-
	8	Production and industrial facilities	-	-	-	-	+	-
	9	Agricultural and aquaculture facilities	-	-	-	+	-	-
	10	Population distribution — demography	-	-	-	-	-	-
	11	Area management/ restriction/regulation zones and reporting units	-	-	-	-	+	+
	12	Natural risk zones	-	-	-	-	-	-
	13	Atmospheric conditions	-	-	-	-	-	-
	14	Meteorological geographical features	-	-	-	-	-	-
	15	Oceanographic geographical features	-	-	-	-	-	-
	16	Sea regions	-	-	-	-	-	-
	17	Bio-geographical regions	-	-	-	-	-	-
	18	Habitats and biotopes	-	-	-	-	-	-
	19	Species distribution	-	-	-	-	-	-
	20	Energy resources	-	-	-	-	+	-
	21	Mineral resources	-	-	-	-	-	-
Annex I	Sum		7.5	6.5	6.5	9	8	7.5
Annex II			2.5	2.5	1	3	3	2
Annex III			1.5	1	1.5	6.5	5	4
Total			11.5	10	9	18.5	16	13.5

Table 2. Evaluation of the content of selected free spatial data sources in the context of INSPIRE data themes.

As it can be seen in Table 2, official and administrative data are best represented by government-operated map sources. Physical features are available in virtually all of the analysed data sources, although the quality of their representation may vary. This being said, none of the analysed web map sources contain all of the data types defined by the INSPIRE spatial data themes. The best coverage is provided by the Polish Geoportal, which is to be expected as it represents the national implementation of the INSPIRE Directive. The coverage of themes defined by Annex III is generally worst, but this is expected to gradually improve in the foreseeable future.

2.2 Data source evaluation: availability

Many web mapping services include an Application Programming Interface (API), which enables their integration with other software. The popularization of APIs, catalysed by the Google Maps API, introduced map mashups created by combining various spatial data sources (Batty et al. 2010). While APIs provide access to data and services (such as geocoding) in a way specific to each of them, a higher level of interoperability can be achieved by employing data exchange standards established by the Open Geospatial Consortium (OGC), which include (among others) Web Map Service (WMS), Web Map Tile Service (WMTS), and Web Feature Service (WFS).

A comparison of data acquisition methods provided by the studied sources is presented in Table 3.

	Google Maps/Earth	Bing Maps	OpenStreetMap	Geoportal.gov.pl	Atlas Warmii i Mazur	MSIPMO
Raw data download	Very limited (Map Maker Data Download)	-	All data available	WFS and ATOM (restricted access)	-	Limited
Export of user created drawings	+	+	(editors)	-	-	+
Map tiles	custom TMS	custom TMS	TMS, third-party WMS	WMS	-	-

Table 3. Methods of data acquisition from studied sources.

While available raster map tile formats are relatively similar to each other, raw data differs not only in content, but also in models and formats used to capture, store and transfer it. While most sources provide standard point, line and polygon features (also known as the “spaghetti” data model) in an open format such as shapefile (ESRI® Shapefile Technical Description, 1998), OSM data consist of three types of elements: nodes, ways and relations. Each of them can be described using key-value pairs called tags, however only nodes contain geometry data. While most Open Source GIS libraries offer direct support for reading the OSM XML format, it is also possible to obtain pre-processed OSM data in the form of shapefiles from online services operated by third-party institutions such as Geofabrik GmbH (2017).






2.3 Data source evaluation: quality and accuracy

Because the means of analysing quality of raster maps is quite limited (Ciepluch et al. 2010), the presented research focuses on investigating the quality and accuracy of OSM data for Olsztyn, a mid-sized city in northern Poland. The area of research encompasses the entire city (about 88 km²), including the sparsely built suburbs.

As reference, the analysis uses data from the BDOT10k public sector cadastral database. According to the Polish Act of 1989 on Geodesy and Cartography, the Polish cadastre must be created with an accuracy of 0.1 m, and must reflect all changes as soon as possible under penalty of a substantial fine. In consequence, it constitutes a high quality dataset which can be treated as reference for comparison to maps obtained from other sources.

OSM data has been acquired through the Overpass API and imported to a PostgreSQL database (with PostGIS extension) using `osm2pgsql`, at which point it was reprojected from WGS84 to EPSG:2178 to match the SRS of reference cadastral data. An overview of the studied OSM dataset is presented in Fig 1.

Legend

-  City boundary
-  Area shown in Figure 3
-  OSM buildings successfully matched to cadastral counterparts
-  OSM buildings not matched to cadastral counterparts
-  Location of Olsztyn in Poland (on the map below)

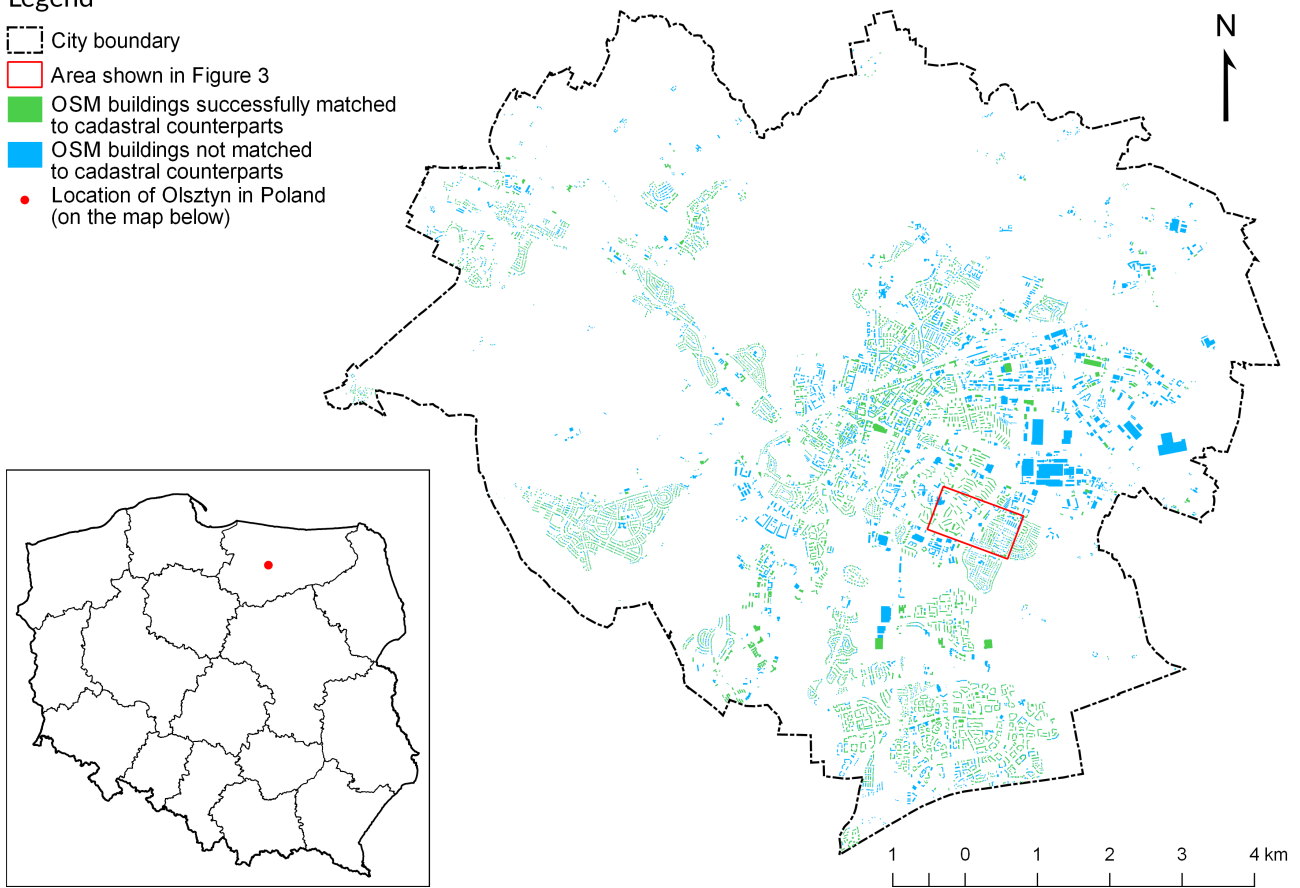


Figure 1. Location and overview of the analysed OSM dataset.

