

OPEN CITIES



D4.4.2 (Platform) Requirements and Tool Specs (Specifications) for Open Data

Project Acronym **Open Cities**

Grant Agreement number: **270896**

Project Title: **OPEN INNOVATION Mechanism in Smart Cities**

D4.4.2 (Platform) Requirements and Tool Specs (Specifications) for Open Data

Revision: v1.1

Authors:

Juan Soto (Fraunhofer)
Jens Klessmann (Fraunhofer)
Edzard Höfig (Fraunhofer)

Project co-funded by the European Commission within the ICT Policy Support Programme		
Dissemination Level		
P	Public	X
C	Confidential, only for members of the consortium and the Commission Services	

Revision History

Revision	Date	Author	Organisation	Description
1.0	1.6.11	Juan, Jens, Ed	Fraunhofer	Initial Version
1.1	10.6.11	Ed	Fraunhofer	Added Introduction and Conclusion

Statement of originality:

This deliverable contains original unpublished work except where clearly indicated otherwise. Acknowledgement of previously published material and of the work of others has been made through appropriate citation, quotation or both.

TABLE OF CONTENTS

1 INTRODUCTION 6

2 ANALYSIS OF DATA MANAGEMENT PROCESSES..... 6

2.1 GUIDING PRINCIPLES..... 6

 2.1.1 *Data Centric*..... 6

 2.1.2 *Trustworthiness and Security*..... 7

 2.1.3 *Service-Oriented Architecture*..... 7

2.2 DATA MANAGEMENT PROCESSES 8

 2.2.1 *Control*..... 8

 2.2.2 *Definition* 8

 2.2.3 *Acquisition* 8

 2.2.4 *Organization*..... 8

 2.2.5 *Provision* 8

 2.2.6 *Archival*..... 9

 2.2.7 *Maintenance*..... 9

 2.2.8 *Interpretation* 9

 2.2.9 *Analysis*..... 9

3 FUNCTIONAL REQUIREMENTS 10

3.1 USABILITY REQUIREMENTS 10

[UR01] LANGUAGE LOCALIZATION..... 10

[UR02] CONCEPTS, TERMINOLOGY, AND INFORMATION TECHNOLOGIES..... 10

[UR03] UI FOR CONTRIBUTORS TO SHARE DATA 10

[UR04] UI FOR CONTRIBUTORS TO SHARE METADATA 10

[UR05] UI FOR CONSUMERS TO ENABLE DATA DISCOVERY 10

[UR06] UI FOR CONSUMERS TO ENABLE DATA CONSUMPTION 10

[UR07] UI FOR CONSUMERS TO ENABLE DATA VISUALIZATION 10

[UR08] UI FOR CONSUMERS TO PROVIDE FEEDBACK AND RECOMMENDATIONS 11

[UR09] UI FOR CONSUMERS TO SHOWCASE APPLICATIONS..... 11

[UR10] UI FOR ADMINISTRATORS TO PERFORM PLATFORM ADMINISTRATION 11

[UR11] UI FOR ADMINISTRATORS TO HANDLE INPUT BY DATA CONSUMERS..... 11

[UR12] UI FOR AUTHORIZED PERSONNEL TO MANAGE DATA ASSETS..... 11

3.2 SECURITY REQUIREMENTS 11

[SR01] DATA INTEGRITY FOR CONTRIBUTORS 11

[SR02] DATA INTEGRITY FOR CONSUMERS 11

[SR03] LOGGING AND TRACING 11

[SR04] IDENTITY AND ACCESS MANAGEMENT..... 12

3.3 DATA TRANSFORMATION REQUIREMENTS..... 12

[TR01] DATA EXTRACTION 12

[TR02] DATA CLEANSING 12

[TR03] DATA FILTERING 12

[TR04] AGGREGATION OF DATA FORMATS..... 12

[TR05] CREATING LINKED DATA..... 12

[TR06] PUBLISHING LINKED DATA 12

3.4 DATA STORAGE REQUIREMENTS..... 12

[DS01] STORE DATA IN VARYING FORMATS 12

[DS02] STORE METADATA 12

[DS03] STORE FEEDBACK AND RECOMMENDATIONS 12

[DS04] STORE LIST OF AVAILABLE APPLICATIONS..... 13

3.5 DATA MODEL REQUIREMENTS 13

[MR01] METADATA CONTENTS 13

[MR02] THESAURI, DICTIONARIES, AND/OR ONTOLOGIES..... 13

3.6 SOFTWARE TOOL REQUIREMENTS 13

- [SW01] AVAILABLE FOR USE, FREE OF COST 13
- [SW02] SUFFICIENT DOCUMENTATION 13
- [SW03] STABLE SOFTWARE 13
- [SW04] SELECTION OF RELEVANT TOOLS 13
- 3.7 APPLICATION DEVELOPMENT REQUIREMENTS 13
- [IR01] SERVICE ACCESS POINTS FOR DATA CONSUMPTION 13
- [IR02] COMPOSE AN API 14
- [IR03] HOSTING APPLICATIONS OR A REGISTRY OF APPLICATIONS 14
- 3.8 MISCELLANEOUS (BACKEND) REQUIREMENTS 14
- [BE01] IDENTIFICATION OF DATA SOURCES 14
- [BE02] DATA NAVIGATION CAPABILITIES 14
- [BE03] DOWNLOADING (DATA CONSUMPTION) SUPPORT 14
- [BE04] RUN DATA ANALYSIS PROCESSES 14
- 4 NON-FUNCTIONAL REQUIREMENTS 14**
- 4.1 DOCUMENTATION REQUIREMENTS 14
- [DR01] PLATFORM USER MANUAL 14
- [DR02] PLATFORM ARCHITECTURAL DESIGN DOCUMENT 14
- [DR03] PLATFORM SOFTWARE DESIGN DOCUMENT 15
- [DR04] DOCUMENT SOURCE CODE 15
- [DR05] LISTING OF AVAILABLE APIS 15
- 4.2 LEGAL REQUIREMENTS 15
- [LR01] DISCLOSURE OF LICENSING TERMS 15
- 4.3 PRIVACY REQUIREMENTS 15
- [PR01] COMPLY WITH EU DATA PROTECTION DIRECTIVES 15
- [PR02] ENSURE/PRESERVE USER PRIVACY 15
- 4.4 (DATA) QUALITY REQUIREMENTS 15
- [QR01] CLEANLINESS 15
- [QR02] TIMELINESS 15
- [QR03] FEEDBACK MECHANISM 15
- 4.5 PERFORMANCE/EFFICIENCY REQUIREMENTS 16
