



## D3.4 VISUALISATIONS, DASHBOARDS AND MULTILINGUAL INTERFACE STATUS REPORT 1

### PROJECT

Acronym: **OpenDataMonitor**  
Title: Monitoring, Analysis and Visualisation of Open Data Catalogues, Hubs and Repositories  
Coordinator: SYNYO GmbH

Reference: **611988**  
Type: Collaborative project  
Programme: FP7-ICT

Start: November 2013  
Duration: 24 months

Website: <http://project.opendatamonitor.eu>  
E-Mail: [office@opendatamonitor.eu](mailto:office@opendatamonitor.eu)

Consortium: **SYNYO GmbH**, Research & Development Department, Austria, (SYNYO)  
**Open Data Institute**, Research Department, UK, (ODI)  
**Athena Research and Innovation Center**, IMIS, Greece, (ATHENA)  
**University of Southampton**, Web and Internet Science Group, UK, (SOTON)  
**Potsdam eGovernment Competence Center**, Research Department, Germany, (IFG.CC)  
**City of Munich**, Department of Labor and Economic Development, Germany, (MUNICH)  
**Entidad Publica Empresarial Red.es**, Shared Service Department, Spain, (RED.ES)

## DELIVERABLE

Number:	<b>D3.4</b>
Title:	<b>Visualisations, dashboards and multilingual interface status report 1</b>
Lead beneficiary:	SYNYO
Work package:	WP3: Concept Design and Software Development
Dissemination level:	Public (PU)
Nature:	Report (R)
Due date:	October 31, 2014
Submission date:	October 31, 2014
Authors:	Ejona Sauli, SYNYO Michael Heil, SYNYO Peter Leitner, SYNYO
Contributors:	Dimitris Skoutas, ATHENA Vassilis Kaffes, ATHENA Tom Heath, ODI Ulrich Atz, ODI Elena Simperl, SOTON Yunjia Li, SOTON Bernhard Krieger, IFG.CC Sirko Hunnius, IFG.CC
Reviewers:	Bernhard Jäger, SYNYO

**Acknowledgement:** The OpenDataMonitor project is co-funded by the European Commission under the Seventh Framework Programme (FP7 2007-2013) under grant agreement number 611988.

**Disclaimer:** The content of this publication is the sole responsibility of the authors, and in no way represents the view of the European Commission or its services.

## TABLE OF CONTENTS

<b>1. Introduction .....</b>	<b>7</b>
Purpose.....	7
Scope .....	7
References.....	8
<b>2. Development Environment.....</b>	<b>10</b>
<b>3. Data Analysis .....</b>	<b>12</b>
3.1. Status Report.....	12
3.2. Processing.....	12
3.3. Synchronization .....	12
3.4. Next Steps.....	15
<b>4. Data Reporting .....</b>	<b>16</b>
4.1. Technological Landscape and Approaches.....	16
4.1.1. Overview of the Solution Approaches.....	16
4.1.2. Template-based Report Generation Tools Overview.....	17
4.1.3. Tool Selection .....	20
4.2. Architecture & Workflow .....	20
4.2.1. Report Designer.....	21
4.2.2. Report Engine .....	23
4.3. Implementation Status.....	24
4.3.1. General Report Structure .....	24
4.3.2. European Dashboard Report.....	24
4.3.3. Data Catalogue Report .....	26
<b>5. Graphic User Interface Status Report .....</b>	<b>27</b>
5.1. Data Catalogues Dashboard .....	27
5.2. Data Catalogues List View .....	28
5.3. Data Catalogue Profile Dashboard .....	29
5.4. Data Catalogue Profile List View .....	30
5.5. Dataset Profile .....	31
5.6. Groups .....	31
5.7. Internationalization (i18n).....	32

<b>6. Know-How .....</b>	<b>35</b>
6.1. Home .....	35
6.2. Knowledge Base .....	36
6.3. Forum .....	38
6.4. FAQ.....	39
<b>7. Conclusion .....</b>	<b>40</b>
<b>Appendix I. Glossary of acronyms and abbreviations.....</b>	<b>41</b>
<b>Appendix II. Terminology definition .....</b>	<b>42</b>

## LIST OF FIGURES

Figure 1: Forum/OpenDataMonitor platform category .....	11
Figure 2: Google Trend comparison .....	19
Figure 3: The architecture of Eclipse BIRT from a workflow point of view. ....	20
Figure 4: The BIRT Report Designer interface .....	22
Figure 5: The BIRT Chart Designer interface .....	23
Figure 6: Page two of the European dashboard report .....	25
Figure 8: Data Catalogues Dashboard .....	27
Figure 7: Data Catalogues List View .....	28
Figure 9: Data Catalogues Profile Dashboard.....	29
Figure 10: Data Catalogues Profile List View.....	30
Figure 11: Dataset Profile.....	31
Figure 12: Groups .....	32
Figure 13: Language selector of the ODM platform.....	32
Figure 14: Table schema of the i18n table .....	33
Figure 15: Possible entry in i18n table .....	34
Figure 16: Knowledge Base Home.....	35
Figure 17: Knowledge Base .....	36
Figure 18: Article View .....	37
Figure 19: Forum .....	38
Figure 20: FAQ.....	39

## LIST OF TABLES

Table 1: References to other resources .....	8
Table 2: Development environment specifications .....	10
Table 3: Possible values for cron jobs .....	13
Table 4: Basic procedure of template based report generation. ....	17
Table 5: List of all report generation solutions .....	17

## 1. Introduction

### Purpose

The purpose of this document is to provide a status report on the implementation of Open Data Monitor end-user interfaces including planned dashboards, visualisation elements, reporting capabilities and other basic features. This report requires knowledge of software engineering and general information about the domain of application (Open Data Landscape). The target audience of this report include project partners, the European Commission and relevant audiences with technical knowledge of open data and software development concepts.

The work being undertaken in work package 3, consist on the actual implementation of the platform and related supporting services and products. The technical implementation of the platform and supporting services will be extended throughout the two project years, while producing the first prototype after the first project year. This report will give some light to the implementation status of the Presentation layer of the platform. This report will take for granted the implementation of Metadata Harvester, Metadata Repository, Harmonisation Engine, Catalogue Registry and Job Manager. These components are defined in D3.3.