The comparison was based on the surface distance method first devised by Vauglin (1997), and later applied e.g. for investigation of building geometries (Bel Hadj et al., 1999) as well as lake outlines (Girres & Touya, 2010). For every feature of the investigated dataset, the method involves calculation of surface distance (dS) between the feature and its equivalent in the reference dataset. dS is calculated as the ratio of the features' symmetrical difference area to their union area. A graphical interpretation of the applied data quality analysis method is presented in Figure 2.

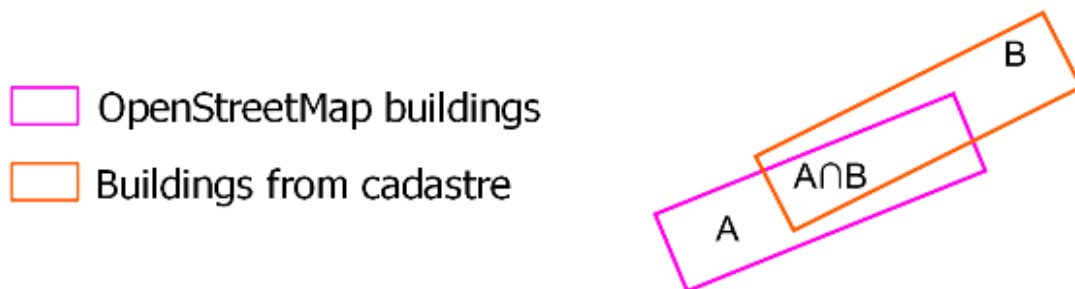


Figure 2. Graphical representation of the applied data quality analysis method, showing OSM features overlaid on reference cadastre data.

The OSM and cadastral datasets contained 12 872 and 23 058 polygons representing buildings, respectively. The building features in the reference dataset have been spatially joined

with attributes of 11 312 address points that had been obtained alongside the cadastral data. To ensure that the proper polygons are matched, cadastral building addresses have been compared with the ones stored in OSM tags. Moreover, buildings represented in radically different ways in both data sets (e.g. by using multiple polygons) have been filtered out, resulting in a sample of 7456 buildings (which amounts to 58% of the original OSM dataset).

The presented feature matching process was not effective for certain buildings deep within industrial (eastern part of the city) or military areas, as well as for utility buildings and garages, which are frequently without their own addresses. As it can be seen in Fig. 3, buildings of the latter category were also often completely missing from the OSM dataset.

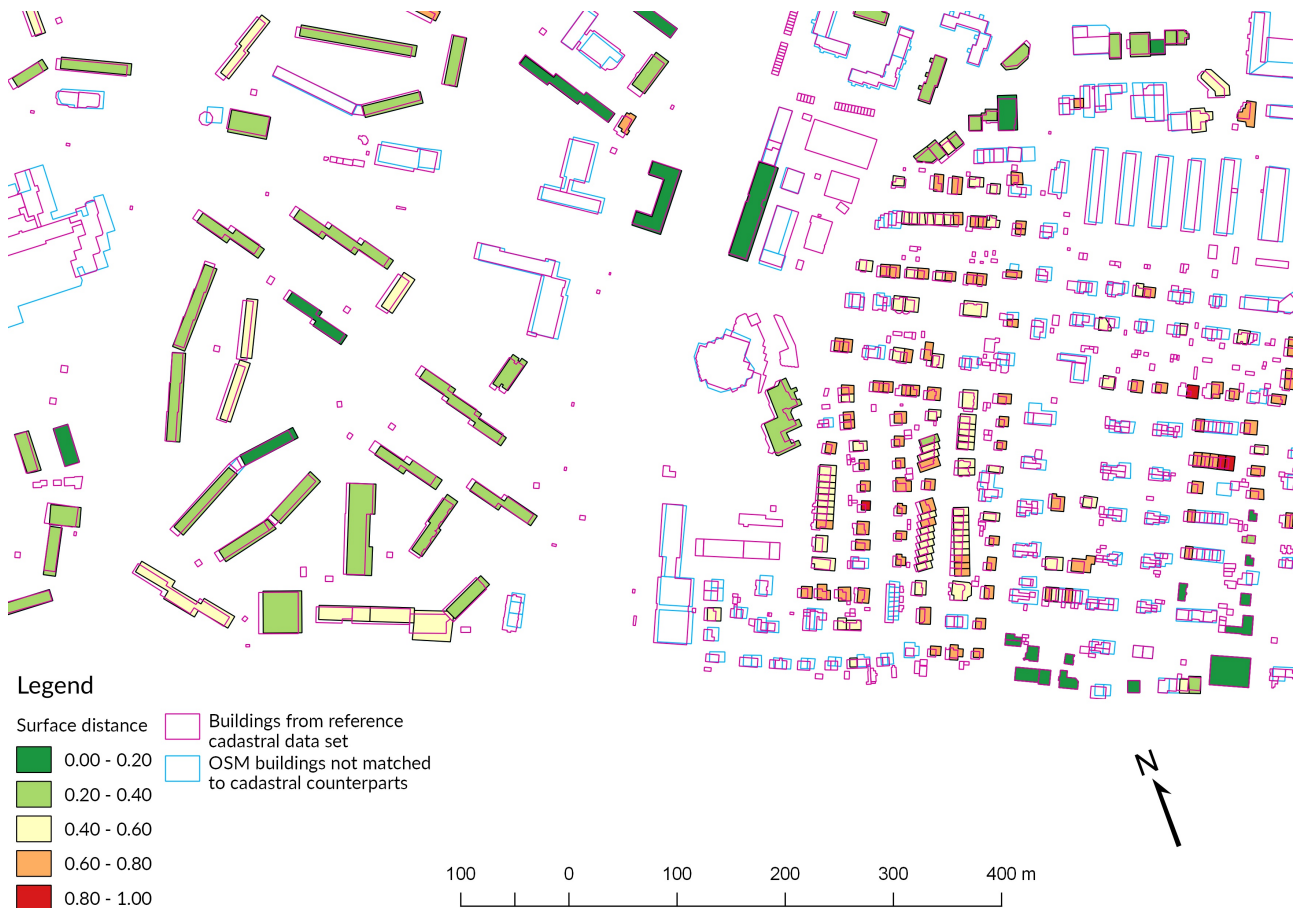


Figure 3. A fragment of the studied area, showing differences between OSM data and the reference dataset.

Fig. 3 also cartographically presents the calculated surface difference values for matched building features. Small differences are marked with shades of green, while large values are depicted in red colour. As it can be seen, the investigated OSM dataset shows a large diversity in feature accuracy. The largest differences are usually found in the smallest features, which could indicate that their OSM representations were created using lower resolution reference imagery. In case of features closely resembling those from the cadastre, use of more precise mapping methods can be suspected. A histogram of results obtained for the whole sample is presented in Fig. 4.