- [ER01] AVAILABILITY 16
- [ER02] DEVICE-INDEPENDENT ACCESSIBILITY 16
- [ER03] FEEDBACK FROM STAKEHOLDERS 16
- [ER04] SCALABILITY 16
- 5 TOOL SPECIFICATIONS 17**
- 5.1 GOOGLE REFINE 17
- 5.2 IBM MANY EYES 18
- 5.3 TRIPLIFY 19
- 5.4 JAX-RS 20
- 5.5 JENA 20
- 5.6 SESAME 21
- 5.7 TRDF/TSPARQL 21
- 5.8 VIRTUOSO 22
- 5.9 CKAN 23
- 5.10 ONTOWIKI 24
- 5.11 APACHE TIKI 24
- 6 CONCLUSION 26**
- 7 BIBLIOGRAPHY 26**

Synopsis: This document contains a list of requirements for the *Open Cities* open data platform and contains a list of tool specifications. In *Section 1*, we present a general overview of typical data management processes. *Sections 3* and *4* contain a listing of functional and non-functional requirements for the open data platform, respectively. In *Section 5* we list the specifications for varying tools that we are currently considering for use in the open data platform.

1 INTRODUCTION

Within the Open Cities project we are designing and implementing a prototypical platform for the provisioning of Public Service Information (PSI). This technology will be used by novel applications, fostering innovation in this sector. It is planned to organise a number of competitions to trigger application developments activities in the user community and we intend to utilise the platform prototype as a basis for these competitions. Therefore, it is necessary that we agree upon, and document, the existing requirements on this system and this is the purpose of this document. These requirements have been compiled by questioning the relevant stakeholders for PSI data (mainly the cities contributing to the project) and we choose to do this by means of a questionnaire, as well as by discussion using email, phone and face-to-face meetings.

In addition to the identified requirements, we are also documenting a number of tools that have been surveyed in the course of the first half-year of the project. As it is not feasible to implement such a complex system as the Open Cities data platform from scratch, we are relying on the integration and facilitation of existing technologies and tools. The choice of tools has been motivated by our own experience with the necessary technology, but also through meetings and discussions together with the LOD2 project¹.

2 ANALYSIS OF DATA MANAGEMENT PROCESSES

2.1 GUIDING PRINCIPLES

We want to develop a platform for the management of information flows with respect to the trustworthiness of the content and the participating stakeholders. The goal is to achieve a new level of transparency for processes in the public sector. In this Section we will clarify the terminology and basic concepts that underlie such a platform.

2.1.1 Data Centric

The platform mainly deals with the management of data and (structural) metadata as well.

We distinguish between three classes: *raw data*, *data asset*, and *information set*. Figure 1 illustrates these classes, along with a number of processes, which will be described in Section 2.2. A data source is an entity that provides raw data to the Open Cities platform. The *raw data* is stored in a pre-defined structure within the platform, yielding a data asset, which is maintained by the platform. The *data asset* can be combined

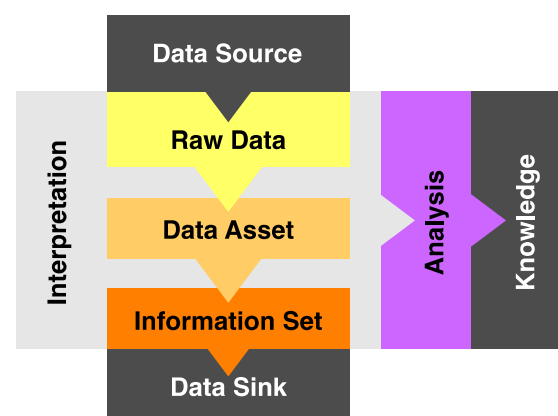


Figure 1) Data Artifacts

¹ <http://lod2.eu/Welcome.html>

with other assets in a meaningful way, forming an information set. The *information set* is provided to data sink entities using appropriate formats. Data sinks use the information over the natural course of their particular business processes; they might also contribute the information as data to the platform, again. During the overall data processing steps, there are a number of interpretation activities that introduce subjective input into the processes (e.g., deciding on the structuring of the data assets, the choice of linked data that yields the information sets, the assignment of semantics to the data assets). An additional analysis process uses the interpreted information sets to build up knowledge (e.g., to assess the trustworthiness of a data source). In the following text, we will align concepts and terminology to the ISO/IEC 11179 and DAMA DMBOK standards (DAMA International, 2009; ISO/IEC, 2003; ISO/IEC, 2005).