### Scope

The main objective of this project is to create sophisticated methods to monitor data catalogues, harvest and analyse the metadata of the datasets published in them, and provide comprehensive visualisations to compare existing open data resources. Using standardised APIs (e.g. CKAN2), it will be possible to analyse data usage, file formats, updates, licenses and further metadata to statistically describe and visualise it. This information will be used to identify trends, gaps and potentials of open data resources, and also to build a scalable open data monitoring concept using metadata, parameters and key-indicators.

The end product of this project will mainly consist of an open data platform that will collect and harmonize the metadata of existing open data, meanwhile developing flexible feature for easily and automatically future open data harvesting. The unique values of this project in comparison to other existing open data platforms include:

- Aggregate metadata of open datasets from all European countries, mainly member states, and build dashboards upon the metadata harvested.
- Creation of graphical and user friendly dashboard based on main level of data abstraction.
- Automatically harvest metadata of open data published by existing data catalogues either from published APIs or via scraping techniques.
- Use state of the art visualization user interface elements to better represent the metadata and their specific context in order to address the target audiences accordingly.

## References

This report is based on the technical decision documented in the previous deliverable. Additionally, references to official online resources are being used to support the concept and terms used in this document. The main references to the other project's deliverables are listed in the following table, the references link type are stated as a footnote in each page and a full list of academic references can be found at the end of this report (section **Fehler! Verweisquelle konnte nicht gefunden werden.**).

**Table 1: References to other resources**

Deliverable	Description
D2.2 Monitoring methods, architectures and standards analysis report <i>July 2014</i>	Deliverable 3.2 makes use of the analysis results of D2.2 to proper define the use cases, related business requirements and non-functional requirements related to performance and software quality. The work was conducted in parallel, in order to obtain consistency of technical concepts and decisions made.
D2.3 Best practice visualisation, dashboard and key figures report <i>July 2014</i>	This report was used as a base input for conceptualizing and creating functionalities, user interfaces and respective use cases. The full report was taken into account to make proper decisions on the business requirements, specifically defining the software interfaces between different architecture layers. The work undertaken for D2.3 was conducted in parallel with the work for the present deliverable, in order to obtain consistency of project business logic and best cases to represent data.
D2.4 Open data stakeholder requirement report 1 <i>September 2014</i>	The output of the research conducted for this deliverable has been synchronised with the business requirements of D3.2. The synchronisation required is very essential in order to keep the business logic consistent.
D2.5 Open data resources, platforms and APIs collection 1 <i>September 2014</i>	Related to, and in conjunction with, D2.1.
D3.1 Scalable open data monitoring concept and framework design <i>May 2014</i>	In this deliverable the platform architecture has been defined, and this design and technical planning is used in D3.2 to further analyse and specify requirements and interfaces.
D3.2 Tool specifications, use cases, mockups and functionalities status report 1	D3.2 is a complementary document to the current one, focusing rather at the system level and the product perspective, describing overall requirements and functionalities, user roles and interfaces, while the



<i>July 2014</i>	current document details the implementation status of the visualisation of dashboards and interfaces.
D3.3 Tool architecture and components/plugins programming status report 1 <i>October 2014</i>	The purpose of this document is to present the overall architecture of the ODM platform, and to report the status of the implementation for each of the main components involved

## 2. Development Environment

During the development the technical group was working and two separate development environment. The first environment, referenced later in this document as the backend, is used to perform the harvesting and harmonisation process (find additional information in D3.2, sec 2.3 and D3.3) of the open data in Europe. The second environment, referenced later in this document as the front end consists of the analysis and visualisation layer applied to harmonised data. The specifications for this environment can be found in **Fehler! Verweisquelle konnte nicht gefunden werden..** The programming languages, data source and other libraries are selected based on the Open Source license attribute. The communication between these two environments is being achieved through RESTful services described in detail in D3.2, section 4.2.

**Table 2: Development environment specifications**

Resource Type	Name	Version	License
<b>HTTP server</b>	Apache	2.4	Apache 2.0 License
<b>Programing Language</b>	PHP	5	PHP License v3.01,
<b>Database</b>	MYSQL	5.5.36	MySQL Community Edition
<b>PHP Framework</b>	Yii Framwork	2.0.0	GNU Free Documentation License (GFDL)
<b>HTML, CSS &amp; JS Framework</b>	Bootstrap	3.3.0	MIT License
<b>JavaScript Library</b>	D3.js	3.4.13	Open Source
<b>JavaScript Library</b>	NVD3	1.1.15	Apache 2.0 License
<b>JavaScript Library</b>	jQuery	2.1.1	MIT License
<b>Translator Library for raster and vector geospatial data formats</b>	gdal	1.11	X/MIT
<b>GeoJson extension</b>	TopoJson	1.1	Open Source
<b>Programing Language</b>	python	2.7	PSF LICENSE
<b>Operating System</b>	Linux Fedora	14	LICENSE AGREEMENT FEDORA(TM) 14
<b>Version Control Software</b>	Bitbucket	101	Atlassian Customer Agreement
<b>Browser Extension</b>	Firebug	1.4	BSD License

The front end is being set up in three instances: Development, Testing and Production. The development instance is set up at SYNYO's development domain, and additionally in Bitbucket as a private repository, in order for all developers to access it and have a safer version control process. The repository will be opened to the broad public after the usability tests and code review is finished. The repository will be available to the community through github and bitbucket.

After a mature prototype, the front end instance will be published into SYNNO's testing domain for a selective range of end users to test it and send feedbacks. Except for the planned development tasks, the technical team will also fix the bugs found or develop new requirements received from the project stakeholders. Through the Forum section of Know-how application the project will be able to interface with the community and receive feedback on potential bugs or enhancements.