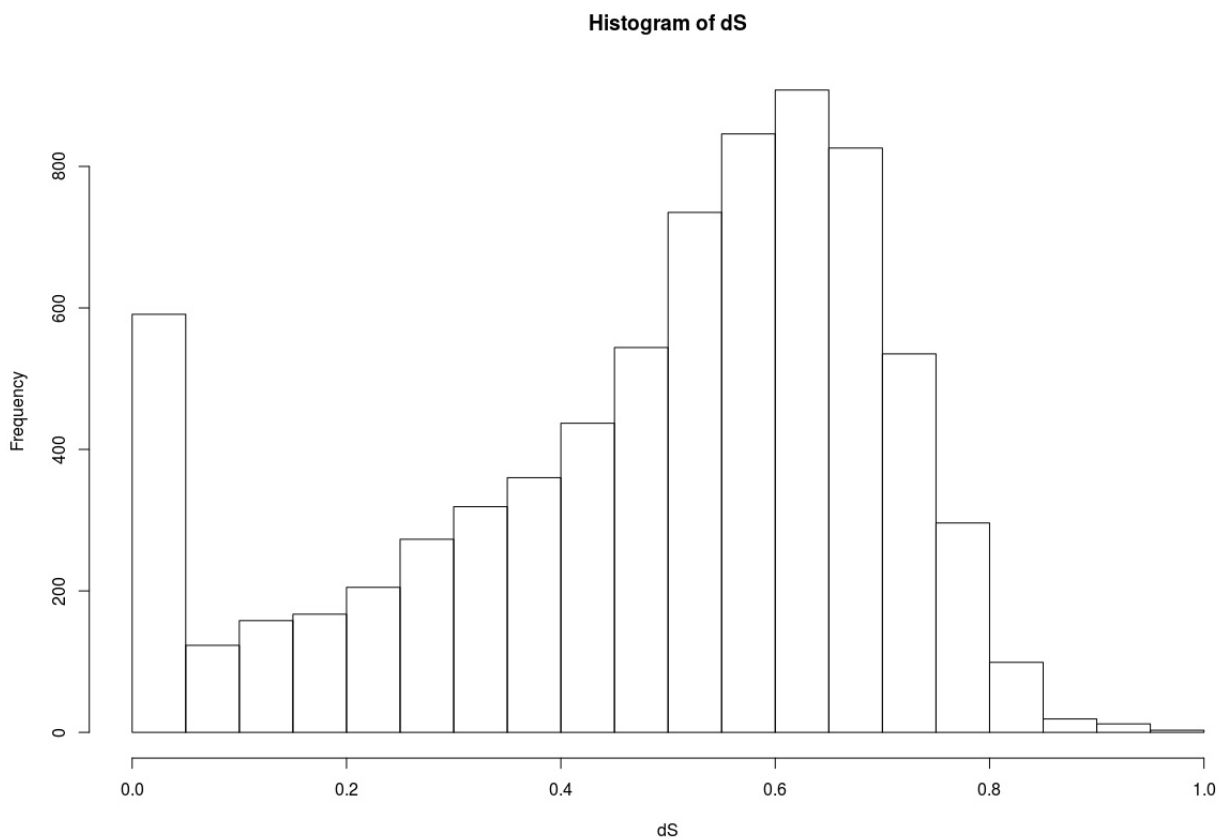


Figure 4. Distribution of surface distance (dS) between OSM features and reference cadastre data.

The histogram shown in Fig. 4 reveals a bimodal distribution of surface distances (dS), with peaks in the ranges of 0-0.05 m and 0.6-0.65 m. This supports the theory that at least two different approaches to data collection were employed. The minimum recorded distance from reference data was only 0.00019 m, while the maximum was 0.97888 m. The average dS for the whole sample was 0.48 m, and the standard deviation was 0.22 m. While there has been some concern that the process of conversion between EPSG:4326 (used by OpenStreetMap data), and EPSG:2178 (used by the reference cadastre) could introduce some errors, the results have shown otherwise. In particular, the value of an error introduced by the coordinate conversion process over a relatively small area would be similar for every processed feature. Yet, research has shown that for many features the difference is negligibly small, while being over an order of magnitude larger for others.

The obtained results generally confirm the findings of researchers from other countries, particularly that the accuracy of the OSM dataset can vary noticeably, even in relatively small and contained areas. At the same time, clusters of features mapped with a similar level of precision, be it high or low, can also be identified. While the accuracy of the investigated data is not ideal, with an average below 0.5 m it is still acceptable for many applications.

3. Architectures of Open Source Web-GIS for integration and processing of spatial data

A Web-GIS is built on a client-server architecture, which may adhere to either the thin-client or thick-client paradigm. In the former case, spatial data is stored and processed on the server, with results being sent to the client using a lightweight (usually raster tile) format. A thick client, on the other hand, will often process data directly on the client machine, which means that it first needs to download and later upload the data back in a vector format. A general comparison of those two paradigms is shown in Fig. 5.



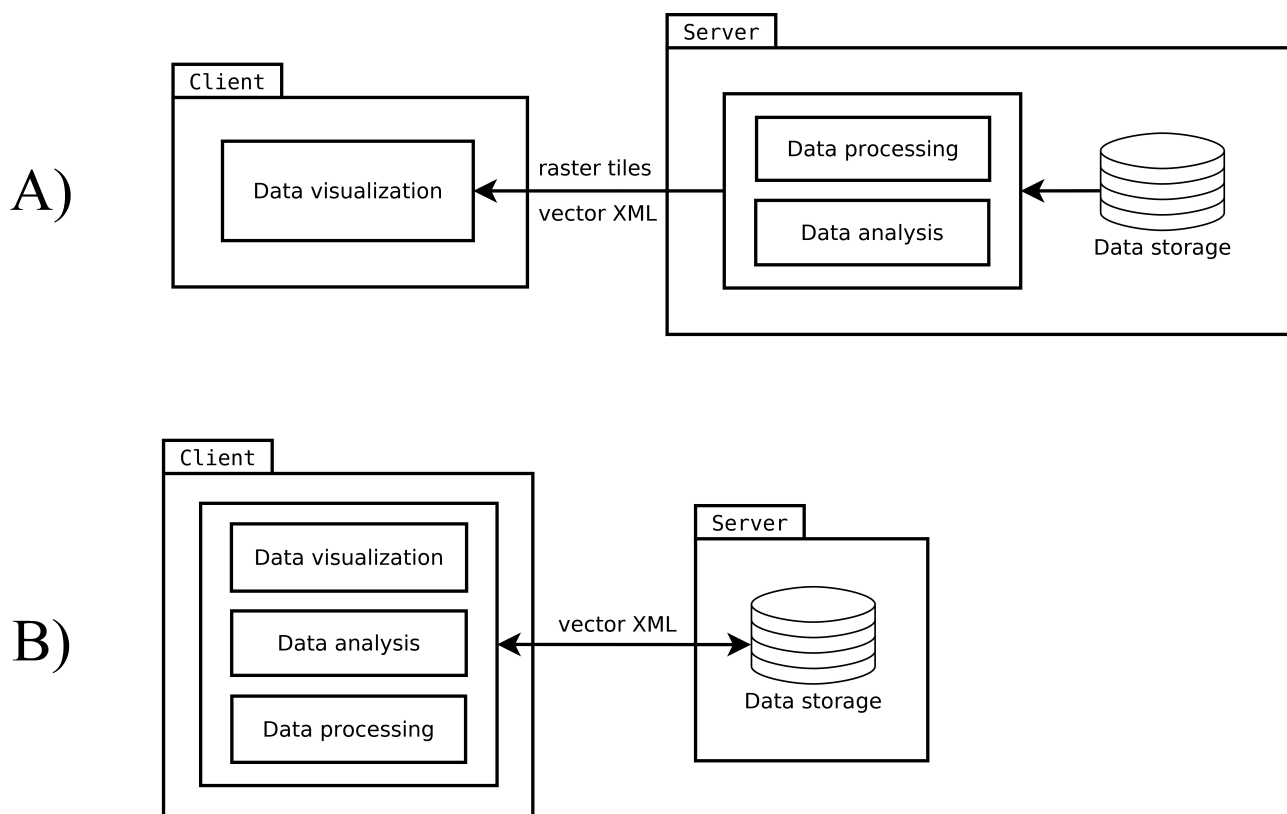


Figure 5. General overview of the thin client (A) and thick client (B) architectures.

In general, when a Web-GIS is used purely for data presentation, the optimal solution is to use a thin client which displays raster tiles, as this combination ensures high performance and low usage of bandwidth as well as CPU resources on both the client and server (Agrawal et al., 2014). This is possible due to the tiles being pre-generated, optimized for small size and cached on the server. However, generation and caching of raster tiles is a time and resource-consuming process, which in general should not be applied to dynamically evolving datasets. The latter are disseminated primarily in vector format, which ensures that all changes are quickly visible to all users. This characteristic may be observed e.g. when using OpenStreetMap services, as there is a visible time delay between making a change to OpenStreetMap data via its vector API, and the change being reflected in its Slippy Map raster tiles (OpenStreetMap, 2018). If both data processing methods use the same file format, it can be assumed that the average transfer time of this file between the client and server under given network conditions will be similar regardless of whether server-side or client-side data processing is employed. Thus, when analysing the network traffic differences between client-side and server-side data processing approaches, it is important to consider not only the time required to transfer a file of certain size between the client and server, but also focus on the number of times these transfers will take place. In this context, it is necessary to consider the differences between those data processing scenarios.