2.1.2 Trustworthiness and Security

Although trust management is considered a part of security management, there are notable differences when compared to conventional security approaches. Conventional security systems usually employ mechanisms that aim at preventing users from executing certain operations or accessing certain information. Trust-based mechanisms do not prohibit users from performing access or execution functions, but continuously assess the behavior of both parties, as well as interpretation processes, to decide on the trustworthiness of each. This information can then be used in conjunction with conventional security mechanisms (e.g., refined access controls via the use of policies or the use of encryption for message confidentiality) to achieve higher levels of system security.

Trust-based mechanisms are especially useful in open-world situations where the user group is too large or too diverse to be sufficiently controlled using a closed mechanism, or where such an approach would be considered to be intrusive (e.g., with respect to privacy -or- for reasons of convenience).

2.1.3 Service-Oriented Architecture

The Open Cities platform will likely follow a service-oriented architecture (SOA). Platform components may belong to separate organizations, may be globally distributed, may fail at any time, or may need to be replaced. To support this architectural style, the platform components should be loosely coupled. For example, this may require components to rely on messages in order to communicate (or exchange) information. We envision the platform will be based on web technologies and as such, services will be accessed using HTTP based mechanisms (such as RESTful interfaces, Web Service endpoints, or SOAP messaging). Another requirement that follows from postulating loose coupling, concerns the discovery of components: the system components need to find each other (e.g., by means of a lookup in a service registry or through discovery mechanisms built into the communication framework).

2.2 DATA MANAGEMENT PROCESSES

We identified nine processes employed for *data management* (DM), as depicted in Figure 2. The majority of these DM processes were directly taken from (DAMA International, 2009), but we also found that it necessary to augment these with additional ones.

2.2.1 Control

The *control process* ensures that the overall operation of the DM system is working correctly. It regulates the system behavior in accordance with the operational policies as set forth by a system operator and the responsible legal authorities.

2.2.2 Definition

The *definition process* encompasses the identification of data sources and definition of data types, structure and technologies used to store the data assets. The outputs of this process are clearly defined structures for data assets and the data sources used for acquiring the raw data.

2.2.3 Acquisition

During the *acquisition process*, raw data is entered into the structure created by the definition process, resulting in data assets. Based on type and source of the raw data this process uses conversion operations to transform the data from the raw format into a suitable storage format. The outputs of the process are data assets, stored in pre-defined data structures.

2.2.4 Organization

The *organization process* constructs information sets from the data assets by constructing meaningful links between individual data values or sets. The outputs of this process are information sets.

2.2.5 Provision

The *provision process* enables a distribution of the linked information created by the organization process. The output of this process is an information set, in a format suitable for consumption by clients of the platform. The output (i.e., the information) may also serve as input data to the acquisition process.

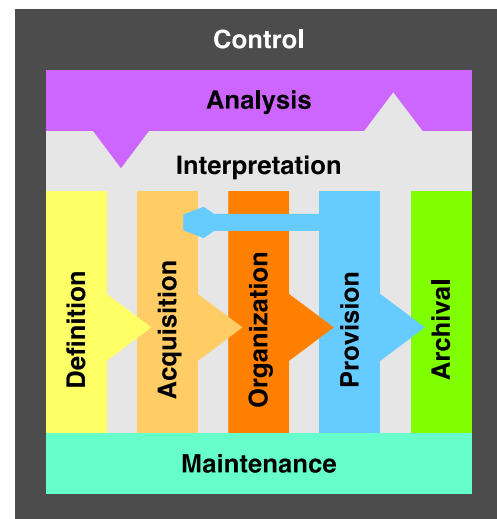


Figure 2) Data Management Processes

2.2.6 Archival

The *archival process* is used at the end of the lifetime of an information set to guarantee a required retention period, where the information is not provided through the platform anymore, but still available to a system operator. This process has no output.

2.2.7 Maintenance

The *maintenance process* encompasses all of the activities necessary to guarantee the proper operation of the platform. This includes activities like backup, restore and data validation.

2.2.8 Interpretation

The *interpretation process* is a vertical process that encompasses activities where subjective input can influence five other DM processes, namely, definition, acquisition, organization, provision, and archival. This does not only refer to the processes themselves, but also to the artifacts shared between processes. Typical interpretation activities are visualization, transformation, or computation using the prescribed artifacts. The output of the interpretation process can be used to assess the system's operation.

2.2.9 Analysis

The *analysis process* evaluates the operation of the system and assesses the interpretation of the stored data assets and information sets. The results of the analysis process can be used to influence the interpretation process and therefore, the platform's operation.

3 FUNCTIONAL REQUIREMENTS

It may be helpful to keep in mind the varying stakeholders when compiling a list of requirements for the creation of an open data platform. In particular, these stakeholders include: (a) *open data evangelists*, (b) *data stewards*², (c) *data providers*, (d) *platform architects/engineers*, (e) *data consumers*³, and (f) *data privacy officers*.

For further descriptions surrounding these stakeholders and others, reference Section D.

3.1 USABILITY REQUIREMENTS

[UR01] Language Localization

The platform's user interfaces (UIs) will be presented in English and in German. Support will be provided to translate the UIs to Catalan, Dutch, Finnish, French, and Castilian.