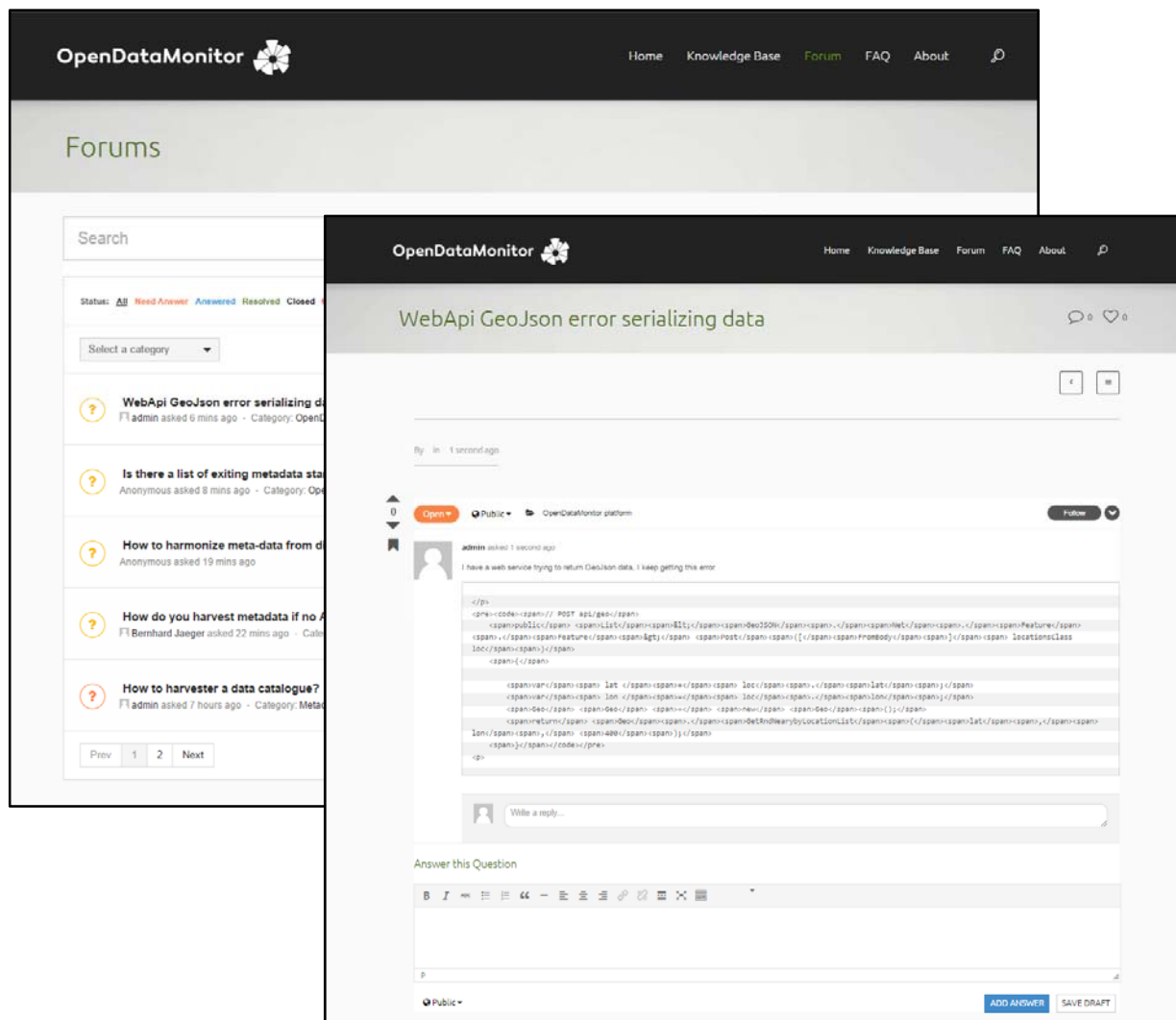


Figure 1: Forum/OpenDataMonitor platform category

The know-how application is described in section 6 of this document. Interest to this section is the possibility to post questions under “OpenDataMonitor” platform category and availability to be replied from registered and anonymous users.

The production environment will be set after positive results from several technical and user tests. The final product will be published until month 22 of the project timeframe under the reserved domain name **opendatamonitor.eu** which at the current stage it redirect to the project website (project.opendatamonitor.eu).

## 3. Data Analysis

### 3.1. Status Report

As a result of WP2, 217 data catalogues have been identified by the consortium members. Of these **217** items, **31** have been processed and harvested using various techniques described in earlier deliverables. In total, around **130,000** datasets are covered by them, which results in a representative portion of the whole European open data landscape already.

Using these datasets, reasonable metrics can already be calculated. Of the **70** metrics defined in deliverable D2.3, **32** have already been implemented, ready to be used for visualizations on the web page or a report. In the second iteration, when visualizations and charts are created, some more metrics might be defined.

### 3.2. Processing

As specified, the actual conduct of the data analysis is done in the Analysis Engine which resides outside the Demonstration Site and the CKAN module. The Analysis Engine draws the data from the Data Storage Unit of the CKAN module, performs the calculations (like for instance counts the datasets per license, date, catalogue, etc.) and exposes the results via a REST web service in the JSON format.

In order to reduce load time and network traffic, these results are stored in a database within the Demonstration Site. Therefore, the REST service is queried in regular intervals, daily at the moment using a cronjob which triggers a Java respectively PHP console application, and the results are stored, thus implementing a caching-functionality. Another issue that covered by doing so is availability: if the Analysis Engine would not be available due to any reason, the metrics would still be available, despite being slightly out of date. Additionally, the harmonized metadata is also synchronized from time to time to even decrease latency.

Technically, both the metrics and metadata are stored within the same database as there is an intersection (e.g. data catalogues or country information). However, the metrics part is organized as a star-schema, meaning that the values are stored in a fact table and around that, the dimensions are referenced as foreign keys. Current dimensions are time, country, city, state, organisation, catalogue and a generic dimension which is used for various things like license, file type, etc.

### 3.3. Synchronization

The briefly mentioned synchronization of the metrics and the metadata in the previous section is investigated in more detail in this section.

Basically there are three different synchronization jobs implemented at the current stage of the project. As mentioned before, those jobs are currently executed on a daily basis via cron jobs defined on the server. In essence all jobs use the provided REST service of the Analysis Engine in order to synchronize the desired data.

The different jobs are listed and discussed below:

- Data Catalogues Job

The first cron job is defined to retrieve data about the catalogues harvested within the ODM project. Data retrieved are for example: url, title, description or country.

- **Datasets Job**

With the execution of the second job the datasets get synchronized. The retrieved datasets are directly matched to a certain data catalogue and stored in the database. Some important values for the dataset are a unique identifier created in course of the harvesting of the datasets, the title, license, and the resources of the datasets.