In the case of client-side processing, the minimum number of data transfers required to edit a file, process it (with an operation such as buffering) and save the result are as follows:

1. The layer data is downloaded during client initialization, or on user demand;
2. Layer data is edited and processed entirely on the client side;
3. Layer data is uploaded and stored on the server on user demand.

In the above context, the layer data is downloaded once and uploaded once, so the minimum number of data transfers is 2.

As far as server-side processing is concerned, the minimum number of data transfers required to

edit a file, process it (with an operation such as buffering) and save the result are as follows:

1. Layer data is downloaded during client initialization, or on user demand;
2. Layer data is edited on the client, after which it must be uploaded back to the server;
3. The client orders data processing on the server, after which the processed layer must be re-downloaded.

In the above context, the layer data is downloaded twice and uploaded once, which means the minimum number of data transfers is 3. As it can be seen, the number of dataset transfers is likely to be higher in the thin client architecture. However, this disadvantage may well be offset by faster processing of data on the server side. In this context, the presented comparison of client-side and server-side data processing methods involves performance testing as well as analysis of data transfer times.

3.1 Web-GIS architecture test environment

Comparing server-side and client-side data processing is not straightforward. The server and client machines usually constitute very diverse software and hardware environments. However, the hardware differences between servers and desktop computers have been gradually decreasing over the last decade, and it is not uncommon for server-grade machines to have similar single-core CPU performance as desktop computers. Thus, it can be argued that at this time the majority of performance disparity between server-side and client-side data processing is caused by the respective software environments. On the server side, spatial data is processed using well-established and optimized programming languages such as Java, .NET or C++ (Kulawiak et al., 2010). Meanwhile, if the client software is to be universally portable, data must be processed in a HTML5-compliant environment using technologies such as HTML Canvas, WebGL and JavaScript (Moszynski et al., 2015). A fair comparison of such disparate computing environments can only be made by implementing and testing the exact same data processing algorithms in both of them. Fortunately, the appropriate libraries already exist in the form of Java Topology Suite (JTS) and JavaScript Topology Suite (JSTS). JTS is a Java library for creating and manipulating vector geometry, which implements common geoprocessing functions such as union, intersection or buffer (Java Topology Suite, 2017). JSTS, on the other hand, is a JavaScript port of JTS, made using automatic translation of Java sources (Harrtell, 2017). Because both libraries implement virtually the exact same algorithms, using them presents a unique opportunity to compare the performance of the same code in two very different software environments.

For the purpose of the presented research, a simple thick-client Web-GIS using JSTS 1.5.0 and OpenLayers 4.6.4 has been implemented. The service loads a set of test data layers and performs the selected processing algorithm on them. Because server-side processing involves additional latencies induced by the applied frontend and framework, JTS has been tested in the form of Web Processing Services (WPS) implemented as part of GeoServer 2.12.1. The architecture of the test environment is shown in Fig. 6.



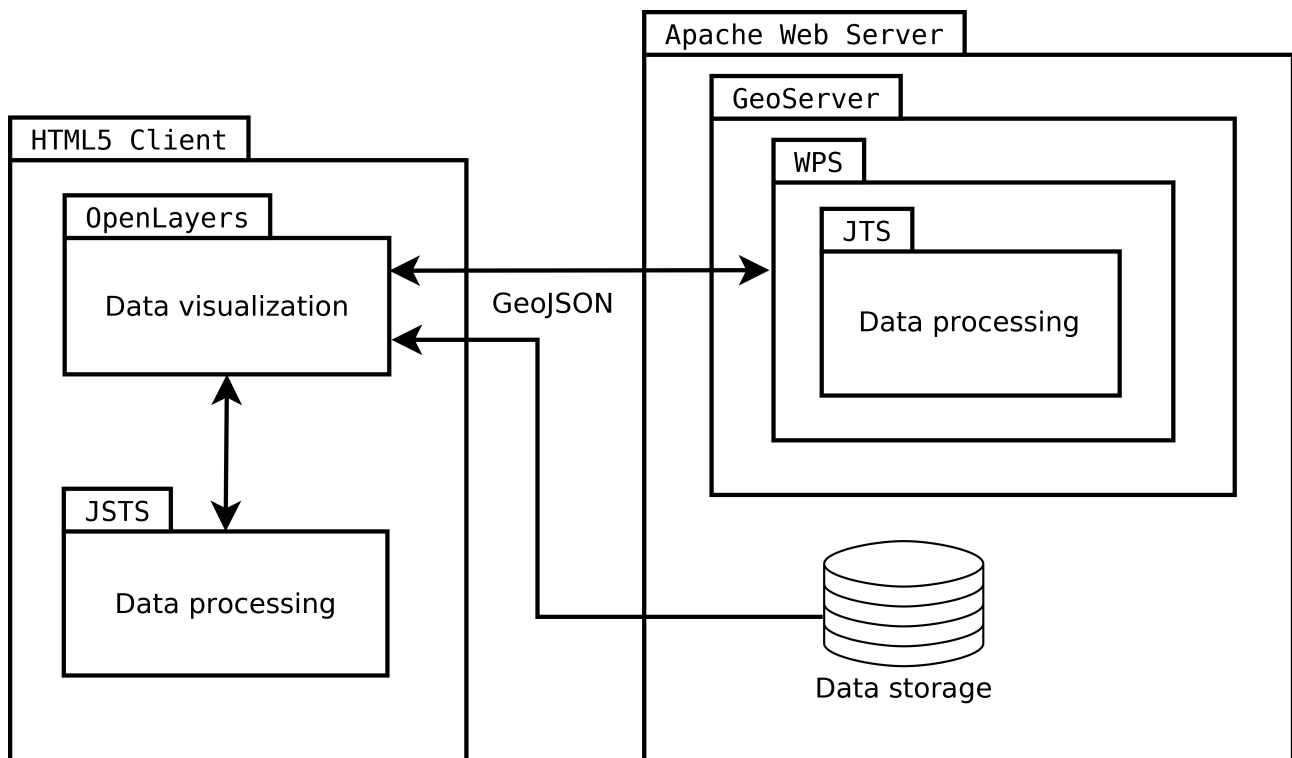


Figure. 6. Architecture of the Web-GIS used for comparing the performance of thick and thin client data processing.

The presented system operates on vector data in GeoJSON format. A simple OpenLayers GUI enables the processing of selected data layers via JTS WPS queries or respective JSTS procedures. The system has been tested on a desktop PC equipped with a quad-core Intel Core i5 2500K processor operating at 4.2 GHz and 16 GB of DDR3-1600 RAM, controlled by 64-bit Windows 8 Operating System. The chosen CPU offers a combination of IPC and clock speed that provides a good middle ground between single-core performance of modern desktop and server processors, while the single-threaded nature of performed tests ensures that the low number of cores (in comparison to server machines) will not impact the results.

The software used for tests comprised of the following 64-bit programs:

- Mozilla Firefox 57.0.4
- Google Chrome 63.0.3239.132
- Java SE 8u152

The system was operated from within the Apache 2.4.29 web server. The client module of the system has been run using Google Chrome and Mozilla Firefox, which at the time of writing are first and second most popular cross-platform Open Source web browser, respectively. JTS operations were tested using Google Chrome, while JSTS procedures were also run in Mozilla Firefox in order to test the impact of JavaScript interpreter performance on the end result.

3.2 Web-GIS architecture testing procedure

The tests have been performed using real-world scenarios involving geoprocessing of spatial data for the purposes of researching hazards in marine as well as municipal environments.

The former involved the analysis of location as well as areal impact of illegal oil discharges in the context of marine cadastral parcels, which may be used by responsible administration bodies for further study (Dawidowicz, Kulawiak, 2017). For this purpose, illegal oil spills in the Polish exclusive economic zone were identified by means of intersection with the collection of oil discharges in the entire Baltic Sea. Consecutively, the area of each spill was approximated by using

distance buffering.

Analysis of municipal environment hazards involved finding residential buildings whose residents may be exposed to high levels of railway noise. In the event of lack of in-situ measurements, noise may be mapped using established location-distance relationships (Brons et al., 2003)(Chew, 1998). In this context, areas in Tricity, northern Poland, characterized with high noise have been identified using a 100 m distance buffer of train tracks.

