PUBLICAMUNDI

SCALABLE
REUSABLE
OPEN
GEOSPATIAL
DATA

DELIVERABLE 1.1

REQUIREMENTS AND SYSTEM ARCHITECTURE

PROJECT NUMBER: 609608

START DATE OF PROJECT: 2013/11/01

DURATION: 24 months

PublicaMundi is a research project funded by European Commission's 7th Framework Programme.

Dissemination Level	Public
Due Date of Deliverable	Month 4, 28/02/2014
Actual Submission Date	28/02/2014
Work Package	WP1, Architecture and Integration
Task	T 1.1, T1.2
Type	Report
Approval Status	Submitted for approval
Version	1.0
Number of Pages	122
Filename	D1.1 Requirements and System Architecture.pdf

ABSTRACT

This document describes the user requirements and system architecture of PublicaMundi, a system which will support the entire life-cycle of open geospatial data. First, a brief introduction of the PublicaMundi project is provided. In the following, we present the processes applied (interactions, interviews and surveys) in order to collect user requirements from the open data community and elaborate on our findings. PublicaMundi's architecture is described in detail, in terms of software components and modules. Finally, we present the open source geospatial software components that will be developed, extended and used within PublicaMundi. This document will be updated throughout the duration of the project to reflect the actual state of PublicaMundi's implementation.

HISTORY

version	date	reason	revised by
0.1	15 Jan 2014	Initial Draft	Angelos Tzotsos
0.2	31 Jan 2014	Improvements on draft	Alan Beccati, Michalis Alexakis, Yannis Kouvaras, Gerald Fenoy, Thodoris Stratiotis, Georgia Papadaki, Simos Kamilieris,
0.3	14 Feb 2014	Version prepared for internal peer-review	Thodoris Stratiotis, Michalis Alexakis, Yiannis Kouvaras, Angelos Tzotsos.
0.5	21 Feb 2014	Peer Review	Spiros Athanasiou
1.0	27 Feb 2014	Final Version	Thodoris Stratiotis, Michalis Alexakis, Yiannis Kouvaras, Angelos Tzotsos

AUTHOR LIST

organization	name	contact
IMIS	Michalis Alexakis	alexakis@imis.athena-innovation.gr
IMIS	Spiros Athanasiou	spathan@imis.athena-innovation.gr
Rasdaman	Alan Beccati	beccati@rasdaman.com
Geolabs	Gerald Fenoy	gerald.fenoy@geolabs.fr
GET	Simos Kamilieris	skamilieris@getmap.gr
IMIS	Yiannis Kouvaras	jkouvar@imis.athena-innovation.gr
GET	Georgia Papadaki	gpapadaki@getmap.gr
IMIS	Thodoris Stratiotis	stratiot@imis.athena-innovation.gr
IMIS	Angelos Tzotsos	tzotsos@imis.athena-innovation.gr

EXECUTIVE SUMMARY

This document describes the user requirements and system architecture of PublicaMundi, a system which will support the entire life-cycle of open geospatial data. First, a brief introduction of the PublicaMundi project is provided. In the following, we present the processes applied (interactions, interviews and surveys) in order to collect user requirements from the open data community and elaborate on our findings. Our focus is on highlighting and documenting the challenges data publishers, end users, and developers encounter, across the life-cycle of geospatial data.

The user requirements are reported based on consolidated results of surveys and personal interviews, performed to more than 70 end users and stakeholders of the open data community. The user requirements are categorized into three levels: end user, publishers/catalogue owners, and developers.

In the following, we outline the software architecture for the PublicaMundi system. PublicaMundi will be based on the CKAN open data catalogue, heavily extended with inherent improvements (*CKAN core*), new software extensions (CKAN *plugins*), and integration with open source geospatial software. The system will support the entire lifecycle of open geospatial data: from harvesting and data publishing, to scalable data APIs for integration in value added services.

The layout of the document is the following:

- *Introduction*, which introduces the System Requirements and Architecture Document (SRAD) for PublicaMundi to its readers.
- *System Requirements*, which provides the use cases and the requested features, as compiled through WP1 and more specifically the surveys and interviews of the domain experts.
- *System Architecture*, which provides a high level description of the PublicaMundi system.
- *Component Architecture*, which documents the major software components of PublicaMundi, their responsibilities, and their relationships.
- *Tool Stack*, which provides an overview of the Free and Open Source Software (FOSS) stack that will be developed, used and maintained as part of this project in order to implement parts of the proposed architecture.

- *Conclusions*, which summarizes the document and
- *References*, which enumerates external resources needed to compile this document.

This document will be updated throughout the duration of the project to reflect the actual state of PublicaMundi's implementation.

ABBREVIATIONS AND ACRONYMS

OGC	Open Geospatial Consortium
WMS	Web Map Service
WFS	Web Feature Service
CSW	Catalogue Service for Web
WPS	Web Processing Service
WCS	Web Coverage Service
WCPS	Web Coverage Processing Service
SQL	Structured Query Language
API	Application Programming Interface
SDI	Spatial Data Infrastructure
FOSS	Free and Open Source Software
OSGeo	Open Source Geospatial Foundation
CRS	Coordinate Reference System
GIS	Geographical Information Systems

TABLE OF CONTENTS

TABLE OF FIGURES

1 INTRODUCTION

This document outlines the user requirements and the component architecture of the PublicaMundi system. In the following sections, the PublicaMundi system is presented in detail.

The goal of PublicaMundi is to research and develop methodologies, as well as scalable, reusable tools and technologies to ***facilitate the publication, discovery and reuse of open geospatial data*** by re-purposing and leveraging open standards and technologies of the geospatial domain.

For this purpose we will extend the CKAN open data catalogue to fully support the publishing, curation and management lifecycle of geospatial data. The system will integrate tools enabling the interlinking of geospatial data and provide multilinguality support in a cross-boundary context. Our goal is to provide scalable technologies and services to create and reuse on-demand maps from open geospatial data. PublicaMundi also aims to provide analytics services to accurately monitor the usage of open geospatial data. Finally, the project will develop scalable technologies and reusable data APIs supporting querying, processing, and analysis of open geospatial data.

1.1 REQUIREMENTS AND ARCHITECTURE DOCUMENT OBJECTIVES

This document has the following objectives:

- To present and analyze the use cases and system requirements of the system.
- To collect, present and adapt to the architecture, all features that have been requested by the interviews and surveys of data publishers, end-users and developers.
- To document the software components of the developed system in a way that can be further analyzed into modules and sub-systems.
- To document the logic behind the important decisions made during the brain-storming phase of the first months of the project and build upon the ideas presented by the Consortium.
- To act complimentary to the Document of Work (DOW) and help reviewers to follow the implementation procedures and status.

1.2 INTENDED AUDIENCES

This document is intended for the following audiences:

- Project Management
- EU Reviewers
- PublicaMundi users
- PublicaMundi developers including:
    - Software Engineers, who will follow the requirements and features proposed to meet the architecture as documented in this document.
    - UI Designers, whose design must conform to the architecture documented.
    - QA Engineers/Testers, which must validate the architecture documented in this document.

1.3 REFERENCES

This document references to the following documents:

- OGC Standard Specifications [OGC]:
    - OGC CSW 2.0.2 [CSW]
    - OGC WPS 1.0 [WPS]
    - OGC WCS 2.0 [WCS]
    - OGC WMS 1.3 [WMS]
    - OGC WCPS 1.0 [WCPS]
- Project Documents:
    - Description of Work, which documents the project deliverables as well as a high-level overview of PublicaMundi's components and features.

1.4 DOCUMENT OVERVIEW

This document is organized into the following chapters:

- *Introduction* (Chapter 1), introduces the System Requirements and Architecture Document for PublicaMundi to its readers.
- *System Requirements* (Chapter 2), provides the use cases, as compiled through WP1 and more specifically Task 1.1. During this task, three surveys were created and distributed to selected users and

geospatial organization members, as well as decision makers from the private and public sector. This process included personal web surveys targeted to data publishers and open web surveys for any interested publisher/producer/user. At the same time, interview sessions were performed with several domain experts to gather information about the challenges of open geospatial data publishing, as well as to obtain information about requested features for PublicaMundi.

- *System Architecture* (Chapter 3), provides a high level description of the PublicaMundi system. An overview of the PublicaMundi system is provided, as well as a description of the sub-components and their correspondence to the project's work packages. In this chapter, our work in Task 1.2 is presented and the initial plan for the system integration (Task 1.3) is described.
- *Component Architecture* (Chapter 4), documents the major hardware and software components of PublicaMundi, their responsibilities, and their relationships. This chapter describes the proposed system architecture. The system's sub-components are analyzed in detail and their role, relationships and interfaces between them are explained, in order for system interactions to be determined. Since PublicaMundi is heavily based on CKAN and extending it in core functionality, throughout this document the CKAN terminology and API references are preserved. At the same time, since the project is aiming towards the full geospatial extension of CKAN, the sub-components are following well defined geospatial standards and their implementation through Free and Open Source Geospatial Software (FOSS4G) is presented.
- *Tool Stack* (Chapter 5), provides an overview of the FOSS stack that will be developed, used and maintained as part of this project in order to implement parts of the proposed architecture. The project's contributions to OSGeo and its community are presented in detail.
- *Conclusions* are presented in Chapter 6.
- *References* are given in Chapter 7.

2 USER REQUIREMENTS

User requirements for PublicaMundi have been collected through these channels:

- Interviews with prominent publishers of geospatial data and open data. These included national, EU and international stakeholders. In total representatives from 16 publishers were interviewed.
- Web surveys targeted to publishers and consumers of geospatial data. These included national, EU and international stakeholders.
    - Invitational web surveys were addressed to selected publishers of geospatial data. In total, we received 12 responses from this category.
    - Open participation in the web surveys was possible through our web site and disseminated through various channels. In total we received 59 responses from this category.

2.1 INTERVIEWS

2.1.1. Profile of participants

We have targeted prominent publishers of geospatial data and open data, and in particular representatives from:

- Cadastral and mapping agencies, with publishing activities in the context of Open Data policies and/or the INSPIRE Directive [INSP07]
- Governmental bodies, with open data publishing activities in the context on Open Data policies
- SMEs and web entrepreneurs interested in reusing open geospatial data for value added services
- NGOs active in promoting Open Data Policies and open data publishing
- Members of the Open Data community. We focused on active members of local, national, and international community efforts for publishing open data.
- Members of the Open Source Community. We focused on developers and ICT integrators of open source projects for open data publishing and geospatial data management.

2.1.2. Methodology

The methodology followed for organizing and conducting the interviews was:

- A list of potential candidates was jointly established by members of the Consortium and interviews were assigned to select Consortium partners.
- The candidates were contacted by email/phone in order to schedule the interview and provide background information. The candidates were requested to reserve at least 1 hour of their time. Further, the candidates were informed that their responses were to be kept *anonymous and confidential* in order to ensure *an open discussion*.
- A scenario was created to serve as a guideline for the interview (provided in Appendix I). This included major discussion points needed to extract user requirements. The interviewers were instructed to follow the scenario, making sure all topics were discussed, but also explore additional points, remarks and issues raised by the participants.
- The interviews were performed either by face-to-face meetings, or through tele-conferencing (Skype, Google Hangout)
- During the interview, minutes were kept by the responsible partner. These were analyzed to extract the user requirements presented in the following section.
- We have originally organized and performed 12 interviews from 3/12/2013 up to 28/1/2014. Another 4 interviews were conducted on 30/1/2014 during a Workshop organized by the Ministry of Public Reform and eGovernance, in which PublicaMundi was invited and represented by its Coordinator.

2.1.3. Summary of Results

The issues, points and ideas produced during the interviews were analyzed and integrated in the System Requirements section of this Deliverable (§ 2.3). In the following we provide a high-level overview of major topics raised during the interviews, which are grouped into the following areas:

- Data catalogues and geospatial data (vector/raster data, metadata)
- Services: search, download, visualization, web mapping
- Analytics, multilinguality, interlinking
- Recommendations for PublicaMundi

A few general remarks follow:

- All the interviewees were members of bodies, agencies, organizations, enterprises and communities/foundations which are involved in the field of open geospatial data publishing either as one of their main everyday activities, or as part of specific projects.
- Some of the interviewees represented organizations which publish their own geospatial data; this means data which have been produced by the organizations themselves and can be available to the public through their catalogues. In some cases, geospatial data publishing consists part of Spatial Data Infrastructures (SDIs) which have been implemented in accordance to the provisions of National Laws which harmonize the INSPIRE Directive (2007/2/EC) [INSP07].
- All the interviewees were familiar with EU and national efforts regarding open data publishing. However, support for these efforts is evaluated as *low* among public sector stakeholders with existing revenue streams from marketing data products. For historically established producers of geospatial data (e.g. Cadastral and Mapping Agencies), the availability of their data with an open license is considered a policy/business issue. In contrast, for other data producers the major issues are technical.

2.1.3.1. Data catalogues and geospatial data (vector/raster data, metadata)

For the few of the interviewees who use data catalogues and are familiar to CKAN, it is considered important to collaborate with current projects in order to enhance CKAN's spatial capabilities. The most requested features of CKAN-spatial concern Data Visualization and Web Services consumption (WMS, GeoJSON, etc.). Based on the discussion with CKAN experts, the CKAN core model is not to change until CKAN 3.0 and a new plugin system will be implemented for data viewers. Since CKAN is one of the main software which will be used for the development of PublicaMundi, it would be beneficial to integrate extra accessibility tools for CKAN UI. It was also recommended that catalogs should support more types of data. Moreover, harvesting external CSW and CSW 2.0/3.0 requires improvements for the existing catalogue services.

Overall, CKAN adoption from the geospatial community is still low, since in most cases repurposed GIS-related software is used (e.g. GeoNetwork) to provide a catalogue with their data. Interviewees who are not familiar to CKAN are interested to be informed and to use its spatial extensions for geospatial data publishing, Regarding URI patterns for data these are not followed in the majority of cases.

Vector data are more frequently published than raster data. Raster data usually have issues with their size, cost/benefit barriers, technical concerns and their potential use. In cases where raster data are available, simple data transfer protocols are used (ftp).

It is preferred by all the interviewees that metadata and data follow standardized schemata (e.g. according to the INSPIRE Directive). However schemas for geospatial data are mostly proprietary so in many cases conformance to the INSPIRE Directive is extremely low.

Significant effort is required from users who are not familiar with the data in order to understand its structure, semantics and purpose of use. For metadata, a metadata editor with core subset of ISO, use of code-lists and capacity to keep free text to minimum, are some of the indicative features which seem to be still problematic in the geospatial data publishing process. Regarding standards, it is considered as very important feature to support ISO 19115 XML, INSPIRE and OGC CSW. For the time being, OGC standards are used mainly from established geospatial data producers whereas data publishers with no relevant background have limited experience.

2.1.3.2. Services: search, download, visualization, web mapping

Most data publishers do not provide visualization services for their data. In the few cases where visualizations are available, these are mostly specific-purpose (e.g. for a single dataset/application) or have been developed in the context of another project/initiative (e.g. Census data).

Although web mapping is considered to be one of the most important services for geospatial data, there are still challenges regarding web mapping capabilities, reusable map building APIs and automatic procedures. Most of the interviewees have explored possible solutions and rejected them mainly due to the lack of “knowhow” and expected high costs. Also, where OGC services are already available, their adoption is very low beyond the core GIS community (mainstream developers). Further, where CKAN is used, the data APIs are still considered highly problematic and are avoided by developers.

It is commonly accepted that published data are used for third parties in applications and services. All data publishers are familiar with specific use cases and commercial applications where their data are used. However, the known cases only cover a small fraction of the actual use.

Availability of geospatial data for download is considered as insufficient since most users and developers require actual maps and services to understand and use the data respectively. Nowadays, most data publishers

do not provide interactive maps for geospatial data due to cost/infrastructure concerns. In cases where interactive maps are available, these are Google Maps-based or offered from an existing SDI.

On the service provision level, current web platforms for geospatial data provide OGC services, more often WMS and WFS, and less often WCS, CSW and WPS. As the stated in the interviews, several improvements should be implemented in order to offer users with easier data search. Some of the proposed necessary improvements are: stricter compliance to standardized OGC services, implementation of improved architecture, need for client tools (web or desktop), plugins (e.g. INSPIRE QGIS plugin).

Analysis services are typically not provided due to cost/infrastructure constraints. There are specific concerns regarding the scalability of such attempts, the effort required to sustain a high level of service, and the unknown reception from users. The consensus is that currently experts prefer desktop tools for analysis and WPS for simple processing.

2.1.3.3. Analytics, multilinguality, interlinking

Some of the interviewees use in-house analytics to monitor how their data is used. However, in the majority of cases, available analytics are at best provided by Google Analytics (or a similar service). For web mapping applications, tile-level and OGC-service requests are kept in system logs, but are typically not analyzed.

In most cases, catalogues, data and metadata are not available in other languages. Despite the fact that it is not a priority, it is believed that it is necessary to extend multilinguality in order to facilitate data searching. In some cases multilinguality has been partially achieved with the use of extra field in the underlying database or through a third party translator facility. Requests for help in understanding the content/purpose of published data and/or translated versions from non-native speakers are more frequent in the past years. However this trend is not enough at this point in time to drive concrete actions and policies for multilinguality.

Regarding interlinking with other data, it is only performed during data production/curation and is based on standardized national/international vocabularies. The issue of inconsistencies between different datasets remains a factor which hinders the process of publishing and discovering geospatial data. There are inconsistencies even in data originating from the same producer. In most cases errors result from proprietary vocabularies and different reference data (e.g. addresses, place names, administrative boundaries). The majority considers interlinking with semantic web technologies and the use of vocabularies and ontology as positive options.

However, the additional technical complexity is not currently balanced with suitable benefits. Based on the discussion with interviewees who are involved in CKAN maintenance and upgrade, currently interlinking is only provided in CKAN's Datastorer through simple linking in SQL API, but soon there will be improvements. Another issue raised from the interviews is that interest for applying Linked Data technologies is extremely low and the additional.

In the process of geospatial data publishing, limited measures are taken for improving the quality of data. Other than highlighting potential problems in the metadata (or in FAQs), no other measures are taken due to the associated costs. Towards this, distributed ETL/cleansing, or heavily curated crowd-sourced cleansing is favorably considered.

2.1.3.4. Recommendations for PublicaMundi

All the interviewees believe that PublicaMundi will be very useful for providing scalable, reusable open geospatial data. Overall, they proposed a number of interesting ideas, recommendations and features:

- Collaboration with developer teams of tools used in PublicaMundi development (e.g. CKAN) in order to be informed for the effort and new features implemented, and to avoid duplication of work.
- User-friendly Interface for publishers and users.
- Data preview, data citation, preservation for long time access, peer review in order to capture required processes for data publishing.
- Indicators distinguishing which data are complete and which not.
- Tools to search data from the desktop (open source and proprietary GIS) in order to bring open data to the desktop of each user.
- Footprints on maps and support complex spatial queries.
- Advanced spatial searches and spatial types.
- Catalogues supporting more types of data.
- Integration with OpenLayers.
- Additional interfaces to catalogue, e.g. CSW, OpenSearch etc.
- Linking data with (DOI) Digital Object Identifiers
- PublicaMundi as a web platform for collaboration on future data collection (crowdsourcing “marketplace”)

2.2 SURVEYS

The purpose of the surveys was to solicit feedback from stakeholders involved in the entire lifecycle of open geospatial data and guide us towards establishing the system requirements, architecture and services provided by PublicaMundi.


PublicaMundi - User Survey

PublicaMundi is an FP7 research project aiming to make open geospatial data easier to publish, view, and reuse. Regardless of how you are involved in the open data life-cycle, we would like to hear your thoughts!

Your responses, ideas, and insights can help us devise faster, more efficient ways to use and reuse open geospatial data.

You shouldn't need more than 10min to complete the survey. Thank you for your time!

PUBLICA MUNDI

Continue »

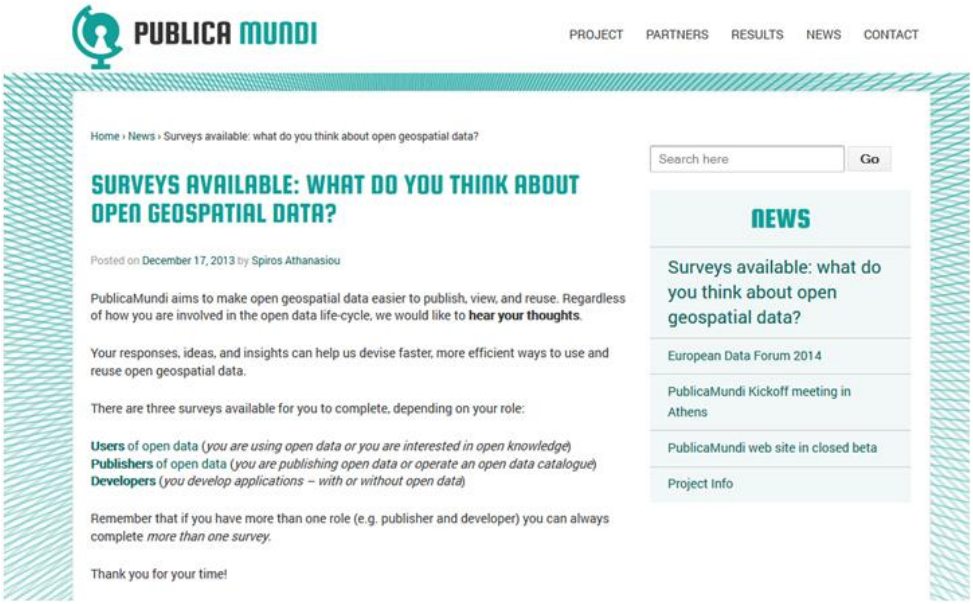
25% completed

Powered by Google Drive

This content is neither created nor endorsed by Google.

Report Abuse - Terms of Service - Additional Terms

Figure 1: Screenshot from the first page of the User Survey


PUBLICA MUNDI

PROJECT PARTNERS RESULTS NEWS CONTACT

Home › News › Surveys available: what do you think about open geospatial data?

SURVEYS AVAILABLE: WHAT DO YOU THINK ABOUT OPEN GEOSPATIAL DATA?

Posted on December 17, 2013 by Spiros Athanasiou

PublicaMundi aims to make open geospatial data easier to publish, view, and reuse. Regardless of how you are involved in the open data life-cycle, we would like to **hear your thoughts**.

Your responses, ideas, and insights can help us devise faster, more efficient ways to use and reuse open geospatial data.

There are three surveys available for you to complete, depending on your role:

Users of open data (*you are using open data or you are interested in open knowledge*)
Publishers of open data (*you are publishing open data or operate an open data catalogue*)
Developers (*you develop applications – with or without open data*)

Remember that if you have more than one role (e.g. publisher and developer) you can always complete *more than one survey*.

Thank you for your time!

Search here Go

NEWS

Surveys available: what do you think about open geospatial data?

European Data Forum 2014

PublicaMundi Kickoff meeting in Athens

PublicaMundi web site in closed beta

Project Info

Figure 2: Screenshot from the article in the project's website announcing the surveys

2.2.1. Survey Design

The surveys aim to capture use cases and requirements from users outside the PublicaMundi consortium. Given the highly diverse skills and expertise

matrix of stakeholders involved in open data publishing, we prepared three (3) surveys, each one targeting a specific category of stakeholders involved in the open data lifecycle. The three surveys prepared target **Users**, **Publishers** and **Developers**.

The three surveys include a set of common questions, with the following questions focusing on the specific stakeholder needs, focus and expertise.

2.2.1.1. Common section

All stakeholders, regardless their category, are asked a few common questions, which is presented in Table 1.

No	Question	Type
Q1	How are you involved in the open data life-cycle?	Nominal, Multiple Choice
Q2	Please rate the importance of open geospatial compared to other open data	5-Scale Choice
Q3	Please rate your technical skills regarding geospatial data and Geospatial Information Systems	5-Scale Choice

Table 1: Common section questions

2.2.1.2. Users

Aiming to identify the users' relation with open geospatial data and their corresponding needs, a targeted set of questions has been prepared, presented in Table 2.

No	Question	Type
UQ4	Please rate how easy you find searching for open geospatial data	5-Scale Choice
UQ5	What are the major problems you encounter when searching for open geospatial data	Nominal, Multiple Choice
UQ6	Please provide some ideas on how searching for open data geospatial data could be improved	Free text, Open
UQ7	Please rate how easy you find using downloaded open geospatial data	5-Scale Choice
UQ8	For what purpose are you downloading open geospatial data?	Nominal, Multiple Choice
UQ9	In what file format do you typically download open geospatial data?	Nominal, Multiple Choice
UQ10	What would be your preferred file format for downloading open geospatial data?	Nominal, Multiple Choice
UQ11	After you have downloaded an open geospatial data set, what are the most common problems you encounter?	Nominal, Multiple Choice
UQ12	After you have downloaded an open geospatial data set,	Nominal, Multiple

No	Question	Type
	what is the typical way you use it?	Choice
UQ13	Open geospatial data should be available in all possible file formats. Do you agree?	5-Scale Choice
UQ14	Open geospatial data should be available in all possible Coordinate Reference Systems. Do you agree?	5-Scale Choice
UQ15	Open geospatial data should be available in various languages. Do you agree?	5-Scale Choice
UQ16	Open geospatial data should be available in interactive maps. Do you agree?	5-Scale Choice
UQ17	I should be able to upload my own data and create custom maps. Do you agree?	5-Scale Choice
UQ18	According to your estimates, what is the cost for creating a data set containing the entire road network for Athens, Greece?	Ordinal, Single Choice

Table 2: Users' section questions

2.2.1.3. Publishers

In order to receive the data publishers' experiences and the state of the art, a special set of questions was formed, which is presented in Table 3.

No	Question	Type
PQ4	How easy is it for you to publish an open geospatial data set?	5-Scale Choice
PQ5	What is the average time for publishing an open geospatial data set? (please take into account the entire workflow you currently follow, e.g. extract from a geospatial database, transform, create metadata, create a catalogue entry, loading them to interactive maps, publish)	Nominal, Multiple Choice
PQ6	What are the major problems you typically encounter?	Nominal, Multiple Choice
PQ7	Can you give us an example of a data set that was extremely time-consuming, and why?	Free text, Open
PQ8	How easy is it for you to create metadata for a geospatial data set?	5-Scale Choice
PQ9	Do you provide metadata for geospatial data following standard schemas? If yes which?	Nominal, Multiple Choice
PQ10	How easy is it for you to transform the data in other formats?	5-Scale Choice
PQ11	How easy is it for you to transform the data in other Coordinate Reference Systems?	5-Scale Choice
PQ12	What types of geospatial data do you publish?	Nominal, Multiple Choice
PQ13	If you do not publish raster data, what are the main reasons?	Nominal, Multiple Choice

No	Question	Type
PQ14	How many of the open geospatial data you provide are available in interactive maps?	Ordinal, Single Choice
PQ15	If not all of your open geospatial data are available in interactive maps, what are the main reasons?	Nominal, Multiple Choice
PQ16	What APIs do you provide for your geospatial data?	Nominal, Multiple Choice
PQ17	If you do not provide any APIs, what are the main reasons?	Nominal, Multiple Choice
PQ18	What of the following analytics do you keep?	Nominal, Multiple Choice
PQ19	Open geospatial data should be available in all possible file formats. Do you agree?	5-Scale Choice
PQ20	Open geospatial data should be available in all possible Coordinate Reference Systems. Do you agree?	5-Scale Choice
PQ21	Open geospatial data should be available in various languages. Do you agree?	5-Scale Choice
PQ22	Open geospatial data should be available in interactive maps. Do you agree?	5-Scale Choice
PQ23	Open geospatial data should be provided to developers through open APIs. Do you agree?	5-Scale Choice

Table 3: Publishers' section questions

2.2.1.4. Developers

Finally, pursuing to assess the developers' skills, their familiarity with geospatial data processing and their needs in relation with the development of applications which consume open geospatial data, a special set of questions only for developers was prepared. The questions targeted to developers are presented in Table 4.

No	Question	Type
DQ4	For which platforms do you develop applications?	Nominal, Multiple Choice
DQ5	What is your level of expertise regarding the use of web mapping frameworks?	5-Scale Choice
DQ6	Please rate your familiarity with this task: Insert a shapefile into a geospatial database	5-Scale Choice
DQ7	Please rate your familiarity with this task: Write geospatial SQL queries	5-Scale Choice
DQ8	Please rate your familiarity with this task: Perform spatial analysis in a desktop GIS	5-Scale Choice
DQ9	Please rate your familiarity with this task: Transform a geospatial data set in another format	5-Scale Choice

No	Question	Type
DQ10	Please rate your familiarity with this task: Transform a geospatial data set in another Coordinate Reference System	5-Scale Choice
DQ11	Please rate your familiarity with this task: Create a Google Maps mashup	5-Scale Choice
DQ12	Please rate your familiarity with this task: Install a map server	5-Scale Choice
DQ13	Consume OGC services (e.g. Web Feature Service)	5-Scale Choice
DQ14	Have you used open data in your applications?	Nominal, Yes/No Choice
DQ15	What type of open data would do you use, or you are currently using, in your applications?	Nominal, Multiple Choice
DQ16	Have you encountered any problems when using open data?	Free text, Open
DQ17	Do you use maps in your applications?	Nominal, Yes/No Choice
DQ18	If no, why?	Nominal, Multiple Choice
DQ19	If yes, what map tiles are you using?	Nominal, Multiple Choice
DQ20	What is the map server for the applications you currently develop?	Nominal, Multiple Choice
DQ21	If you had free access to an API providing querying and analysis services for open geospatial data would you use it?	5-Scale Choice
DQ22	What kind of a data API would you prefer?	Nominal, Multiple Choice
DQ23	What of the following data API capabilities would you use?	Nominal, Multiple Choice
DQ24	Any other potential capabilities for the data API that you would like?	Free text, Open
DQ25	Regarding the data API, please rate the importance of the following	Categorized, 3-Scale Choice
DQ26	If you had free access to a web mapping framework for developing interactive maps, what would be you use it?	5-Scale Choice
DQ27	What kind of a web mapping framework would you prefer?	Nominal, Multiple Choice
DQ28	What of the following web mapping framework capabilities would you use?	Nominal, Multiple Choice
DQ29	Any other potential capabilities for the web mapping framework that you would like?	Free text, Open
DQ30	Regarding the web mapping framework, please rate the importance of the following	Categorized, 3-Scale Choice

No	Question	Type
DQ31	Open geospatial data should be available in all possible file formats. Do you agree?	5-Scale Choice
DQ32	Open geospatial data should be available in all possible Coordinate Reference Systems. Do you agree?	5-Scale Choice
DQ33	Open geospatial data should be available in various languages. Do you agree?	5-Scale Choice
DQ34	Open geospatial data should be available in interactive maps. Do you agree?	5-Scale Choice
DQ35	Open geospatial data should be provided to developers through open APIs. Do you agree?	5-Scale Choice
DQ36	Final question! If you had to ask for ONE feature to be developed from PublicaMundi, regardless of time, budget and current technology, what would that be?	Free text, Open

Table 4: Developers' section questions

2.2.2. Survey Creation and Distribution

For the survey creation, Google Forms was used. Google Forms allows collecting information from users via a personalized survey or quiz in an easy, streamlined way. A Google Form can be connected with a Google spreadsheet and responses can be automatically sent to the spreadsheet.

The survey was disseminated to more than 1,000 recipients through email lists. The majority of the recipients are members of the free and open source software, open data and geospatial data communities and members of the academia. Participants in other related EU projects were notified as well. Furthermore, a related news article was published in the project's website and it was communicated through twitter. More details can be found on Table 5.