- **Metrics Job**

The defined metrics are also imported into the database of the Demonstration Site. The synchronization is also handled via a cron job. As mentioned before the metrics tables of the database are organized according to a star-schema which helps to retrieve data in an effective way.

Based on the type of job, either a PHP script implemented within the Yii Framework or a Java console application is executed. Both data catalogues and datasets are retrieved via PHP and the metrics are retrieved via Java. This situation can be reasoned with the fact that the synchronization of the data was developed simultaneously for the data reporting and the data visualization. Since the reporting component, which is addressed in section 4 of this document, is implemented in Java, the synchronization for the metrics was also written in this language. In a later stage of the development the decision for either Java or PHP for the synchronization of the data is taken and then the respective code is translated in order to have a structured methodology to address the issue of data synchronization.

As mentioned earlier the jobs run on a daily basis at the moment. This interval must be monitored and analyzed in order to find the right amount of executions for providing the best possible results concerning the Demonstration Site of the ODM project. The execution itself is maintained via the Linux Cron utility. With the help of this utility it is easy to define a schedule for the synchronization task. The command to utilize this functionality is “crontab”. With the help of this command cron jobs can be added, removed and edited. The command follows a simple syntax consisting out of six fields, which are: <minute>, <hour>, <day of month>, <month>, <day of week>, <command>. Table 3: shows the different possible values for each field.

**Table 3: Possible values for cron jobs**

Name	URL
minute	0-59
hour	0-23
day of month	1-31
month	1-12 (or names)
day of week	0-6 (0 is Sun, or names)

The following command example would execute the job daily at 6 AM.

```
00 6 * * * <command>
```

In order to provide better insight into the topic of synchronization, a code snippet of the initial retrieval of all datasets implemented within PHP is displayed in the following. Note that the provided code is simplified in order to show the important parts of the data retrieval process.

```
public function actionDatasets_all() {
    $url = 'http://.../api/v1.0/_find?count=1'; // URL of API
    $content = file_get_contents($url);
    $json = json_decode($content, true);

    $connection = \Yii::$app->db;

    $batch_size = 100;
    $target = ceil(intval($json['count']) / $batch_size);

    for ($i = 0; $i < $target; $i++):
        $offset = $i * $batch_size;
        $url = 'http://.../api/v1.0/_find?batch_size='.$batch_size.'&offset='.$offset;
        $content = file_get_contents($url);
        $json = json_decode($content, true);

        $transaction = $connection->beginTransaction();
        foreach ($json['results'] as $item):
            $dataset = new \common\models\Dataset;

            $dataset->odm_id = $item['odm_id'];
            $dataset->title = ...
            $dataset->notes = ...
            .
            .
            .
            $dataset->save();
        endforeach;
        $transaction->commit();
    endfor;
    echo 'End of import.';
}
```

First of all the API is called with parameter “count=1” which provides the total amount of datasets. On basis of this count and the batch\_size the target value for the loop is calculated. This step is performed in order to avoid errors because of a memory overload. Within the loop the offset for the API call is calculated so that with each call of the API new datasets are loaded. The API provides the data in form of a JSON array which is read and looped through within the for each loop. In this loop an Active Record object is created, filled with the gathered data and in a final step saved and therefore stored to the database. The transaction is used to increase the performance of this code snippet, since database calls are collected and sent in bulk. After running through all loops the import is finished and a notification is displayed.

### 3.4. Next Steps

Concerning the data analysis the basic functionalities are implemented at the current stage of the project. The next step concerning this part of the project will focus on the quality improvement in the different areas. Examples for these improvements are as follows:

- Unify synchronisation of data  
As pointed out before, the current synchronization process is done via PHP and Java for explained reasons. This is not a perfect situation which will be addressed in the future.
- Cron job logging  
The cron jobs are running and so far no issues were presents. Nevertheless this part needs a profound logging in case errors occur.
- Monitoring of the synchronization process  
The synchronisation is not monitored at the moment. In order to be aware of changes and retrieved metadata respectively metrics, this part of the application should be enhanced with monitoring capabilities.
- Register the new implemented metrics and visualise the results using the best practice from the standard Visual Analytics Digital Library.

## 4. Data Reporting

The reporting component represents an integral part of the ODM project as it allows creating reports optimized for the paper format. These reports are intended to comprise the same information as is conveyed via the web page and hence makes use of the same data basis, but presented a slightly different layout and format. This includes taking into consideration structural elements such as headers, footers, page numbers, and also a front page or bookmarks. Such report can easily be used for information reuse as no database, or no data in general needs to be accessed on reading the document. Further, spreading information is facilitated as pre-generated PDF reports can be distributed as convenient as possible for example via mail with the guarantee that the content is not subject to alteration.

As described in the specification, there will be reports for each major view in the web page, including the geographical and data catalogue dashboards and list views of catalogues and datasets, respectively. The current implementation which will be discussed in detail later encompasses two reports: the European dashboard and the generic data catalogue report.

### 4.1. Technological Landscape and Approaches

In this section an overview about the different approaches that are available to create a report in very general is presented. After this overview the selected approach is discussed in more detail, and eventually the tool which is used in the ODM project is selected.

#### 4.1.1. Overview of the Solution Approaches

In deliverable D3.1 section 4.5 an analysis was conducted which put light on the different strategies that are available in order to close the gap between the data source and a visually appealing report, most preferably in the PDF format. They were grouped into three groups: imperative (report is built from scratch programmatically), declarative (a template that has been created before is filled) and transformation-based (a different visualisation is converted) strategies. Those different strategies included:

- **Low-level APIs:** creating PDF documents imperatively with high flexibility but to the cost of high complexity
- **High-level Layouting:** reports are created and filled with data according to predefined templates
- **Document Templates:** making use of well-known text processing programs such as Microsoft Word, LibreOffice Writer, etc.
- **XSL-FO Processing:** XML-transformation based PDF generation
- **HTML/CSS Converting:** transforming HTML web sites directly into PDF documents according to printing-optimized CSS sheets
- **LaTeX:** another template based dynamic report generation system.