[Required]

[UR02] Concepts, Terminology, and Information Technologies

Commonly occurring *concepts*, *terminologies*, and *information technologies* that are widely used in the creation and use of an open data platform will be summarized in an accompanying document, in order to enable effective communication among all stakeholders. **[Required]**

[UR03] UI for Contributors to Share Data

Design and implement a user interface to enable contributors to perform management functions, such as: *upload*, *replace*, *delete*, *transform*, and *classify* their data within the (data) catalog. **[Required]**

[UR04] UI for Contributors to Share Metadata

Design and implement a user interface to enable contributors to *insert*, *edit*, or *delete* metadata associated with the data. This includes the assignment of categories, versions and source information. **[Required]**

[UR05] UI for Consumers to Enable Data Discovery

Design and implement a user interface to ease data discovery. The UI should provide facilities for *displaying* data (particularly, existing open government data), *browsing* data, *searching* by keyword, and *filtering* by data source, data class, tags, regional reference, topic and date of publication, among others. **[Required]**

[UR06] UI for Consumers to Enable Data Consumption

Design and implement a user interface to facilitate consuming (downloading) data, in a prescribed data format (e.g., .pdf, .xls, .txt, .xml, .docx, etc.). If the data is not available, the UI should indicate that the data is unavailable. **[Required]**

[UR07] UI for Consumers to Enable Data Visualization

Design and implement a user interface to enable data visualization analytics. **[Optional]**

² The *data steward* role refers to an individual or group that is tasked with maintaining data assets. Data stewards are also responsible for ensuring data quality consistency.

³ The *data consumer* refers to any individual, group, or mobile application developer interested in data access.

[UR08] UI for Consumers to Provide Feedback and Recommendations

Design and implement a user interface to provide user feedback concerning individual data sets, the platform, requests for new data sets, or recommend ideas for new applications. **[Required]**

[UR09] UI for Consumers to Showcase Applications

Design and implement a user interface to showcase applications that utilize from the open data platform. The UI should provide mechanisms for storing details about an application (e.g., URL, etc.). **[Required]**

[UR10] UI for Administrators to Perform Platform Administration

Design and implement a user interface to ease conducting platform administration (e.g., create user accounts, declare user roles, define user rights, or perform platform audits, among others). **[Required]**

[UR11] UI for Administrators to Handle Input by Data Consumers

Design and implement a user interface to allow an administrator to *edit* and *publish* descriptions of data applications and requests for new data sets as provided by data consumers. **[Required]**

[UR12] UI for Authorized Personnel to Manage Data Assets

Provide a user interface to enable authorized personnel to maintain their data assets. Representative maintenance capabilities include: (a) the purging of errors in data sets, (b) ensuring data consistency between replicated mirror sites, and (c) archiving, in order to support versioning and the retirement of data assets. **[Required]**

3.2 SECURITY REQUIREMENTS

[SR01] Data Integrity for Contributors

Provide a mechanism to enable data contributors to create *message digests* to enable data integrity. **[Required]**

[SR02] Data Integrity for Consumers

Provide data consumers with a mechanism (e.g., to *recompute a hash and compare it against the initial pre-computed., reference hash, using a standardized hash function, such as SHA-2*) to validate data integrity, in order to ensure that the data has not been tampered with and that it originated from a reliable source. **[Required]**

[SR03] Logging and Tracing

Introduce auditing capabilities to facilitate periodic analysis of platform activities in order to discourage foul-play. All data access operations must be logged for security and troubleshooting purposes. **[Required]**

[SR04] Identity and Access Management

The platform should provide a mechanism for the creation of user accounts and associated roles for access management, in order to exercise greater control regarding who can contribute data to the platform. **[Required]**

3.3 DATA TRANSFORMATION REQUIREMENTS

[TR01] Data Extraction

Provide mechanisms to extract (specific) data from data sets using open source tools. **[Required]**

[TR02] Data Cleansing

Provide mechanisms to cleanse data sets using open source tools. **[Optional]**

[TR03] Data Filtering

Provide mechanisms for filtering data according by keyword (e.g., publisher, date of publication, regional reference, topic, tags, or data source). **[Required]**

[TR04] Aggregation of Data Formats

Provide mechanisms for harmonizing data across varying digital formats to facilitate its utility. **[Required]**

[TR05] Creating Linked Data

Provide mechanisms for creating linked using open source tools. **[Required]**

[TR06] Publishing Linked Data

Provide mechanisms for publishing linked data using open source tools. **[Required]**

3.4 DATA STORAGE REQUIREMENTS

[DS01] Store Data in Varying Formats

Provide the ability to store data in varying formats, including the RDF (resource description framework) format. **[Required]**

[DS02] Store Metadata

Provide the ability to store (structural) metadata information for supported data sets. **[Required]**

[DS03] Store Feedback and Recommendations

Provide the ability to collect, store, and disseminate feedback and recommendations to appropriate stakeholders. **[Required]**

[DS04] Store List of Available Applications

Provide the ability to compile and store a list of all of the available applications that were developed. **[Required]**

3.5 DATA MODEL REQUIREMENTS

[MR01] Metadata Contents

Data assets should carry a standardized set of metadata that includes provenance, license, and categorization information. **[Required]**

[MR02] Thesauri, Dictionaries, and/or Ontologies

An analysis of appropriate thesauri, dictionaries or ontologies must be conducted. If needed, a new data model will be developed for use in the platform. **[Required]**