Communication Channel	Detail	
Mailing List	OSGeo-opendata (https://lists.osgeo.org/mailman/listinfo/geodata) OSM-talk (https://lists.openstreetmap.org/listinfo/talk) CKAN-users (http://lists.okfn.org/mailman/listinfo/ckan-discuss) ordnancesurvey.co.uk ellak.gr	geodatagov.gr dbhellas.gr rasdaman community geoknow.eu PublicaMundi.eu
Personal e-mail to key experts from	okfn	

various organizations	data.gov (USGS and NOAA)
News article	A news article was published in the project's website announcing the release of the surveys (http://publicamundi.eu/?p=273)
twitter	PublicaMundi's twitter account (@PublicaMundi) was used to communicate the release of the surveys

Table 5: Communication channels used to promote the web surveys

2.2.3. Analysis of responses

Surveys went public on 17/12/2013. In this document we analyze responses provided up to 21/01/2014. During this period we had 71 responses, 12 of them coming from data publishers, 28 from developers and 31 from geospatial data users. The feedback was analyzed and integrated in the System Requirements section of this Deliverable (§2.3). We will keep the surveys open to receive more responses throughout the duration of the project. In the following we provide an analysis of the responses received for each type of survey.

2.2.3.1. Common section

2.2.3.1.1. How are you involved in the open data life-cycle?

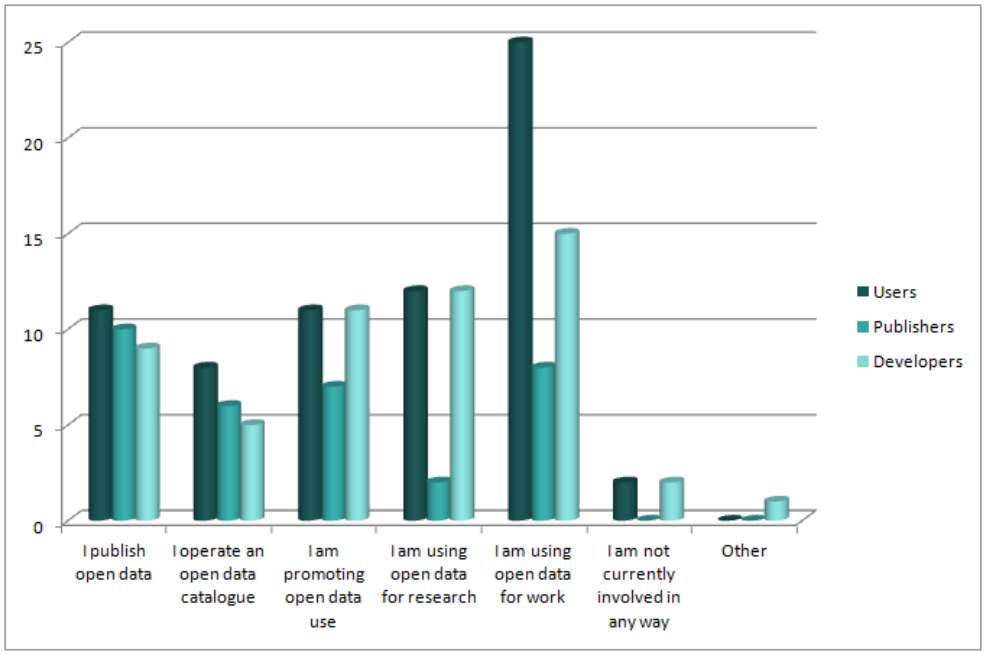


Figure 3: Chart of response distribution on question "How are you involved in the open data life-cycle?"

As Figure 3 indicates the majority of the respondents are knowledgeable on open data. Besides the data producers, most participants appear as

advocates of open data either promoting them or using them for work/research.

Other Answers:

- Open data are used for teaching

2.2.3.1.2. Please rate the importance of open geospatial compared to other open data

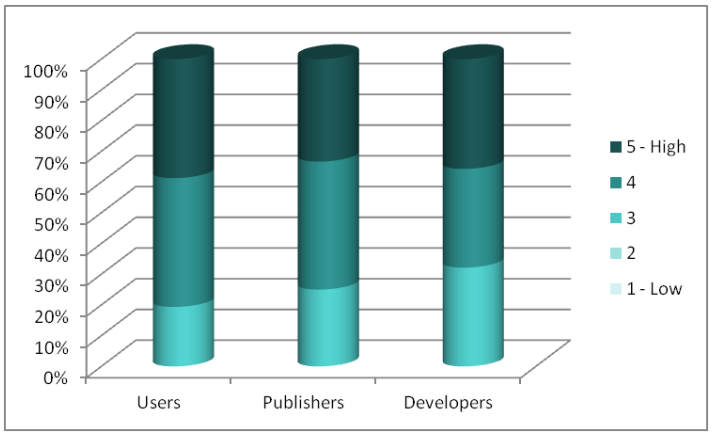


Figure 4: Chart of response distribution on question "Please rate the importance of open geospatial compared to other open data"

As depicted, open geospatial data is considered as one of the most important kinds of open data by the majority of the respondents. All replies are between medium [3] and high importance [5].

2.2.3.1.3. Please rate your technical skills regarding geospatial data and Geospatial Information Systems

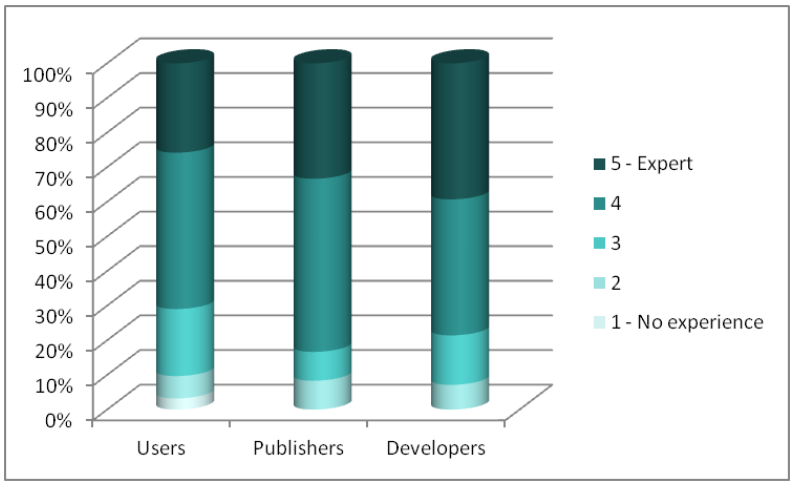


Figure 5: Chart of response distribution on question "Please rate your technical skills regarding geospatial data and Geospatial Information Systems"

The majority of the respondents are *self-assessed* as familiar with geospatial data and GIS technologies. It is important to highlight the increased number of No/limited experience responses for Users and Developers.

2.2.3.2. Users

2.2.3.2.1. Please rate how easy you find searching for open geospatial data

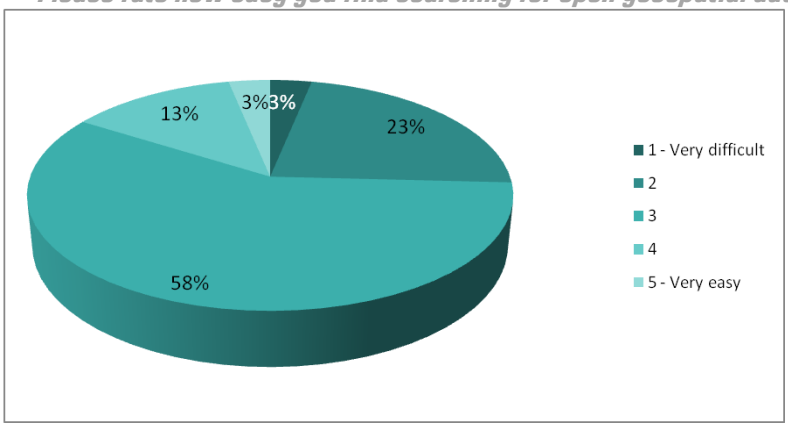


Figure 6: Chart of response distribution on question "Please rate how easy you find searching for open geospatial data"

Searching for open geospatial data appears to be challenging for most users. Only 16% of the users consider it "easy" [4] or "very easy" [5], while another 26% considers it "difficult" [2] and "very difficult" [1]. The vast majority of the respondents consider it a task of "medium difficulty" [3].

2.2.3.2.2. What are the major problems you encounter when searching for open geospatial data

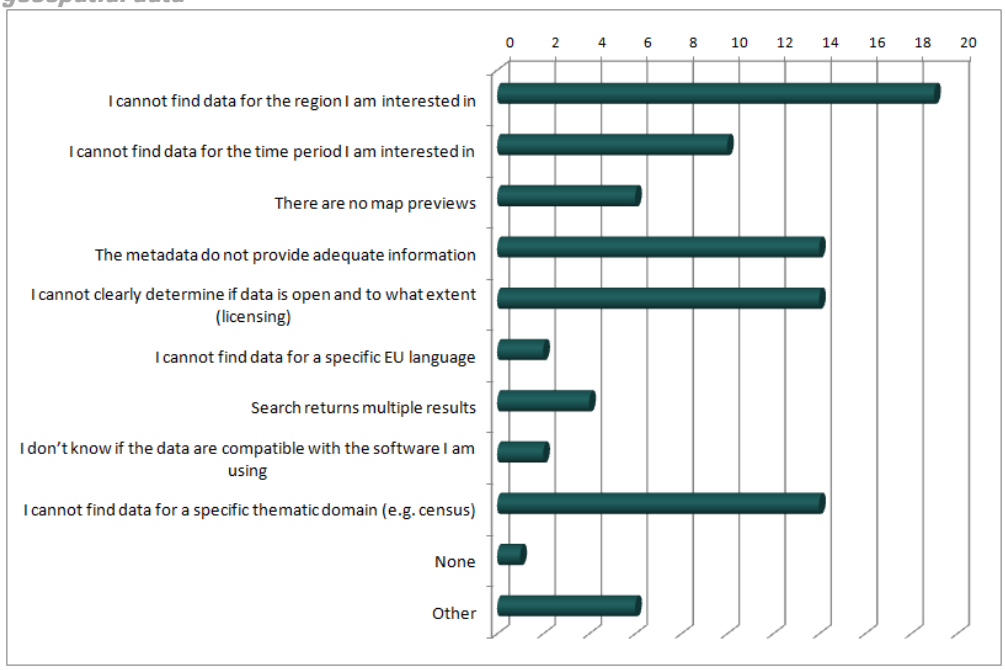


Figure 7: Chart of response distribution on question "What are the major problems you encounter when searching for open geospatial data"

The most usual problems are "*difficulty to find data for the region of interest*", "*poor metadata*", "*uncertainty regarding their compatibility with the software used*" and "*difficulty to find data for the thematic domain of interest*".

Other Answers:

- Metadata don't link to data
- Existing data are not open or freely available (2)
- Available raster data do not have enough resolution for areas of interest or they are not free (2)
- Searching is complicated

2.2.3.2.3. *Please provide some ideas on how searching for open data geospatial data could be improved*

Some of the most prominent ideas proposed were the following:

- Better thematic catalogues and search keywords
- Follow the OSM paradigm
- Rigid data classification
- Use of better metadata standards
- Spatial relevance ranking

2.2.3.2.4. *Please rate how easy you find using downloaded open geospatial data*

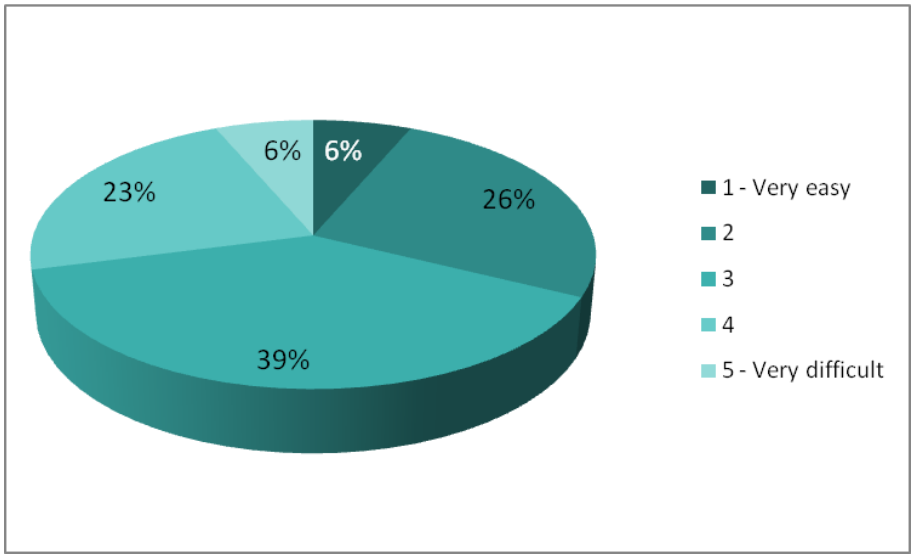


Figure 8: Chart of response distribution on question "Please rate how easy you find using downloaded open geospatial data"

The majority of the responses indicate that using downloaded open geospatial data is a task of "medium difficulty" [3].

2.2.3.2.5. For what purpose are you downloading open geospatial data?

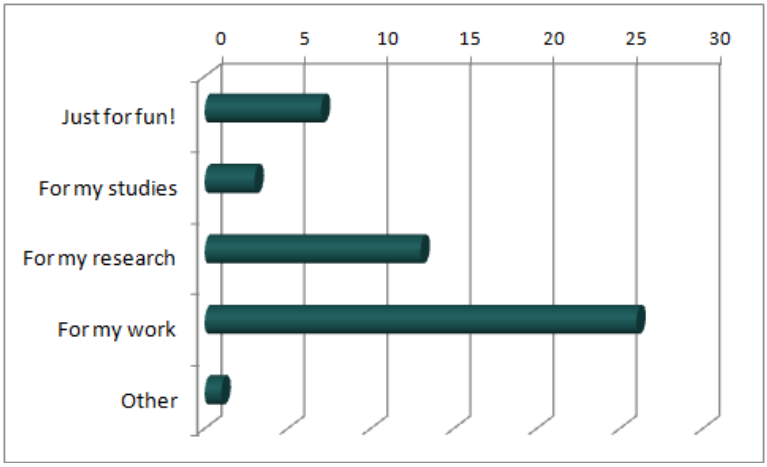


Figure 9: Chart of response distribution on question "For what purpose are you downloading open geospatial data?"

The great majority of the respondents use open geospatial data for professional reasons ("*For my work*").

Other Answers:

- Use for teaching

2.2.3.2.6. In what file format do you typically download open geospatial data?

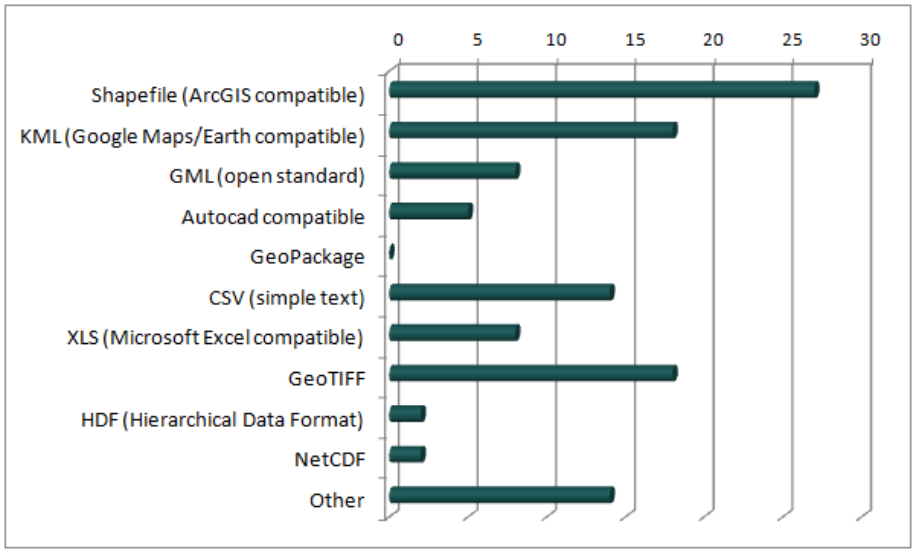


Figure 10: Chart of response distribution on question "In what file format do you typically download open geospatial data?"

Users are accustomed to downloading open geospatial data in various file formats. Most popular choices though appear to be "Shapefile", "KML" and "CSV" for vector data and "GeoTIFF" for raster data.

Other Answers:

- OSM format (4),
- geoJSON (3)
- RDF/Turtle or RDF/Ntriples
- Spatialite
- e00
- jp2

- PBF (2)
- ESRI ASCII

2.2.3.2.7. What would be your preferred file format for downloading open geospatial data?

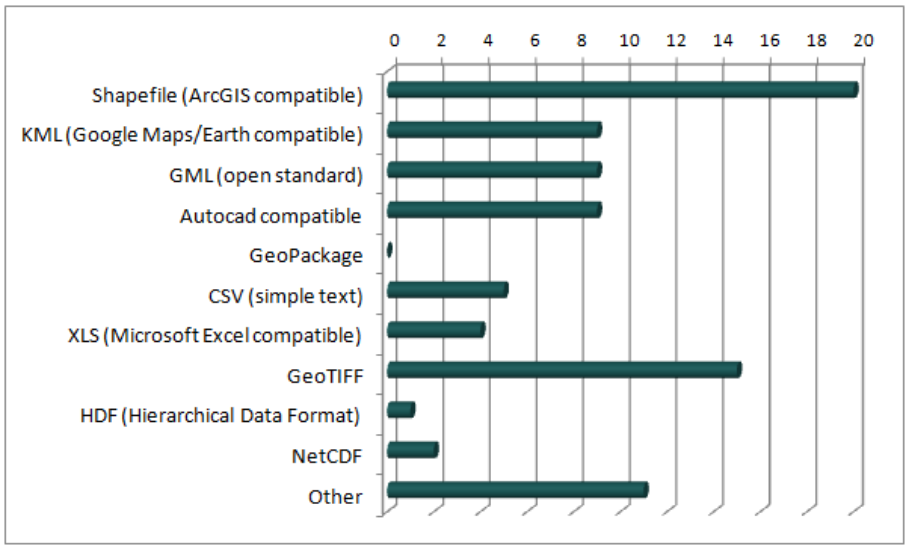


Figure 11: Chart of response distribution on question "What would be your preferred file format for downloading open geospatial data?"

Most popular choices are "Shapefile" and "GeoTIFF". KML, GML and CSV are popular as well

Other Answers:

- OSM format (3),
- RDF/Turtle or RDF/Ntriples
- PBF
- GeoJSON (3)
- All GDAL compatible
- Spatialite (2)

2.2.3.2.8. After you have downloaded an open geospatial data set, what are the most common problems you encounter?

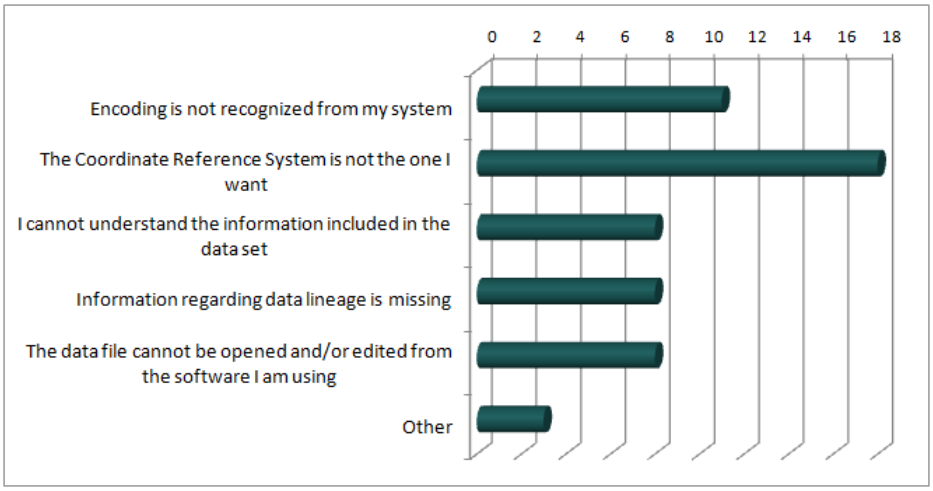


Figure 12: Chart of response distribution on question "After you have downloaded an open geospatial data set, what are the most common problems you encounter?"

As the users report, they frequently encounter problems related with the Coordinate Reference System and encoding. They also report problems related with data's missing linage, and with software incompatibilities.

Other Answers:

- Poor metadata in general (2)
- Data require cleanup

2.2.3.2.9. After you have downloaded an open geospatial data set, what is the typical way you use it?

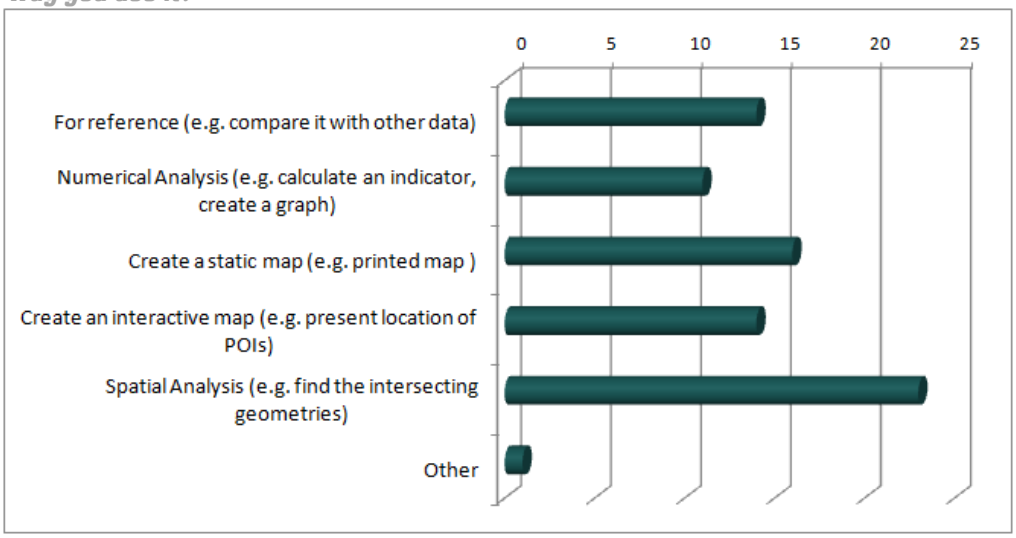


Figure 13: Chart of response distribution on question "After you have downloaded an open geospatial data set, what is the typical way you use it?"

Usually data are used for Spatial Analysis, for (static/dynamic) map building, for reference and for Numerical Analysis.

Other Answers:

- Interlinking

2.2.3.2.10. Open geospatial data should be available in all possible file formats. Do you agree?

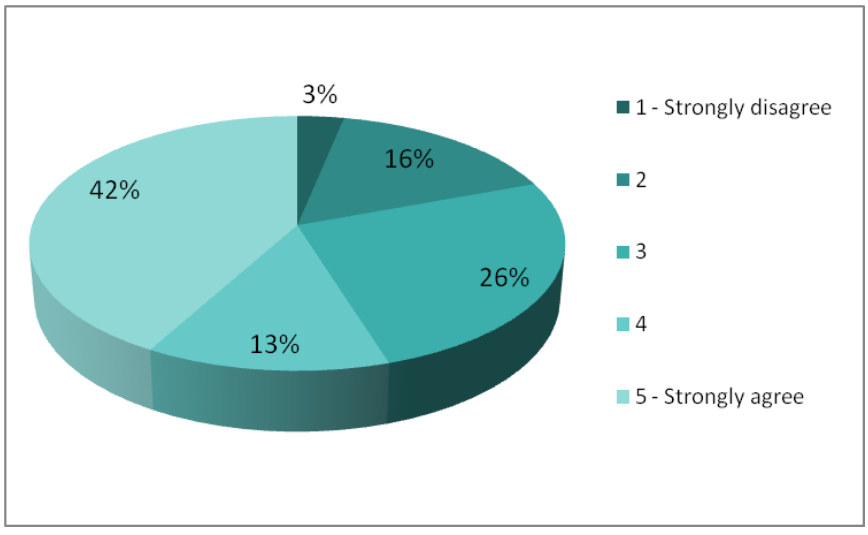


Figure 14: Chart of response distribution on question "Open geospatial data should be available in all possible file formats. Do you agree? (Users)"

As expected the users' great majority (58% agrees or agrees strongly) ask for open geospatial data which are published in all possible file formats.

2.2.3.2.11. Open geospatial data should be available in all possible Coordinate Reference Systems. Do you agree?

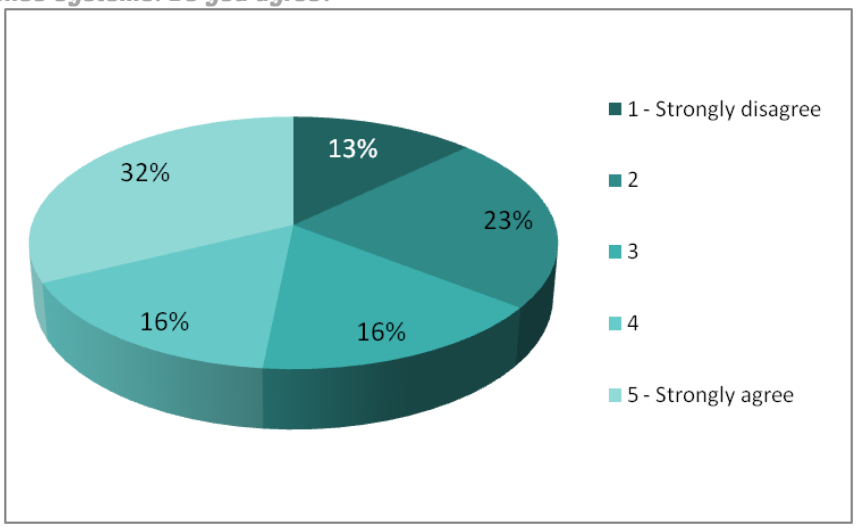


Figure 15: Chart of response distribution on question "Open geospatial data should be available in all possible Coordinate Reference Systems. Do you agree? (Users)"

The users' majority (48%) agree that open geospatial data should be available in all possible Coordinate Reference Systems. On the other hand 36% of the respondents disagree with that. Apparently it is understood that **there is no use into providing spatial data into all possible CRSs**, apart from the national CRSs and certain global ones (e.g. WGS84).

2.2.3.2.12. Open geospatial data should be available in various languages. Do you agree?

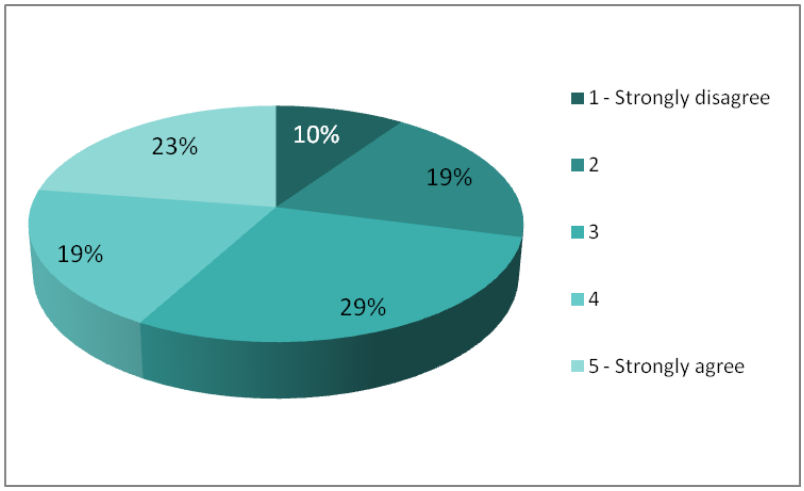


Figure 16: Chart of response distribution on question "Open geospatial data should be available in various languages. Do you agree? (Users)"

Most respondents (42%) agree that open geospatial data should be available in various languages, while a great part of them (29%) is indecisive. Possibly once again, users think that **there is not much benefit from providing data into a great number of languages** apart from

the national and a few others characterized as "*lingua francas*" (e.g. English, German, French etc.).

2.2.3.2.13. Open geospatial data should be available in interactive maps. Do you agree?

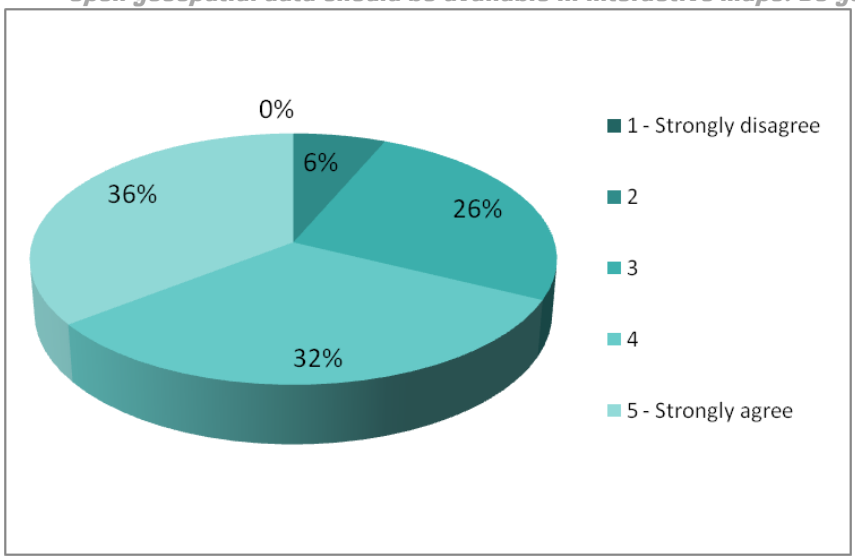


Figure 17: Chart of response distribution on question "Open geospatial data should be available in interactive maps. Do you agree? (Users)"

The majority (68%) understands the importance of interactive maps and agrees that open geospatial data should be available through them. Only 6% of them disagree and none of them disagrees strongly.

2.2.3.2.14. I should be able to upload my own data and create custom maps. Do you agree?

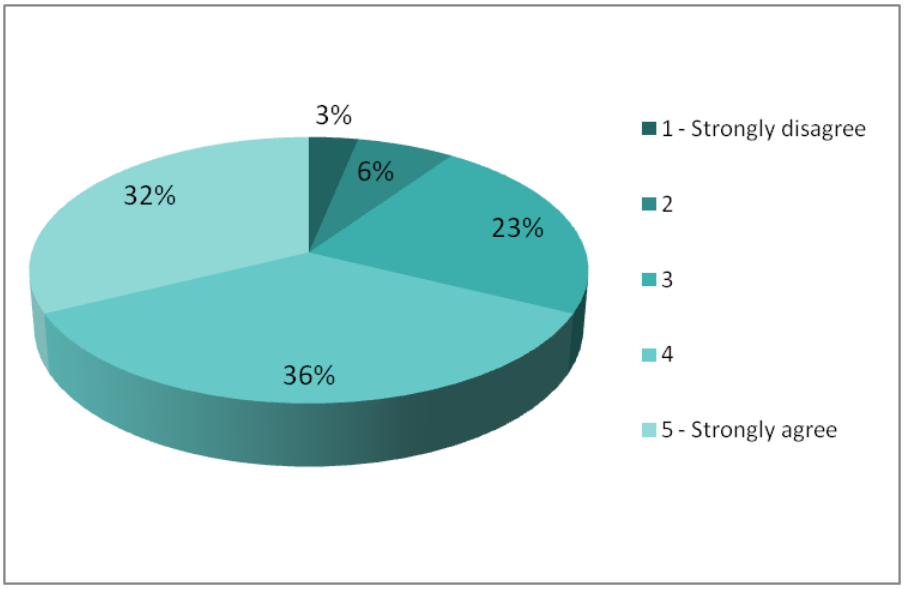


Figure 18: Chart of response distribution on question "I should be able to upload my own data and create custom maps. Do you agree? (Users)"

Most of the users (68%) ask to be able to upload their own data and create custom maps. Only 9% of them disagree with that possibility.

2.2.3.2.15. According to your estimates, what is the cost for creating a data set containing the entire road network for Athens, Greece?