All data used during the tests has been obtained from freely available sources. Marine environment hazards have been analysed using the following datasets (Kulawiak, 2019):

- collection of illegal oil discharges observed in the Baltic Sea region during aerial surveillance in the years 1998-2015. The dataset contains 4337 point features and takes up 186 KB in GeoJSON format. The data has been obtained from the HELCOM Baltic Sea Environmental Monitoring database (HELCOM, 2017);
- boundaries of Polish exclusive economic zone, which is a region of the Baltic Sea under jurisdiction of Polish Marine Offices. The dataset contains 1 polygon feature comprised of 943 points and takes up 41 KB in GeoJSON format. The data has been obtained from the Polish Centre of Geodesic and Cartographic Documentation (CODGIK, 2017).

Municipal environment hazards have been analysed using the following datasets:

- collection of buildings in the City of Gdynia, located on the Northern shores of Poland. The dataset contains 18 355 polygon features comprised of 144 704 points and takes up 7155 KB in GeoJSON format. The data has been obtained from the OpenStreetMap database in shapefile format using the conversion site operated by Geofabrik GmbH (2017);
- collection of buildings in the Tricity, which is a metropolitan complex comprised of Gdynia, Sopot and Gdansk, located in Northern Poland. The dataset contains 64 582 polygon features comprised of 463 942 points and takes up 22 963 KB in GeoJSON format. The data has been obtained from the OpenStreetMap database using the conversion site operated by Geofabrik GmbH (2017);
- a map of railway-related noise in the Tricity area, obtained by distance-buffering the railway network feature dataset. The dataset contains 1 polygon feature comprised of 10708 points and takes up 459 KB in GeoJSON format. The original railway track data has been obtained from the OpenStreetMap database using the conversion site operated by Geofabrik GmbH (2017);

Prior to testing, all datasets have been converted into single GeoJSON files containing multi-feature objects which can be processed by JTS and JSTS in a single request. In both cases the operations have been timed with millisecond accuracy between starting of the procedure and reception of results. Every operation was run 10 times, with the browser being restarted and cache-cleaned between every test. Because a GeoServer WPS operation is compiled on its first run, the time of the first response for every type of JTS operation (1 for “intersection” and 1 for “buffer”) was not counted towards the average value.

3.3 Thick client vs thin client performance results

The performed tests consisted of the following operations:

- intersection1 - the intersection between Polish exclusive economic zone (1 polygon, 943 points) and illegal oil discharges in the Baltic Sea area (4337 points).
- intersection2 - the intersection between buildings in Gdynia (18 355 polygons, 144704 points) and the Tricity train noise polygon (1 polygon, 10 708 points).
- intersection3 - the intersection between buildings in the whole Tricity (64 582 polygons, 463942 points) and the Tricity train noise polygon (1 polygon, 10 708 points).

- buffer1 - the buffer of illegal oil discharges in the Polish exclusive economic zone (357 points).
- buffer2 - the buffer of illegal oil discharges in the entire Baltic Sea area (4337 points).

JTS operations were tested in Google Chrome, while JSTS operations were also tested in Mozilla Firefox. Results of the tests can be found in Table 4.

Operation	Average time of execution [ms]	Mean absolute deviation [ms]	Uncertainty [%]	Percentage of JTS execution time [%]	Average time difference vs JTS [ms]
JTS intersection1	44	4	9.29	100	0
JSTS intersection1 Firefox	183	1	0.80	414	139
JSTS intersection1 Chrome	186	1	0.30	420	142
JTS intersection2	3 137	33	1.05	100	0
JSTS intersection2 Firefox	10 405	258	2.50	332	7268
JSTS intersection2 Chrome	8 734	33	0.40	278	5 596
JTS intersection3	14 769	476	3.22	100	0
JSTS intersection3 Firefox	57 009	1389	2.40	386	42 239
JSTS intersection3 Chrome	47 416	184	0.40	321	32 646
JTS buffer1	56	3	6.08	100	0
JSTS buffer1 Firefox	197	7	3.80	352	141
JSTS buffer1 Chrome	168	4	2.60	300	112
JTS buffer2	383	4	0.97	100	0
JSTS buffer2 Firefox	1 030	18	1.70	268	646
JSTS buffer2 Chrome	920	15	1.60	240	537

Table 4. Comparison of server-side and client-side spatial data processing performance.



The test results presented in Table 4 reveal some interesting characteristics of both client-side as well as server-side spatial data processing. As expected, server-side data processing by means of JTS WPS was always faster than respective operations run in JSTS. At the same time, however, JTS exhibited the highest measurement uncertainty in the majority of tests (9.29 % in intersection1, 6.08 % in buffer1 and 3.22 % in intersection3). In most tests, the uncertainty of JTS time measurements was caused by server-side servlet caching and optimization, which caused consecutive responses to the same query to be slightly faster. In the case of intersection1 (9.29 %) and buffer1 (6.08 %) the high uncertainty values were likely caused by very short execution times of the actual operations, in which case small latencies introduced by the testing environment (web browser and WPS servlet) had a noticeable impact on the measured execution times. A similar effect can be noticed in the case of JSTS tests, where measurements performed using Mozilla Firefox (which is known to be the less optimized browser) regularly exhibit higher uncertainty than those made using Google Chrome. In consequence, running the same test in Google Chrome usually provided not only around 20% higher performance (in particular for larger datasets) over the same code executed in the Mozilla Firefox JavaScript interpreter, but also offered considerably more stable execution time.

As far as performance is concerned, server-side processing appears to have an advantage when it comes to memory-intensive operations. As it has been shown by the intersection tests, iterating over thousands of polygons in Java has shown to be on average over 3 times faster than performing the same operation in JavaScript. When it comes to CPU-intensive operations like buffering, however, the performance difference was noticeably smaller.

As far as data processing times are concerned, theoretical analysis of server-side and client-side data transfer patterns suggested that server-side processing on average may require more dataset transfers. Thus, whether its performance difference will be maintained in practice may strongly depend on the time required to transfer the dataset between the client and server. Estimation of average data transfer time is a complex process which involves analysis of several factors, including bandwidth, the physical distance between the start point and endpoint as well as routing between those two points (Gummadi et al., 2002)(Akella et al., 2003). However, the minimum times required to transfer data between two points on the network may be established using only the nominal data download speeds. Thus, a theoretical study of download speeds for datasets of different sizes has been conducted using transfer rates of 0.1 Mbps, 0.5 Mbps, 1 Mbps and 5 Mbps, assuming that 1 Mbps = 1024*1024 bits/s = 131072 bytes/s. The results, which show minimum data transfer times for every file, are presented in Table 5.

Dataset name, size [bytes]	Download speed at 0.1 Mbps [ms]	Download speed at 0.5 Mbps [ms]	Download speed at 1 Mbps [ms]	Download speed at 5 Mbps [ms]
Result of intersection1, 6362	485	97	48	10
Result of intersection2, 278471	21 245	4 249	2 124	425
Result of intersection3, 749211	57 160	11 432	5 716	1 143
Result of buffer1, 328521	25 064	5 012	2 506	501
Result of buffer2, 3986488	304 145	60 829	30 414	6 082

Table 5. Minimum time required to transfer the products of analysed operations in the context of

different transfer speeds.

The results of server-side data processing times shown in Table 4 have been adjusted by adding the projected dataset download times for the given transfer speeds from Table 5. The results, presented as percentage of execution time of the same operation in JSTS measured in Chrome, are shown in Table 6.

Operation (in Chrome)	% of JTS processing time at 0.1 Mbps	% of JTS processing time at 0.5 Mbps	% of JTS processing time at 1 Mbps	% of JTS processing time at 5 Mbps
JSTS intersection1	35	131	200	344
JSTS intersection2	35	118	165	245
JSTS intersection3	66	180	231	298
JSTS buffer1	0.6	3	7	30
JSTS buffer2	0.3	1.5	3	14

Table 6. Execution time of JSTS operations in Chrome shown as percentage of the execution time of the relevant JTS operation after adjustment for download time.

As it can be seen, JTS maintains its performance advantage in all intersection operations if the client-server connection offers download speed of at least 0.5 Mbps. This is because the resulting dataset is relatively small, the processing times are long and the performance advantage of JTS before adjustment is in the 2x/3x range. However, the situation is entirely different in the case of the buffer operations, which culminate relatively quickly, but produce relatively large results. Even on a 5 Mbps connection, performing the buffer1 operation server-side and downloading results to the client lasts over 3 times longer than performing the same operation in JSTS. For the same connection speed, the buffer2 operation performed server-side lasts 7 times longer than its JSTS equivalent. This is despite the initial JTS performance advantage of 300 % for buffer1 and 240 % for buffer2.