From these solutions, following a declarative or transformation based approach turned out to be the best way of implementing the required functionality. However, in course of the refinement of the requirements and the definition of the use-cases, which included a closer consideration of the

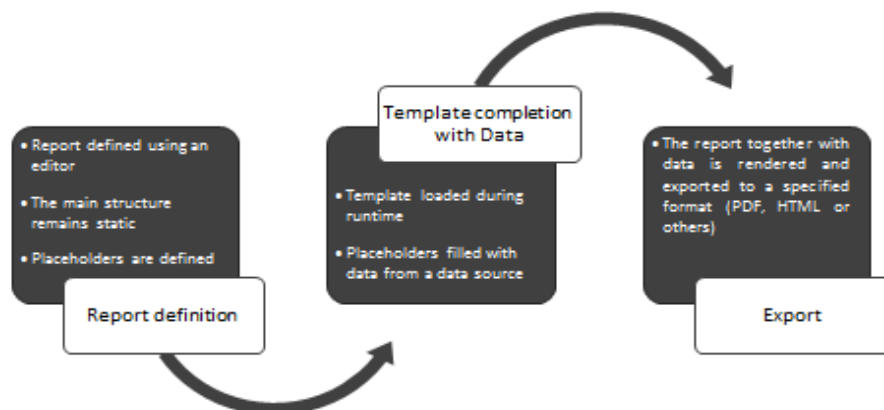


elements that the reports shall contain and their layouts, the decision was eventually made to make use of a separate reporting component that follows its own report definition, instead of taking the web page as the template. By that, flexibility is kept at an acceptable level and while not losing much of the convenience in definition.

#### 4.1.2. Template-based Report Generation Tools Overview

In the following, the working principles of this technology are briefly summarized, followed by an overview of the tools that are available. In the end, the most promising tools are compared to provide a basis for the final decision.

The basic workflow of report creation is shown in Table 4:. First of all, the report itself, including the layout, the arrangement of elements and the data scheme that is used as the source is defined using a designer that is part of the respective solution. This declarative definition, which in most cases is a simple XML file, is then processed at runtime by the report generator where the information of the data sources that are involved is used to populate tables and generate charts. Finally, the result is rendered and exported to a specific format.



**Table 4: Basic procedure of template based report generation.**

The following Table 5: lists all solutions that were found to be relevant to our context in alphabetic order.

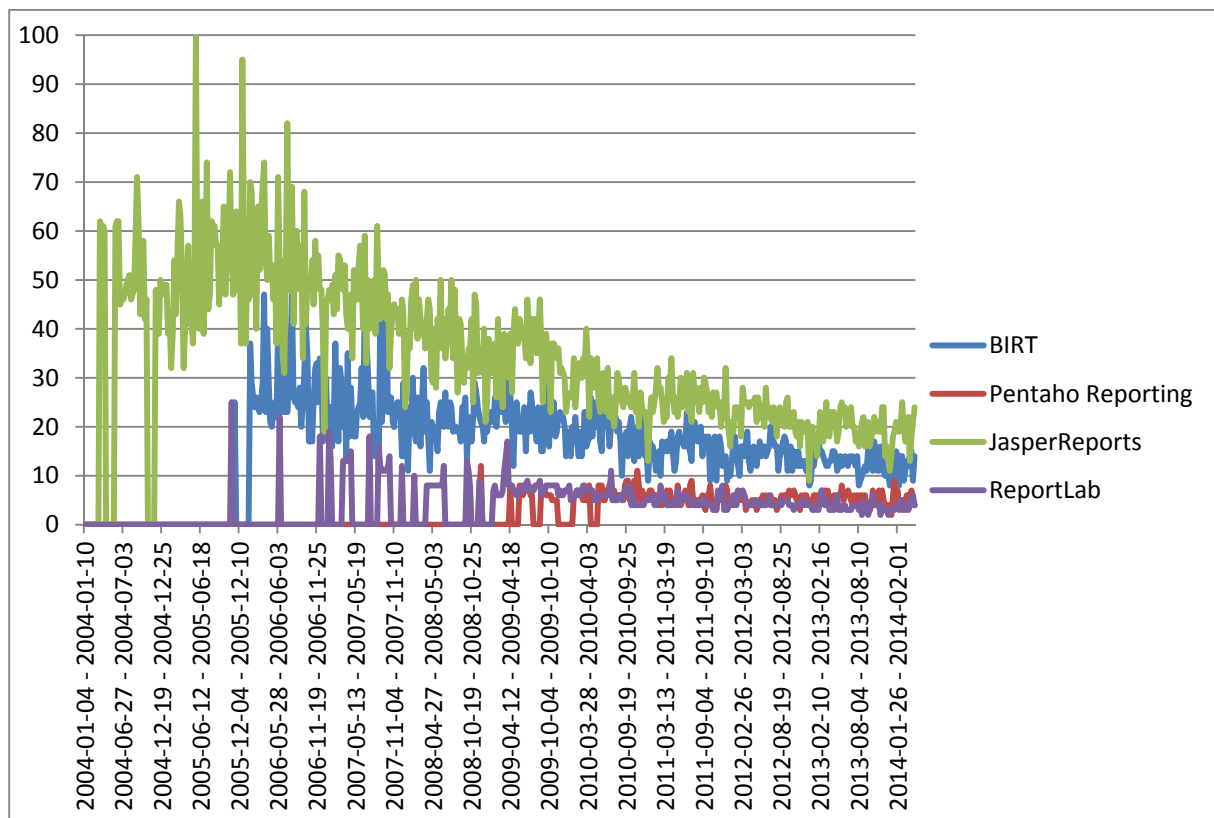
**Table 5: List of all report generation solutions<sup>1</sup>**

Name	URL	OS	Platform	Comments
<b>BIRT</b>	<a href="http://www.eclipse.org/birt/">http://www.eclipse.org/birt/</a>	x	Java	Uses XSL-FO Reports are saved as RPTDESIGN (XML) Eclipse integrated designer
<b>Crystal Reports</b>	<a href="http://www.crystalreports.com/">http://www.crystalreports.com/</a>		.NET/Java	Complete reporting solution - not just for printing Reports are saved as RPT (XML) Formerly integrated in the .NET world, now part of SAP
<b>DataVision</b>	<a href="http://datavision.sourceforge.net/">http://datavision.sourceforge.net/</a>	x	Java	Out of date - last version from 2008! Similar to Crystal Reports
<b>i-net Clear Reports</b>	<a href="https://www.inetsoftware.de/de/products/clear-reports">https://www.inetsoftware.de/de/products/clear-reports</a>		Java/.NET	Complete reporting solution - not just for printing Reports are saved as RPT (XML)

<sup>1</sup> Solutions that have their own definition format and a convenient editor. The coloured column 'OS' stands for open source and green means positive.