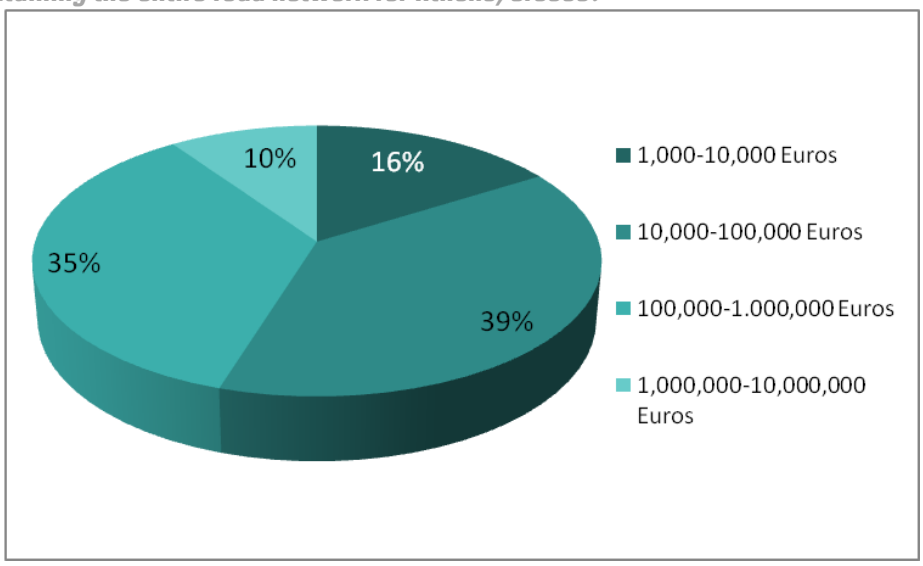


Figure 19: Chart of response distribution on question "According to your estimates, what is the cost for creating a data set containing the entire road network for Athens, Greece?"

The majority of the respondents is divided between "10,000 - 100,000 euros" and "100,000 – 1,000,000 euros" choices. This seems that they are more or less aware of the cost of creating a spatial dataset.

2.2.3.3. Publishers

2.2.3.3.1. How easy is it for you to publish an open geospatial data set?

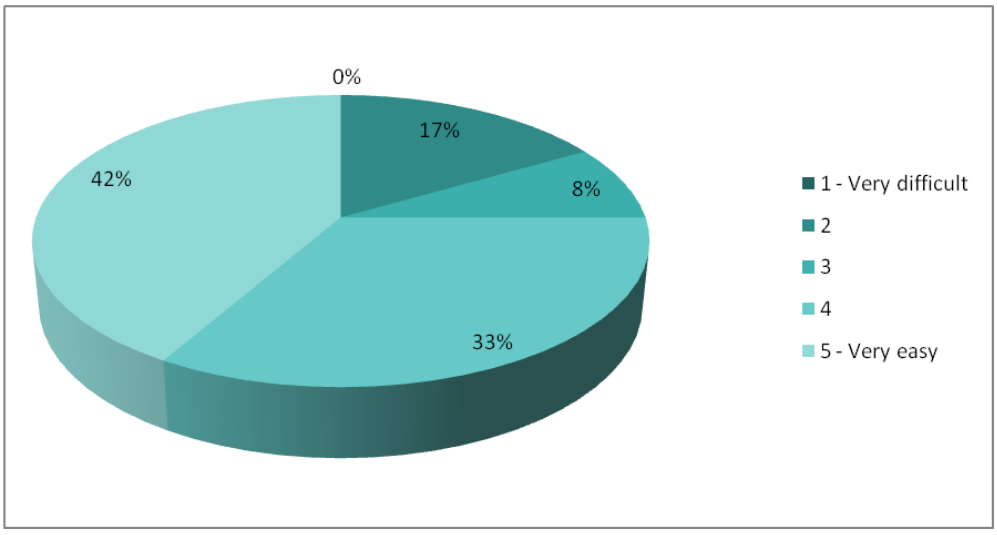


Figure 20: Chart of response distribution on question "How easy is it for you to publish an open geospatial data set?"

As it is shown publishers are skilled regarding publishing open geospatial data. The three quarters of them consider this procedure as "easy" [4] or "very easy" [5].

2.2.3.3.2. What is the average time for publishing an open geospatial data set?

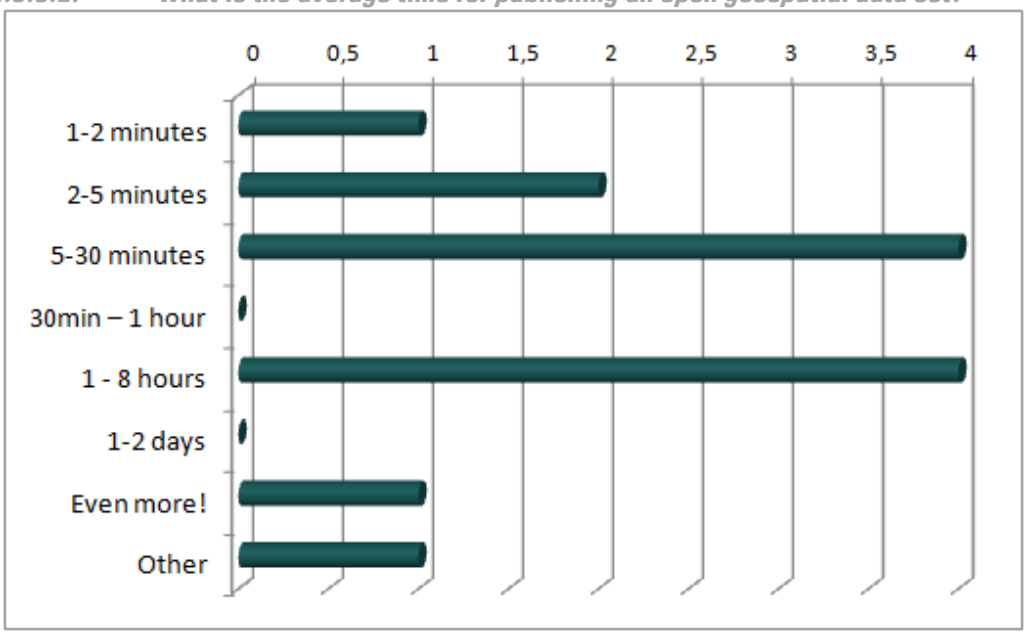


Figure 21: Chart of response distribution on question "What is the average time for publishing an open geospatial data set?"

The results indicate that the time needed varies enormously and depends on the complexity of a given dataset.

2.2.3.3.3. What are the major problems you typically encounter?

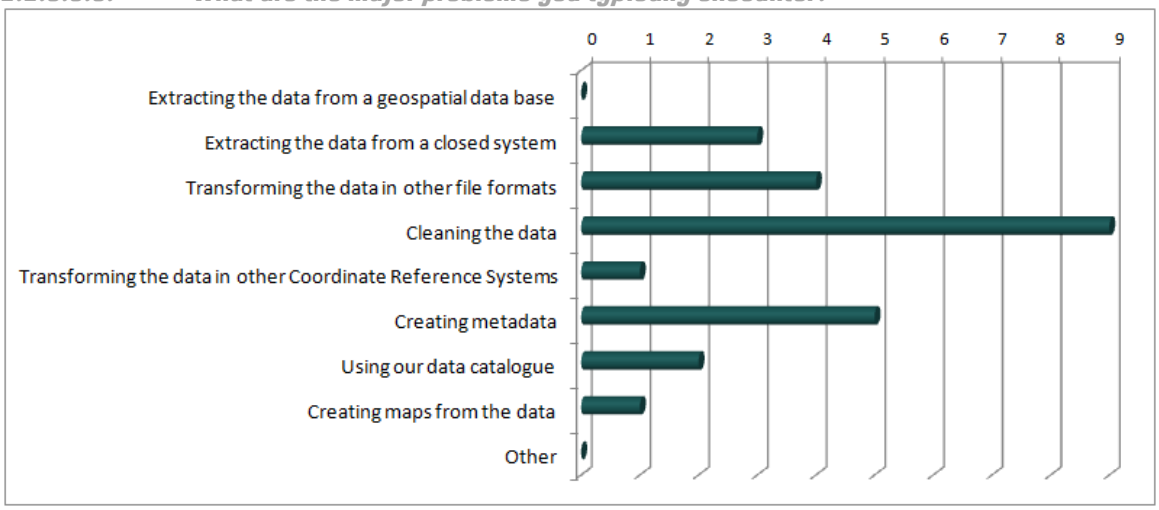


Figure 22: Chart of response distribution on question "What are the major problems you typically encounter? (on publishing an open geospatial data set)"

The most frequent problems reported are related with "data cleaning", "creating metadata", "transforming data in other file formats", and "extracting data from closed systems".

2.2.3.3.4. Can you give us an example of a data set that was extremely time-consuming, and why?

The examples presented were the following:

- Datasets with lack of metadata: *"Data producers don't really want to spend time writing metadata. But for me to create the metadata, I have to ask them a lot of questions, which also takes time."*
- Legacy data: *"Legacy data sets built using dbase with poor or no geocoding in data set"*

2.2.3.3.5. How easy is it for you to create metadata for a geospatial data set?

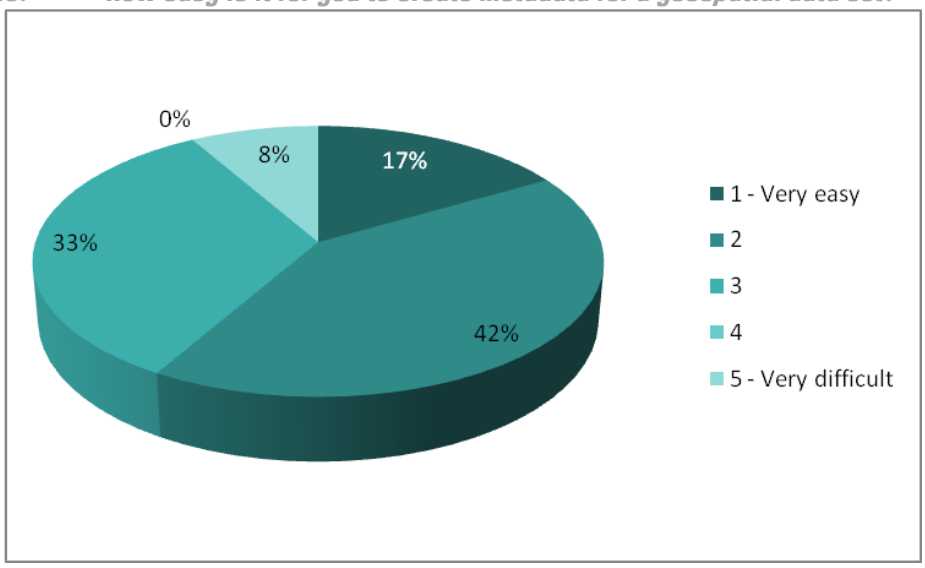


Figure 23: Chart of response distribution on question "How easy is it for you to create metadata for a geospatial data set?"

The responses show that the task of producing metadata is a rather easy task. 59% of the respondents consider it "easy" [2] or "very easy" [1]. Another 33% of the respondents consider it a task of medium difficulty.

2.2.3.3.6. Do you provide metadata for geospatial data following standard schemas? If yes which?

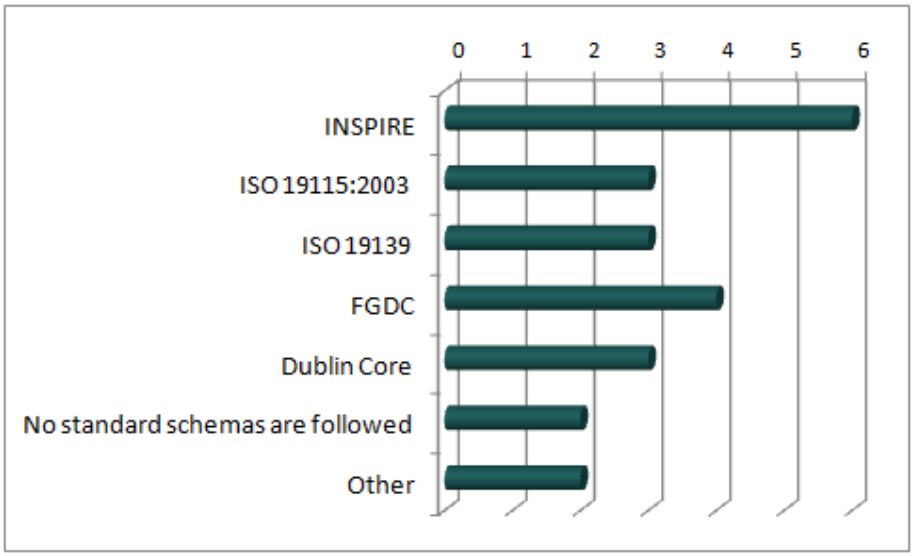


Figure 24: Chart of response distribution on question "Do you provide metadata for geospatial data following standard schemas? If yes which?"

The most popular metadata schema appears to be INSPIRE. FGDC, ISO 19115:2003, Dublin Core and ISO 19139 follow in popularity.

Other Answers:

- OpenStreetMap schema
- UK Gemini

2.2.3.3.7. How easy is it for you to transform the data in other formats?

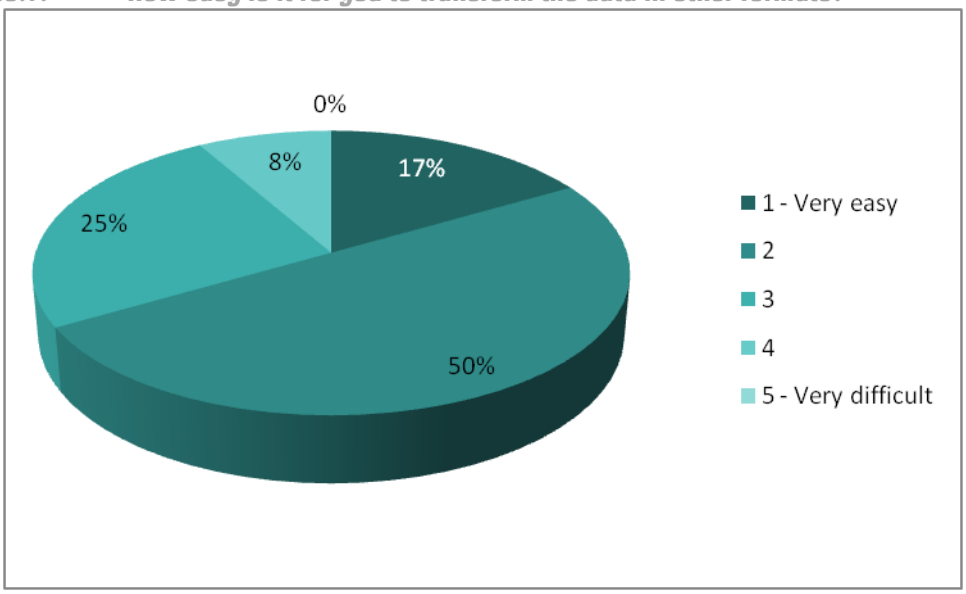


Figure 25: Chart of response distribution on question "How easy is it for you to transform the data in other formats?"

The majority of the respondents (67%) appears **capable in transforming data in other formats** and considers it an easy task. Only 8% of them consider it difficult.

2.2.3.3.8. How easy is it for you to transform the data in other Coordinate Reference Systems?

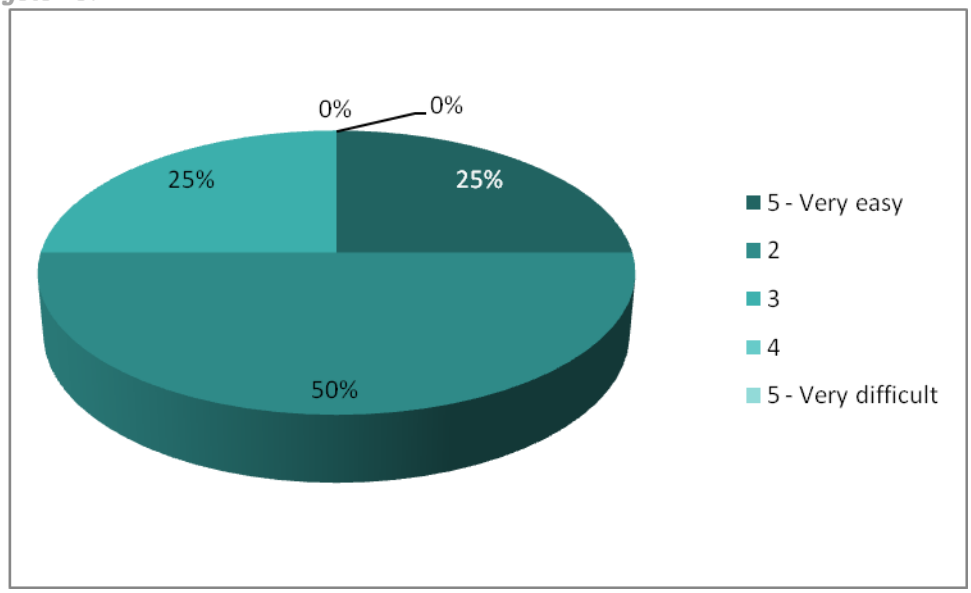


Figure 26: Chart of response distribution on question "How easy is it for you to transform the data in other Coordinate Reference Systems?"

Transforming data in other Coordinate Reference Systems, once again **is considered as an easy task** by the majority of the publishers (75%). None of the respondents consider this task as "difficult" or "very difficult".

2.2.3.3.9. What types of geospatial data do you publish?

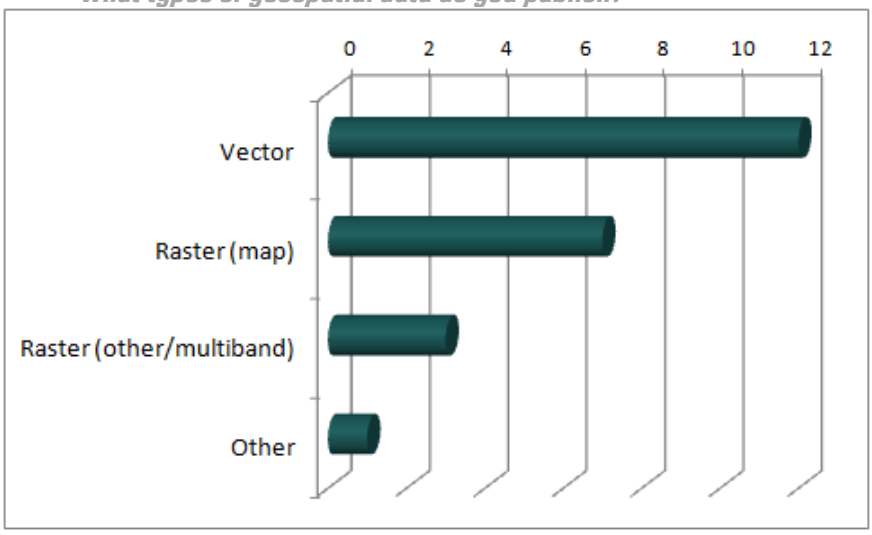


Figure 27: Chart of response distribution on question "What types of geospatial data do you publish?"

As it is shown Vector data are more frequently published than Raster data (of any type).

Other Answers:

- Linked data

2.2.3.3.10. If you do not publish raster data, what are the main reasons?

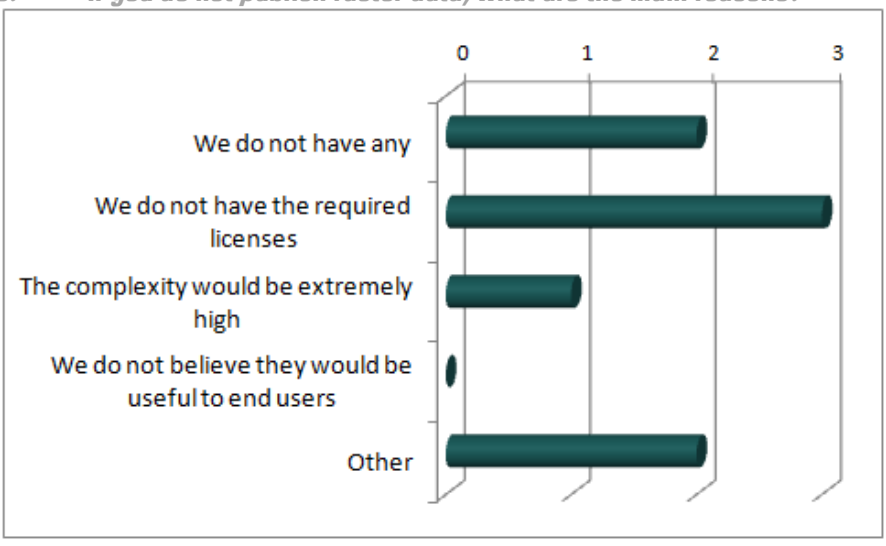


Figure 28: Chart of response distribution on question "If you do not publish raster data, what are the main reasons?"

The main reasons for not publishing raster data are "Licensing problems", "high complexity" and unavailability.

Other Answers:

- Publishing of raster data is not the respondent's duty (2)

2.2.3.3.11. *How many of the open geospatial data you provide are available in interactive maps?*

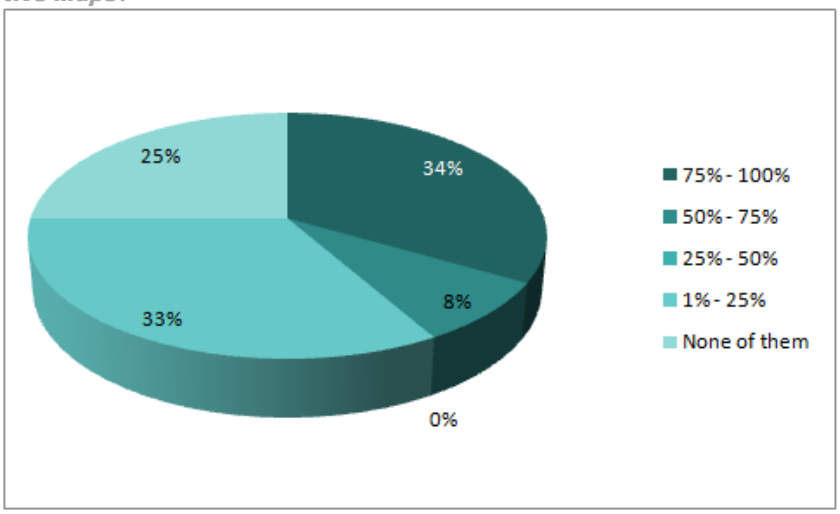


Figure 29: Chart of response distribution on question "How many of the open geospatial data you provide are available in interactive maps?"

The majority of the data publishers (58%) do not make their data available in interactive maps (at least the greater part of them). One does not provide any data through interactive maps.

2.2.3.3.12. *If not all of your open geospatial data are available in interactive maps, what are the main reasons?*

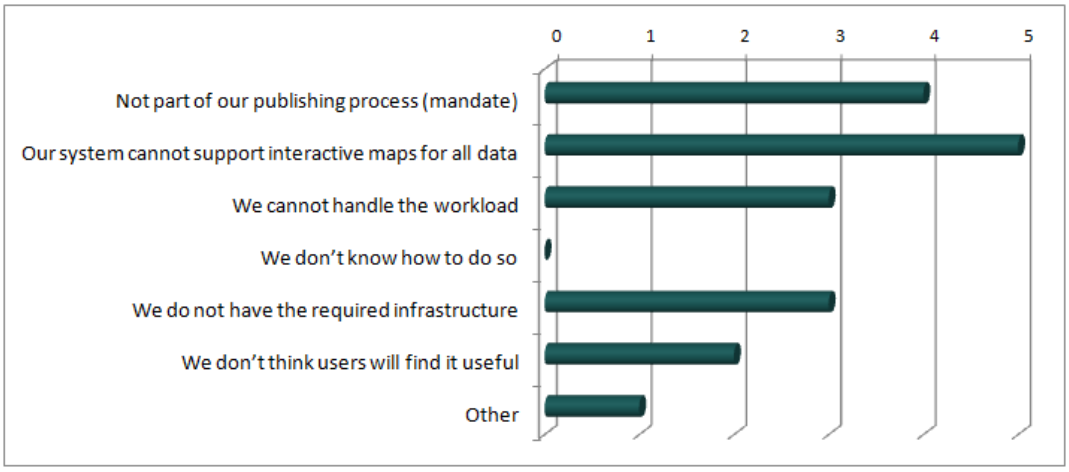


Figure 30: Chart of response distribution on question "If not all of your open geospatial data are available in interactive maps, what are the main reasons?"

The main reasons for not making open geospatial data available in interactive maps usually are related with "system capacity limitations" and with the fact sometimes this "is not part of the publishing process". Other reasons are related with "inadequate resources/infrastructure to handle the workload". Surprisingly enough, some publishers believe that "it is not useful for the users".

Other Answers:

- "This service is currently under implementation"

2.2.3.3.13. *What APIs do you provide for your geospatial data?*

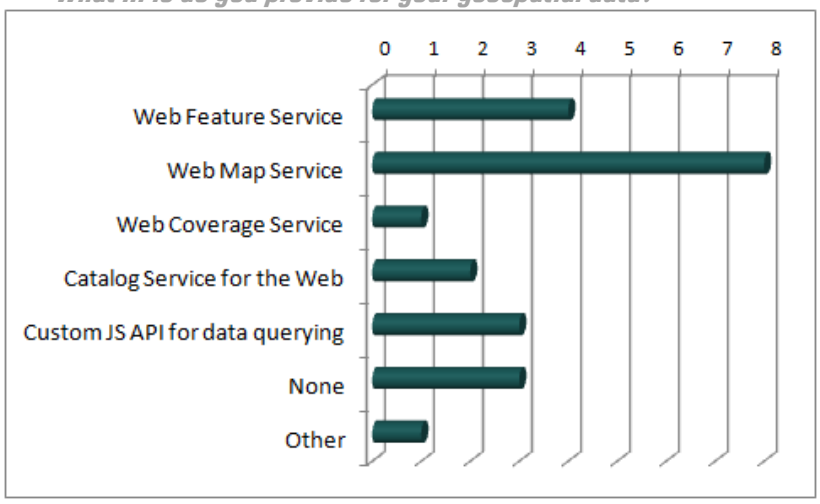


Figure 31: Chart of response distribution on question "What APIs do you provide for your geospatial data?"

Most of the publishers provide the WMS API. Other standard popular APIs are WFS, CSW and WCS. Also custom JS APIs are provided. In some cases no API at all is provided.

Other Answers:

- SPARQL

2.2.3.3.14. *If you do not provide any APIs, what are the main reasons?*

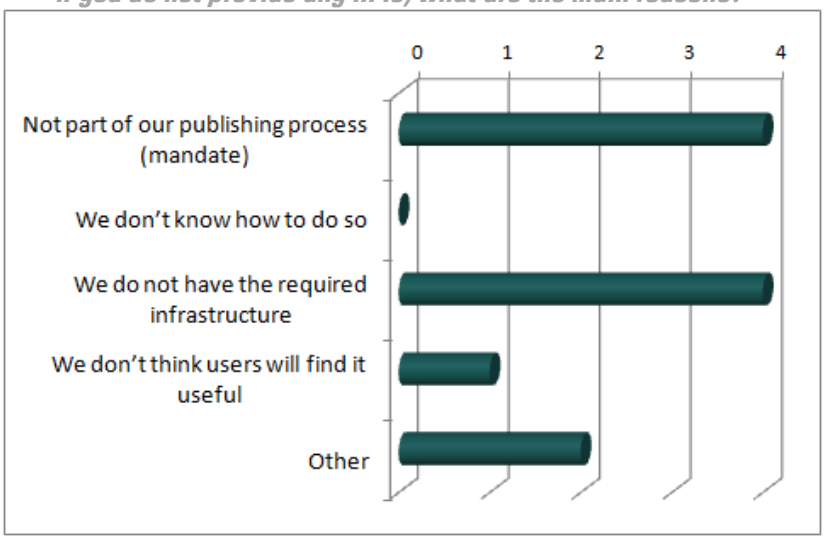


Figure 32: Chart of response distribution on question "If you do not provide any APIs, what are the main reasons?"

Most of the publishers do not provide APIs because "it is not part of the publishing process" and because of "inadequate infrastructure". Once again some publishers believe that "APIs are not useful for the users".

Other Answers:

- "API provision is currently under implementation"

2.2.3.3.15. *What of the following analytics do you keep?*

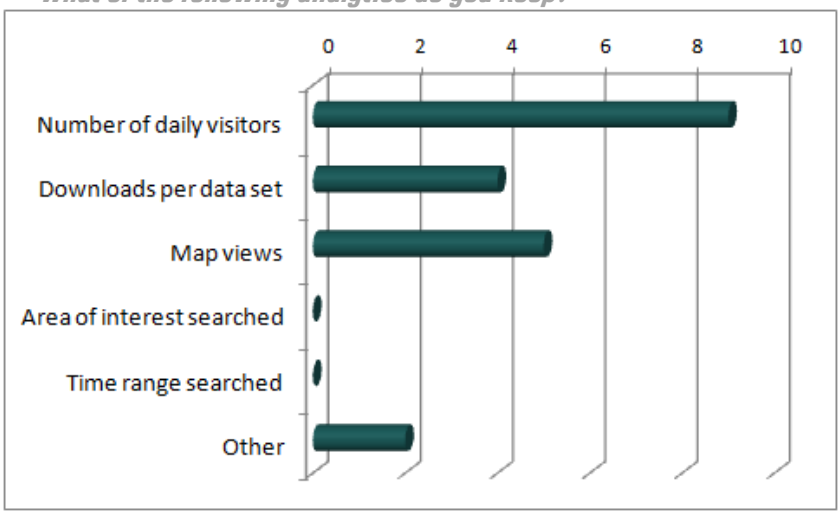


Figure 33: Chart of response distribution on question "What of the following analytics do you keep?"

Popular types of analytics are about the "number of daily visitors", "map views", and "downloads per dataset".

Other Answers:

- Unique visitors
- Formats downloaded

2.2.3.3.16. *Open geospatial data should be available in all possible file formats. Do you agree?*

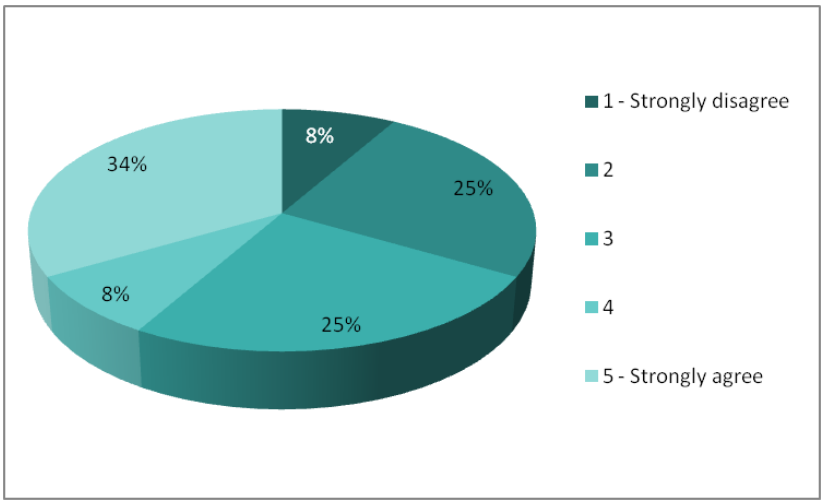


Figure 34: Chart of response distribution on question "Open geospatial data should be available in all possible file formats. Do you agree? (Publishers)"

The respondents are divided. 42% of them agrees or strongly agrees with the possibility of open geospatial data being available in all possible file formats. In the same time 33% disagrees or strongly disagrees with that, while 25% do not lean towards any side. It seems that a great part of the publishers consider it a tedious task and/or think that it is not part of their publishing process.