Even if the download times are not taken into account, the relative execution time of JSTS versus JTS compares quite favourably. While intersection1 ran over 4 times faster on the server side, the same operation in JSTS ended on average only 0.1 s later, a difference which would be entirely unnoticeable to the end user. The same may be said about buffer1 (0.1 s) and buffer2 (0.6 s). The intersection2 operation, which involved processing nearly 8 MB of data, lasted around 6 s longer when performed by JSTS. It is a noticeable but not inconvenient difference. In reality only the time difference of performing the intersection3 operation by JSTS could be considered as detrimental to the user experience.

At the same time, it is important that the presented results be put into context. While the JTS operations have proven to execute between 2 and 4 times faster in comparison to client-side processing, in a real-world scenario additional time would be necessary for constructing and sending the WPS query. To put things into perspective, when constructed using GeoServer's WPS query builder servlet, the intersection3 query was completed in 110 190 ms (total time required for building and executing the intersection3 WPS query, as measured using Mozilla Firefox debug console). Additional tests have also shown that encoding the end result contributes a significant part to the total processing time. When the WPS output format was set to GML instead of the simpler JSON, average processing time of the intersection3 query has shown to be 30% longer.

While it would be possible to optimize the execution time e.g. by bypassing WPS and running the operation directly in JTS, there still remains the issue of returning the results back to the client.

Using server-side data processing assumes a thin-client architecture where features are sent to clients in a lightweight format such as WMS image tiles. However, registering a new feature in a web map server and generating image tiles for it is a time-consuming process as well.

Basing on the obtained results, it may be concluded that for datasets with size less than 10 MB, client-side data processing may be a viable alternative to server-side operations.

3.4 Thick client vs thin client performance scaling

The fact that JTS and JSTS share a lot of code allows for an analysis of performance scaling in the function of the number of input features for client-side and server-side data processing. For this purpose it was necessary to choose a method which would process every input feature (in order to properly illustrate scaling with the number of input features) but at the same time offer a performance profile which would be consistent between different input datasets (which cannot be said about intersection, which sometimes needs to slice and reshape features). The buffer operation fulfils those requirements very well, and thus it has been chosen for testing the performance of JTS and JSTS under different data loads. For the same purpose, three datasets containing 1000, 10 000, 30 000 and 50 000 point features, respectively, have been generated in the Polish Exclusive Economic Zone area using Quantum GIS Random Points tool. The measurements have been performed in Google Chrome due to its superior speed and reliability. The performance scaling results may be found in Table 7.

Operation	Average time of execution [ms]	Mean absolute deviation [ms]	Uncertainty [%]
JTS buffer of 1000 features	102	8	8
JTS buffer of 10 000 features	1 168	12	1
JTS buffer of 30 000 features	10 306	87	0.8
JTS buffer of 50 000 features	34 214	294	0.9
JSTS buffer of 1000 features	280	3	1
JSTS buffer of 10 000 features	1 800	17	1
JSTS buffer of 30 000 features	13 040	215	1.6
JSTS buffer of 50 000 features	40 258	350	0.9

Table 7. Comparison of server-side and client-side spatial data processing performance scaling.

For every input feature (be it point, line or polygon), the JTS buffer algorithm computes a raw offset curve at a given distance from the source. Because this curve may contain self-intersections, the final buffer polygon is computed by forming a topological graph of all created curves and tracing its outside contours (JTS Javadoc, 2016). This operation is executed even if the input data consists of a single point, and performance profiling has shown that it is responsible for the polynomial growth characteristic of processing time versus input size, which may be observed in Fig. 7.



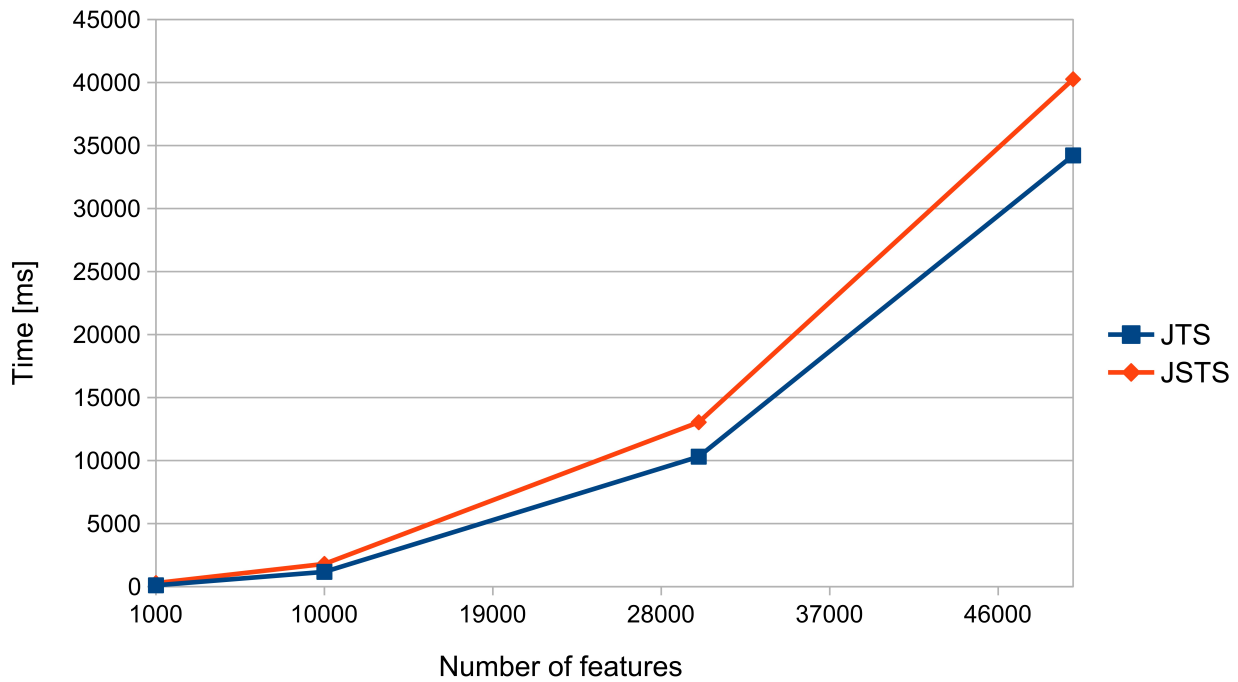


Figure. 7. Scaling of processing time versus input size for the buffer operation in JTS and JSTS.

As JTS and JSTS basically run the same code on different platforms, it is not surprising that they exhibit similar scaling of processing time versus input size. What is interesting, however, is how the relative performance of both libraries changes with the increasing number of input features. For the 1000 point dataset, JTS delivers 276% of JSTS performance. However, for the 10 000 point dataset this is reduced to 154%, and further down to 126% for the 30 000 point dataset. As far as the 50 000 point dataset is concerned, the performance advantage of JTS is only 17%. This is likely caused by the way both methods handle external data. Both JTS and JSTS convert input data into an internal format for processing, and then convert the results back into a format readable by other applications. However, whereas JTS needs to first parse and later encode the output data in a text-based format such as GeoJSON, JSTS directly interfaces with a data presentation library (OpenLayers), and thus reads input data and returns results in the form of binary objects. In consequence, JSTS not only reads and generates results faster than JTS, but those results are also smaller in size (and thus can be copied to the recipient much quicker). As the number of processed features increase, this advantage appears to have an increasingly prominent influence on the difference in processing time between those two methods.