3.6 SOFTWARE TOOL REQUIREMENTS

[SW01] Available for Use, Free of Cost

Software tool recommendations must be available for use by public authorities and application developers for free. This includes either software available under an open source license or as freeware. **[Required]**

[SW02] Sufficient Documentation

Software tool recommendations must have sufficient documentation to enable third parties to easily use the tools. **[Required]**

[SW03] Stable Software

Software tool recommendation must be stable enough for prototyping. **[Required]**

[SW04] Selection of Relevant Tools

Software tool recommendations must be relevant and address the needs of platform stakeholders. For example, software tools supporting (meta)data input should be carefully chosen so as to meet the needs of data providers. Similarly, software tools should be selected such that to ensure the needs of data consumers are satisfied. **[Required]**

3.7 APPLICATION DEVELOPMENT REQUIREMENTS

[IR01] Service Access Points for Data Consumption

Application developers will consume their data via web services (e.g., REST, SOAP). Service endpoints should be self-descriptive and at the very least, it should be implemented using REST principles. A listing of representative API for consideration may be found at <http://www.deutschland-api.de>. **[Required]**

[IR02] Compose an API

Provide an API (e.g., <http://developer.yahoo.com/yql/>) that may be used by mobile application developers to create and send queries (e.g., expressed in SPARQLZ or tSPARQL) to retrieve the particular data of interest. **[Required]**

[IR03] Hosting Applications or a Registry of Applications

Provide a mechanism to host applications and/or maintain a registry of available applications. **[Required]**

3.8 MISCELLANEOUS (BACKEND) REQUIREMENTS

[BE01] Identification of Data Sources

Provide a mechanism to facilitate identifying data sources. **[Required]**

[BE02] Data Navigation Capabilities

Provide mechanisms to facilitate data navigation, such operations include *browsing*, *searching* (by keyword), and *filtering* (by data source, data class, tags, regional reference, topic and date of publication, among others). **[Required]**

[BE03] Downloading (Data Consumption) Support

Provide mechanisms to facilitate data consumption. For example, downloading identified data sets as in their *raw data* form or as *linked data*. **[Required]**

[BE04] Run Data Analysis Processes

The backend logic needs to be able to run custom analysis processes against the stored data assets. **[Required]**

4 NON-FUNCTIONAL REQUIREMENTS

4.1 DOCUMENTATION REQUIREMENTS

[DR01] Platform User Manual

Write a user manual that addresses the needs of *data contributors*, *data consumers*, and *platform administrators*. **[Required]**

[DR02] Platform Architectural Design Document

Write a platform architectural design document to provide details suitable for platform engineers, who aim to customize and/or further extend the capabilities of the platform. **[Required]**

[DR03] Platform Software Design Document

Write a platform software design document, in order to provide support for platform engineers, who aim to customize and/or further extend the capabilities of the platform. In particular, listing all software tools (e.g., name, version, and additional reference information) and their dependencies that make up the platform should be provided. **[Required]**

[DR04] Document Source Code

For any source code produced, proper documentation (according to an established standard) should be included to ease maintainability of the code. **[Required]**

[DR05] Listing of Available APIs

Provide a list of available APIs (e.g., <http://www.programmableweb.com/>). **[Required]**

4.2 LEGAL REQUIREMENTS

[LR01] Disclosure of Licensing Terms

All data sets, should, preferably must, include associated licenses that clearly specify the licensing terms that govern the usage of the data, by third parties. **[Required]**

4.3 PRIVACY REQUIREMENTS

[PR01] Comply with EU Data Protection Directives

The platform must meet EU data protection directives. **[Required]**

[PR02] Ensure/Preserve User Privacy

The platform must preserve user privacy. **[Required]**

4.4 (DATA) QUALITY REQUIREMENTS

[QR01] Cleanliness

The data should (ideally, must) be clean in order for it to be usable and reliable. **[Required]**

[QR02] Timeliness

The data should be timely for it to be suitable in critical situations. Historical data, would naturally be excluded from this requirement. **[Required]**

[QR03] Feedback Mechanism

The platform should provide a feedback function to facilitate the collection of user recommendations for improvements. **[Required]**

4.5 PERFORMANCE/EFFICIENCY REQUIREMENTS

[ER01] Availability

Incorporate measures to track the availability of data sources. **[Optional]**

[ER02] Device-Independent Accessibility

Incorporate measures to enable device independence. **[Optional]**

[ER03] Feedback from Stakeholders

Provide a mechanism to collect recommendations on how to improve the platform from all stakeholders (e.g., a repository of ideas and applications that rely on government data). **[Required]**

[ER04] Scalability

The platform must be federated across several administrative boundaries and across several data repositories. **[Optional]**

5 TOOL SPECIFICATIONS

To date, we have reviewed numerous tools that we deem are worth considering for use in the Open Data platform. These tools include:

- Google Refine
- IBM Many Eyes
- Triplify
- JAX_RS
- Jena
- Sesame
- tRDF / tSPARQL
- Virtuoso
- CKAN
- OntoWiki
- Apache Tika

In the following sections, we will summarize the major points for each of the aforementioned tools.

5.1 GOOGLE REFINE

Creator: David Huynh, Stefano Mazzocchi and contributors

License: New BSD License <<http://www.opensource.org/licenses/bsd-license.php>>

Architecture layer: It can be employed in the City Data Cloud Layer 2 “data storage and description” to prepare particularly tabular spreadsheet data and to discover & remove potential inconsistencies. It can change and clean up in bulk large chunk of data matching some particular criteria. Furthermore it allows transformation of data sets from one format into another and to extend them with new data from external web services or other databases.