2.2.3.3.17. Open geospatial data should be available in all possible Coordinate Reference Systems. Do you agree?

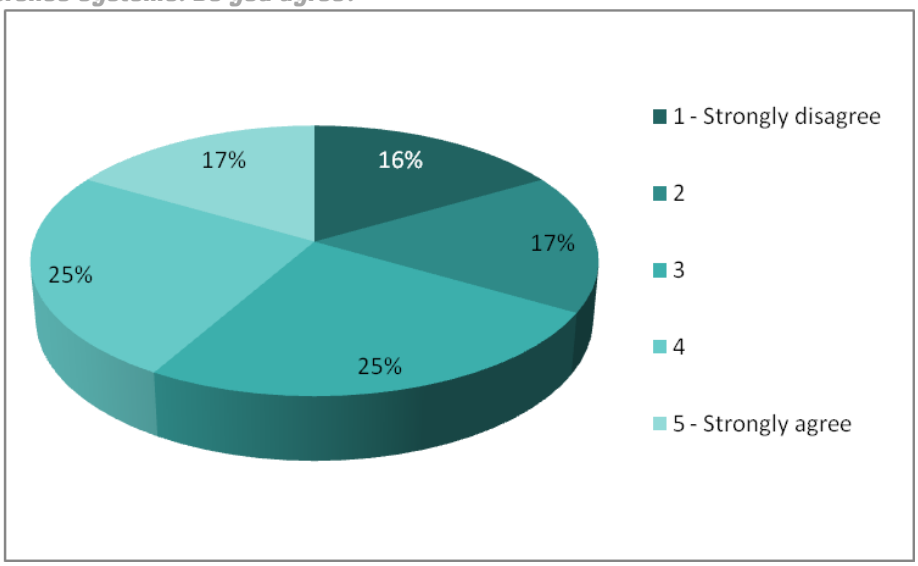


Figure 35: Chart of response distribution on question "Open geospatial data should be available in all possible Coordinate Reference Systems. Do you agree? (Publishers)"

The respondents are divided as well. 50% of them agrees with the possibility of open geospatial data being available in all possible CRSs. In the same time 33% disagrees or strongly disagrees with that, while 16% do not lean towards any side. Obviously data publishers along with the users agree that there is not much benefit to be gained into providing spatial data into all possible CRSs, apart from the national CRSs and selected global ones (e.g. WGS84).

2.2.3.3.18. Open geospatial data should be available in various languages. Do you agree?

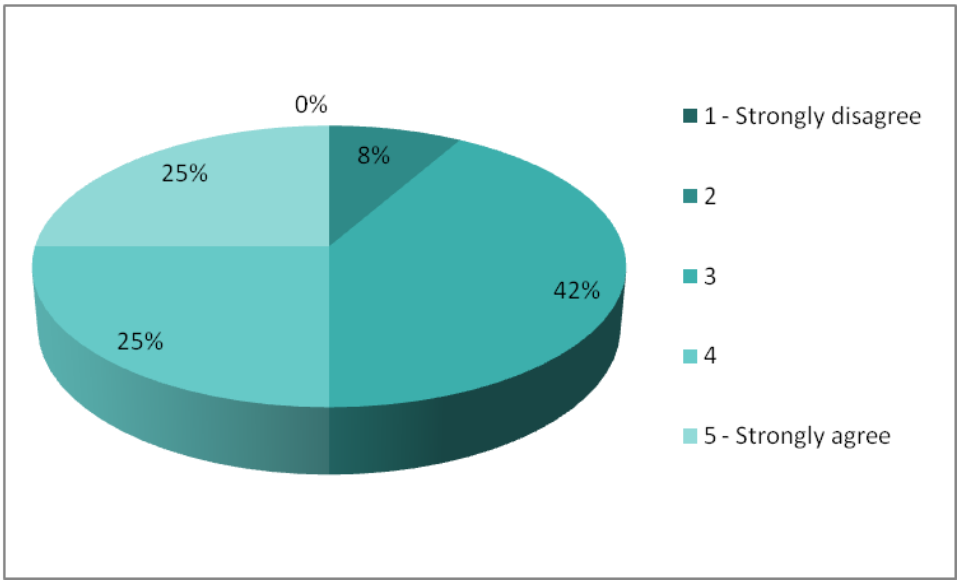


Figure 36: Chart of response distribution on question "Open geospatial data should be available in various languages. Do you agree? (Publishers)"

Half of the publishers agree that open geospatial data should be available in various languages. Only 8% of them disagree with that possibility.

2.2.3.3.19. Open geospatial data should be available in interactive maps. Do you agree?

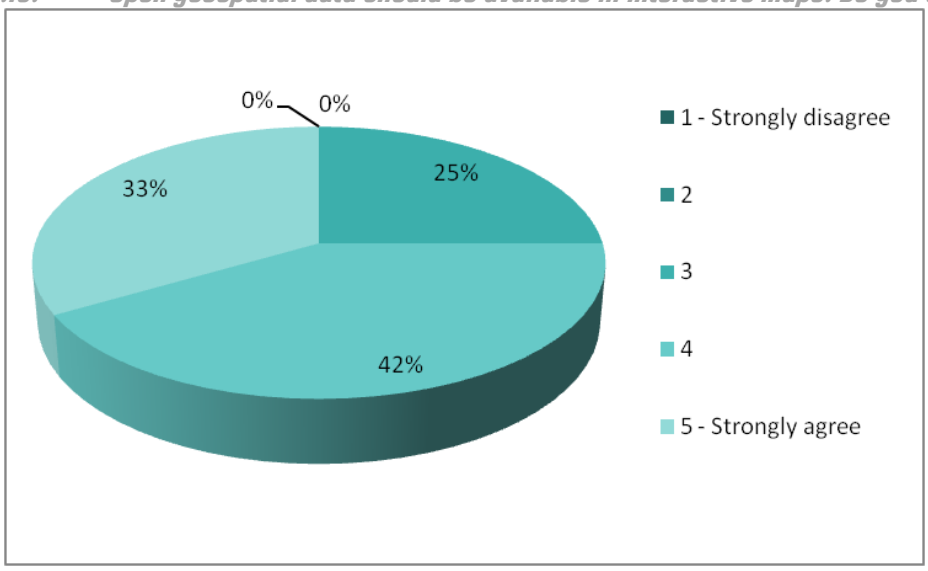


Figure 37: Chart of response distribution on question "Open geospatial data should be available in interactive maps. Do you agree?"

The great majority of the publishers (75%) seem to support open geospatial data's availability in interactive maps. Apparently all publishers understand the importance of interactive maps.

2.2.3.3.20. Open geospatial data should be provided to developers through open APIs. Do you agree?

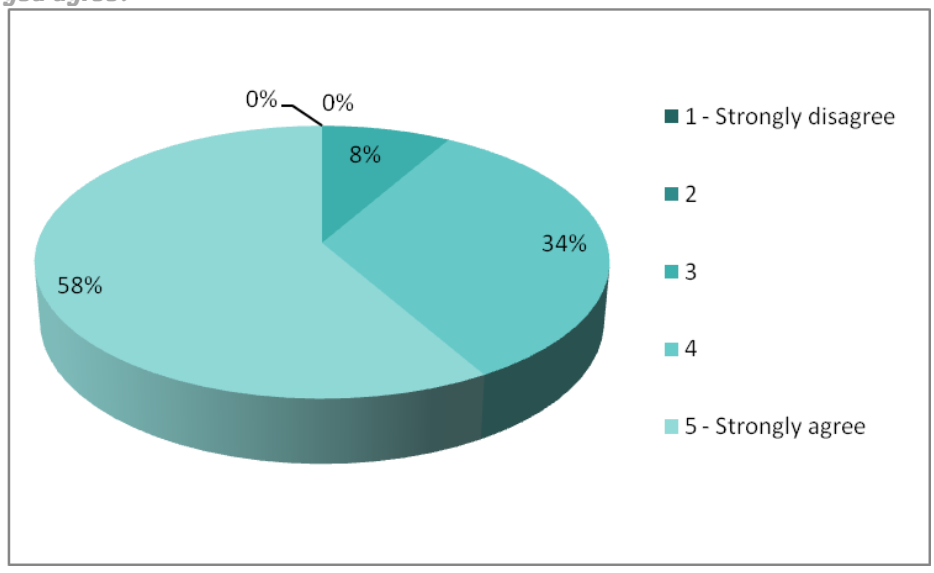


Figure 38: Chart of response distribution on question "Open geospatial data should be provided to developers through open APIs. Do you agree? (Publishers)"

The vast majority (92%) believes in open data APIs and agrees that open geospatial data should be available should be provided to developers through them. None of them disagrees.

2.2.3.4. Developers

2.2.3.4.1. For which platforms do you develop applications?

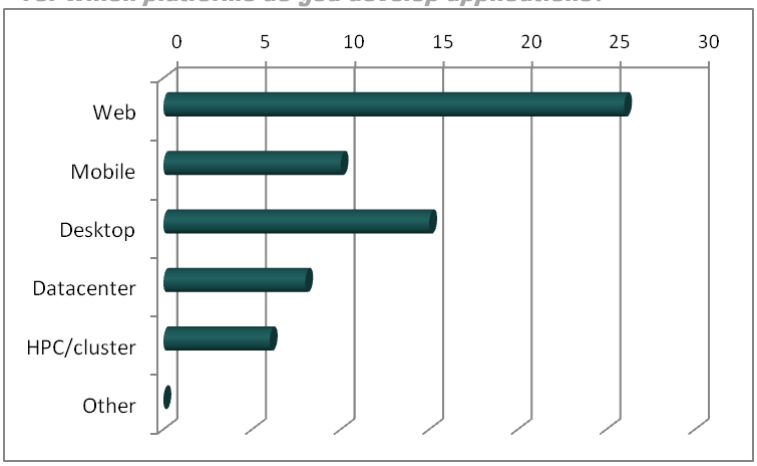


Figure 39: Chart of response distribution on question "For which platforms do you develop applications?"

The respondents are developers who primarily develop web applications. Then desktop and mobile applications follow in popularity. Some of the developers even develop applications for datacenters and HPCs.

2.2.3.4.2. What is your level of expertise regarding the use of web mapping frameworks?

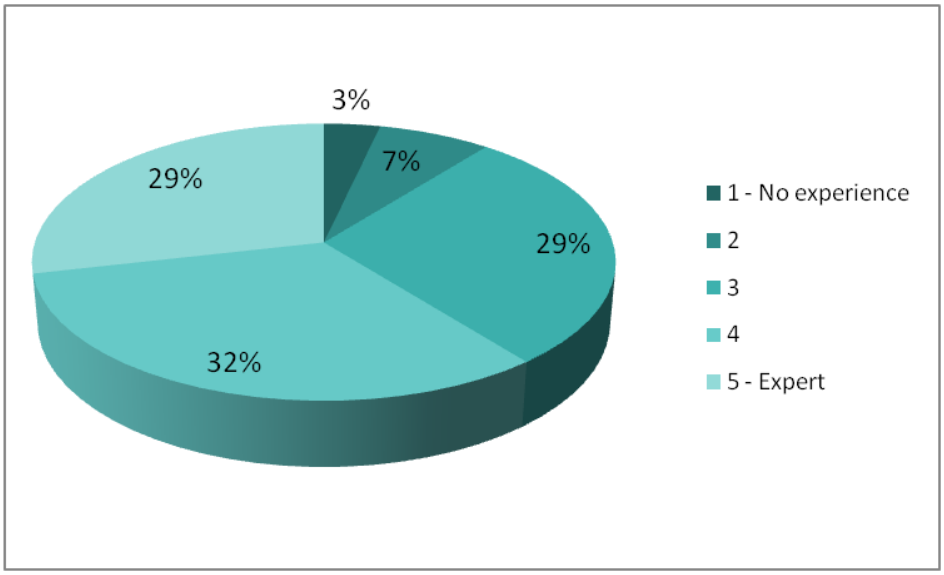


Figure 40: Chart of response distribution on question "What is your level of expertise regarding the use of web mapping frameworks?"

Most respondents are quite experienced in using web mapping frameworks. Only the 3% of them declare having "No experience".

2.2.3.4.3. *Please rate your familiarity with this task: Insert a shapefile into a geospatial database*

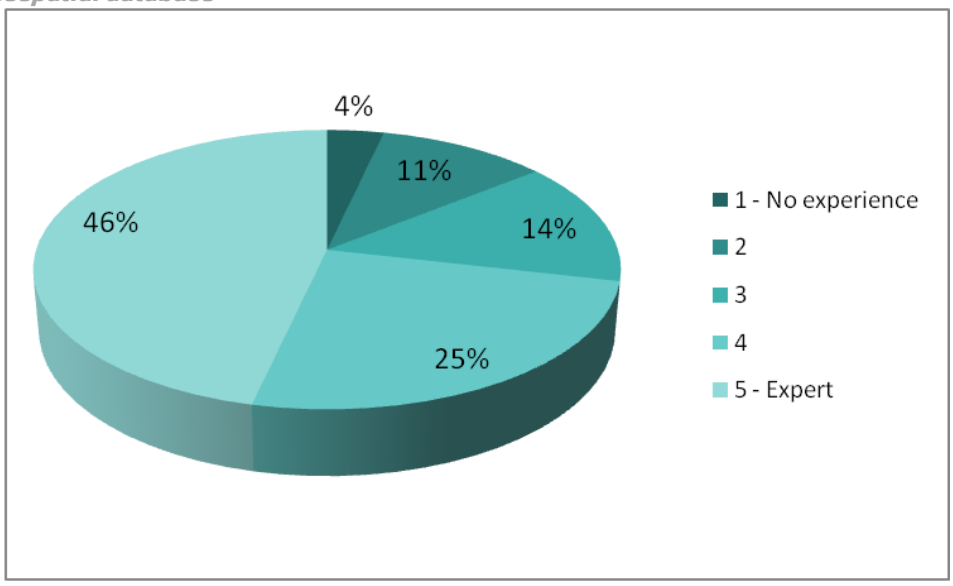


Figure 41: Chart of response distribution on question "Please rate your familiarity with this task: Insert a shapefile into a geospatial database"

The majority of the developers (71%) are **familiar with tasks like inserting a shapefile into a spatial database**. Only 4% of them declare having "No experience".

2.2.3.4.4. *Please rate your familiarity with this task: Write geospatial SQL queries*

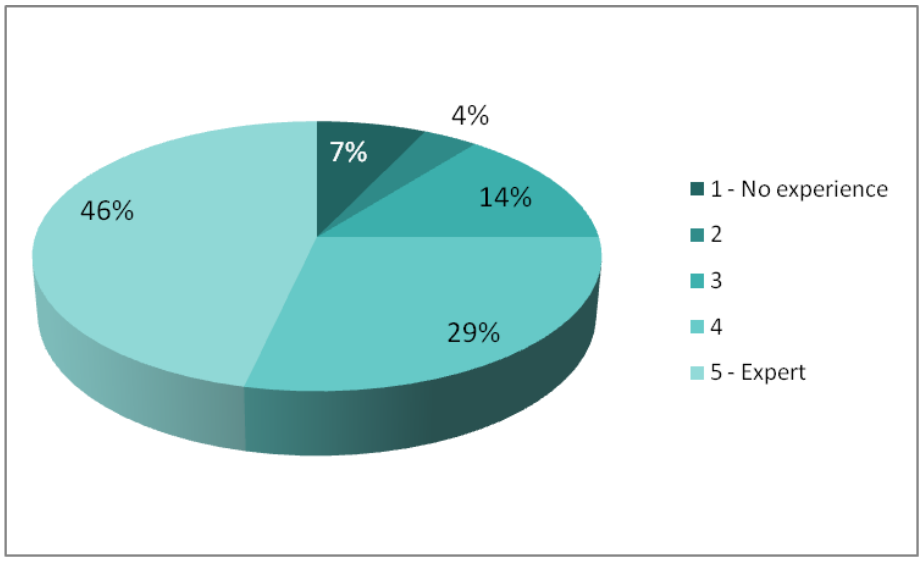


Figure 42: Chart of response distribution on question "Please rate your familiarity with this task: Write geospatial SQL queries"

Three quarters of developers are **familiar with writing geospatial SQL queries**. 46% of them declare themselves "experts" Only 7% of them declare having "no experience".

2.2.3.4.5. Please rate your familiarity with this task: Perform spatial analysis in a desktop GIS

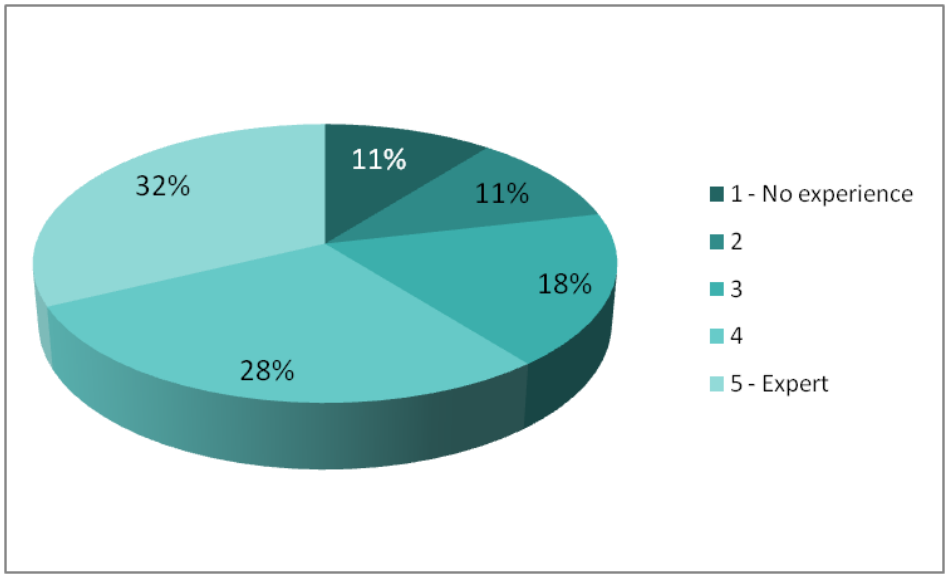


Figure 43: Chart of response distribution on question "Please rate your familiarity with this task: Perform spatial analysis in a desktop GIS"

The majority of the developers (60%) are **familiar with performing spatial analysis in a desktop GIS**. In the same time, 22% of them declare having little or no experience.

2.2.3.4.6. Please rate your familiarity with this task: Transform a geospatial data set in another format

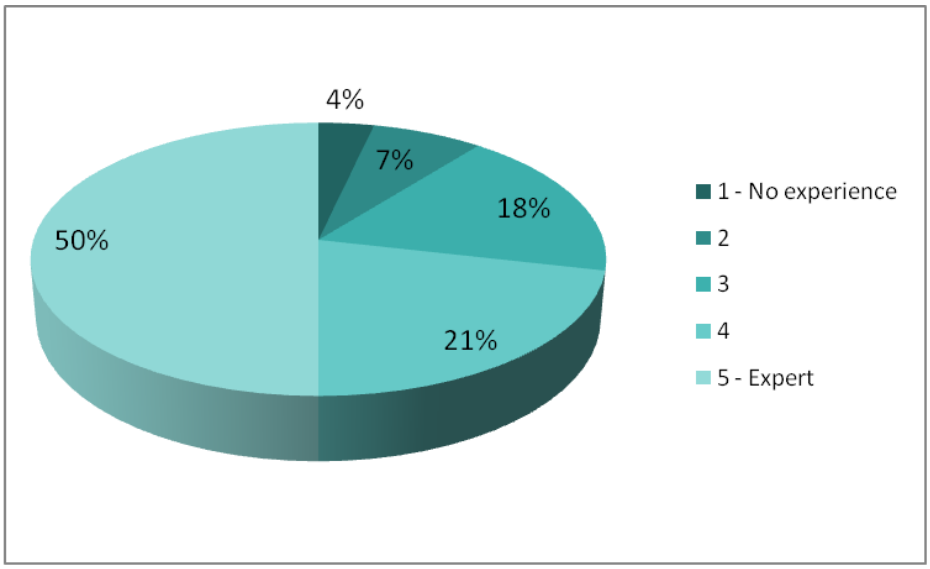


Figure 44: Chart of response distribution on question "Please rate your familiarity with this task: Transform a geospatial data set in another format"

The greater part of the developers (71%) are **familiar with transforming a geospatial data set in another format** (half of the developers declare themselves "experts").

2.2.3.4.7. Please rate your familiarity with this task: Transform a geospatial data set in another Coordinate Reference System

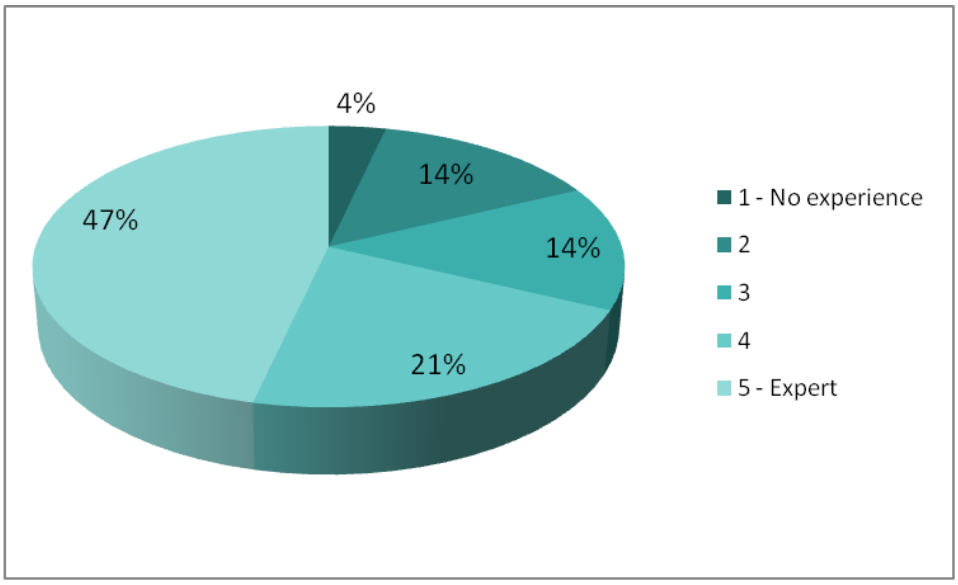


Figure 45: Chart of response distribution on question "Please rate your familiarity with this task: Transform a geospatial data set in another Coordinate Reference System"

Most of the developers (58%) are familiar with transforming a geospatial data set in another Coordinate Reference System. 18% of them declare no or little experience.

2.2.3.4.8. Please rate your familiarity with this task: Create a Google Maps mashup

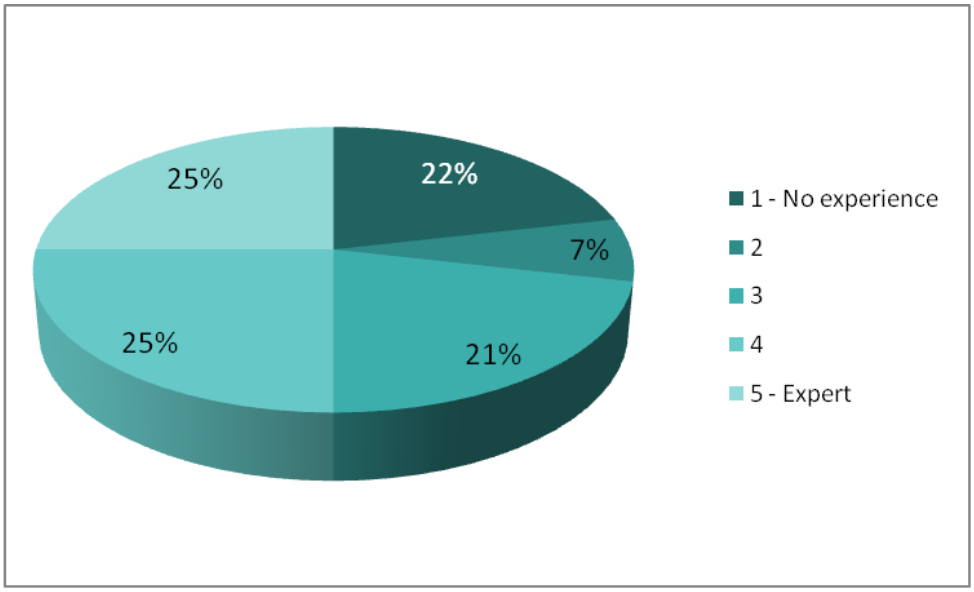


Figure 46: Chart of response distribution on question "Please rate your familiarity with this task: Create a Google Maps mashup"

Half of the developers have good experience in creating a Google Maps mashup. In the same time 22% of them declare having "no experience" and 7% of them "little experience".

2.2.3.4.9. *Please rate your familiarity with this task: Install a map server*

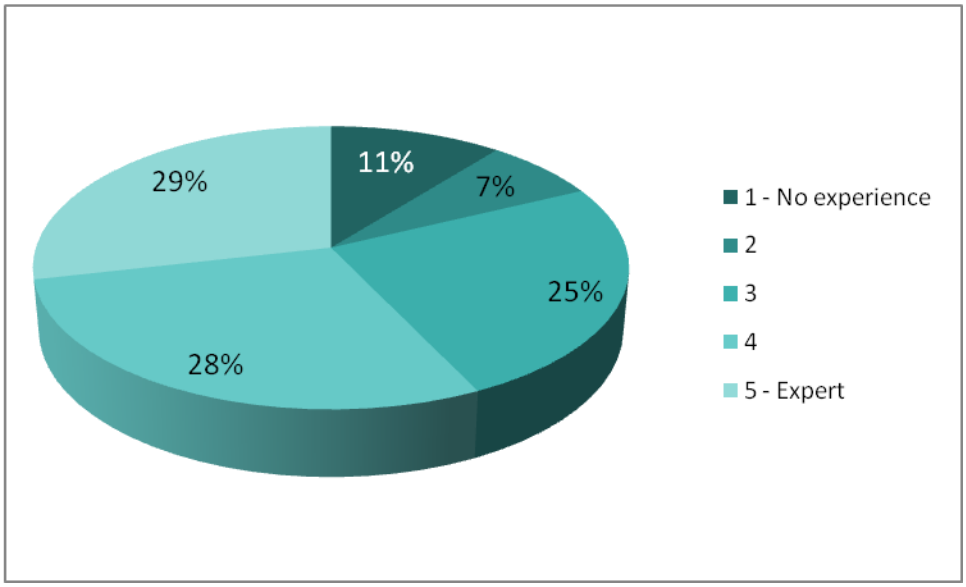


Figure 47: Chart of response distribution on question "Please rate your familiarity with this task: Install a map server"

More than half of the developers (57%) have **good experience in installing a map server**. 28% of the developers have no or little experience.

2.2.3.4.10. *Consume OGC services (e.g. Web Feature Service)*

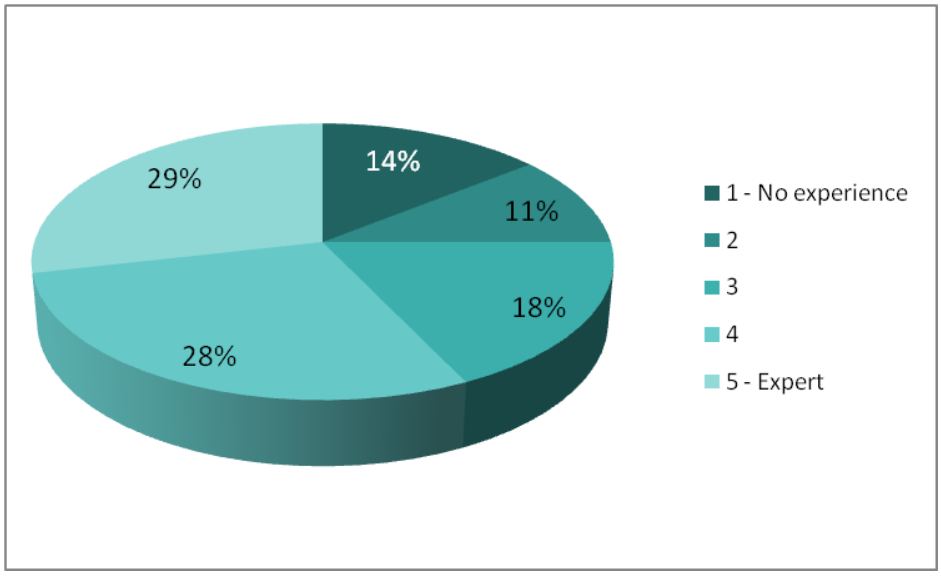


Figure 48: Chart of response distribution on question "Please rate your familiarity with this task: Consume OGC services (e.g. Web Feature Service)"

A similar situation is observed with familiarity regarding the consumption of OGC services. **57% of the developers have good experience** while 28% of them have no or little experience.

2.2.3.4.11. Have you used open data in your applications?

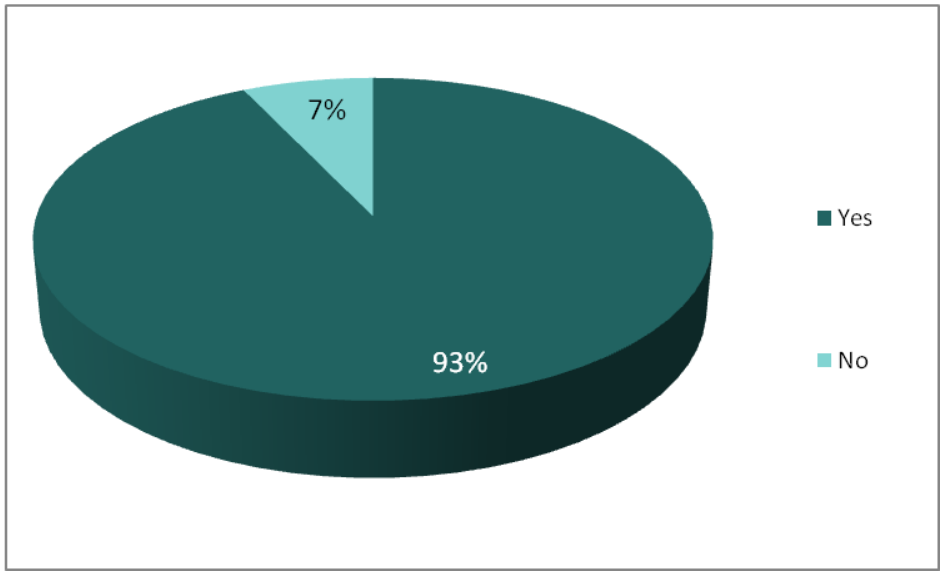


Figure 49: Chart of response distribution on question "Have you used open data in your applications?"

Open data appears to be a valuable asset for developers. The great majority of them (93%) report that they have them in their applications.

2.2.3.4.12. What type of open data would do you use, or you are currently using, in your applications?

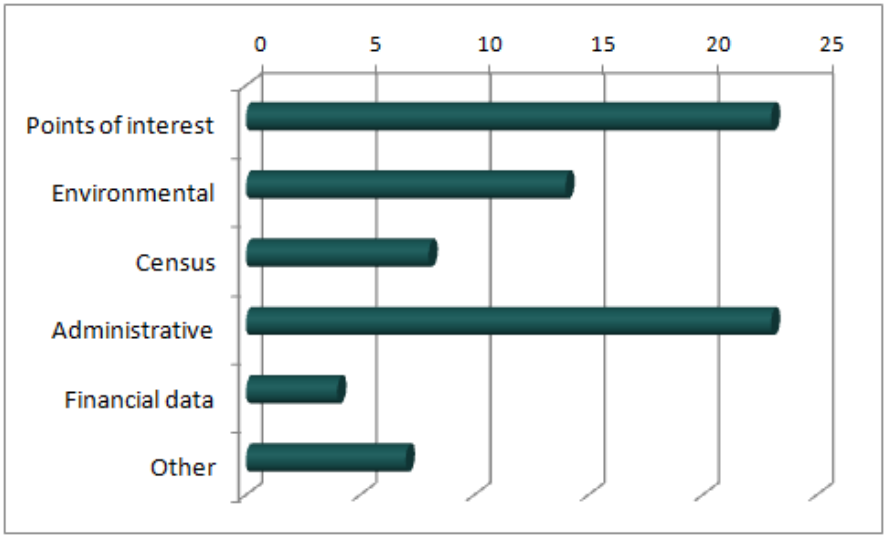


Figure 50: Chart of response distribution on question "What type of open data would do you use, or you are currently using, in your applications?"