<b>JasperReports</b>	<a href="http://community.jaspersoft.com/">http://community.jaspersoft.com/</a>	x	Java	Complete BI solution Creation of diagrams using JFreeChart Reports are saved as JRXML (XML) Eclipse integrated designer (Jaspersoft Studio) Uses iText Considered best open source high-level solution for paper reports
<b>Microsoft SQL Server</b>	<a href="http://www.microsoft.com/de-de/server/sql-server/2012/default.aspx">http://www.microsoft.com/de-de/server/sql-server/2012/default.aspx</a>		.NET	Report Services Engine Reports are saved as RDL (XML)
<b>OpenReport</b>	<a href="http://sourceforge.net/projects/openreport/">http://sourceforge.net/projects/openreport/</a>	x	Python	Supposedly out of date RML2PDF converter
<b>Oracle BI Publisher</b>	<a href="http://www.oracle.com/technetwork/middleware/bi-publisher/overview/">http://www.oracle.com/technetwork/middleware/bi-publisher/overview/</a>		Oracle	
<b>Pentaho Reporting</b>	<a href="http://community.pentaho.com/projects/reporting/">http://community.pentaho.com/projects/reporting/</a>	x	Java	Complete BI solution Formerly known as JFreeReport Reports are saved as PRPT (XML)
<b>PL/PDF</b>	<a href="http://www.plpdf.com/">http://www.plpdf.com/</a>		PL/SQL	Works with Oracle databases only
<b>ReportLab</b>	<a href="http://www.reportlab.com/">http://www.reportlab.com/</a>	(x)	Python	Used by Wikipedia for instance Creation of charts as bitmap or vector formats Reports are saved as RML (XML) RML can be used with PLUS version only
<b>Scriptura Engage</b>	<a href="https://www.inventivedesigners.com/products/scriptura">https://www.inventivedesigners.com/products/scriptura</a>		Java	
<b>XF Rendering Server</b>	<a href="http://www.ecrion.com/products/xfrenderingserver/overview.aspx">http://www.ecrion.com/products/xfrenderingserver/overview.aspx</a>		Java/.NET/WS	XF Designer to design XSL-FO documents
<b>XSLfast</b>	<a href="http://www.xslfast.com/">http://www.xslfast.com/</a>		Java	Uses FOP
<b>z3c.rml</b>	<a href="https://pypi.python.org/pypi/z3c.rml">https://pypi.python.org/pypi/z3c.rml</a>	x	Python	RML alternative

There exist some commercial suites (for which the list above most likely is incomplete) which have been on the market for quite some time already, as for instance Crystal Reports (which is now part of SAP), the Microsoft SQL Server Report Services or Oracle BI Publisher for Oracle databases. However, over the time, mainly in the first half of the 2000 decade, also open source solutions found their way on the market. In many cases, reporting represents only one single component of a more comprehensive business intelligence solution offering for instance ad-hoc-reporting and OLAP access. Figure 2 shows a Google Trend comparison among the most influential solutions of the last 10 years. Due to its dominance, Crystal Reports has been excluded from the diagram as it makes it impossible to obtain reasonable figures for the other tools then. Nevertheless, it has lost some part of its popularity, whereas JasperReports, BIRT and Pentaho Reporting are coming closer today.



**Figure 2: Google Trend comparison<sup>2</sup>**

Among these open source tools, JasperReports can be considered the most popular one to date.<sup>3</sup> However, as these tools are rather similar, a lot of comparisons have been made which shall be summarized briefly here. First of all, all tool vendors (Actuate, Jaspersoft and Pentaho) offer commercial suites around their open source protégés, which provide further features and convenience (as for instance SaaS solutions). BIRT's design paradigm orientates on web page design where elements are arranged in general layout structures and the rendering engine is allowed a little flexibility concerning the final arrangement. In contrast, in the JasperReports and Pentaho designers elements are placed pixel-precisely with the guarantee that the resulting report looks exactly the same, which can turn out more convenient when creating paper reports. Concerning the elements supported, the solutions are rather similar – among the most discriminating issues is that Pentaho does not support tables, multi-column layouts are supported by JasperReports only and CSS styling capabilities by BIRT only. JasperReports' approach to include multiple data sources into one report has been criticised as it is not supported inherently but requires a workaround by using sub-reports. The data source support is also solid for all three tools, but Pentaho supports most systems, followed by JasperReports. Further, common chart types like bar, pie, XY, line, scatter plots, etc. are supported by all tools but beyond that, each of them supports some special types like pyramid, radar,

<sup>2</sup> Comparison of all major reporting solutions without Crystal Reports. In order to obtain reasonable figures, a combination of various search terms has been formulated for each tool.

<sup>3</sup> <http://www.innoventsolutions.com/jasper-review.html>

thermometer, Gantt, etc. Graphs are exported as scalable vector graphs instead of images by BIRT and JasperReports.<sup>4</sup>

Generally, BIRT and JasperReports are considered best for reports with a medium to high complexity whereas Pentaho Reporting is appropriate for simpler reports. This is due to its designer which is easy to use even for non-technicians, but when it comes to more complex elements like charts, its usability and flexibility drops sharply. In fact, the Pentaho reporting facilities, as they represent a sub-component of the overall BI solution only, are not designated to be integrated into other applications at all. The situation is different with JasperReports, which allows for integration despite being a BI suite, and especially different with BIRT as its core functionality is reporting itself. JasperReports is the oldest solution and, as a consequence, also widely spread, but this also implies that the designer is not that convenient and a little outdated compared to the other solutions. In turn, BIRT offers a user-friendly designer and a very convenient chart creation wizard. It further puts emphasis on reusing elements defined previously and offers the most comprehensive documentation.

#### 4.1.3. Tool Selection

As a consequence, the tools that seem most appropriate for meeting the requirements are JasperReports and BIRT. Whereas the former offers a widely spread and approved solution that allows for a pixel-precise definition of the elements, BIRT path is more progressive in some aspects by supporting formatting via CSS for instance. Further, the creation of a report is more convenient due to the higher usability of its designer on the one hand and it's relaxed positioning approach. If needed, both also offer a small tool to visualize the report in HTML within a server component. Eventually, the decision was made for BIRT.