Functionalities: Google Refine has a local web interface, is available for all platforms (Windows, Mac, Linux) and integrates filtering and faceted browsing mechanisms. Data is kept local and can be imported from CSV (Character Separated Values), Excel (.xls, .xlsx), XML, RDF as XML, JSON and Google Spreadsheets. Data can be exported to CSV, HTML table, Excel and customized formats via templating function and as whole project file. It comes with an extended history functionality which allows reuse of working steps through history im- and export (JSON).

Purpose: Google Refine is a power tool for working with messy data, cleaning it up, transforming it from one format into another, extending it with web services, and linking it to databases like Freebase.

Maturity: Google acquired Metaweb and relaunched their tool Freebase Gridwork as Google Refine. It is currently in version v.2.0 and seems to be stable. It is a high activity project on Google code Project Hosting and future versions are in development. It has been used by ProPublica and data.gov.uk.

References: The project is hosted at <http://code.google.com/p/google-refine/>. Good links of usage examples can be found in the official announcement of the Google blog at <http://google-opensource.blogspot.com/2010/11/announcing-google-refine-20-power-tool.html> such as the use at data.gov.uk <http://www.jenitennison.com/blog/node/145>.

Comments: It seems to be a real powerful tool for data cleansing, format transformation and extension through external RESTful web services such as a geocoding service. It is extensible and there are by now two extensions available – one of them is a RDF extension by DERI: <http://lab.linkeddata.deri.ie/2010/grefine-rdf-extension/>.

5.2 IBM MANY EYES

Creator: Visual Communication Lab, part of Collaborative User Experience group, at IBM Research

License: IBM Research Labs Services Use Agreement (the “Agreement”) <http://www-958.ibm.com/software/data/cognos/manyeyes/research_agreement.html>


Architecture Layer: It can be employed in the City Data Cloud Layer 5 “services and applications” as a platform service offered to visualize tabular spreadsheet data (tab delimited) provided by the platform. However data has to be uploaded to the Many Eyes server to get the possibility to visualize it afterwards.

Functionalities: IBM Many Eyes is offered for registered users through the browser requiring Adobe Flash Player and Java plug-in to see its visualizations of uploaded data sets.

Procedure: (1) Register an account, (2) Copy/paste tab delimited data like spreadsheet data and upload it, (3) Create visualizations on it. Different visualization types can be accessed through PNG image or interactively embedded through a JavaScript via the <script> element. Users can browse people's data sets, their attached visualizations and other users' comments. They can visualize data sets of others and being notified of changes by watchlisting other data sets or visualizations.

Purpose: IBM Many Eyes is a site which allows the community to upload data, visualize it, and also talk about it.

Maturity: It can be used to quickly visualize some tab delimited data. However data sets cannot be modified after upload – one has to delete and re-upload the data in order to get it changed with the implication that all built visualizations and comments will be also deleted. The project seems to be not that active.

References: The site can be found at <http://www-958.ibm.com/software/data/cognos/manyeyes/>. An interesting German tutorial for displaying government data with the visualization Country Map  Germany Map showing the federal states of Germany <http://www.datenjournalist.de/tutorial-interaktive-kartenvisualisierung-mit-many-eyes-am-beispiel-rechter-straftaten/>.

Comments: Data will be stored on the IBM server publicly available and always connected to community functionality like other people and their comments on the Many Eyes site. Some of the project's community concepts seem to be interesting to think about.

5.3 TRIPLIFY

Creator: Research Group Agile Knowledge Engineering and Semantic Web (AKSW) at the University of Leipzig

License: LGPL <<http://www.gnu.org/licenses/lgpl.html>>

Architecture Layer: It can be employed in the City Data Cloud Layer 2 'data storage and description' for relational stored data in web applications which can be selectively published as RDF, JSON or Linked Data via customized SQL queries.

Functionalities: The Triplify PHP scripts – deployed in Triplify folder in root folder of the web app – generate database views by performing some customized SQL queries against the web app's relational database. It needs to have access to the relational database on chosen data which has to be 'semantified' via RDFS vocabulary terms such as defined in foaf (Friend Of A Friend), SIOC (Semantically-Interlinked Online Communities Project), self-defined ones or others. In this way it reveals the semantic structures of the relational database by making its content available as RDF, JSON or Linked Data (needs URL rewriting).

Purpose: Exploiting structured relational representations behind web applications to create a critical mass of semantic representations on the Web. Make web applications usable for mashups.

Maturity: It's currently only implemented in PHP. First Version published in 02-2008 and last version update (v0.8) was in 03-2010. It's declared to be beta grade software. The customization of the SQL queries has to be done by hand by editing the Triplify php configuration file, but there are also some community-contributed Triplify configurations for common web applications like Drupal, WordPress, Joomla!, phpBB.

References: Useful information about Triplify and how it is used can be found at <http://triplify.org/About> and <http://triplify.org/Documentation>. Code is hosted at <http://sourceforge.net/projects/triplify/> and one can jump directly into Triplify's php config file at <http://triplify.svn.sourceforge.net/viewvc/triplify/triplify-php/config.dist.php?view=markup>.