Popular datasets are POIs and administrative. Then environmental, census and financial data follow.

Other Answers:

- Transport network data (4)
- Health data
- Education data
- Base maps

2.2.3.4.13. Have you encountered any problems when using open data?

The most recurrent problems encountered are the following:

- Not sufficient granularity: *"Data are not accessible in the granules I need", "They have insufficient spatiotemporal coverage for my purposes"*
- Not up to date: *"They are outdated", "Slow release cycles"*
- Insufficient metadata: *"Not enough info describing the source schema"*
- Bad quality: *"Errors in data sets, "Incompleteness", "Heterogeneity"*
- Not really available: *"Not available for free and for the regions of interest", "Institutions don't want to make public their data. They have fear to show errors in their data, among other things"*

2.2.3.4.14. Do you use maps in your applications?

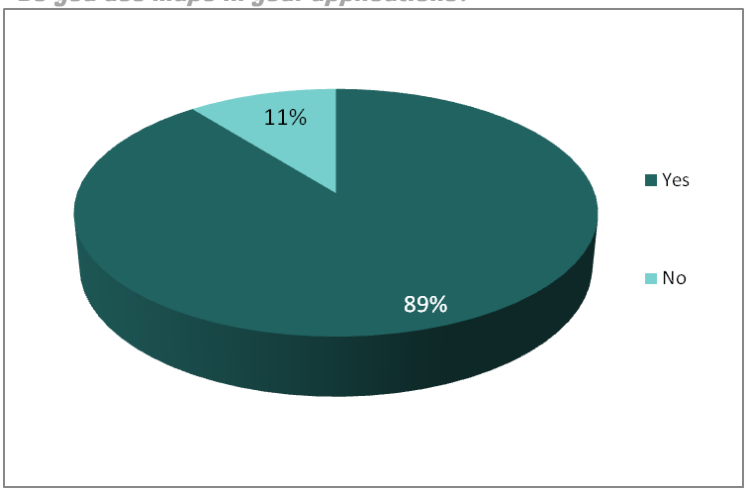


Figure 51: Chart of response distribution on question "Do you use maps in your applications?"

It seems that maps are integral parts of a modern application as most of the developers (89%) declare their use in their applications.

2.2.3.4.15. If no, why?

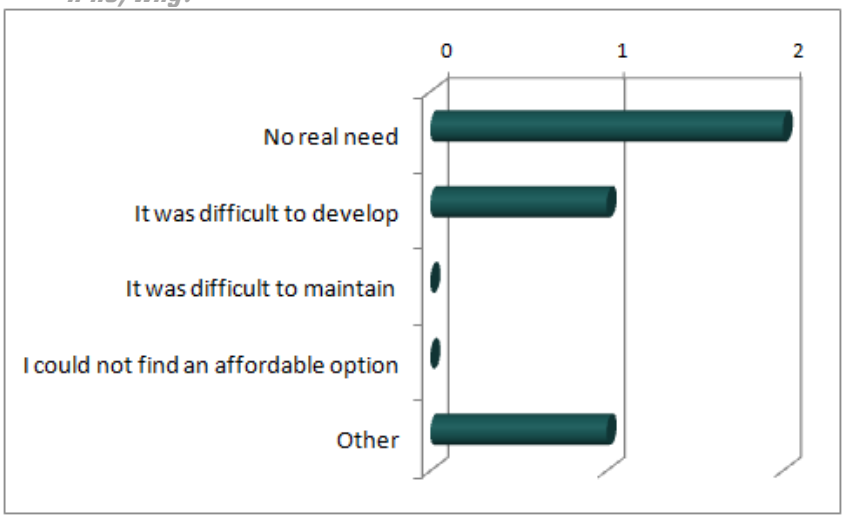


Figure 52: Chart of response distribution on question "If no, why? (Do you use maps in your application)"

From the few developers reported not using maps in their application the most frequent reason was that it was not required. Another reason was the difficulty.

Other Answers:

- "No experience, do not know where to start"

2.2.3.4.16. If yes, what map tiles are you using?

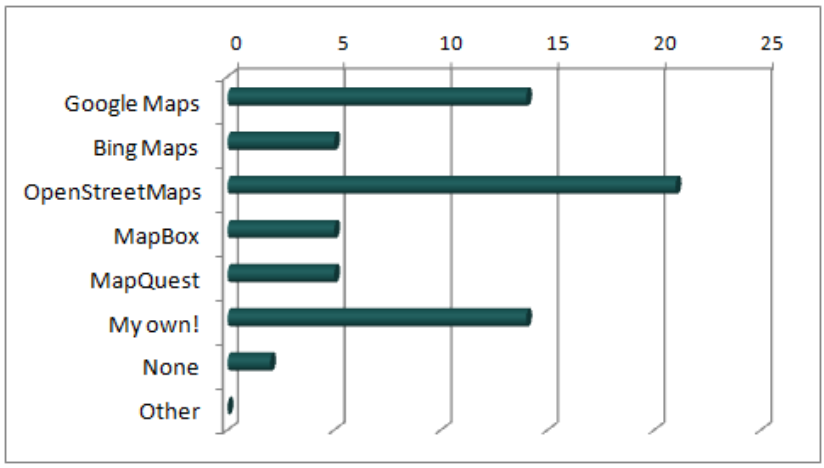


Figure 53: Chart of response distribution on question "If yes, what map tiles are you using? (Do you use maps in your application)"

The most popular maps tiles are from OSM. Google Maps tiles are another popular choice, while Bing Maps, MapBox and MapQuest choices follow. A lot of developers are reporting using their own tiles, while a few of them do not use any map tiles at all. Overall it seems that more than the half of the respondents are avid users of open map tiles.

2.2.3.4.17. What is the map server for the applications you currently develop?

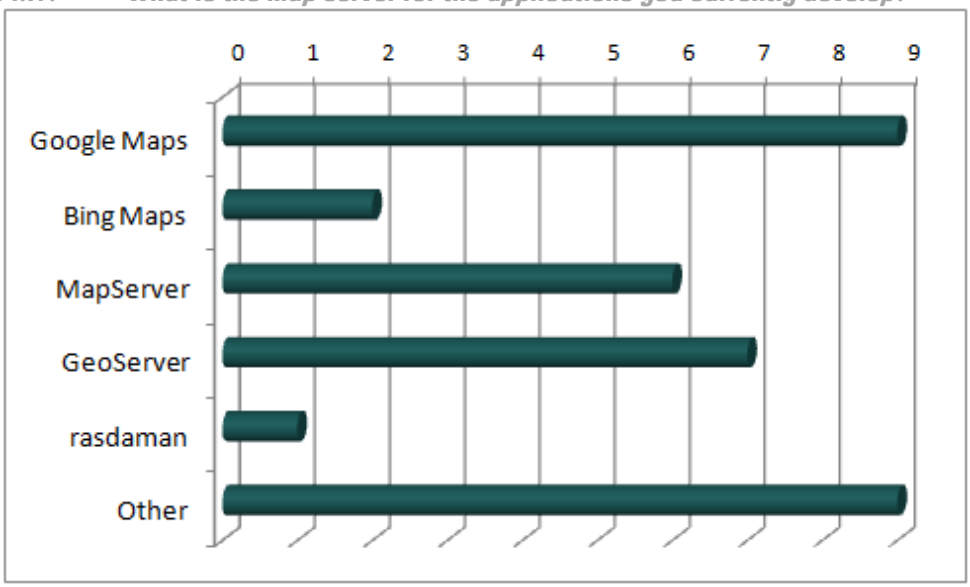


Figure 54: Chart of response distribution on question "What is the map server for the applications you currently develop?"

Google Maps server is on the most popular map server, along with GeoServer and MapServer. More rare choices are the Bing Maps and the rasdaman servers.

Other Answers:

- OpenStreetMap
- mod_tile/mapnik (2)
- LuciadFusion Server
- mod_tile/rendered
- tilestache/mapnik
- ArcGIS Server
- TileMill
- Own server (2)

2.2.3.4.18. *If you had free access to an API providing querying and analysis services for open geospatial data would you use it?*

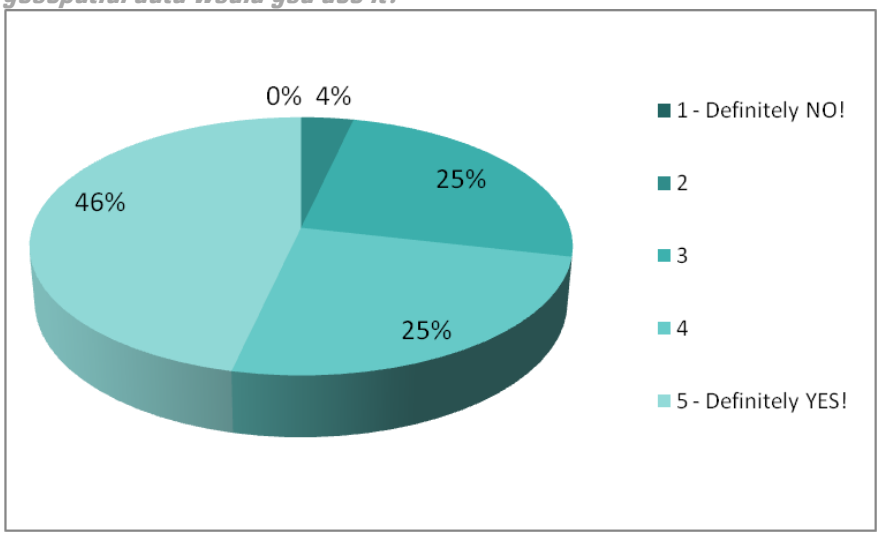


Figure 55: Chart of response distribution on question "If you had free access to an API providing querying and analysis services for open geospatial data would you use it?"

As it was expected, the great majority of the developers (71%) **believe that they would benefit from the use of an API**, providing querying and analysis services for open geospatial data if it was freely available.

2.2.3.4.19. *What kind of a data API would you prefer?*

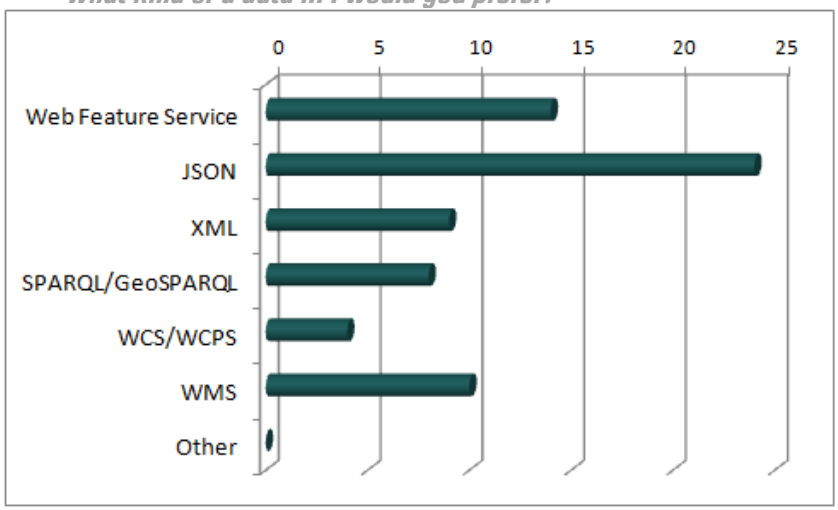


Figure 56: Chart of response distribution on question "What kind of a data API would you prefer?"

The most preferred APIs are JSON APIs which are followed in popularity by WFS, WMS, XML,SPARQL/GeoSPARQL. WCS/WCPS is less popular.

2.2.3.4.20. What of the following data API capabilities would you use?

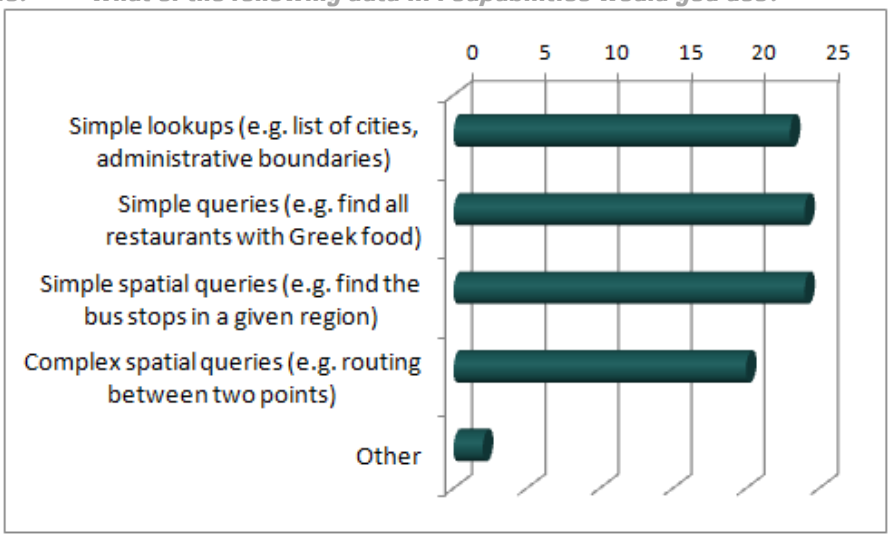


Figure 57: Chart of response distribution on question "What of the following data API capabilities would you use?"

More or less all the respondents would use all typical API capabilities as "simple lookups", "simple queries", "simple spatial queries", and "complex spatial queries".

Other Answers:

- "Providing relationship between features (e.g. list of residential areas without public transport or local supply facilities)"
- Geocoding service

2.2.3.4.21. Any other potential capabilities for the data API that you would like?

The most popular suggestions are the following:

- Fusion of data sets of different origin
- Ad-hoc analytics
- Calculating differences (diff) diff between current and previous versions of objects
- Raster analysis and LiDAR geoproceses
- Information provision for data licenses and implications

2.2.3.4.22. Regarding the data API, please rate the importance of the following

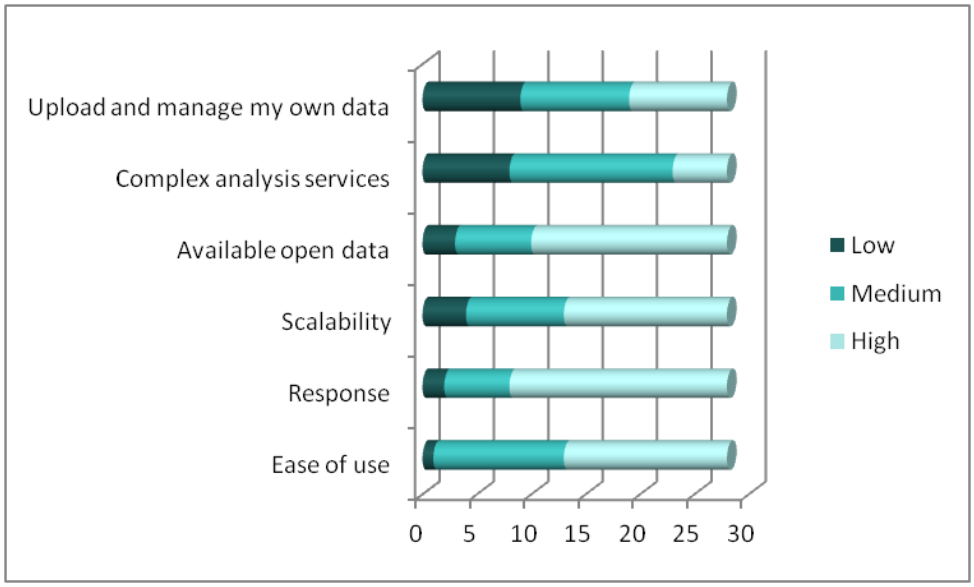


Figure 58: Chart of response distribution on question "Regarding the data API, please rate the importance of the following"

As it is shown the most important properties of the data API are its **response**, the fact that it **serves open data**, its **scalability** and its **ease of use**. The capability of performing complex analysis and upload of data are not considered that important. Once again the importance of open data APIs is shown.

2.2.3.4.23. If you had free access to a web mapping framework for developing interactive maps, what would be you use it?

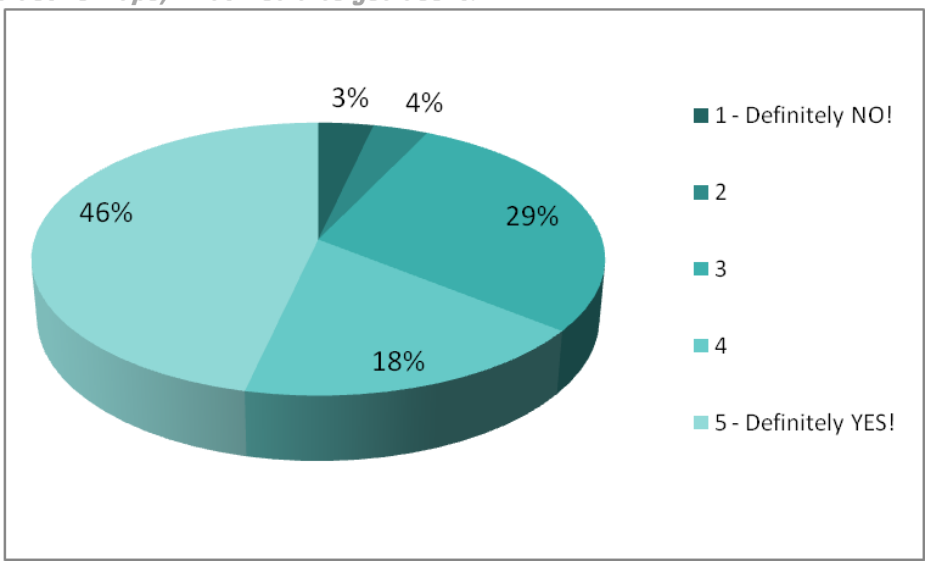


Figure 59: Chart of response distribution on question "If you had free access to a web mapping framework for developing interactive maps, what would be you use it?"

As it is depicted, the majority of the developers **wish to use a web mapping framework for interactive maps** if they had free access to it. Only 29% are indecisive while 7% would not use it.

2.2.3.4.24. What kind of a web mapping framework would you prefer?

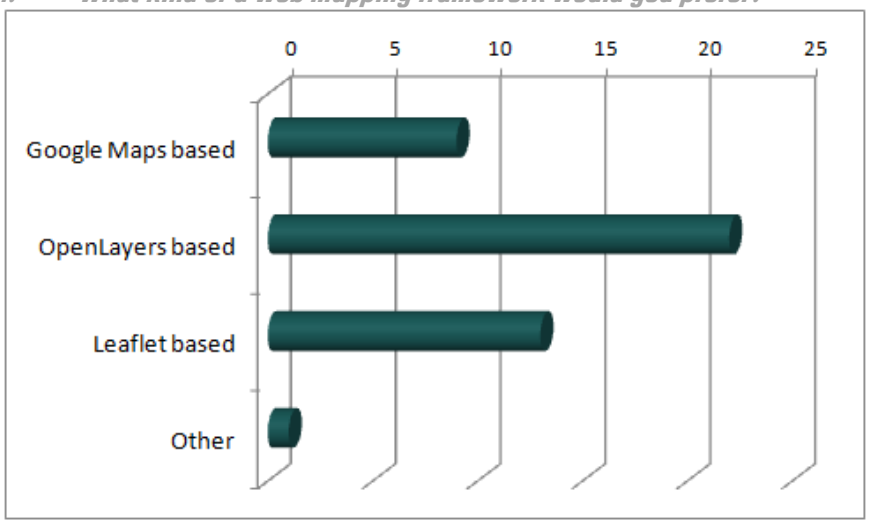


Figure 60: Chart of response distribution on question "What kind of a web mapping framework would you prefer?"

Most of the developers would prefer an OpenLayers based mapping framework, then a Leaflet based and finally a Google Maps based.

2.2.3.4.25. What of the following web mapping framework capabilities would you use?

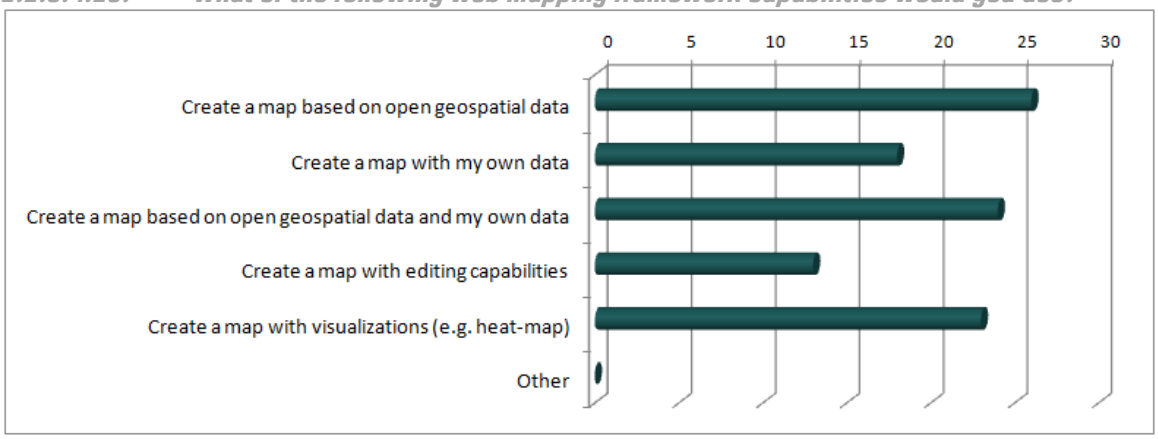


Figure 61: Chart of response distribution on question "What of the following web mapping framework capabilities would you use?"

More or less all web mapping framework capabilities are important for the developers while "create a map based on open geospatial data", "create a map based on open geospatial data and my own data" and "create a map with visualizations" are the most important ones

2.2.3.4.26. Any other potential capabilities for the web mapping framework that you would like?

The most popular suggestions are the following:

- Provision of WPS
- 3D maps
- Time series data and visualization
- Custom map building

2.2.3.4.27. Regarding the web mapping framework, please rate the importance of the following

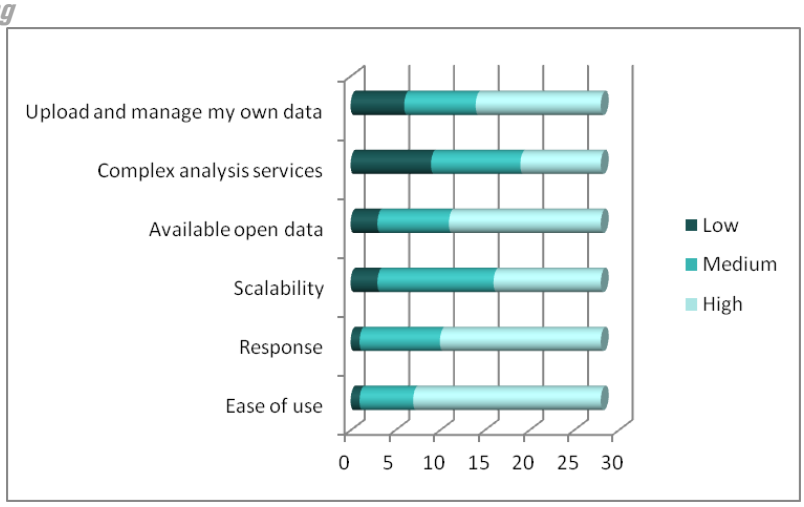


Figure 62: Chart of response distribution on question "Regarding the web mapping framework, please rate the importance of the following"

Figure 62 shows that the most important properties of the web mapping framework are once again, its **ease of use**, its **response** and the fact that it **provides open data**. The least important property is the capability of performing complex analysis.

2.2.3.4.28. Open geospatial data should be available in all possible file formats. Do you agree?

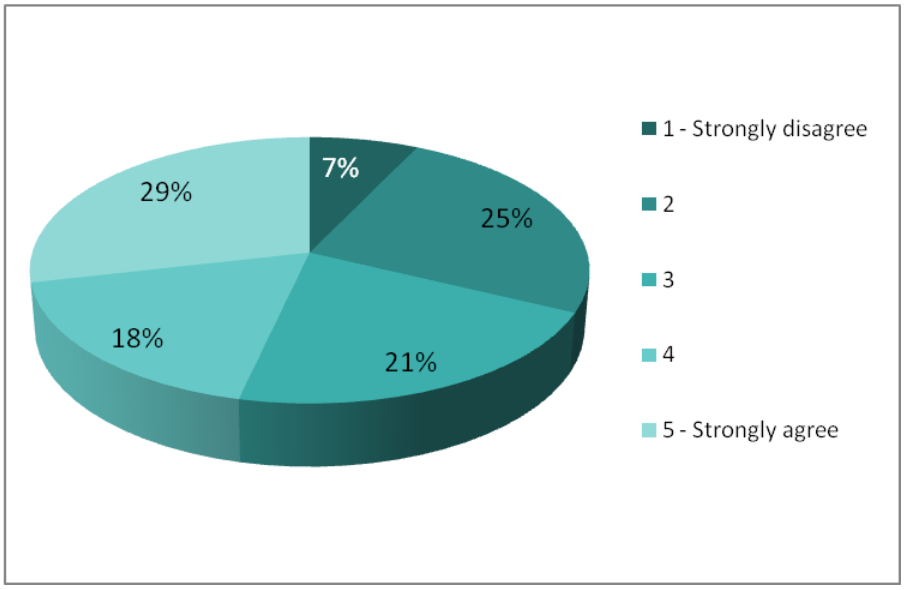


Figure 63: Chart of response distribution on question "Open geospatial data should be available in all possible file formats. Do you agree? (Developers)"

The developers are divided on the possibility that open geospatial data should be available in all possible file formats. 47% of them agree but another 32% disagree. Apparently they believe that it is not always required since they have the capacity to perform their own transformations.

2.2.3.4.29. Open geospatial data should be available in all possible Coordinate Reference Systems. Do you agree?

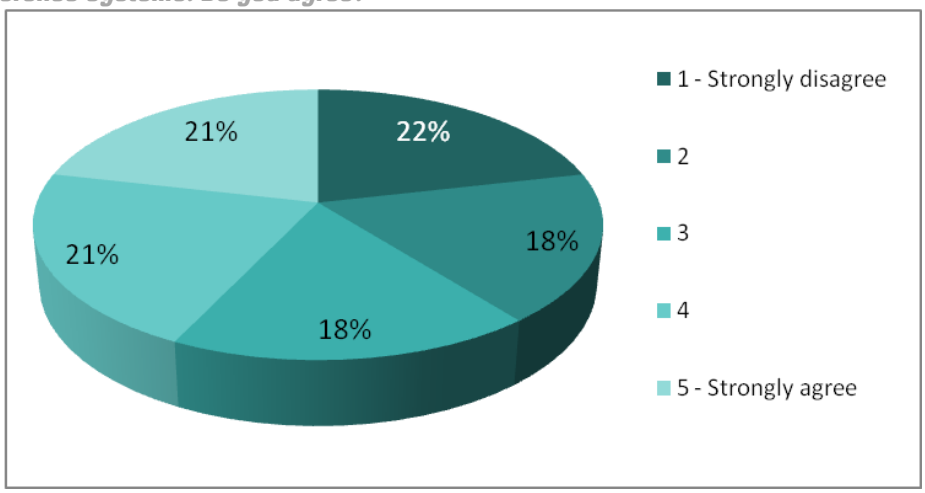


Figure 64: Chart of response distribution on question "Open geospatial data should be available in all possible Coordinate Reference Systems. Do you agree?"

The developers are divided once again. 42% agree that open geospatial data should be available in all possible Coordinate Reference Systems while 40% disagree and 18% of them are indecisive. Once again the developers do not rate important that feature since they have the expertise to carry out these transformations themselves

2.2.3.4.30. Open geospatial data should be available in various languages. Do you agree?

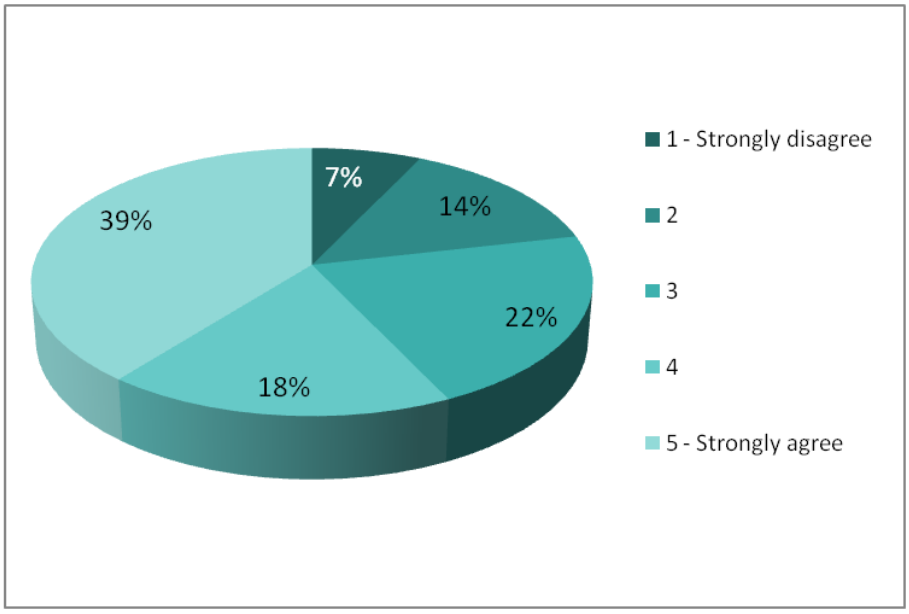


Figure 65: Chart of response distribution on question "Open geospatial data should be available in various languages. Do you agree? (Developers)"

In this case a trend is more observable. 57% of the developers agree that open geospatial data should be available in various languages, while 21% of them disagree.

2.2.3.4.31. Open geospatial data should be available in interactive maps. Do you agree?

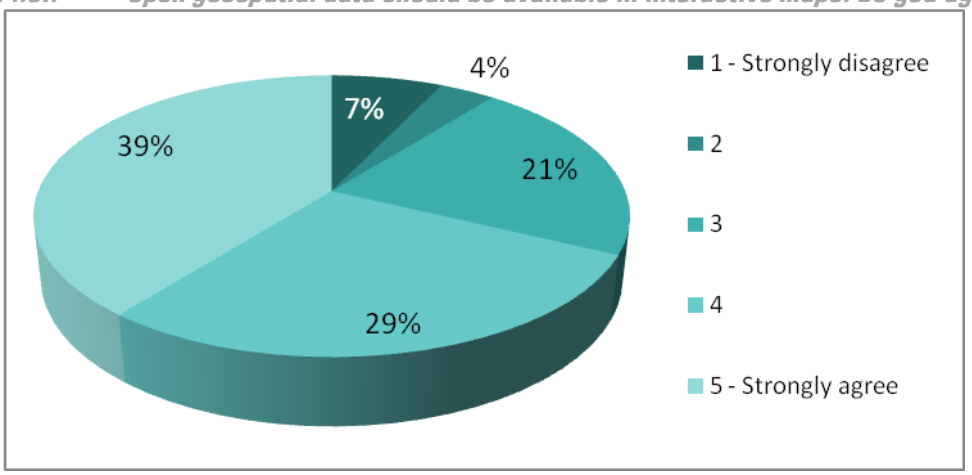


Figure 66: Chart of response distribution on question "Open geospatial data should be available in various languages. Do you agree? (Developers)"

The majority of the developers (68%) **believes in interactive maps and agrees that open geospatial data should be available in them**. Only 11% of them express their disagreement.