## 4.2. Architecture & Workflow

In this section an overview about the architecture and workflow of report generation using BIRT is provided. The basic architecture is depicted in Figure 3.

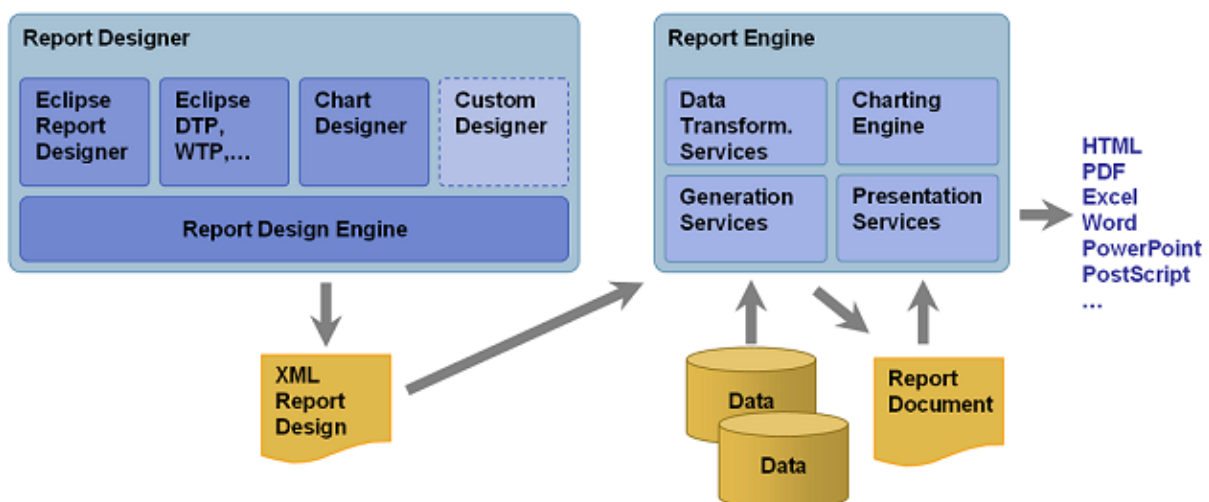


Figure 3: The architecture of Eclipse BIRT from a workflow point of view.<sup>5</sup>

<sup>4</sup> <http://www.innoventionsolutions.com/comparison-matrix.html>

<sup>5</sup> <http://eclipse.org/birt/about/architecture.php>

#### 4.2.1. Report Designer

First of all, the idea of the layout of the report, as inspired from the corresponding web site, has to be turned into a report definition. Within BIRT, reports are defined as plain XML file following a specific scheme. Instead of directly implementing the layout in XML, however, the BIRT Report Designer provides the means for a convenient arrangement of the elements. As depicted in the figure above, the Report Designer consists of various modules, including the Chart Designer, which facilitates the creation of charts. The following chart types are currently supported (version 4.4.1): bar, line, area, pie, meter, scatter, stock, bubble, difference, Gantt, radar (spider) and some others. Another rather important outcome of the Report Designer is the data connection and query definition. Hence, the database connection string is contained in the report design so that it can be generated afterwards without any further information.

The following Figure 4 shows the BIRT Report Designer interface. On the left hand side on the top there are the different items that can be placed on the report. The most important ones are *Label* (static text), *Text* and *Dynamic Text* (text that can hold markup language), *Image*, *Grid* (used for arranging other elements), *Table* and *Chart*. Below, there is the outline which visualises the meta-information of the report and the actual content in a tree view. An additional element that is shown becomes important especially when fine-tuning the reports, that is styles. They are defined for the report as a whole and then applied to different elements in order to have a consistent look. From a technical perspective they are similar to CSS style definitions. The meta-information includes the *Data Source* (in terms of connection string and information; a report can have multiple data sources), the *Data Sets* (data queries based on a data source), *Data Cubes* (for advanced information processing), *Report Parameters* and *Variables*.

On the right hand side the report layout is depicted in an abstract way, meaning that no actual data from the data sources is involved. It is possible to switch to the XML source-view which shows the report in raw XML, which is sometimes necessary when looking for report elements. Eventually, there is also a preview button which renders the report with an extract of the data.

Eventually on the right bottom side the Property Editor allows to alter the properties of a single element on a report, as for instance a grid or text element. Depending on the type of element, different settings are available.

Elements that rely on data drawn from one of the data sources make use of a binding to a respective data set. This binding means that the element is logically connected to the data set, providing parameters where needed as input and obtaining the result as output. Further, a filter can be applied which basically is equally powerful as the SQL where-statement. This binding can be edited using the Property Editor as well for normal elements and the Chart Designer for charts (Figure 5).