Comments: D2RQ seems to be a similar tool. A comparison as stated by the makers of

Triplify: „D2RQ has basically the same aim as Triplify, although it is more difficult to deploy. It includes its own mapping language to map DB content to ontologies, whereas Triplify just uses SQL. In contrast to Triplify, D2RQ also contains SPARQL endpoint functionality.“. However Triplify is not that easy to configure, too. Often one needs to know the semantic vocabulary terms and / or SQL. If there isn't a community-contributed config file one needs to know the vocabulary terms for the manual mapping of SQL data fields to their corresponding semantic vocabulary property terms out of vocabularies such as foaf, SIOC or dc and a basic understanding of SQL. However if a config file is provided for the given web application like WordPress 2.7, there will remain the question whether this file will be running also with the actually used version of WordPress like currently 3.1.2 and whether the exposed data fulfills the desired degree of publicness / privacy.

5.4 JAX-RS

URL: <http://jsr311.java.net/>

License: CDDL license

Purpose: JAX-RS is a standard (JSR 311) for binding Java classes to REST interfaces. Programmers use Java annotations to declare a mapping between a method and a web request (e.g. a POST request carrying JSON data).

Installation: There are competing implementations for JAX-RS. We found that it is pre-installed in version 1.1 with the JBoss 6 environment.

Integration: As JAX-RS is a standard, it is men to server integration.

Extensibility: Extension is done through new versions of the standard. The current version is 1.1 and the project is still evolving.

Project Background: The project started due to the popularity of REST APIs as a means for RPCs. As REST is one of the most popular ways for data exchange in web based applications and Java might be the implementation language of choice, JAX-RS is relevant for helping to easily bind implementation to interface.

5.5 JENA

URL: <http://jena.sourceforge.net/>

License: Open source, with what looks like a BSD-style license. Originally developed by HP.

Purpose: Jena is a Java framework for building Semantic Web applications. It provides a programmatic environment for RDF, RDFS and OWL, SPARQL and includes a rule-based inference engine.

Installation; We had no problems installing JENA in an eclipse environment by copying the respective jar files.

Integration JENA supports interchangeable backends and there are bindings to other triple stores, e.g. Virtuoso.

Extensibility: Full extensibility given, due to open source.

Project Background: JENA is one of the most popular frameworks for dealing with RDF data and many other applications are engineered to work on top of it. JENA can be considered mature.

5.6 SESAME

URL: <http://www.openrdf.org/>

License: Sesame is made available under a BSD-style license. It is freely available, but copyrighted by a company: Arduina Software.

Purpose: Sesame is a toolbox for handling RDF data. It includes an API for access to the RDF data, a triple store and querying facilities with support for SPARQL and SerQL. There are also other tools, for example an annotation based mapper (“Elmo”) for binding Java classes to RDF.

Installation: We installed Sesame using eclipse. This is straightforward and boils down to copying a number of jar files.

Integration: Sesame can be integrated with other backends and we tried this with the Virtuoso server via JDBC. It is also designed to be used in conjunction with web application servers, but we did not verify this.

Extensibility: Extensibility is generally given through the open source approach.

Project Background: This project can be considered stable. It is now in version 2.x and currently applied in a number of existing systems.

5.7 TRDF/TSPARQL

URL: <http://trdf.sourceforge.net/>

License: GNU GPL v3

Purpose: tRDF is a framework that allows to assess the trustworthiness of RDF data. It defines a vocabulary for expressing statements about the trust someone has towards an RDF datum. tSPARQL is an extension of SPARQL, that enables users to query tRDF data while taking trustworthiness calculations into account.

Installation: The framework is based on the JENA project and we had no issues installing the software using eclipse and an existing JENA installation.

Integration: tRDF is a vocabulary defined in OWL, which makes it compatible with the rest of the semantic web world. tSPARQL is defined as an extension of the SPARQL language and currently only implemented on top of the JENA framework.

Extensibility: Both tRDF, as well as tSPARQL, are freely extensible under the GPL.

Project Background: This seems to be the work of mostly one person: Olaf Hartig of HU Berlin. The current version is 0.12 and marked as “pre alpha” - indicating it being far from a mature status. The last update of the project has been in September 2009.

5.8 VIRTUOSO

URL1: <http://virtuoso.openlinksw.com/> (commercial version)

URL2: <https://sourceforge.net/projects/virtuoso/> (community version)

License: Dual License: GPL for non-commercial and a proprietary commercial license

Purpose: Acts as a general-purpose server. Implements an RDF triplestore with querying capabilities via SPARQL. Also encompasses a WebDAV system, a relational database (uses SQL), XML storage, a document server and a web application server

Installation: We tried the installation using the community / open source version. It is straightforward and follows the usual “configure / make / make install” routine.

Integration: Virtuoso offers JDBC / ODBC connectors and REST / SOAP APIs. It has specific modules for integrating with Sesame, JENA and the Hibernate persistence layer (supports JPA).

Extensibility: The open source version can be easily extended. There are a number of options for customization of Virtuoso: at the end of the day, this is a full blown RDBMS system, supporting stored procedures, etc...

Project Background: Virtuoso is mature and we did not encounter any issues during our initial tests. It is recommended by the folks from LOD2. It seems to be rather fast.

5.9 CKAN

Creator: Open Knowledge Foundation: The Open Knowledge Foundation (OKF) is a not-for-profit organization founded in 2004 and dedicated to promoting open knowledge. It is incorporated in England & Wales.

License: GNU Affero GPL < <http://www.fsf.org/licenses/licenses/agpl-3.0.html>>

Architecture layer: CKAN is a metadata store to gather information about many open data sources. It is used as a central instance for storing, searching and discovering different data sources and reveals all necessary information to use the data. It is not about accessing data to use in services, but to give an overview of all available data in the cloud.