2.2.3.4.32. Open geospatial data should be provided to developers through open APIs. Do you agree?

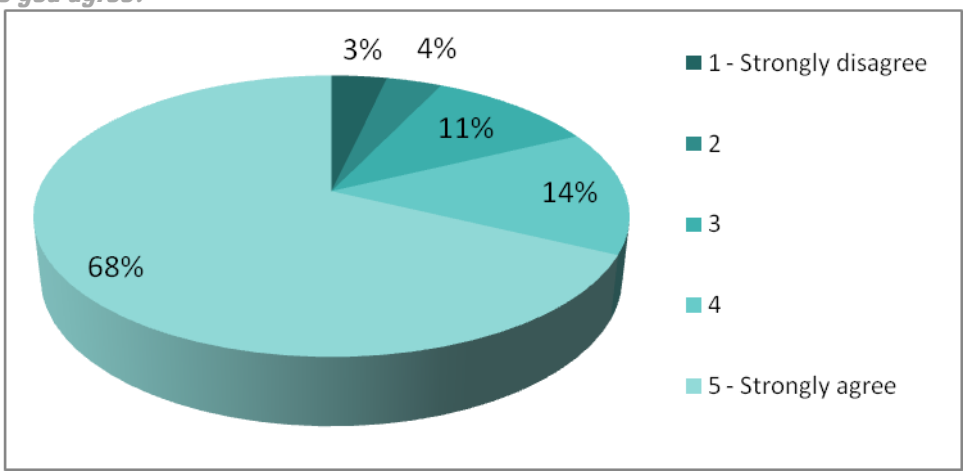


Figure 67: Chart of response distribution on question "Open geospatial data should be provided to developers through open APIs. Do you agree? (Developers)"

As it was expected **developers are avid supporters of open data APIs**. The great majority of them (82%) agree that open geospatial data should be available through open APIs.

2.2.3.4.33. Final question! If you had to ask for ONE feature to be developed from PublicaMundi, regardless of time, budget and current technology, what would that be?

Some of the suggestions are the following:

- "A query language that integrates all kinds of data (meta, vector, raster, meshes, social network graphs, etc), with a scalable processing engine underneath and a wealth of trivially pluggable client components for any platform and any purpose."

- "Adopt ODC-PDDL OpenDataCommons Public Domain License and Dedication as the single license for all government published open data sets."
- "Linked open data API (but WITH data)"
- "Make data available with a license suitable for other developers."
- "OpenStreetMap usage"

2.2.4. Conclusions

As a whole, survey participants are knowledgeable in open data and proponents of open data publishing. Further, they are experienced in using open data and rate open geospatial data as one of the most important open data kinds.

2.2.4.1. Users

Users report that searching for open geospatial data is sometimes challenging. They face difficulties in finding data for specific region and thematic categories of interest, and they attribute this difficulty to poor metadata. They suggest the use of better metadata standards, more rigid data catalogues and data interlinking.

Further, users declare that using open geospatial data is somewhat easier than searching for them but still a cumbersome task. The most common problems they encounter are related with character encoding, Coordinate Reference Systems, and software compatibility. Also, lack of rich and valid metadata hinders data's use.

Most users exploit geospatial data for professional reasons. Data is used for Spatial Analysis, reference, (static/dynamic) map building and Numerical Analysis. Their preferred file formats are "Shapefile" and "GeoTIFF" while data are frequently requested as "KML" and "CSV" files as well.

The majority of users believe that open geospatial data should be available in all possible file formats, Coordinate Reference Systems, in various languages and in interactive maps. They also strongly support the idea of being able to upload their own data and create their own custom maps.

Finally, users participated in this survey appear to be well informed regarding the cost of producing open geospatial data.

2.2.4.2. Publishers

Open geospatial data publishers declare that publishing an open geospatial data set is a relatively easy task. However, the time needed may vary

depending on the dataset particularities and complexity. Usually problems are encountered while data are cleaned, metadata are created and secondarily, when transformations in other file formats are performed.

Publishers who participated in the survey declared that they publish both vector and raster data. The most common reasons for not publishing raster data are licensing problems, and data's high complexity regarding the underlying infrastructure. Publishers also report that transforming data in other formats and Coordinate Reference Systems are easy tasks.

Publishers provide metadata for geospatial datasets in various standards with INSPIRE being the most prominent. They also report that the metadata creation process itself is rather easy.

Data Publishers do not tend to make the greater part of their data available in interactive maps. The most frequent reasons are system capacity limitations, inadequate resources/infrastructure to handle the workload and the fact that publishers are not obliged to do so. Some of them believe that this is not useful for the users.

Data are typically available through a WMS API and secondarily through WFS, CSW, WCS and custom JS APIs. When APIs are not provided the reasons are: publishers are not obliged to do so, they have inadequate infrastructure, and they believe that this is not useful for the users.

Concerning analytics, publishers report that they mostly keep typical types of analytics, as "number of daily visitors", "map views", and "downloads per dataset".

Publishers are divided over the possibility that open geospatial data should be available in all possible file formats and Coordinate Reference Systems. For the former case it is probably believed that it is a tedious task and not part of their publishing process. For the latter, publishers seem to believe that there is not much use in providing datasets into all possible CRSs apart for the national ones and certain global ones (WGS84). On the other hand, the majority agrees that open geospatial data should be available in various languages and available in interactive maps and through open APIs.

2.2.4.3. Developers

Developers participated in the survey develop primarily web applications and secondarily desktop and mobile ones. They are experienced using of web mapping frameworks and with the basic tasks of handling geospatial data (working with shapefiles, transforming geospatial data in other file formats and CRS, performing geospatial SQL queries and spatial analysis with desktop GIS). They also declare experience in creating Google Maps mashups, installing mapping servers and consuming OGC services.

Developers appear to be avid supports of open data as they report using quite frequently open data in their applications (e.g. POIs, administrative, environmental, census financial and transportation data).

Concerning the problems they face when using open data, they highlight that data usually do not have sufficient granularity, are not up to date, are accompanied by poor metadata, and their quality is bad in general.

The developers also report that they frequently use maps in their applications and in the rare cases in which they don't is because it is not deemed necessary, implementation difficulties or due to lack of experience. Usually developers prefer Open Street Map tiles and secondarily Google Maps, Bing Maps, MapBox and MapQuest tiles. Their favorite Map servers are Google Maps, GeoServer and Map Server while custom solutions are also used.

Developers express their willingness to use a free API providing querying and analysis services for open geospatial data. For them it is most preferable to be a JSON and secondarily a WFS, WMS, XML or SPARQL/GeoSPARQL one. The API's most desired capabilities are executing simple lookups, simple queries, simple spatial queries, and complex spatial queries. The developers rate a data API primarily from the fact that it provides open data, from its scalability and its ease of use.

Developers are willing to access and use a free web mapping framework for developing interactive maps. Most preferably this framework should be based on OpenLayers and secondarily on Leaflet and Google Maps. The most preferred capabilities of this framework are the ability to create a map based on a combination of open geospatial data and their own data, and the ability to create visualizations. Developers also would appreciate additional capabilities such as WPS and 3D maps. The developers rate a web mapping framework based on its ease of use, its response and the fact that it provides open data.

Developers, just like data publishers, are divided over the idea of open geospatial data being available in all possible file formats and CRS. On the other hand they agree with the idea that open geospatial data should be available in various languages, in interactive maps, and through open APIs.

2.3 SYSTEM REQUIREMENTS

The feedback received by the interviews and the surveys was formulated into specific system requirements.

2.3.1. Searching for datasets and retrieving information

	Title	Description
REQ1.1	Searching data for a region	Searching for datasets which are related with a user specified geographical region should be supported.
REQ1.2	Searching data for a thematic domain	Searching for datasets which are related with a user specified thematic domain should be supported.
REQ1.3	Searching data for a time period	Searching for datasets which are related with a user specified time period should be supported.
REQ1.4	Searching data in a specific language	Searching for datasets in a specific language should be supported.
REQ1.5	Determining license type	A dataset's license type in particular and whether it is open or not in general should be clearly stated.
REQ1.6	Determining file type	The file types of the resources of a given dataset should be clearly stated.
REQ1.7	Providing map previews	A map preview should be provided for a given geospatial dataset.
REQ1.8	Provision of interlinked data	The data catalogue should be able to cater with interlinked datasets.

Table 6

2.3.2. Using geospatial data resources

	Title	Description
REQ2.1	Resources' recognizable encoding	Data resource files should have recognizable character encoding depending on the software used for their processing.
REQ2.2	Metadata in various standards	Geospatial datasets' metadata should be available in various metadata standards.
REQ2.3	Resource provision into various formats.	Data resources (vector or raster) should be provided into all possible file formats.
REQ2.4	Resource provision into various CRSs.	Data resources should be provided into all possible Coordinate Reference Systems.
REQ2.5	Resource provision into various languages.	Data resources should be provided into various languages.
REQ2.6	Data available in interactive maps	Data should be available in interactive maps enabling users to combine them with other datasets and geospatial layers.
REQ2.7	Creating custom maps	Users should be allowed to upload their own data and create custom maps.
REQ2.7	Provision of data APIs	Datasets (vector or raster) should be provisioned through various data APIs
REQ2.8	Data processing	Users should have the ability process and download a copy of an original dataset.

	Title	Description
REQ2.9	Web mapping framework	A web mapping framework should be available to developers to develop their own interactive maps.

Table 7

2.3.3. Publishing geospatial data sets

	Title	Description
REQ3.1	Facilitating metadata creation	The dataset catalogue should provide all required forms for metadata creation for a number of supported standards. Publishers should be able to create (or import) metadata in the standard of their preference and metadata could be automatically transformed to the other supported standards thereupon.
REQ3.2	Multilingual metadata creation	Publishers should be able to create multilingual versions of metadata
REQ3.3	Multilingual data creation	The data catalogue should provide the publishers the means to create multilingual versions of the dataset resources.
REQ3.4	Data interlinking	The data catalogue should provide the publishers the means to interlink their data with other datasets.
REQ3.5	Automatic data resource transformation in other file formats	Data resources should be automatically transformed in other file formats without the publisher's intervention.
REQ3.6	Automatic data resource transformation in other file Coordinate Reference Systems	Data resources should be automatically transformed in other Coordinate Reference Systems without the publisher's intervention.
REQ3.7	Semi-automatic map creation	A dataset's import into the catalogue's interactive map should be semi-automatic requiring only minor intervention by the publishers as setting map styling parameters.
REQ3.8	Raster data publication robustness	System should be robust enough to handle the workload of raster data publication.
REQ3.9	API provision robustness	System should be robust enough to support the provision of various data APIs.

Table 8

2.3.4. Analytics

	Title	Description
REQ4.1	Analytics	The overall system (data catalogue, interactive maps, data APIs) should keep detailed analytics on the users' behavior and preferences.

Table 9

3 SYSTEM ARCHITECTURE

This chapter provides the System Architecture of PublicaMundi, which consists of several *application levels* described below.

Within the PublicaMundi project we envision seven application levels, from the top level client applications to low level data storage components. In between, the system follows a multi-tier, multi-level and multi-user approach aiming to provide tools for all stages of the open data lifecycle.

The purpose of this multi-layer architecture is to be able to adapt PublicaMundi to specific deployment needs (*e.g. different map servers, deploy other CKAN extensions if needed, deploy on top of a database cluster*), or even to be able to change a specific component of the system without affecting the rest of the components. This makes the current architecture extensible for future improvements. We envision a system that will be built on many loosely coupled layers, in order to be able to scale and be deployed in a modular way, according to the end user needs.

More specifically, the application levels are:

- Data Storage. This layer includes all the storage units, database clusters and cloud storage infrastructures that will be used/developed in PublicaMundi.
- Data Processing. This layer includes all the spatial software that is responsible to transform and process spatial data (vector, raster and their metadata) between storage and the core CKAN application that publishes those data on the web.
- CKAN (core). The core application is based on the well-known and widely used open data catalogue CKAN. CKAN is based on the Pylons Web Framework, which follows the *Model, View, Controller* approach (MVC). This approach gives the advantage of loose coupling of resources and software components, which leads to great re-usability of the source code and better management of the stack. This layer includes only the core CKAN application. It is the base catalogue that will be used and deployed on the PublicaMundi system.
- Application Modules. This layer consists of all the CKAN extensions that are going to be developed in order to spatially extend CKAN but also to provide new functionality such as multilinguality, interlinking etc. Also, some extra modules are going to be developed in this layer to enhance scalability of the application (caching, proxy and analytics).

- Web Services. This layer consists of all the geospatial web services that are going to work along with CKAN for providing geospatial data on the web. These web services are OGC compatible and will be implemented using software from the OSGeo stack.
- Developer API's. This layer consists of all the development tools and API's that are provided to developers in order to: create maps, re-use data, process data and publish/search/harvest metadata.
- Client Applications. This layer consists of external applications that will use PublicaMundi's APIs and interfaces in order to interact with open data.

Figure 68 provides a detailed overview of the system architecture, from the lower level (data storage) up to the higher level of the client applications and system APIs.

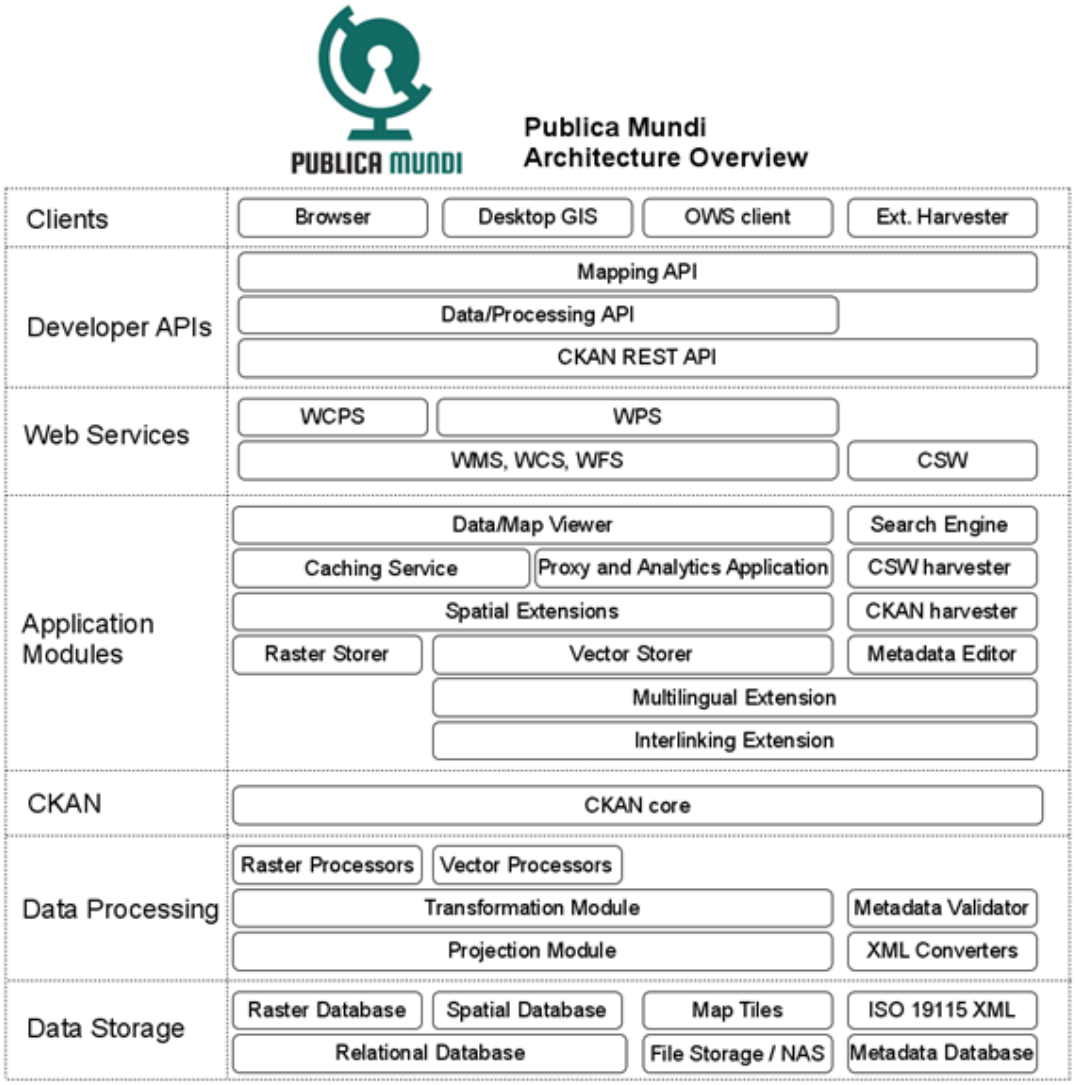


Figure 68: Logical Architecture Overview

In the following we elaborate the specific modules and components identified at each layer of the PublicaMundi architecture.

3.1 DATA STORAGE

The first layer of PublicaMundi components is the data storage layer. This layer consists of the following modules:

- Relational Database. This is the primary RDBMS of the system which will mainly store CKAN's application data model and array database storage elements. As this is a core part of the whole system, the deployment of the database server should follow a proper replication/failover scheme (master-slave) to ensure scalability and high-availability.
- Vector Database. It is based on the Relational Database which will be spatially-enabled to handle vector data. The vector data are stored in native geometry types and attribute tables in their original form and reference system. The database is OGC-SFSQL compatible, supporting all the basic geometry types and a variety of spatial operators and stored procedures for spatial analysis.
- Raster Database. This is an array database engine on top of an RDBMS, in order to be able to store and query multi-dimensional data arrays. Under this component is the storage of (raster) coverage data in an efficient and scalable manner. A key feature of the array database is the configurable tiling scheme for its backend storage, which can be used to provide tailored performance enhancement, enabling the execution of fast search and retrieval of parts of n-D raster datasets in any dataset dimension.
- File Storage. This is the simplest type of storage, based on a Network File System or a Cloud SaaS (Storage as a Service) solution. The uploaded datasets (spatial or not) are stored as plain binary files (blobs) to the storage and they are linked to the metadata entries.
- Metadata Database. This is a document-oriented database system for storing metadata files (XML based). The XML documents are stored and indexed into this database component, and associated to a unique identifier by a key-value pair, in order to be efficiently retrieved and translated in the system. Additionally, parts (i.e. fields) of the document are analyzed/tokenized, and indexed in order to be subject to several types of queries (e.g. full-text, numerical).

3.2 DATA PROCESSING

The second layer of PublicaMundi components is the data processing layer. This layer consists of the following modules:

- Projection module. This is a software library and command line interface utility which provides a way to re-project geospatial data between map projections and reference systems. It is used as a base library in many software components but also as a separate component to provide on-the-fly data re-projections.
- Transformation module. This is a software library and command line interface utility which provides a way to support many geospatial data formats and a utility to transform a dataset to the required format by the user. It is part of many server and client software components and is also used as standalone transformation and processing module.
- Raster Processors. These are software components, libraries, scripts and algorithms that are linked with the WPS services in order to provide a platform for raster data processing within PublicaMundi. A collection of image processing algorithms are provided in the form of linked open source software. The option to have WCPS service to encapsulate these algorithms is envisaged for investigation.
- Vector Processors. These are software components, libraries, scripts and algorithms that are linked with the WPS service in order to provide a platform for vector data processing. Through its OWS interface, WPS offers users a variety of vector processing tools to select and apply to geospatial data.
- ISO XML Converters. This module consists of a python metadata library, supporting the model of ISO 19115 and its mappings to all other known metadata standards (FGDC, DCAT, INSPIRE etc). A CKAN plugin will extend the CKAN database schema in order to support ISO 19115, and will also provide convenient wrappers to convert between metadata (sub) schemas.
- Metadata Validator. This component is able to retrieve the ISO XML metadata files from the core system's database and validate them against XSD schemas. The official standard schemas are supported. At the same time, this module provides validation results to the metadata editor UI, in order to help the end user add/edit metadata records.

3.3 CORE APPLICATION

The third and main layer of PublicaMundi is the core CKAN application component. This layer consists of the following Pylons MVC components:

- Models. These are the database schemas translated to pure object-oriented python classes, so to be able to interact with the database from the application level without the need of SQL scripting. These classes are defined by the application (or the extensions) and control the complete lifecycle of database objects (update, select, delete, insert etc.).
- Views. These include the XML/HTML/Text templates used by the application (or the extensions) in order to define the User Interface (UI). The templates are created in a way to be re-usable and to achieve separation of the UI from business logic. They are modified, in a controlled manner, with injected python-like code and generate the final output sent to the end-user.
- Controllers. These are python classes that implement the business logic of the application and the extensions. The controllers are invoked when a new HTTP request has reached the server, and through URL mapping (i.e. routing), a controller acts like an entry point to the application. The controller interacts with the models and the views (templates) in order to process the request and render the result to be sent back to the end user.

3.4 APPLICATION MODULES

This layer consists of the following modules:

- Interlinking extension. This module is implementing the interlinking support for the data/metadata records in the CKAN database. It implements new database schemas in order to facilitate the interlinking algorithms but also user-defined linking. At the same time, this module defines the interfaces that the JavaScript Data API should implement in order to exchange information on the client side.
- Multilingual extension. This extension implements multilingual support for metadata and certain kinds (e.g. tabular) of data. Implies modifications to the database schema in order to hold translated metadata field values and data columns. This also includes a versioning schema in order to track different versions of the fields and compare changes. On top of the database, several controllers

handle the way a user interacts with the UI while providing translations, and also the way to recreate full versions of data columns or metadata fields, based on the selected language and the completeness of the records.

- Raster storer. This is an extension to the queued processing of uploads, in order to be able to handle raster/coverage data files. This extension hooks into this processing, recognizes spatial attributes and invokes the needed procedures so that the uploaded resource is stored in the raster database and/or optimized in a tile cache. Beyond 2D "layers", the coverage data model allows for higher dimensionality, including time series; thereby 3D raster coverage storage will be investigated for inclusion.
- Vector storer. As with the raster storer, accepts uploads of vector files and analyzes them in order to extract metadata and information before importing them to the spatial database.
- Spatial extensions. This module is implemented by two spatial extensions, the default CKAN spatial extension [CKAN-SP] and a PublicaMundi spatial extension, which adds full geospatial functionality to the catalogue. This is one of the major contributions of this project to the CKAN system. The new extension incorporates well known spatial libraries working as system controllers, in order to identify, locate, discover and view spatial information.
- Metadata editor. This is the component of the PublicaMundi system where the end users can edit metadata records for datasets and resources (either uploaded or simply referenced as URLs). This editor supports all metadata schemas the system is aware of, and can be extended through an API to support new developer-defined schemas. This extension provides a way to define metadata fields and accompanying sets of conversion rules that must be applied at ingress/egress in a per-field or per-record context. It is also heavily linked with the metadata validation module and the translation engines in order to make it easier to the user to provide only the needed fields, before creating valid (according to schema standards) metadata documents.
- CKAN harvester. This extension [CKAN-HRV] of the CKAN system is used to implement a generic metadata harvester, supporting several metadata standards. The purpose of this module is to harvest records from third party catalogues and include the results in local searches that the user defines. The harvested records are stored

within the core database and can be translated, validated, converted and edited (yielding new versions). Generally, the CKAN harvester should be able to recognize the spatial nature of a given (external) catalogue and delegate to the CSW harvester.

- CSW harvester. This is a specific kind of harvester that supports the well-defined and very popular spatial catalogue specification. It is used as the default harvester for spatial metadata and can directly harvest from other CSW compatible catalogues. Also, it provides the functionality of federated searching through other catalogues, acting as a client and presenting the remote search results as if they were local. For reasons of efficiency and to avoid needless database synchronizations, the CSW component interacts (through a similar ORM layer) with the same database objects as core CKAN application.
- Search Engine. This component is tightly coupled with the metadata database and its main purpose is to speed up queries on metadata. Thus, among its core tasks is to maintain type-aware indices and preprocess textual fields (e.g. tokenization, stemming) in order to participate in full-text queries
- Proxy and Analytics application. This module is a middleware application that serves the following basic purposes:
    - Logs and analyzes client requests. The access log is analyzed by a daemon process to extract valuable information on what is being requested and on which frequency (e.g. which map layer if WMS, which coverage-id if WCS, what was the bounding box). The statistics are focused on geospatial web services and will determine if certain results should be cached or if scalability parameters have to be tuned (e.g. by starting new server instances). The analytics component is aware of the type of contained services, i.e. it needs to understand part of the incoming queries in order to extract requested information (e.g. WCPS sub-setting, both in domain and range)
    - Acts as an authentication layer to backend services
    - Acts as a reverse proxy and a load balancer. The proxy distributes incoming requests to backend services based on their (estimated) current workload.
    - Monitors backend services and provides a failover/failback solution.
- Caching service. This module serves as a middleware, tightly coupled with the proxy and analytics application. It creates a local cache of computed results (mainly coming from the web service backend) to

provide instant responses to requests commonly asked by users (e.g. WMS layers). This service is also coupled with the file storage, where a caching scheme especially for raster datasets (Tile Cache) will be implemented.

- Data and Map Viewer. A JavaScript application that provides the UI required for visualizing raster, vector and tabular data on the client side. It has advanced mapping capabilities and full OWS support. Also used as a UI for data/metadata interlinking. Existing 3D visualization libraries that can connect to PublicaMundi services will be investigated for inclusion (e.g. X3DOM based tools).

3.5 WEB SERVICES

The fifth layer of PublicaMundi components is the Geospatial Web Services layer. This layer consists of the following modules:

- WMS Interface. Open source applications able to serve raster and vector data through a Web Map Service interface. WMS 1.1 and 1.3 are supported. The technology behind this module can be any compliant WMS server. For PublicaMundi deployment, a configuration interface is required (preferably RESTful) to change service contents and layer configuration dynamically, including server-level processing capacity scaling.
- WFS Interface. Open source applications able to serve vector data through a Web Feature Service interface. WFS 2.0 is supported. Any WFS compliant server with REST configuration interface can be the backend. This service can also work as a transformation service, providing data at the required format.
- WCS Interface. Open source applications able to serve coverage data through a Web Coverage Service interface. WCS 2.0 is supported on raster coverages. This is a user download and viewing service for raster data, allowing download of data in their original form (e.g. a multiband dataset with rich metadata) and also RGB map tiles. For PublicaMundi deployment a configuration interface is required (preferably RESTful) to change service contents and coverages configuration dynamically, including server-level processing capacity scaling.
- CSW Interface. Geospatial Catalogue component, fully implementing CSW specification, providing the CSW interface of PublicaMundi. The CSW server is tightly integrated into the CKAN database, searching

and serving based on the same data. Also provides transactional interface for third party applications to be able to edit the catalogues records if required.

- WPS Interface. The Web Processing Service module is an independent component that provides the processing functionality for all geospatial data included in PublicaMundi database. It supports several programming languages and can execute processes on the server side of the application, removing the need of the user to download the data and process them locally. WPS abstracts the processing engines through the OWS interfaces, with no restrictions on the backend spatial technologies used. WPS can also process non spatial data.
- WCPS Interface. The Web Coverage Processing Service is an extension to the WCS service, specifically designed to enhance coverage services with query language processing on (raster) coverage data. This module is tightly linked with the raster database, adding the capability to execute complex raster and image processing algorithms on top of the original data stored within the database. It also provides multi-dimensional support and time series analysis. For PublicaMundi deployment, a configuration interface is required (preferably RESTful) to change service contents and coverages configuration dynamically, including server-level processing capacity scaling.

3.6 APPLICATION DEVELOPMENT APIS

The sixth layer of PublicaMundi components is the Application Development APIs layer. This layer consists of the following modules:

- Data and Processing API. This module is a JavaScript wrapper library around the data and processing interfaces, provided by the corresponding modules. For example, the WPS JavaScript API can be called directly by the PublicaMundi API through its wrapper classes. This way an application developer only needs to know one API that invokes processing on the data, regardless of the server side implementation of the processing algorithms. Also, this module heavily extends CKAN's Data API and provides geospatial functionality to it.
- Mapping API. This is a wrapper JavaScript library around well-established mapping frameworks used within PublicaMundi. This

mapping API is designed to be very simple to use, with a few lines of code and easy to embed into a custom web application. Including more than one base framework, solves the problem of scalability, and will support multi-modal application development/integration.

- CKAN API. This is an HTTP API provided by the core of CKAN application. It will be extended to handle spatial functionality (e.g. query criteria based on OGC filtering).

3.7 CLIENTS

The seventh layer of PublicaMundi components is the Client layer. This layer consists of the following use cases:

- Web Browser and SDIs. Any web browser can access CKAN's and PublicaMundi UI, without restrictions on technology, or need for external plugins. Especially SDIs can access the catalogue's open data for discovery, harvesting, downloading or viewing through the OGC OWS interfaces. Also, maps can be embedded to web applications through the mapping API.
- OWS Clients. All OGC compliant applications can access the open geospatial data offered from PublicaMundi. Following the well-established open standards and implementing them on their latest version, makes PublicaMundi widely connectable to any spatial technology available today.
- Desktop GIS. Traditional GIS desktop applications can benefit from download, discovery and view services as provided by PublicaMundi. GIS specialists can download, or even process online the data they need for their application, and use PublicaMundi facilities for serving scalable maps and tiles.
- External Harvesters. Since PublicaMundi and CKAN are open data catalogues, it is expected that they will be harvested for their data and metadata by other catalogues. This will be achieved through the Data API and the CSW interface.

4 COMPONENT ARCHITECTURE

This section documents the major system components, their responsibilities, how they are related, their interfaces, the associated strategic design decisions and their rationales.

4.1 SOFTWARE COMPONENTS

This subsection documents the major software components of the system

4.1.1. CKAN

Definition
CKAN is a powerful data management system that makes data accessible – by providing tools to streamline publishing, sharing, finding and using data. CKAN is a Python web application with some basic components.

Responsibilities
CKAN will be the catalogue of PublicaMundi, integrating the majority of components and sub-systems developed. Its basic responsibilities are to:

- Provide CRUD functionality (UI/API) with datasets and resources.
- Provide a core set (extensible, globally or per-dataset) of metadata.
- Provide grouping and tagging for datasets
- Suggest a data-publishing workflow (private, draft, public, deleted) for datasets.
- Provide storage capabilities (UI/API) for resources (as blobs)
- Provide a distributed organization-centric authorization model
- Provide search capabilities (UI/API) for metadata and certain kinds of data (e.g. tabular).
- Provide a harvesting class interface as a base for implementing specialized harvesters.
- Track history of changes (datasets and resources as versioned entities)
- Provide unique identifiers (UUIDs) that accompany both datasets and revisions.

Relationships to externals and other components

- Geospatial Catalogue Service (see §4.1.6)

- CKAN extension Layer (see §4.1.5, §4.1.7, §4.1.8, §4.1.10 and §4.1.11)
- Application Development API (see §4.1.13 and §4.1.14)
- Client Layer (see §4.1.23, §4.1.25 and §4.1.26).

Internal interfaces

- CKAN API: is a powerful, RPC-style API that exposes all CKAN's core features to API clients.
- Pylons MVC Components Model: manages the behavior and data of the application domain, responds to requests for information about its state and responds to instructions to change state.
- SQLAlchemy: is used to model our data into a relational database and handle them in an object-oriented manner.
- Views: manages the presentation of the model. The view is the interface the user sees and interacts with. Template rendering engines are a popular choice for handling the task of view presentation.
- Directly-supported template engines: Mako, Genshi, Jinja2
- Controllers: interpret requests from the user and calls the appropriate methods and generates the view to be sent as output to the user.

Strategic design decisions and their rationales

Basic data catalogue

4.1.2. Proxy/Balancer/Analytics module

Definition

This module is a middleware application that acts as a frontend to the provided web services.

Responsibilities

- Log incoming requests to backend services and provide aggregate usage analytics
- Monitor backend services and provide alerts and a failover/failback solution
- Proxy and distribute requests according to monitored workload
- Scale instances of backend services (dynamically) according to monitored workload
- Enforce authentication

- Decide which requests are cacheable in terms of 3 basic factors: (a) popularity, (b) computation load and (c) density of input space.

Relationships to externals and other components

- Proxies requests to contained web services (see §4.1.6, §4.1.15, §4.1.16 and §4.1.17)
- Drives the caching service (see § 4.1.24), based on collected statistics processed by its internal analytics engine
- Drives scalability factors (leveraging control interfaces of elements behind proxy)

Interfaces

- (optional) API for collection of processing metrics from the other components responding to queries
    - executed query (to match with logged entries)
    - Actual stats regarding the host (e.g. memory usage, disk usage and throughput, cpu usage, number of processes/threads) and the service itself (e.g. average response time per type of request)

Strategic design decisions and their rationales

- An API for collecting processing statistics will be used by components behind a proxy so that the analytics engine can leverage such extra information in deciding for scalability configuration and load balance.