4. Discussion

With the freely available data and software libraries discussed in the preceding sections, the price of creating a fully-featured Web-GIS can be reduced to the cost of computer hardware. The contents and coverage of freely available data sources have shown to be good enough to find their application not only in standard mapping solutions, but also in systems which provide critical services to municipalities (Kulawiak & Lubniewski, 2014) as well as government administrative bodies (Dawidowicz & Kulawiak, 2017). Although public-sector geographic data (freely available e.g. from EU Member States under the INSPIRE Directive or from USGS resources in the USA) is generally regarded to have the highest quality and accuracy, research conducted in the city of Olsztyn has shown that the respective features in the OSM dataset are usually within 0.5 m from their cadastre equivalents. This level of accuracy enables OSM data to be effectively used e.g. for the purpose of navigating the visually impaired in urban environments (Kaminski & Bruniecki,

2012). Moreover, freely available geographic data can be integrated and disseminated through a system built solely with Open Source technologies such as OpenLayers and GeoServer. The choice of Open Source libraries enables Web-GIS data processing to be realized either on the server side (eg with the use of JTS WPS) as well as on the side of the client (e.g. using JSTS). Conducted research has shown that under appropriate conditions (which include a modern Desktop-class PC equipped with an optimized web browser, and datasets which do not exceed the size of 10 MB in general) the performance of both data processing paradigms is comparable and thus they may be used interchangeably without negatively impacting the user experience. With good quality data and flexible software solutions being freely accessible, the only component of a modern Web-GIS which still requires financial investment is the hardware. While the hardware costs of building a Web-GIS are not likely to ever disappear completely, they can be optimized depending on the system's designated work environment. The thin-client architecture assumes the use of a strong server machine together with weak terminal-level clients, while the thick-client paradigm sees the server used primarily for data storage. Depending on the circumstances, either architecture may prove to be more cost-effective.

The idea of server-side data processing was originally conceived from the efficiency point of view. It has been argued that with client-side processing, the clients sometimes need to download large amounts of data, which may result in network overload with many simultaneous client requests. (Hirschfeld, 1996). While average network bandwidth has increased tremendously since, in recent years the paradigm has shifted to optimizing power efficiency. Best practices for constructing client-server applications involve using a thin client coupled with cloud computing and virtualization in order to optimize power usage (Agrawal et al., 2014). This approach assumes that the system's clients will be using low power terminals, which require less energy due to lacking hard drives and expansion slots and using less powerful components (Davis, 2007). Thus, the thin-client architecture is best suited to use within institutions which completely operate in the Software as a Service (SaaS) model, where data is stored and processed in a private or external cloud computing system, and all employees use only low-power terminals. In such cases, the thin-client architecture may provide both better utilization of network infrastructure (due to limited exchange of information) as well as more economical use of electricity.

On the other hand, a thick-client architecture may be better suited for institutions which are obliged to store their data on internal server systems, such as national administration offices or hospitals, where the employees (who are also clients of the system) use standard Desktop PC's.

Despite the architectural improvements which cause the average power consumption of similarly-performing computers to slowly decrease (Bohr, 2014), such computers are known to use 40-60% of their maximum power draw when remaining in idle mode (Berl & De Meer, 2010) (Tiwari et al., 1998) (Wasson, 2015). Combined with the fact that the power draw rises linearly with the number of clients connected to a cloud computing system (Jung et al, 2010), and providing ample cooling to the servers rises the power requirements by an additional 25-45% (Shuja et al., 2012), implementing client-side data processing may result in increased global power efficiency due to better load balancing between clients and servers. Such an architecture may also be applied to systems whose clients are located outside of their operating institution (Dawidowicz & Kulawiak, 2017). Because the users of the system are primarily clients located outside of the hosting institution, offering cloud-based data processing would only increase the server-side operating costs without any benefits to the system operators. In this context, transferring the costs of data processing and analysis operations to the clients may result in a more optimal use of available server-side resources, e.g. for the purpose of supporting a larger number of simultaneously connected clients.

5. Conclusions

The aim of the presented work was to analyse the current state of Open Source technologies



and publicly available geographic data sources in the context of creating cost-effective Web-GIS applications for integration and processing of spatial data. The conducted analysis of freely available geospatial data sources has shown a large collection of public, commercial and crowd-sourced services which deliver raster maps of comparable quality. For the purpose of processing and analysis, vector geographic data should preferably be obtained from public sector cartographic resources. If those are not freely available, however, the OSM database has shown to be a good secondary source, with comparable coverage (particularly in urban areas) and positional accuracy suitable for many applications. As far as technology is concerned, the available Open Source GIS libraries allow for construction of fully-featured Web-GIS solutions which provide server-side as well as client-side data processing functionalities. The latter, in particular, shows new opportunities for cost optimization of Web-GIS development and deployment. The introduction of HTML5 technologies has permitted for construction of platform-independent thick clients which offer data processing performance which under the right circumstances may be close to that of server-side solutions. In this context, institutions which already use Desktop PC's for other tasks should consider implementing Web-GIS with client-side data processing, which could result in cost savings without negative impacts on the user experience.

Computer code availability

Data and code samples used to perform tests described in section 3 are available in (Kulawiak, 2019).

References

Agarwal S., Nath A. (2014) Desktop Virtualization and Green Computing Solutions. In: Babu B. et al. (eds) Proceedings of the Second International Conference on Soft Computing for Problem Solving (SocProS 2012), December 28-30, 2012. Advances in Intelligent Systems and Computing, vol 236. Springer, New Delhi

Akamai Research, 2018. The State of the Internet. Available at: <https://www.akamai.com/us/en/about/our-thinking/state-of-the-internet-report/> [Accessed on 29.01.2019]

Akella, A., Seshan, S. and Shaikh, A., 2003, October. An empirical evaluation of wide-area internet bottlenecks. In Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement (pp. 101-114). ACM.

Azzam, T. (2013). Mapping data, geographic information systems. *New Directions for Evaluation*, 140(2013), 69-84.

Batty, M., Hudson-Smith, A., Milton, R. and Crooks, A., 2010. Map mashups, Web 2.0 and the GIS revolution. *Annals of GIS*, 16(1), pp.1-13.

Bel Hadj Ali, A., & Vauglin, F. (1999). Geometric matching of polygons in GISs and assessment of geometrical quality of polygons. In Proceedings of International Symposium on Spatial Data Quality (pp. 33-43).

Berl, A. and De Meer, H., 2010, April. A virtualized energy-efficient office environment. In Proceedings of the 1st International Conference on Energy-Efficient Computing and Networking (pp. 11-20). ACM.

Bohr, M., 2014. 14 nm Process Technology: Opening New Horizons. In Intel Developer Forum (IDF'14) Available at: <https://www.intel.com.tw/content/dam/www/public/us/en/documents/technology-briefs/bohr-14nm-idf-2014-brief.pdf> [Accessed on 11.10.2017]

Brons, M., Nijkamp, P., Pels, E. and Rietveld, P., 2003. Railroad noise: economic valuation and policy. *Transportation Research Part D: Transport and Environment*, 8(3), pp.169-184.

Camboim, S. P., Bravo, J. V. M., Sluter, C. R. An Investigation into the Completeness of, and the Updates to, OpenStreetMap Data in a Heterogeneous Area in Brazil. *ISPRS International Journal of Geo-Information* 4.3 (2015): 1366-1388.

Chew, C.H., 1998. Vertical directivity pattern of train noise. *Applied Acoustics*, 55(3), pp.243-250.

Chybicki, A., Kulawiak, M., Lubniewski, Z., Dabrowski, J., Luba, M., Moszynski, M., Stepnowski, A. 2008. GIS for remote sensing, analysis and visualisation of marine pollution and other marine ecosystem components. In 2008 1st International Conference on Information Technology, IT 2008, pp. 1-4. doi: 10.1109/INFTECH.2008.4621628

Ciepluch, B., Mooney, P., Jacob, R. and Winstanley, A.C., 2009. Using openstreetmap to deliver location-based environmental information in ireland. *SIGSPATIAL Special*, 1(3), pp.17-22.

Ciepluch, B., Jacob, R., Mooney, P. and Winstanley, A.C., 2010, July. Comparison of the accuracy of OpenStreetMap for Ireland with Google Maps and Bing Maps. In *Proceedings of the Ninth International Symposium on Spatial Accuracy Assessment in Natural Resources and Environmental Sciences 20-23rd July 2010* (p. 337). University of Leicester.