Functionalities: CKAN has a web interface to access all functions like searching and publishing new data sources. It also offers a mechanism to query and enter new data over a REST-API to access the data from other services and remote clients. It stores the data in a MySQL database and is completely written in Python.

Purpose: As a store for metadata it doesn't store the data itself, but holds information like, which organization releases what kind of data, where to find it, how to access and what licenses it has. The goal is to install a central point of reference where the developer can find all known open data sources and get some basic information about the published data. It also can be used to tag data and organize datasets into groups.

Maturity: CKAN is currently in version v1.3.2 and in a quite stable release. It has an active community and is under constant development. There are several open data projects in many countries like Great Britain, Germany and Austria which use CKAN as a platform to install an open data portal. The API is at version 2 and provides a good mechanism to automatize querying from other services.

References: The CKAN instance from OKF can be found at <http://ckan.net/>. Great Britain and the city of London use a CKAN backend for their open data portals at <http://data.gov.uk> and <http://london.gov.uk/>. A German instance is provided under <http://offenedaten.de>.

Comments: As a metadata store it provides a good entry point to many open datasets. As well as publishing data on the official OKF CKAN instance, the user can set up its own instance. Because of the limited possibilities to customize the design, some projects prefer CKAN as a backend and export the data to a content management system which handles the layout and the integration of other content. Therefore a Drupal plugin is available too.

5.10 ONTOWIKI

URL: <http://ontowiki.net/>

License: The Agile Knowledge Engineering and Semantic Web group of the University of Leipzig develops OntoWiki and publishes it under GNU GPL v2.

Purpose: OntoWiki is a web application for collaborative editing of semantic data. The intended mode of usage is that a (potentially large) number of co-workers, contribute facts to semantic knowledge bases through a web interface, i.e. using the browser. Conversely, OntoWiki's also aims to provide intuitive and visual access to semantic data. For that widgets are in place to input/output especially dates and places in calendars and maps.

Installation: The installation is straightforward and requires only a functional XAMP system, i.e. any operating system with an Apache webserver, a MySQL database server and a PHP installation. This can be provided within minutes using for instance Debian GNU/Linux, the XAMPP-package.

Integration: OntoWiki can use a Virtuoso triple store as a data backend alternative to a relational database. That in turn can be populated/queried by other tools. If simultaneous editing of one triple store by both OntoWiki and other tools remains to be studied, but threatens to give rise to difficulties.

Extensibility: As this application is written in PHP and with extensibility in mind, it possible with reasonable effort to create domain specific widgets and other means of input and visualization.

Project Background: This tool has a solid academic foundation promises a certain degree of maturity and is continuously developed by team of several experts. It is hosted a Google code.

5.11 APACHE TIKI

URL: <http://tika.apache.org/>

Purpose: Apache Tika is an open source content analysis and detection toolkit to support the extraction of data and metadata for varying file formats. In order to setup a local version of your computing system, you will need to download and install both *Maven2* and *Tika*, and possibly "*Java SE 6 Update 24*" (JDK), if not already installed.

The Tika toolkit supports the following file formats:

File Format	File Format
<ul style="list-style-type: none"> Hyper Text Markup Language (* .html) 	<ul style="list-style-type: none"> Compressed & Packaged (* .ar, *.cpio, *.tar, *.zip, *.gzip, *.bzip2)
<ul style="list-style-type: none"> XML & Derivations (* .xml) 	<ul style="list-style-type: none"> Text (* .txt, ...)
<ul style="list-style-type: none"> Microsoft Office Documents (* .docx, *.xlsx, ...) 	<ul style="list-style-type: none"> Audio (* .au, ...)
<ul style="list-style-type: none"> Open Documents (* .odf) 	<ul style="list-style-type: none"> Image (* .jpg, ...)
<ul style="list-style-type: none"> Portable Documents (* .pdf) 	<ul style="list-style-type: none"> Video (* .wmv, ...)
<ul style="list-style-type: none"> Electronic Publication (* .epf) 	<ul style="list-style-type: none"> Java Class Files and Archives (* .class, *.jar)
<ul style="list-style-type: none"> Rich Text (* .rtf) 	<ul style="list-style-type: none"> mbox (...)

License: Apache License, Version 2.0, <http://www.apache.org/licenses/LICENSE-2.0>

6 CONCLUSION

We presented basic concepts in regard to a platform for the provisioning of PSI in the Open Cities project. We then continued to document and classify functional, as well as non-functional requirements that are relevant for the design of the platform, which are serving as the foundation to the design and implementation processes.

In the second part of this document we give an overview of a number of tools that are relevant for creating and maintaining the platform and its data assets. We would like to point out that the motivation for this Section has been to generally identify tools and to survey a relevant part of the technology landscape. As such, these tools are only potential candidates for an utilisation in the project. We will now evaluate these tools during the creation of the platform prototype and will find out how well they match to our needs and the resulting software.

7 BIBLIOGRAPHY

DAMA International. (2009). The DAMA Guide to the Data Management Body of Knowledge (DAMA-DMBOK). 1. Technics Publications, LLC.

ISO/IEC. (2003). International Standard 11179-3, Information technology — Metadata registries (MDR) — Part 3: Registry metamodel and basic attributes.

ISO/IEC. (2005). International Standard 11179-6, Information technology — Metadata registries (MDR) — Part 6: Registration.