4.1.3. Spatial extensions

Definition

PublicaMundi will extend CKAN's spatial capabilities by improving the current spatial extension [CKAN-SP] but also provide a new spatial extension with extra functionality. The CKAN spatial extension adds a spatial field to the default CKAN dataset schema. This allows performing spatial queries and displaying the dataset extent on the main CKAN application. It also provides harvesters to import geospatial metadata into CKAN from other sources, as well as commands to support the CSW standard. It also includes plugins to preview spatial formats. The PublicaMundi spatial extension will provide a full OGC OWS client integration, as part of integration with external OGC Servers, allowing the users to upload, manage, discover, view and publish geospatial data within CKAN.

Responsibilities

- Implementation of the spatial publishing workflow.

- Add and consume spatial fields and functions to the database schema.
- Previewing of spatial datasets
- Discovery of spatial datasets though CKAN searches
- Extends CKAN API with spatial functionality

Relationships to externals and other components

- Interaction with raster (see §4.1.10) and vector storers (§4.1.11) to publish vector and raster data to the backend OGC Servers
- Depends on Harvester extension (see §4.1.5)
- Depends on Data/Map Viewer (see §4.1.12) to visualize geospatial data within CKAN
- Linked with Metadata Editor (see §4.1.9) for the metadata publishing workflow
- Heavy use of Spatial Database (see §4.2.1.1, §4.2.1.2, §4.2.1.3, and §4.2.1.4) for storing spatial data and metadata

Interfaces

- Provides Python API for OWS client capabilities
- Spatial CKAN API
- Configuration REST API

Strategic design decisions and their rationales

- The spatial extensions will tightly integrate and extend current OWS client libraries.
- OGC web services support will be generic in order to support many back-ends (MapServer, GeoServer, Rasdaman etc.)

4.1.4. Search Engine

Definition

The search engine is a background service that is responsible to efficiently handle expressive queries on metadata fields. The engine is based on a suitable document-oriented database that also understands the notion of fields. The set of known CKAN fields is mapped (not mirrored) to a set of searchable fields.

Responsibilities

- Maintain type-aware indices of known fields
- Provide Full-Text search capabilities for textual fields
- Provide near real-time indexing

- Provide index configuration options that follow supported metadata schemata (i.e. map CKAN fields to searchable fields)
- Deploy in a replication/failover schema to provide scalability

Relationships to externals and other components

- This service heavily depends on the defined metadata schemata, hence has a strong (though indirect) connection to the metadata-editor CKAN extensions (see §4.1.9 and §4.1.22).
- This service will mainly be invoked by the CKAN API (see §4.1.1) as part of its search functionality.

Interfaces

- Exposes an HTTP API that synchronously replies to queries by exchanging standard XML or JSON objects.
- Is configurable through regular configuration (e.g. XML or INI) files read at service startup

Strategic design decisions and their rationales

- The service should be unaware of the fact that is invoked from the CKAN API. It only understands field specifications declared in its own configuration.

4.1.5. Harvester extension

Definition

The harvester extension can automatically import ("harvest") datasets from multiple CKAN websites into a single CKAN website, and also provides a framework for writing custom harvesters to import data from non-CKAN sources. The CKAN harvester plugin makes it really easy to import datasets from a remote CKAN instance into a new CKAN instance. It is highly customizable, allowing defining default tags, groups, users and permissions for the imported datasets. In PublicaMundi the harvester will be extended to support more target sources (CKAN and OGC compliant web services).

Responsibilities

- Identify the nature of targeted sources (spatial or non-spatial)
- Harvesting of non-spatial metadata sources
- Harvesting of data from external sources

Relationships to externals and other components

- Delegates on CSW Harvester (see §4.1.6) for spatial harvesting operations
- Based on PublicaMundi spatial extension (see §4.1.3) for the OGC client capabilities

- Based on core CKAN application (see §4.1.1) since it is an official extension

Interfaces

- The harvesting extension provides a class interface for building custom harvesters. The interface defines 3 stages that must be implemented (as a corresponding method):
    - The *gather* stage compiles all the resource identifiers that need to be fetched in the next stage.
    - The *fetch* stage gets the contents of the remote objects and stores them in the database.
    - The *import* stage performs any necessary actions on the fetched resource.

Strategic design decisions and their rationales

- Due to limitations on spatial functionality of existing ckanext-harvest extension, it was decided to remove the spatial harvesting from the core extension and to create a new extension based on pycsw in order to perform OGC compliant CSW harvesting.
- The CKAN harvest will remain the core harvesting extension but will delegate to the new CSW harvester when a spatial source is detected.

4.1.6. CSW Interface/Harvester

Definition

This extension will provide a native CSW interface endpoint for CKAN. The OGC CSW 2.0.2 specification will be implemented with APISO profile and INSPIRE support. This interface will be tightly coupled with the CKAN database and will not be based on synchronization methods as previously done in the current CKAN spatial extension. This means that all native and harvested spatial metadata included in CKAN will be available through the CSW interface.

For the harvesting part, a full implementation of CSW-T and CSW harvesting will be provided in order for this to be used by CKAN in cases where a third party spatial catalog is about to be harvested. In that case the CSW interface will be triggered from the core CKAN harvest extension and the metadata will be imported to the database.

Responsibilities

- Provide advanced spatial search capabilities
- Support OGC standards
- Implement CSW harvesting for spatial catalogues.
- On-the-fly transformation between schemata

Relationships to externals and other components

- Metadata Editor (see §4.1.9) will provide the UI for metadata editing before the metadata are available through the CSW interface
- Metadata Validator (see §4.1.22) will provide validation of metadata records before they are imported to CSW
- Metadata Database (see §4.2.1.4) will store the metadata records, both CSW queryables and full XML records. The CSW interface will be serving data directly from that database.
- Metadata Converters (see §4.1.224.2.1.4) are used in case of an incoming CSW request asking for metadata records in alternative format (Dublin Core, DCAT etc.)
- The Harvester Extension (see §4.1.54.2.1.4) calls the CSW Harvester to harvest from external CSW catalogues.
- CKAN core database schema (see §4.2.1.14.2.1.4) will be used and extended so that the extra CSW queryables are stored in the object-relational model.

Interfaces

- HTTP interfaces (CSW 2.0.2, OWS 1.0)

Strategic design decisions and their rationales

- APISO profile will be used to store native metadata, since it is the most complete metadata specification available worldwide.
- DCAT specification support will be added, since it is becoming popular among CKAN deployments
- CSW-3.0 (currently in draft) will be investigated and implemented if finalized on time.

4.1.7. Interlinking extension

Definition

The interlinking extension will support the merging of two vector datasets based on semantically common fields, hence enabling users to explore their data, find relationships and gain valuable insight into it.

Responsibilities

- Support merging of resources that can be represented in tabular format e.g. relational database tables or comma delimited text files.
- Develop a set of scalar and spatial similarity functions for use by the interlinking process. Scalar functions will implement numeric expressions and string similarity metrics. Spatial functions will include distance and analysis operations.

- Configure, execute and manage interlinking process stages, namely, (a) resource selection, (b) declaration of a 1-1 mapping between fields of the selected resources, (c) selection of one or more similarity functions and assignment of the mapping fields to the functions parameters and (d) execution and management of the interlinking process. These stages and the corresponding parameters will be reflected to the interlinking library interface.
- Extend the database schema for storing the interlinking process results. Users will be able to review the generated dataset and optionally publish it to the CKAN catalogue.
- Expose the interlinking functionality to the client through an HTTP API. For the latter a JavaScript wrapper will be also developed.

Relationships to externals and other components

The interlinking component will interact with the following components:

- Vector Storer (see §4.1.11): Access to catalogue's vector data
- CKAN & CSW Harvester (see §4.1.5 and 4.1.64.2.1.4): Access to external data
- CKAN (see §4.1.14.2.1.4): Implementation of the controller for the HTTP API
- (Optionally) Metadata Editor (see §4.1.94.2.1.4): Support for publishing the results of an interlinking process as a new dataset
- Mapping API (see §4.1.144.2.1.4): Expose interlinking functionality to the CKAN data/map viewer
- (Optionally) Data Processing API (see §4.1.134.2.1.4): Expose the interlinking functionality as a WPS service operation.

Interfaces

- Programmatic interface in Python for executing interlinking tasks
- HTTP API for accessing interlinking functionality from the client

Strategic design decisions and their rationales

- Interlinking will be supported only for resources that are stored on the servers managed by the PublicaMundi system. If an interlinking process selects remote resources, these will have to be replicated to the PublicaMundi servers using the available harvest (CSW or CKAN) functionality. Processing data locally will increase performance and allow for better optimization at the vector store level.
- Interlinking process should support asynchronous requests since the merging of large datasets may be time consuming.

4.1.8. Multilingual extension

Definition

The multilingual extension will offer localization support for dataset metadata and tabular data resources. The appropriate user interface and controllers will be implemented for handling the translation process. In addition, the CKAN API for accessing metadata and data will be extended for supporting locale parameters in queries.

Responsibilities:

- Supports translation of metadata and data for multiple locales.
- Offers the UI components required for translating metadata and data for datasets and resources. Integrates these components to the CKAN UI.
- Handles the logical organization and storage of the translated metadata and data.
- Handles concurrency issues that may arise when multiple users attempt to translate the same resource. Implements a fine grained locking scheme at data row level.
- Supports versioning of translated metadata and data.

Relationships to externals and other components

- CKAN UI (see §4.1.1 and §4.2.1.4) for handling localized datasets/resources
- CKAN HTTP API (see §4.2.1.4) for accessing metadata/data
- CKAN Core/Storage components (see §4.1.1 and §4.2.2.1)

Interfaces

- Exposes an HTTP API as part of the CKAN API: a new set of actions will be defined (or a set of core actions will be overridden) in order to take into account a language of preference passed as part of a request.

Strategic design decisions and their rationales

- In order to avoid cluttering the UI the users will be able to translate a single property (field/column) at any given time. Another option is to set an upper bound for the total number of columns a user can translate at the same time.

4.1.9. Metadata Editor

Definition

Metadata Editor is a CKAN extension aiming to provide a rich UI interface for reading/editing metadata records within CKAN. Also, CKAN API actions will have to properly handle these extended metadata fields. This

application will use a plugin-based logic, so that the UI will adapt to the underlying metadata format declared for the dataset in question. This way, it will be fully extensible to new metadata standards and future developments. A major feature requested by end users is the simplicity of use, and this will be tackled with the implementation of heuristic rules to automatically extract metadata records from data, reducing complexity for the CKAN user.

Responsibilities

- Build the UI for reading/editing metadata records (i.e. datasets).
- Adapt CKAN API actions to take into account any extended metadata
- Provide a native (Python) programmatic interface to include future metadata standards
- Auto-complete metadata fields that can be inferred from data
- Be user-friendly and not require the end user to know about ISO 19115 or similar standards specifications.

Relationships to externals and other components

- Tied to CKAN core (as a CKAN extension) (see §4.1.1).
- Metadata Database (see §4.2.1.4) and Search Engine (see §4.1.4) will have to be aware of defined metadata schemas.
- Metadata Validation module (see §4.1.22) will be used to decide if the record is valid according to the followed metadata schema.

Interfaces

- Python interface to define new metadata schemas

Strategic design decisions and their rationales

- As previously mentioned, ISO 19115 core elements will be used as a base (as a superset) for editing spatial metadata records.
- For non-spatial records, CKAN metadata fields will be extended when necessary.
- Support for INSPIRE records will be implemented since it is a feature requested by many users of CKAN.

4.1.10. Raster storer extension

Definition

This is an extension to the core file uploading mechanism, in order to be able to handle raster/coverage data files.

Responsibilities

- This extension recognizes spatial attributes and invokes the needed procedures so that the uploaded dataset is stored in the raster database and/or optimized in a tile cache.
- Direct upload of 2D geo-enabled raster formats (e.g. GeoTIFF) into 2D coverages (spatial)
- (optional) Direct upload of 2D geo-enabled collocated rasters into time series (3D coverages)
- (optional) Customizable upload of other raster formats as expected from the user requirements analysis

Relationships to externals and other components

- Drives raster database (see §4.2.1.2) ingestion through its geo-enabled interface (either WEB or CLI)
- Invoked by CKAN (see §4.1.1) to handle storage of raster data

Interfaces

- A CKAN plugin interface is provided for loading geospatial rasters as coverages:
    - Input:
        - raster dataset location
        - additional metadata for loading as a coverage (e.g. timestamp, rangeType)
        - target coverage if the dataset is a partial update of a larger one
    - Output:
        - target coverage
        - operation outcome
- A CKAN plugin interface is provided for loading raw rasters as n-D arrays:
    - Input:
        - raw dataset location
        - array index extents
        - target array identifier if the dataset is a partial update of a larger one
    - Output:
        - array identifier
        - operation outcome

Strategic design decisions and their rationales

- Adoption of OGC coverage model for all raster data. Beyond 2D "layers", the coverage data model allows for higher dimensionality,

including time series. 3D raster coverage storage will be investigated for inclusion.

4.1.11. Vector storer extension

Definition

This module extends the upload and store capabilities of CKAN to manage vector data. This module accepts uploads of vector files of any supported type, automatically extracts metadata, detects native language and imports data to the spatial database.

Responsibilities

- Analyzing uploaded vector files/datasets
- Extracts metadata based on heuristics or analysis

Relationships to externals and other components

- Stores vector data to the Spatial Database (see §4.2.1.3)
- Updates OGC interface configuration (see §4.1.15, §4.1.16 and §4.1.17) to read and publish the new vector data as service layers
- Updates multilinguality module configuration (see §4.1.8) to setup required stores for translations

Interfaces

- spatial extension to CKAN API/REST configuration API

Strategic design decisions and their rationales

- All spatial data provided or harvested from download services are stored in the spatial database for reference.
- Vector storer is implemented as a separate CKAN extension so as not to impose storage responsibilities to the current CKAN spatial extensions.

4.1.12. CKAN Data/Map Viewer

Definition

CKAN data/map viewer is a collection of user interface components that will support the visualization of raster, vector and tabular data, as well as, the processing of data through a set of services that will be developed as part of other extensions to CKAN.

Data visualization will be enabled by several types of controls such as map, graph and data grid controls allowing the creation of various data views. A user may select multiple views for each resource. For instance, selecting a WFS endpoint will allow the user to render a map, create a graph, browse feature property values using a data grid or even combine two or more

different data views such as a graph with a data grid in order to create composite views.

Moreover, components for initializing service requests such as interlinking and WPS processes will be provided, allowing users to compose requests in an intuitive manner.

Responsibilities

- Offers controls for data visualization including maps, data grids, graphs etc.
- Provides controls for initializing interlinking processes.
- Supports chaining and execution of WPS requests
- Supports execution of WCPS requests
- (optional) Supports visualization of 3D-enabled data (e.g. Map over DEM)

Relationships to externals and other components

- Interlinking extension (see §4.1.7): Interlinking functionality will be implemented using the interlinking HTTP API
- Data Processing API (see §4.1.13): Support for WPS processing will be implemented using the JavaScript API.

Interfaces

- OGC web services interfaces (WMS,WFS, CSW)
- WPS interface
- WCPS interface
- JavaScript API

Strategic design decisions and their rationales

Currently CKAN relies heavily on the Recline.js library for rendering data. We are planning to extend Recline.js by developing additional backend implementations for new data sources, such as WFS, in order to support rendering of spatial data and to be consistent with the overall CKAN architecture. We are also going to investigate the implementation of new views for Recline.js for rendering WCPS request results.

The implementation of views and backbends will support loose coupling between the current Recline.js implementation and the use of other visualization libraries in the future. Finally, we are going to investigate the design and implementation of a plugin architecture which will be based on data viewers. Each viewer will advertise its capabilities by means of supported data formats and the user will be able to select any of the

available viewers for the dataset at hand. Such an architecture may be realized either as an extension of Recline.js or as an independent library.

Within CKAN there will be an advanced map viewer that will be based on the Mapping API library and will consume published open data through web services.

4.1.13. Data Processing API library

Definition

The Data Processing API Library is a JavaScript API which will allow developers to easily invoke remote processing on datasets published through PublicaMundi from their own applications. The API will expose functions giving the capability to list all available services, get detailed information about each service and execute services. Basic but highly customizable user interface elements will be included within the Data Processing API. As WPS is able to handle long running processes by design, the API will also give the capability to call custom callback functions at each status updates and at the end of the processing. In this way developers can easily integrate the Data Processing API to update their custom web interfaces.

The Data Processing API will contain a Service Designer web interface giving the capability to developers to build new services as a chain of available services (from Projection/Transformation libraries and Vector/Raster Processors) but also by adding logic statements to the chain. Developers can then store and export these services chains for future use in their own applications. The designer will consider the data source needed by the generated service in order to provide the adapted tools integrated as modules. The designer will also run any possible preprocessing on the data prior to executing the actual processing service. Similarly, functions will be exposed to integrate the result in the custom application that invoked the processing service.

Responsibilities

- Provide a JavaScript API to interact efficiently with WPS servers
- Provide basics User Interface Elements
- Provide a Service Designer able to integrate logic in the chain
- Give access to both the Projection and Transformation libraries (internally to PublicaMundi or from third party applications)
- Give access to both the Vector and Raster Processors (internally to PublicaMundi or to third party applications)

Relationships to externals and other components:

- Data Processing API (see §4.1.13) will be used by the Mapping API Library
- WPS, WMS, WFS, WCS (see §4.1.15 and §4.1.16) services offered by PublicaMundi infrastructure
- Projection and Transformation Libraries (see §4.1.18 and §4.1.19)
- Vector and Raster Processors

Interfaces

- JavaScript API

Strategic design decisions and their rationales

- Despite the Data Processing API will be mainly used to request the ZOO-Kernel in PublicaMundi, it will have to be compatible with all other implementations available which is respecting the standard definition, this way we may think of third party applications using mixture of PublicaMundi published services and their own implementation
- As for the Data Processing API itself, the Service Designer will also offer the capability to add and access any WPS Server transparently; in this way developers can integrate and use their own services from the designer.

4.1.14. Mapping API library

Definition

The Mapping API Library will allow developers to easily create and embed maps into custom web applications with just a few lines of code. The API will act as a wrapper around well-established mapping frameworks in order to abstract complexity and make mapping features accessible to developers with low or no expertise on spatial development.

Responsibilities

- Implicit or explicit mapping framework selection. Developers may either explicitly select the library of their choice or let the framework decide. The automatic library selection will be based on the features requested by the map configuration options and client capabilities.
- Dynamic loading of the required JavaScript libraries. Developers will have to reference only the API files from their application.
- Creating a map should be as simple as importing the required JavaScript files in a page and executing a single creation command with a configuration object.

- The internal map object, declared by the underlying mapping framework, will be available to the developer, hence allowing for fine-grained map configuration.
- Support multiple types of data sources/stores.

Interfaces

- JavaScript API for composing maps

Relationships to externals and other components

- Any mapping framework used by CKAN (see §4.1.1)
- WMS, WFS, WCS services (see §4.1.15) offered by PublicaMundi infrastructure

Strategic design decisions and their rationales

The objectives of the Mapping API Library are twofold: to expose a simple, yet rich interface for creating maps and to be independent from the underlying mapping framework. Hence, the API will declare a common interface and offer a separate implementation for each of the existing mapping frameworks used by CKAN. Thus, CKAN, as well as 3rd party applications, become less dependent on the underlying map rendering framework.

The most common scenario will involve searching the CKAN catalogue for required datasets and exporting the appropriate parameters as a configuration file for embedding a map representation of the dataset. We are planning to make such configuration files accessible through unique URLs, which in turn will be used for embedding the map with just a few lines of code.

4.1.15. WMS, WFS, WCS Interfaces

Definition

Web services (WMS/WFS/WCS) will be published by an open source map server. The data dissemination through the View Services is based on the OGC's Web Map Service (WMS) standard, version 1.3.0. The WMS standard is a standardized method for exchanging geo-referenced images (maps) using HTTP. This standard is supported by web and desktop GIS applications (proprietary and open), offering increased interoperability in viewing geospatial data.

The Download Services are based on the Open Geospatial Consortium's (OGC) Web Feature Service (WFS) standard. The WFS standard provides the interface for sending requests for spatial data through the web. The difference with the View Services is that through the Download Services the

user gains access to "real" data and not to "images" of the data. The Download Services make it possible to download copies of spatial data sets, or parts of such sets, to be downloaded and, where practicable, accessed directly.

The Web Coverage Service (WCS) service provides access to coverage data in forms that are useful for client-side rendering, as input into scientific models, and for other clients. The WCS may be compared to the OGC Web Feature Service (WFS) and the Web Map Service [WMS]. As WMS and WFS service instances, a WCS allows clients to choose portions of a server's information holdings based on spatial constraints and other query criteria. Unlike WMS, which portrays spatial data to return static maps (rendered as pictures by the server), WCS provides available data together with their detailed descriptions, it defines a rich syntax for requests against these data, and returns data with its original semantics (instead of pictures) which may be interpreted, extrapolated, etc., and not just portrayed.

Unlike WFS, which returns discrete geospatial features, WCS returns coverages representing space/time-varying phenomena that relate a spatiotemporal domain to a (possibly multidimensional) range of properties. As such, WCS focuses on coverages as a specialized class of features and, correspondingly, defines streamlined functionality.

Responsibilities

- The WMS service will be used to display geospatial data in image formats.
- The WFS service will be used to download geospatial data in vector formats.
- The WCS service will be used to provide access to coverage data in forms that are useful for client-side rendering or further processing

Relationships to externals and other components

- The WMS/WFS/WCS services will be used internally from the Map Viewer (see §4.1.12) and the CSW server (via proxy) (see §4.1.6).
- The WMS/WFS/WCS services will be used external from Browser and Desktop GIS (via proxy) (see §4.1.23).

Interfaces

- HTTP/OGC standards
- Configuration interface to content parameters (preferably RESTful) to change service contents and layer/coverages configuration dynamically
- Configuration interface to server-level processing capacity scaling

Strategic design decisions and their rationales

- PublicaMundi will integrate with several OGC compliant servers in order to have multiple back-ends as a production system

4.1.16. WPS Interface

Definition

The Web Processing Service (WPS) maintained by the Open Geospatial Consortium (OGC) provides client access across a network to pre-programmed calculations and/or computation models that operate on spatially referenced data. The calculation can be extremely simple or highly complex, with any number of data inputs and outputs. The inputs of services can be passed by value or by reference to an external dataset which can come as a result of a WMS, WFS, WCS (WCPS) or WPS (implying chaining capability) requests. Similarly, outputs can be requested to be returned in its original form or as a stored file on the server-side. The WPS standard offers requests options to ensure that a service will be executed as a background task, to have an up-to-date completeness status of an ongoing process, and then access the final result. This makes integration of real-time progress reporting possible, which is useful for time/resource consuming algorithms.

Responsibilities

- Add capability to execute simple to complex Raster and Vector processing queries on data coming from the Raster/Vector Database or externally as OWS interfaces (WMS, WFS, WCS/WCPS, WPS requests). Such requests can be used for selecting a subset of a dataset (WMS, WFS, WCS) and/or processing (for WCPS and WPS).
- Add capability to build new services by chaining available services
- Add capability to automatically publish OGC OWS Services (WMS,WFS and WCS) resulting of process execution for a future use
- Support for task management and access to up-to-date status of completeness for an ongoing service
- Give access to both the Projection and the Transformations libraries
- Give access to both the Raster and the Vector processors
- Provide transformation service on outputs of WCS/WCPS queries
- Provide projection services on outputs of WCS/WCPS queries

Relationships to externals and other components

- This module is tightly coupled with Projection/Transformation Libraries (see §4.1.18 and §4.1.19) and the Raster/Vector Processors (see §4.1.20 and §4.1.21).

Interfaces

- Provides WPS 1.0.0 [WPS] configuration interface to service implementation and configuration (preferably RESTful) to change service behavior and create new ones

Strategic design decisions and their rationales

- As a core component, WPS will be massively used even for non-geographical services such as interlinking.

4.1.17. WCPS Interface

Definition

The Web Coverage Processing Service is an extension to the WCS service, specifically designed to enhance coverage services with query language processing on (raster) coverage data. The Web Coverage Processing Service (WCPS) standard defines a language for filtering and processing of multi-dimensional raster coverages, such as sensor, simulation, image, and statistics data. This raster query language allows clients to obtain original coverage data, or derived information, in a platform-neutral manner over the Web.

Responsibilities

- Add capability to execute complex raster and image processing queries on top of raster data, stored as coverage, within the raster database
- Provides query language for multi-dimensional (e.g. volumetric) data
- Supports queries for time series analysis

Relationships to externals and other components

- This module is tightly coupled with the raster database (see §4.2.1.2)

Interfaces

- Provides WCPS 1.0 [WCPS] configuration interface to content parameters (preferably RESTful) to change service contents and layer/coverages configuration dynamically
- Configuration interface to server-level processing capacity scaling.

Strategic design decisions and their rationales

- Since the WCPS language is not tied to any particular transmission protocol, the query paradigm can be embedded into other services, such as WCS (*extension*) and WPS. These synergies will be leveraged in the project.
- Although the query language has a high expressive power, allowing complex computations to be specified, external image processing

algorithms provided by external libraries might be considered for inclusion, depending on progress of WCPS 2.0 standardization.

- When this component is deployed in a container server, scalability parameters configuration has to be done at container level

4.1.18. Projection library

Definition

The Projection Library is a WPS Service able to handle coordinates transformations. This projection library will take advantages of the PROJ.4 library and the numerous SRS it supports. Our work will result in an implementation of the INSPIRE WCTS's Transform operation.

Responsibilities

- Provides simple access to the PROJ.4 capabilities as Web Services: projecting from any supported Spatial Reference System to any other one

Relationships to externals and other components

- This service will be exposed as WPS, so it will be tightly coupled with the WPS Interface (see §4.1.16), the underlying transformation module, and the WMS Interface (optional automatic publication of the data as OWS resources).
- This module will use the Data Transformation Library (see §4.1.19) to give access to a different output data formats compared to the original ones)

Interfaces

- WCTS's [WCTS] Transform operation.

Strategic design decisions and their rationales

- Transformation library capabilities will be gathered in a simple set of services providing projection capabilities for WMS, WFS, WCS (WCPS) and WPS, so to support various kinds of raster and vector data as inputs and outputs.

4.1.19. Data transformation library

Definition

The Data Transformation Library is a set of WPS Services giving access to conversions from one supported raster or vector format to another.

Responsibilities

- Add capability to convert Raster and Vector data formats

Relationships to externals and other components

- The service implementation will be tightly coupled with the Projection library (see §4.1.18)

Interfaces

- WPS

Strategic design decisions and their rationales

- A well-known implementation will be used, embracing a wide variety of data formats

4.1.20. Raster Processors

Definition

Raster Processors are self-standing software components, libraries, scripts and algorithms that are linked with the WPS services in order to provide a platform for raster data processing engine within PublicaMundi.

Responsibilities

- Provide a platform for raster data processing engine including (*but not limited to*) the following algorithms: *image re-sampling, extraction of statistics (local and global), watershed modeling, hydrologic analysis, radiometric corrections, ortho-rectification, filtering and feature extraction*
- A bridge between WCPS and WPS is envisaged for the future (extending WCPS raster processing capabilities)

Relationships to externals and other components

- The service implementation will use the Projection and Transformation libraries (see §4.1.18 and §4.1.19).
- The service implementation will be tightly coupled with the GDAL library (see [GDAL]).

Interfaces

- WPS

Strategic design decisions and their rationales

- A collection of image processing algorithms are provided in the form of linked open source software (using the GDAL library).

4.1.21. Vector Processors

Definition

Vector Processors are self-standing software components, libraries, scripts and algorithms that are linked with the WPS services in order to provide a platform for vector data processing engine within PublicaMundi.

Responsibilities

- Provide a platform for vector data processing engine including (*but not limited to*) the following algorithms: *topology, cleaning, network analysis, geometry operations (buffer, segment, generalization),*

triangulation and point-to-polygon conversions, vector statistics, vector queries

Relationships to externals and other components

- The services implementation will use the Projection and Transformation libraries (see §4.1.18 and §4.1.19).

Interfaces

- WPS

Strategic design decisions and their rationales

- A collection of vector processing algorithms will be provided in the form of linked open source software.

4.1.22. Metadata Converter/Validator

Definition

This module will be implemented as a standard (PyPi-compatible) python module providing class representations of metadata specifications. The library will also provide specific methods that will execute validation checks to the metadata classes, in order to discover missing elements or fields that do not conform to the specification definition. This will be linked with the Metadata Editor module in order to provide real time validation results to the calling UI/API. The same library will implement metadata transformations using templates in a Pythonic way (e.g. by using Jinja2 templates) and/or through XSLT transformations. The transformed metadata will be available through the CKAN UI/API.

Responsibilities

- Validate against metadata specifications.
- Transform metadata records to all known metadata schemas, using ISO as a base standard.

Relationships to externals and other components

- The module is used by the Metadata Editor module (see §4.1.9) to provide validations, during the process of creating/updating datasets
- The module is expected to be invoked by several harvester implementations (see §4.1.26) while storing locally harvested metadata. The harvested records are transformed to the ISO standard.
- The CSW interface (see §4.1.6) uses metadata transformations to serve metadata records in different formats.

Interfaces

- Python API (as a standard Python module)

- HTTP interface through CSW GetRecords functionality.

Strategic design decisions and their rationales

- As already mentioned, ISO metadata core fields are going to be used as base for the converter/validator as well as for the database for reasons of completeness.
- Jinja2 templates can be used as they are also natively supported from CKAN 2.x (thus, not introducing a new dependency).
- Existing open source metadata libraries are going to be extended to support more metadata standards.

4.1.23. Desktop GIS

Definition

The software can be used as a geo-visualization tool supporting simultaneous access to multiple data sources. In addition to its interaction with the spatial database, it allows the use of the most widely used vector, raster, and attribute GIS data formats, as well as dissemination and exchange standards of spatial data and maps via web services.

It provides the ability to consume Geospatial Web Services based on WMS, WFS, WFS-T, WCS ISO/OGC standards. The software can allow hyperlinks on table (layer) level with objects such as websites, files (e.g. PDF, DOC or XLS), videos, images. Therefore, it will be able to directly associate geospatial data with information stored in files and websites, facilitating the search for relevant information during geospatial data management operations.

Responsibilities

- Provides catalogue query capabilities
- Displaying raster spatial data on a map
- Displaying vector spatial data on a map
- Provides navigation tools to the end user such as navigation, zooming, choose scale control etc.
- Provides the capability for rendering multiple layers on the map
- Provides an interface for controlling the display of the layers on the map such as reorder layers, show/hide capability of a layer, add/remove layer, controlling the transparency of a layer etc.
- Provides a tool for displaying the attributes (if available) of a spatial object displaying on the map.

- Provides an interface for chaining and executing WPS requests and rendering the results to the map.
- Provides measurement tools on the map
- Provides Features selection tools: by point, circle, rectangle, polyline, freehand and polygon; by location (spatial operators between layers); by attributes; clear selection.
- Identifying features options.
- Measuring distances and areas tools, with options for selecting measurement units and snapping tools.
- Editing data tools: editing geometry with graphic tools; editing attributes. The creation/editing of data require snapping tools and data entry options with point coordinates typing.
- Coordinate transformations between Spatial Reference Systems (re-project).
- Spatial analysis tools: buffer, intersection, clip, union, spatial difference, spatial join.
- Spatial queries capabilities; solve complex problems through a variety of tools: interactive distance measurement, select map features, select features by location or by attribute, clear selections, use identify features dialog box to access layer's attributes etc.
- Spatial queries capabilities; solve complex problems through a variety of tools: interactive distance measurement, select map features, select features by location or by attribute, clear selections, use identify features dialog box to access layer's attributes etc.
- Use of diverse mapping symbols, development of new, user-defined symbols, import symbols from symbol libraries, insert labels (labeling). Insert/store map legend.
- Spatial data generalization tools.
- Insert graph tools for attribute data display in the form of graphs (of type pie, bars etc.).
- Creating maps using predefined map templates (e.g. topographical, cadastral etc.)
- Map composition tools and options for printing maps (e.g. select map scale, page size, and export map to PDF). Options for adding, editing and displaying the map scale, margins, legend, images, and titles for the map composition.