CODGIK, 2017. Dane z panstwowego rejestru granic i powierzchni jednostek podzialow terytorialnych kraju - PRG. Available at: <http://www.codgik.gov.pl/index.php/darmowe-dane/prg.html> [Accessed on 01.10.2017]

Dabrowski, J., Kulawiak, M., Moszynski, M., Bruniecki, K., Kaminski, Ł., Chybicki, A., Stepnowski, A. 2009. Real-time web-based GIS for analysis, visualization and integration of marine environment data. In *Information Fusion and Geographic Information Systems*, Springer, Berlin, Heidelberg, pp. 277-288. doi: 10.1007/978-3-642-00304-2_19

Dawidowicz A, (2015) Analysis of Polish SDI within the context of needs of real estate developers, GIS Forum Croatia, Zagreb

Dawidowicz, A., Kulawiak, M. 2017. The potential of Web-GIS and geovisual analytics in the context of marine cadastre. *Survey Review*, pp. 1-12. doi:10.1080/00396265.2017.1328331

Davis, E., 2007. Green Benefits Put Thin-Client Computing Back On The Desktop Hardware Agenda. Energy. Available at: http://www.hp.com/canada/products/landing/thin_clients/files/Forrestor_-_Green_and_TCs.pdf [Accessed on 11.10.2017]

Directive 2007/2/WE of the European Parliament and Council, of March 14th 2007, establishing the infrastructure of Spatial Information in the European Community (INSPIRE) (OJ L 108, 25.4.2007, p. 1-14)

ESRI® Shapefile Technical Description, 1998. EDN: ESRI® Developer Network, Environmental Systems Research Institute, Inc., <http://www.esri.com/library/whitepapers/pdfs/shapefile.pdf>, [Accessed on 11.10.2017].

Geofabrik GmbH, 2017, OpenStreetMap Data Extracts - Pomeranian Voivodeship. Available at: <https://download.geofabrik.de/europe/poland/pomorskie.html> [Accessed on 01.10.2017].

Girres, J.F. and Touya, G., 2010. Quality assessment of the French OpenStreetMap dataset. *Transactions in GIS*, 14(4), pp.435-459.

Goodchild, M. F. (2007). Citizens as sensors: the world of volunteered geography. *GeoJournal*, 69(4), 211-221. ISO 690

Gummadi, K.P., Saroiu, S. and Gribble, S.D., 2002, November. King: Estimating latency between arbitrary internet end hosts. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurement* (pp. 5-18). ACM.

Haklay, M. and Weber, P., 2008. Openstreetmap: User-generated street maps. *IEEE Pervasive Computing*, 7(4), pp.12-18.

Haklay, M., 2010. How good is volunteered geographical information? A comparative study of OpenStreetMap and Ordnance Survey datasets. *Environment and planning B: Planning and design*, 37(4), pp.682-703

Harrtell, B. 2017. Javascript Topology Suite. Available at: <https://github.com/bjornharrtell/jsts> [Accessed on 07.10.2017]

Hirschfeld, R., 1996. Three-tier distribution architecture. *Pattern Languages of Programs (PloP)*, pp.1-4.

HELCOM, 2017. Baltic Sea Environmental Monitoring Database. Available at: <http://maps.helcom.fi> [Accessed on 06.05.2017].

Java Topology Suite, 2017. Available at: <https://github.com/locationtech/jts> [Accessed on 06.10.2017].

JTS Javadoc. 2016. JTS Topology Suite 1.15.0 Documentation. Available at: <https://locationtech.github.io/jts/javadoc/index.html> [Accessed on: 29.01.2019].

Jung, G., Hiltunen, M.A., Joshi, K.R., Schlichting, R.D. and Pu, C., 2010, June. Mistral: Dynamically managing power, performance, and adaptation cost in cloud infrastructures. In *Distributed Computing Systems (ICDCS), 2010 IEEE 30th International Conference on* (pp. 62-73). IEEE.

Kaminski, L. and Bruniecki, K., 2012. Mobile navigation system for visually impaired users in the urban environment. *Metrology and Measurement Systems*, 19(2), pp.245-256. doi:10.2478/v10178-012-0021-z

Kulawiak, M. 2019. Client-side versus server-side geographic data processing performance

comparison: Data and code. Data in Brief. *submitted*

Kulawiak, M., Chybicki, A., Moszynski, M. 2010. Web-based GIS as a tool for supporting marine research. *Marine Geodesy* Volume 33, Issue 2 & 3, pp. 135-153. doi:10.1080/01490419.2010.492280

Kulawiak, M., Wycinka, W. 2017. Dynamic signal strength mapping and analysis by means of mobile Geographic Information System. *Metrology and Measurement Systems* Volume 24, Issue 4, pp. 596-606. doi: 10.1515/mms-2017-0057

Kulawiak, M., Lubniewski, Z. 2014. SafeCity - A GIS-based tool profiled for supporting decision making in urban development and infrastructure protection. *Technological Forecasting and Social Change* Volume 89, pp. 174–187. doi: 10.1016/j.techfore.2013.08.031

Lagmay, A.M.F.A., Racoma, B.A., Aracan, K.A., Alconis-Ayco, J. and Saddi, I.L., 2017. Disseminating near-real-time hazards information and flood maps in the Philippines through Web-GIS. *Journal of Environmental Sciences*.

Masumoto, S., Nonogaki, S., Ninsawat, S., Iwamura, S., Sakurai, K., Raghavan, V., Nemoto, T. and Shiono, K., 2008. Development of prototype system for three dimensional geologic modeling based on Web-GIS. *Proc. GISIDEAS2008*, pp.83-88

Moszynski, M., Kulawiak, M., Chybicki, A., Bruniecki, K., Bielinski, T., Lubniewski, Z., Stepnowski, A. 2015. Innovative Web-Based Geographic Information System for Municipal Areas and Coastal Zone Security and Threat Monitoring Using EO Satellite Data. *Marine Geodesy* Volume 38, Issue 3, pp. 203 – 224. doi: 10.1080/01490419.2014.969459

OpenStreetMap. 2018. Featured tile layers/Updates, last modified on 17 January 2018, at 06:44, Available at: https://wiki.openstreetmap.org/wiki/Featured_tile_layers/Updates [Accessed on 20.01.2019]

Schilling, A., Lanig, S., Neis, P. and Zipf, A., 2009a. Integrating terrain surface and street network for 3D routing. *3D Geo-Information Sciences*, pp.109-126

Schilling, A., Over, M., Neubauer, S., Neis, P., Walenciak, G. and Zipf, A., 2009b. Interoperable Location Based Services for 3D cities on the Web using user generated content from OpenStreetMap. *Urban and regional data management: UDMS annual*, pp.75-84

Shuja, J., Madani, S.A., Bilal, K., Hayat, K., Khan, S.U. and Sarwar, S., 2012. Energy-efficient data centers. *Computing*, 94(12), pp.973-994.

Tiwari, V., Singh, D., Rajgopal, S., Mehta, G., Patel, R. and Baez, F., 1998, May. Reducing power in high-performance microprocessors. In *Proceedings of the 35th annual Design Automation Conference* (pp. 732-737). ACM

Tiwari, A. and Jain, D.K., 2013, November. Geospatial Framework For Dengue using Open Source Web GIS Technology. In *Joint International Workshop of ISPRS WG VIII/1 and WG IV/4 on Geospatial Data for Disaster and Risk Reduction* November (pp. 21-22).

Tostes, A.I.J., de LP Duarte-Figueiredo, F., Assunção, R., Salles, J. and Loureiro, A.A., 2013,



August. From data to knowledge: city-wide traffic flows analysis and prediction using bing maps. In Proceedings of the 2nd ACM SIGKDD International Workshop on Urban Computing (p. 12). ACM.

Wasson, S. 2015. „Intel's Core i7-6700K 'Skylake' processor reviewed”, 05.08.2015. The Tech Report. Available at: <http://techreport.com/review/28751/intel-core-i7-6700k-skylake-processor-reviewed/5> [Accessed on 18.10.2017]

Vauglin, F. 1997. Modèles statistiques des imprécisions géométriques des objets géographiques linéaires. PhD Thesis, Université Marne la Vallée.

Vanmeulebrouk, B., Lokers, R. and Bulens, J., 2008. Integration of geo-spatial web services using Adobe Flex. FOSS4G 2008

Zavala-Romero, O., Ahmed, A., Chassignet, E.P., Zavala-Hidalgo, J., Eguiarte, A.F. and Meyer-Baese, A., 2014. An open source Java web application to build self-contained web GIS sites. Environmental modelling & software, 62, pp.210-220.