Relationships to externals and other components

- Consume service from WMS/WFS/CSW CKAN's services (see §4.1.15)
- Executes WPS services and display the results to the software (see §4.1.16)

Interfaces

- Python interfaces through PyQt libraries

Strategic design decisions and their rationales

- User requirements have asked for Desktop clients to access/search the catalogue. This will be investigated as potential plugin for desktop client.

4.1.24. Caching Service

Definition

The caching service is an application service that will be coupled with the Proxy/Analytics application. A certain subset of the HTTP requests that is considered cacheable by the proxy front-end will be sent to the Caching Service, which in turn will store the (computed) response in a suitable manner (generally different per type of requested service). The service will be spatially enabled to be able to store common OGC web service requests, parameters and responses so that the workload will be reduced for the internal OGC interfaces. Also, a tile cache scheme will be implemented so that map tiles can be stored within the Caching Service, and provided back to the user using tile interfaces like TMS etc.

Responsibilities

- Provide cache on demand for supported web services back-ends
- Store Map Tiles for Tile Services but also tiled OGC services.

Relationships to externals and other components

- Coupled with Proxy/Analytics application (see 4.1.2).
- Providing cache to OGC Web Services (WMS, WFS, CSW) (see §4.1.15).

Interfaces

- None

Strategic design decisions and their rationales

- Tile caches speed up map applications and web services, so there will be support for them in PublicaMundi
- A Pythonic implementation of tile cache will be selected as more native to the PublicaMundi architecture.

4.1.25. OWS Clients

Definition

This component represents all the possible OGC OWS clients that can consume the PublicaMundi OGC Services. Those can be web applications, SDIs, Desktop GIS applications, mobile applications or even other OGC Services.

Responsibilities

- Not Applicable

Relationships to externals and other components

- All requests will go through the Proxy and Analytics application (see §4.1.2)
- All requests will run through the Caching Service (see §4.1.24) to determine if the requests are cached within the system.

Interfaces

- OGC OWS interfaces (WMS, WFS, WCS, CSW, WPS, WCPS etc.)

Strategic design decisions and their rationales

- The external OWS clients will be tunneled through the Proxy application in order to log the spatial content they access in order to provide system analytics.
- Specifically for heavy duty Web Services like WPS and WCPS, there will be a dedicated subset of our deployment to perform external processing, in order to prevent over-use of the system resources by third party applications.

4.1.26. External Harvesters

Definition

This component represents all external harvesters. These include external deployments of CKAN or external CSW servers or other harvesters. Those harvesters will be able to harvest metadata from PublicaMundi system through the CKAN API or the CSW interface.

Responsibilities

- Not Applicable

Relationships to externals and other components

- Linked to CKAN API as client (see §4.1.1)
- Linked to CSW interface as client (see §4.1.6)

Interfaces

- OGC CSW
- CKAN API

Strategic design decisions and their rationales

- Similar to the OWS clients, the external harvesters will use the Proxy application to have access to the catalog.
- Harvesting clients will have a performance penalty due to the size of data they request from the system, so that the system performance will not be affected for the rest of the external users.

4.2 DATA REPOSITORIES

This subsection documents the major data repository components of the system.

4.2.1. Databases

This subsection documents the major database components of the system.

4.2.1.1. Relational Database

Definition

This is an SQL-compatible database that stands as the primary RDBMS of the system. The database system runs as a network service, supports concurrency control and must fulfill the ACID set of requirements.

Responsibilities

- Host data model for CKAN application
- Host structured tabular data derived (i.e. analyzed and converted) from user uploads
- Support a replication/failover deployment scheme

Relationships to externals and other components

- CKAN core application and extensions are based on this component.

Interfaces:

- None

Strategic design decisions and their rationales

- None

4.2.1.2. Raster Database

Definition

This is an array database engine building on top of an RDBMS, a domain agnostic array engine allows handling raster data of any size and also allows configurable tiling of the storage backend for tailored performance tuning on expected retrieval workload.

Responsibilities

- store and query multi-dimensional data arrays (good data model for raster data)
- storage of raster data in an efficient and scalable manner
- configurable tiling scheme for storage layout (tailored performance enhancement)

Relationships to externals and other components

- Uses a relational DBMS for storing arrays (tiles)
- Is driven by the raster storer to ingest raster data coming from CKAN uploads
- Is accessed by W*S Web service interfaces to deliver requested datasets
- Is used by WCPS to efficiently execute SQL-like operators on arrays

Interfaces

- Provides raster query language (rasql)

Strategic design decisions and their rationales

- Raster data tends naturally to be modeled into arrays: usually bi-dimensional ones for mapping applications but further dimensions can come in handy for more complex scenarios. An array database is thus a perfect match for storage of raster data.
- This component will store all raster datasets uploaded by users or imported by system administrators

4.2.1.3. Vector Database

Definition

This is a spatial database that stores the user's data through CKAN interface to a vector spatial format.

Responsibilities

- stores vector spatial data
- perform spatial queries using SQL spatial syntax

Relationships to externals and other components

- adds support for geographic objects allowing location queries to be run in SQL
- it stores the vector data that will be used by the WFS service
- is driven by the vector storer that ingest vector data coming from CKAN vector uploads

Interfaces

- SQL interface
- Python interface

Strategic design decisions and their rationales:

- None

4.2.1.4. Metadata Database

Definition

The metadata database is the back-end for the metadata storage in PublicaMundi. Following the implemented relational scheme in CKAN core, the metadata database will be implemented on top of the same RDBMS as a first iteration. The core CKAN schema will be extended with the core ISO elements and queries needed to deploy the CSW interface. At the same time the database will implement a mechanism for translations and versioning of the metadata. On later stages of the project, a document-based database (NoSQL) solution will be investigated as a scalable back-end to the metadata store. This will provide the capacity to store millions of metadata documents without the hardware limitations imposed by SQL solutions and indexes.

Responsibilities

- Store the full XML documents of the original metadata.
- Store the original XML documents of all harvested metadata records.
- Store the spatial extent of the metadata records for spatial queries
- Store the core ISO queryables to perform fast searches on top of a CSW interface to the metadata records
- Implement Full Text Search mechanism
- Provide schema for translation of metadata and resources
- Store URIs for all metadata records (native or harvested)

Relationships to externals and other components

- Used by the CSW interface and harvester
- Used by Metadata Editor, Validator and Transformation modules to insert, delete, update metadata records.

Strategic design decisions and their rationales

- The basic implementation will extend the current CKAN database schema to include the core ISO elements. This is to support CSW integration to the system.
- Future implementation will focus on scalable NoSQL solution (document-based)

4.2.2. Files

This subsection documents the major file components of the system.

4.2.2.1. File storage

Definition

The file system controls how data is stored and retrieved from permanent storage.

Responsibilities

- Provide a typical file system permanent storage
- Provide Network File Storage (NFS) for certain areas and use cases (e.g. Tile Cache)
- (optional) Provide a BLOB storage layer on top of

Relationships to externals and other components

- Every PublicaMundi component will use some kind of File Storage

Strategic design decisions and their rationales:

- None

4.2.2.2. Metadata Files

Definition

These are XML files that need to be stored in their original form in the file system for historical purposes. The files are also going to be stored within the metadata database to be usable from the PublicaMundi system and available to the end users through the UI and the several APIs.

Responsibilities

- File storage of metadata records
- Backup of the system metadata records

Relationships to externals and other components

- Linked with Metadata Database, since those files are the raw data needed to populate the database.

Strategic design decisions and their rationales

- The metadata files will be stored in their original form, using the original language that were produced and published initially. Separate versions of the files will also be stored within the metadata database.

5 TOOL STACK

This section documents the tools and applications which will be used and extended in PublicaMundi, in order to fulfill the system architecture and design constraints. An overview of these tools is provided in Figure 69.

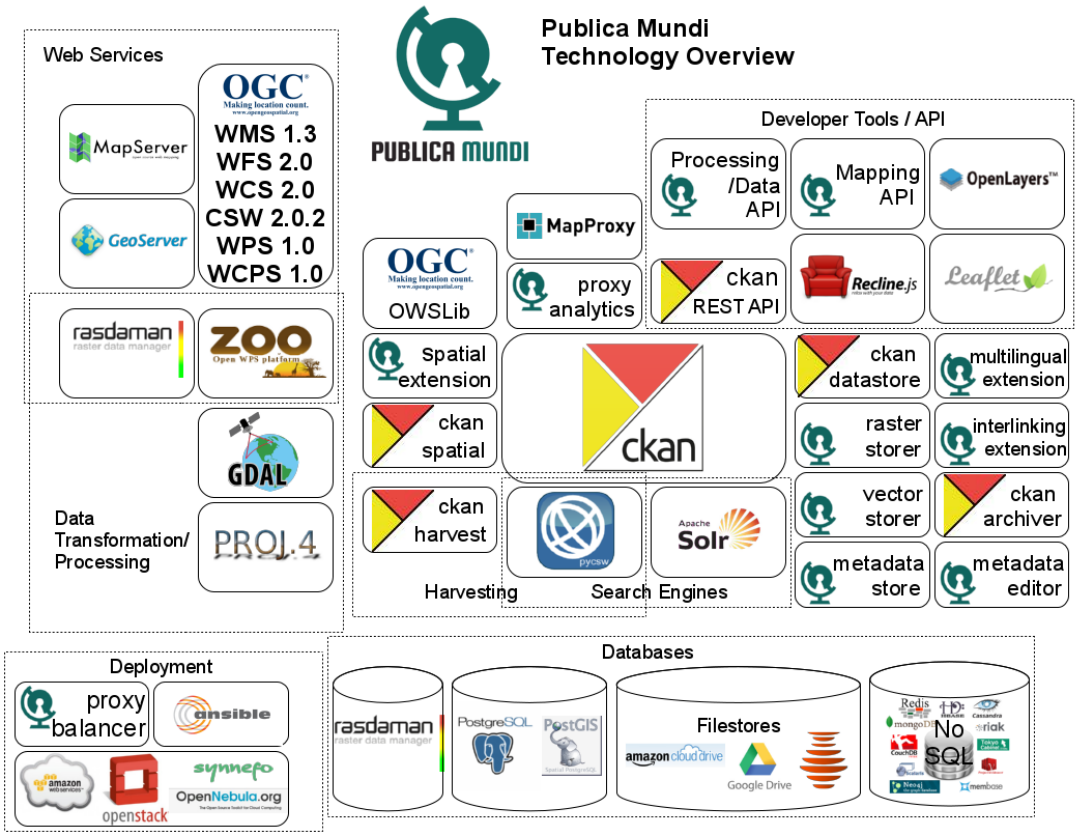


Figure 69: Technology Overview

5.1 CKAN

CKAN [CKAN] is a fully-featured open-source data management solution providing an efficient way to make open data publishable, discoverable and presentable. It deals with datasets i.e. with collections of resources along with their relevant metadata. CKAN aims at data publishers and is used by national and local governments, research institutions, and other organizations worldwide. It provides tools to publish, share, search, use and transform data. Based on CKAN's advanced search capabilities (facets, full-text, expressive query language) users are able to locate and browse the resources they need, download them or preview them using visualization tools as maps, graphs and tables.

CKAN is built with Python on the server-side and JavaScript on the client-side. It is essentially a WSGI application based on the Pylons web framework and using SQLAlchemy as an ORM layer. Its database engine is PostgreSQL and its search engine is powered by a SOLR backend. It has a modular architecture that allows extensions to be developed and provide additional features (or override existing ones) such as harvesting, data processing and analytics. CKAN defines a minimal metadata schema and allows (provides hooks) for schema extensions defined either globally or per-dataset. A key point of CKAN is that nearly every logical action defined on its entities is not only available in a human-friendly way (UI) but also in a machine-friendly way by means of an HTTP API. Therefore, a number of third-party applications can benefit from this fact as the major part of the catalogue's functionality becomes re-usable.

CKAN authorization model is organization-centric: an administrator can delegate access-control to an organization. Each organization is authoritative for its members and can grant fine-grained access (read, edit, administer). That way, loosely coupled (or even independent) organizations can co-exist and publish their data under the same catalogue system.

As a basic objective of a catalogue is to provide uniform access to a variety of resources (possibly from different origins), an important extension is the CKAN harvester [CKAN-HRV] which gathers and aggregates metadata from various sources providing centralized and uniform access to them.

5.2 ZOO PROJECT

ZOO-Project [ZOO] is an open source implementation of the Open Geospatial Consortium's Web Processing Service (WPS). Based on a server-side core component named ZOO-Kernel, it offers simple mechanisms to create, manage and chain processes in various programming languages. More precisely, the ZOO-Kernel supports 8 different programming languages, namely, Fortran, C/C++, Python, JAVA, PHP, Perl, Ruby and JavaScript. This is a key feature of the ZOO-Project which enables the ZOO-Services developers to reuse existing code with only minor modifications.

The ZOO-Project provides the ZOO-API which is a JavaScript API that offers part of the functionality of OpenLayers [OL] and Proj4js [PROJ], as well as, internal C functions which can be called within the JavaScript environment. The latter feature allows developers to call WPS services implemented in any programming language. Moreover, ZOO-API supports calling remote WPS servers, thus enhancing the chaining capacity intrinsic to WPS specification by adding logic provided by the JavaScript programming language in the process chain.

In the ZOO-Project implementation, there is a distinct separation between the metadata of a Service and its implementation. For each Service there is one metadata file (ZOO Configuration file, ZCFG file) and the service implementation which differs in nature depending on the programming language used (shared library for C/C++ languages, Python module for Python, Java Classes and so on). Nevertheless, the signature of each service implementation is the same for almost (with some limitation due to specific JavaScript limitation in memory gesture) each language: it takes three parameters (main environment variables, inputs and outputs structures) and returns an integer value (which can take the value SERVICE_SUCCEEDED or SERVICE_FAILED depending on the success or failure of the running process). For the three parameters, ZOO-Kernel uses its own internal data structures which are translated to the corresponding data structures for the targeted language and filled with data passed as parameters in the request prior to dynamically loading and executing a function. Hence, all the work which has to be accomplished specifically by WPS, such as downloading a FeatureCollection resulting from a complex WFS query, won't have to be taken into account by the service developer, who has direct access to the internal data structures specific to the language used for both accessing and storing resulting values. The developer's work will be limited to store resulting values of each process into a data structure pre-filled by the ZOO-Kernel, making any potential post-treatment, such as storing the result on the server-side, transparent.

From version 1.3.0, ZOO-Kernel is able to publish process execution results as OGC Web Services by using the MapServer [MSRV], provided that the results are spatially enabled and accessible by the GDAL/OGR [GDAL] library. Publishing is enabled through configuration and requires no modifications to the service implementation.

5.3 RASDAMAN

Rasdaman ("raster data manager") [RASD] is a scalable, multi-dimensional array database engine and analytics server. The PublicaMundi project makes use of the rasdaman community open-source project (www.rasdaman.org). Rasdaman is a domain-neutral Array Database System: it extends standard relational database systems with the ability to store and retrieve multi-dimensional raster data (modeled as arrays) of unlimited size through an SQL-style query language. Array intensive services can be set up using rasdaman, such as those found in data centers offering remote sensing, sensor series, image, simulation, and statistics data. Domains taking advantage of an Array Database can be Earth, Space, and

Life science. The query language offered over arrays enables flexibility, speed, and scalability. Rasdaman community is specifically designed to be smoothly paired with PostgreSQL.

Further, on top of the Array Database System (rasdaman components forming the core array server), a geospatially enabled layer is provided that adds spatiotemporal semantics over array data. Within the geo layer fall the petascope components and the related geo-import tools (currently known as rasgeo and wms-import). Geo-enablement of rasdaman provides access to geo raster standards-based services, including OGC WCS and WCPS, the OGC raster query language, along with WCS-T, and WPS. For several of these, rasdaman aims to be reference implementation A GDAL [GDAL] rasdaman driver is available, and likewise a MapServer [MSRV] integration. A key advantage of rasdaman is its internal management of data based on n-D tiling which allows data to be stored in the most convenient way, depending on the desired performance profile along properties such as fast retrieval along some dimensions and fast data ingestion rates.

5.4 OPENLAYERS

OpenLayers [OL] is a pure JavaScript web mapping library for displaying map data in most modern web browsers, with no server-side dependencies, using the latest features from HTML5 and CSS3. OpenLayers implements a JavaScript API for building rich, highly interactive web-based geographic applications by enabling users to put dynamic maps in web pages, display map tiles and markers and interact with the generated maps. It provides support from several data formats such as GML, KML, GeoJSON and WKT and sources such as WMS and WFS.

5.5 LEAFLET

Leaflet [LEAF] is a lightweight JavaScript library for interactive maps. Leaflet is designed with simplicity, performance and usability in mind and has no external dependencies. It works efficiently across all major desktop and mobile platforms out of the box, taking advantage of HTML5 and CSS3 on modern browsers while still being accessible on older ones. It can be extended with numerous plugins and an easy to use API.

5.6 PYCSW

pycsw (http://pycsw.org) [PCSW] is an OGC CSW server implementation written in Python. Since 2013 it is certified OGC Compliant and OGC Reference Implementation.

pycsw implements clause 10 (HTTP protocol binding (Catalogue Services for the Web, CSW)) of the OpenGIS Catalogue Service Implementation Specification, version 2.0.2. pycsw allows for metadata publishing either from its built-in data model, or through configuration to bind to an existing metadata mode. pycsw is Open Source, released under an MIT license, and runs on all major platforms.

5.7 OWSLIB

OWSLib [OWSL] is a Python package for client programming with Open Geospatial Consortium (OGC) web service interface standards and their related content models. It offers a wrapper API that greatly simplifies development as it renders most of the implementation details related to formatting requests and parsing responses transparent to the developer.

5.8 MAPSERVER

MapServer [MSRV] is a popular Open Source project whose purpose is to display dynamic spatial maps over the Internet. Some of its major features include:

- Support for display and querying of hundreds of raster, vector, and database formats
- Ability to run on various operating systems (Windows, Linux, Mac OS X, etc.)
- Support for popular scripting languages and development environments (PHP, Python, Perl, Ruby, Java, .NET)
- On-the-fly projections
- High quality rendering
- Fully customizable application output
- Many ready-to-use Open Source application environments

5.9 GEOSERVER

GeoServer [GSRV] is considered as one of the most advanced map-servers, while constitutes the reference implementation for the WFS standard. GeoServer's main features include:

- Support of OGC standards WMS, WFS, WFS-T, WCS,
- Data publishing from the most popular (commercial and FOSS) RDBMSs, namely like MS SQL Server, PostgreSQL/PostGIS, MySQL, Oracle Spatial, ESRI ArcSDE, as well as file-based geospatial data formats (e.g. ESRI shapefile, Vector Product Format, MapInfo MIF/MID, TIFF, GeoTIFF, BigTIFF, GTOPO30, ECW, MrSID and JPEG2000).
- Map publishing is supported in JPEG and PNG formats, with SLD support integrated.

- Support of coordinate reference systems specified in the EPSG database
- Dynamic CRS transformations
- Support of data export in various formats

5.10 MAPPROXY

MapProxy [MPRX] is an open source proxy for geospatial data. It caches, accelerates and transforms data from existing map services and serves any desktop or web GIS client. MapProxy is a tile server (WMS-C, TMS, WMTS, KML, SuperOverlays). It reads data from:

- WMS sources (1.0.0–1.3.0)
- TMS/WMTS sources
- MapServer and Mapnik configurations
- Any Tile Cache, Google Maps or Bing compatible source

5.11 SOLR

Solr [SOLR] is the popular, blazing fast open source enterprise search platform from the Apache Lucene project. Its major features include powerful full-text search, hit highlighting, faceted search, near real-time indexing, dynamic clustering, database integration, rich document (e.g., Word, PDF) handling, and geospatial search. Solr is highly reliable, scalable and fault tolerant, providing distributed indexing, replication and load-balanced querying, automated failover and recovery, centralized configuration and more. Solr powers the search and navigation features of many of the world's largest internet sites.

Solr is written in Java and runs as a standalone full-text search server within a servlet container such as Jetty. Solr uses the Lucene Java search library at its core for full-text indexing and search, and has REST-like HTTP/XML and JSON APIs that make it easy to use from virtually any programming language. Solr's powerful external configuration allows it to be tailored to almost any type of application without Java coding, and it has extensive plugin architecture when more advanced customization is required.

Solr is a standalone enterprise search server with a REST-like API. You put documents in it (called "indexing") via XML, JSON, CSV or binary over HTTP. You query it via HTTP GET and receive XML, JSON, CSV or binary results.

Advanced Full-Text Search Capabilities

- Optimized for High Volume Web Traffic

- Standards Based Open Interfaces - XML, JSON and HTTP
- Comprehensive HTML Administration Interfaces
- Server statistics exposed over JMX for monitoring
- Linearly scalable, auto index replication, auto failover and recovery
- Near Real-time indexing
- Flexible and Adaptable with XML configuration
- Extensible Plugin Architecture

5.12 GDAL

Geospatial Data Abstraction Library [GDAL] is an Open Source library supporting reading and writing access of a wide variety of raster geospatial data formats. Its internal OGR Simple Feature Library provides similar capabilities for vector data formats. This library which is available in a wide range of programming languages presents a single abstract data model to the calling application for all the supported formats. It is an underlying library for numerous other components involved in the PublicaMundi project, such as MapServer, GeoServer, PostGIS Raster.

Both GDAL and OGR use the concept of Drivers which gives access to read and sometimes write operations to a specific format. This modularity implies that new or existing formats can be easily added by implementing a new driver, thus allowing to transform and convert from one format to another. This library comes with numerous command line tools which will be re-used as WPS Web Services in the PublicaMundi project for Projection / Data Transformation Libraries and Vector / Raster Processors.

5.13 PROJ4

PROJ.4 [PROJ] is a widespread library for converting geospatial data between spatial references systems. Originally written by Gerald Evenden then by the USGS, it is now part of the MetaCRS project leaded by OSGeo. This library supports a wide variety of spatial references system and definitions of new ones.

PROJ.4 is a key requirement for numerous other components which will be used in the PublicaMundi project and will be the underlying library used for the creation of the Projection Library.

5.14 POSTGRESQL

PostgreSQL [PSQL] is an object-relational database management system (ORDBMS) based on POSTGRES, developed at the University of California at

Berkeley Computer Science Department. POSTGRES pioneered many concepts that only became available in some commercial database systems much later.

5.15 POSTGIS

PostGIS [PGIS] is an extension to the PostgreSQL object-relational database system which allows GIS (Geographic Information Systems) objects to be stored in the database. PostGIS includes support for GiST-based R-Tree spatial indexes, and functions for analysis and processing of GIS objects.

The PostGIS 2+ series provides:

- Processing and analytic functions for both vector and raster data for splicing, dicing, morphing, reclassifying, and collecting/unioning with the power of SQL
- Raster map algebra for fine-grained raster processing
- Spatial re-projection SQL callable functions for both vector and raster data
- Support for importing/exporting ESRI shapefile vector data via both command line and GUI packaged tools and support for more formats via other 3rd-party Open Source tools
- Packaged command-line for importing raster data from many standard formats: GeoTIFF, NetCDF, PNG, JPG to name a few
- Rendering and importing vector data support functions for standard textual formats such as KML,GML, GeoJSON, GeoHash and WKT using SQL
- Rendering raster data in various standard formats GeoTIFF, PNG, JPG, NetCDF, to name a few using SQL
- Seamless raster/vector SQL callable functions for extrusion of pixel values by geometric region, running stats by region, clipping raster by a geometry, and vectorizing raster
- 3D object support, spatial index, and functions
- Network Topology support
- Packaged Tiger Loader/Geocoder/Reverse Geocoder utilizing US Census Tiger data

PostGIS follows the Open Geospatial Consortium's "Simple Features for SQL Specification" and has been certified as compliant with the "Types and Functions" profile. PostGIS is open source software, released under the GNU General Public License.

5.16 RECLINE.JS

Recline.js [RCL] is a simple, yet powerful, library for building data applications. The library consists of three discrete parts, namely, models, backends and views. Models are used for data manipulation and shaping e.g. handling data as datasets, records/rows fields etc. Backends act as a bridge between the models and the data sources. In general, there is a backend implementation for each supported data source. For instance, a backend may be used for parsing a CSV file and fetching data to the model. Finally, views are user interface components used for rendering data and supporting user interaction. Such controls include data grids, maps and graphs.

6 CONCLUSION

In this deliverable, we compiled the user requirements for PublicaMundi, taking into account surveys and interviews that took place in the first months of the project. In the following, the System Architecture of PublicaMundi was defined, elaborating on its various components, roles interdependencies, and characteristics. Further, we identified a list of open source software which will be extended, adapted, and integrated to implement and deploy PublicaMundi.

Our next steps include the actual development, testing, integration, and evaluation of PublicaMundi. This deliverable will be updated during the course of the project to reflect new development directions and physical deployment options.

7 REFERENCES

[CKAN] CKAN Open Data Catalogue. Available at http://ckan.org/

[CKAN-HRV] CKAN Harvest Extension. Available at https://github.com/ckan/ckanext-harvest

[CKAN-SP] CKAN Spatial Extension. Available at https://github.com/ckan/ckanext-spatial

[CSW] Catalogue Service for the Web 2.0.2 specification (OGC 07-006r1). Available at http://portal.opengeospatial.org/files/?artifact_id=20555

[GDAL] GDAL web site. Available at http://www.gdal.org/

[GSRV] GeoServer web site. Available at http://geoserver.org/

[INSP07] INSPIRE Directive 2007. Available at http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:32007L0002:EN:NOT

[LEAF] LeafletJS web site. Available at http://leafletjs.com/

[MPRX] MapProxy web site. Available at http://mapproxy.org/

[MSRV] MapServer web site. Available at http://mapserver.org/

[OGC] Open Geospatial Consortium web site. Available at http://www.opengeospatial.org/

[OL] OpenLayers web site. Available at http://openlayers.org/

[OWSL] OWSLib web site. Available at http://geopython.github.io/OWSLib/

[PCSW] pycsw web site. Available at http://pycsw.org/

[PGIS] PostGIS web site. Available at http://postgis.net/

[PGSQL] PostgreSQL web site. Available at http://www.postgresql.org/

[PROJ] Proj4 web site. Available at http://trac.osgeo.org/proj/

[RASD] Rasdaman web site. Available at http://www.rasdaman.org/

[RCL] ReclineJS web site. Available at http://okfnlabs.org/recline/

[SOLR] SOLR web site. Available at https://lucene.apache.org/solr/

[WCPS] Web Coverage Processing Service 1.0.0 specification (OGC 08-068r2). Available at http://portal.opengeospatial.org/files/?artifact_id=32319

[WCS] Web Coverage Service 2.0.1 specification (OGC 09-110r4). Available at https://portal.opengeospatial.org/files/?artifact_id=48428

[WCTS] Draft Technical Guidance for INSPIRE Coordinate Transformation Services. Available at http://inspire.jrc.ec.europa.eu/documents/Network_Services/INSPIRE_Draf

t_Technical_Guidance_Coordinate_Transformation_Services_%28version_2%201%29.pdf

[WFS] Web Feature Service 2.0.0 specification (ISO 19142). Available at http://portal.opengeospatial.org/files/?artifact_id=39967

[WMS] Web Map Service 1.3.0 specification (OGC 06-042). Available at http://portal.opengeospatial.org/files/?artifact_id=14416

[WPS] Web Processing Service 1.0.0 specification (OGC 05-007r7). Available at http://portal.opengeospatial.org/files/?artifact_id=24151

[ZOO] ZOO-Project web site. Available at http://www.zoo-project.org/

APPENDIX I: INTERVIEW DISCUSSION POINTS

Note to interviewers: This is only to serve as a rough scenario; if the discussion moves to other directions and potentially interesting insights, feel free to pursue them

- Introduction
    - The purpose of this interview is to collect your feedback and requirements regarding open geospatial data publishing
    - PublicaMundi is an FP7 research project which will deliver geospatially-aware open data catalogues and a number of services supporting the full lifecycle of open geospatial data. Your input will be used to influence our architecture, development directions, software components, and services
    - Your responses will be anonymous and confidential. We will be keeping written minutes of our discussion which will be shared within the Consortium, and analyzed to extract user requirements. The minutes will be discarded after this process (retention for at most 15days); your anonymity will be preserved in our deliverables and reports.
- First of all, can you tell us a bit about your role and experiences in open data publishing in general and open geospatial data publishing in particular?
- Can you shortly describe the process followed in order to publish an open geospatial dataset? If you do not publish geospatial data, why is that? Technical, knowhow, or other reasons?
    - Workflow (who does what? E.g. find the data, create metadata)
    - Format (typical original, ETL, cleansing, interlinking)
    - Publishing (in original format, other formats)
    - Data size (rough estimate of data/metadata published) and growth (e.g. after 5 years). Any scaling problems encountered?
- Data catalogue
    - What data catalogue are you using? Are you familiar with CKAN?
        - If yes, do you use its spatial extensions?
    - Catalogue service? (automatic discovery)
        - Is your catalogue harvested by another service?
    - URI patterns for data
        - Do you follow specific URIs? Are these documented?

- Geospatial data
    - Do you publish vector data? Or perhaps simple tubular data with coordinates?
    - Do you publish raster data?
    - Do the metadata/data follow standardized schemata?
- Search
    - How do users search for data?
    - Any hierarchies, groups or other organizations?
    - Do you believe it is easy for the users to find what they are looking for? Possible improvements?
- Download
    - What options do the users have for downloading metadata? E.g. original only? In different schemata?
    - Are metadata files stored or automatically generated?
    - What options do users have for downloading data? Original files only? Different formats/transformations?
- Raster data
    - Do you provide raster data? Any examples?
    - Is there a need for providing raster data?
- Visualization
    - Any visualization services? For what?
    - Are the visualizations one-off (i.e. based on a snapshot of the data and part of a third application) or live (i.e. they are based on the actual data)
- Web mapping
    - Do you offer web maps?
    - Are these based on Google Maps, Bing, OSM or do you have your own web mapping framework in place?
    - What is the process for creating providing maps? Is it automated? Manual? Are all geospatial data available in the maps?
    - Do you use OGC standards? In what way?
- Developers/APIs
    - Are your data used for third party applications and services? In what way?
    - Do you offer data APIs? Are these standardized?
    - Do you believe that having data APIs would be beneficial for open data uptake?
- Analysis services

- Are analysis services offered for your data? Any examples?
- Are they one-off applications or integrated with live data underneath?
- What software are you using to develop the analysis services?

- Analytics
  - How do you monitor the catalogue use?
  - Do you have access to analytics of how your data are used? E.g. search, download, maps, etc?

- Multilinguality
  - Is your catalogue available in other languages? Is there the need?
  - Are metadata available in other languages?
  - Are data available in other languages?

- Interlinking
  - Do your data follow common vocabularies (international or national)?
  - Are you aware of inconsistencies between different datasets (e.g. codes for municipalities, addresses)
  - Do you apply any processes/tools to improve their quality and/or interlink them?
  - How do you feel about interlinking with Semantic Web technologies?

- Would you like to keep you involved in PublicaMundi?
  - What do you think of the overall idea? Would it be useful?
  - Newsletter for our progress (max 1/month)
  - Would you like to participate in our planned community events? (hackathons)
  - Any requirements/services/ideas to implement are welcomed!
  - Would you be interested to deploy PublicaMundi's components in your organization?

- Closing remarks: Thank you for your participation!
  - Would like us to prepare a blog post, summarizing our discussion? It will be first sent to you for approval and editing.