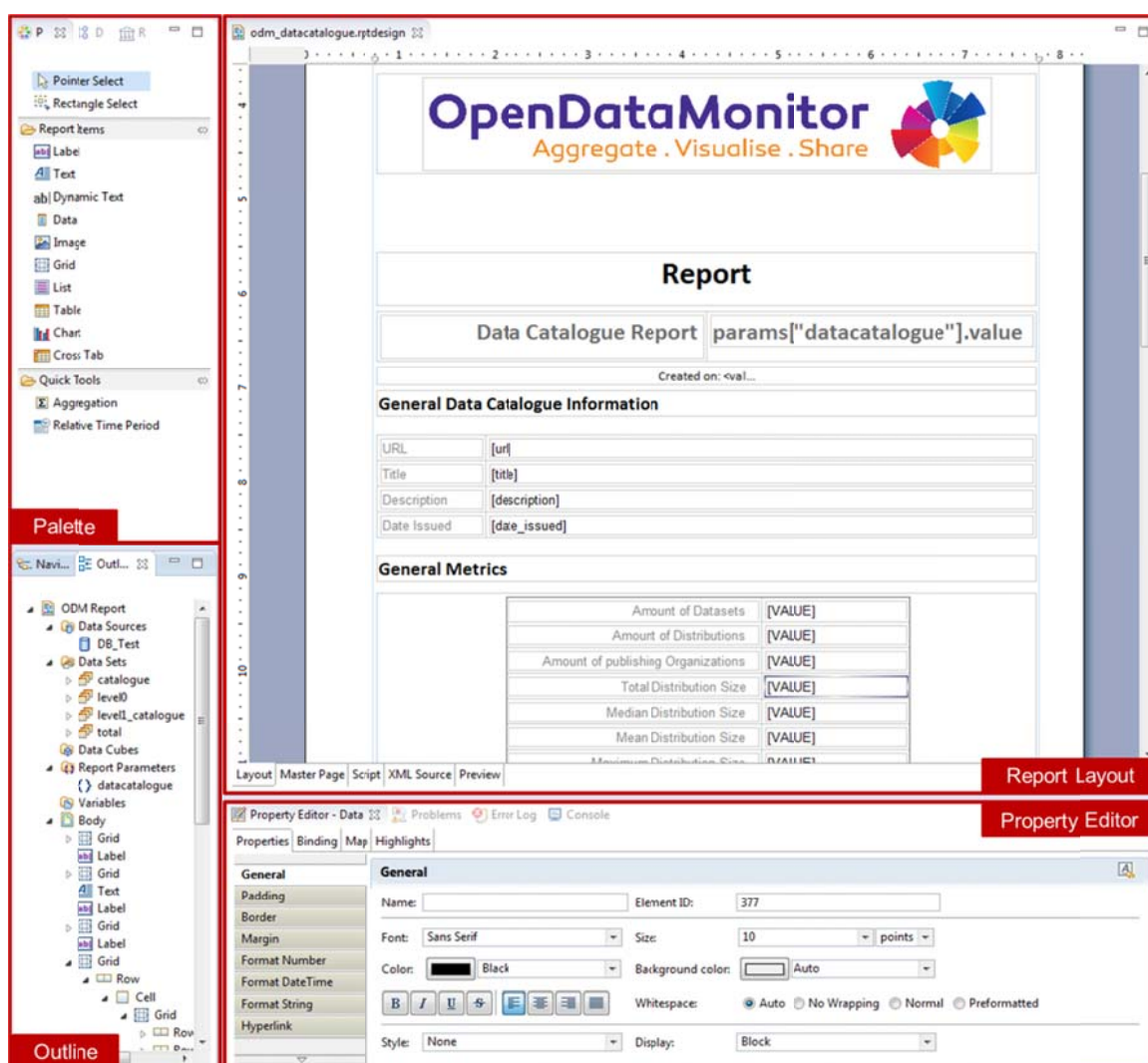


Figure 4: The BIRT Report Designer interface<sup>6</sup>

<sup>6</sup> Showing an ODM report definition (data catalogue report) on the right hand side

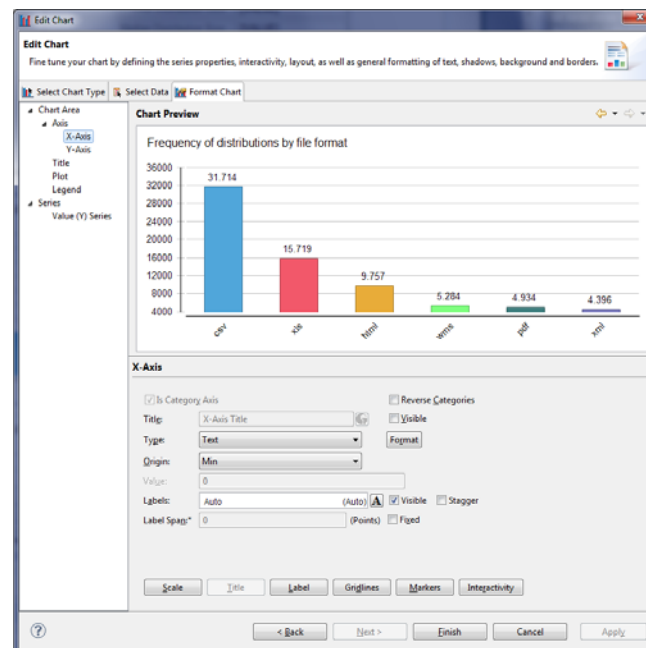


Figure 5: The BIRT Chart Designer interface

#### 4.2.2. Report Engine

After the report design has been created, it can directly be used to create a report from it. Therefore, the BIRT Report Engine is used, which processes the XML file, connects to the data sources, retrieves the data, populates the report elements, creates the charts, etc. Eventually, the report is rendered and exported to one of the various formats, which include PDF, Microsoft Word, Microsoft Excel, or HTML.

Within the ODM context, different strategies for the generation of the reports are relevant, which have their advantages and disadvantages. They include:

- **Ad-hoc report generation:** when a user wants to obtain a report via the web interface, the generation is triggered in an ad-hoc manner, meaning that it is generated whenever needed. The advantages here are that reports are only generated when necessary (reducing the overall system load) and the most recent data can be used, whereas on the other hand the disadvantages are the increased latency (the generation takes at least five seconds), inefficient load distribution (during peak hours a lot of reports will have to be generated) and redundancy (a report is likely to be generated multiple times).
- **Scheduled report generation:** reports are created automatically and periodically during a period of time where load is expected to be low, for instance during the night. The obvious advantages here are the disadvantages of above and vice versa.
- **Hybrid approach:** a combination of the previous approaches which tries to incorporate the advantages of both. In detail, this means that reports that are most likely accessed like European dashboard and the most important data catalogue dashboards are pre-generated at night and the ones that remain are generated on the fly. In order to reduce the load and latency, all reports generated ad-hoc during a specific period, like for instance one day, can be kept in memory.



Apart from the question of the time of generation, there is also the question whether a generated report shall be provided as download directly, meaning that the PDF file is sent to the user, or he/she is redirected to the **BIRT Report Viewer** where the report can be viewed online in HTML format and then manually exported to one of the supported formats. This Report Viewer comes as a web application that can be hosted within an Apache Tomcat webserver, whereas the Report Engine consists of a bundle of Java Archives.

### 4.3. Implementation Status

As already mentioned above, the reports that have been implemented so far are the European dashboard overview and the generic data catalogue report. In this section, a short summary about the elements and structure is presented.

It is to be noted that despite the commitment to make the reports look as close as possible as their web page counterparts, there are quite severe differences up to now. This is to a great extent also the result from the fact that so far the selection of the metrics and their placement on the page has not been defined. However, rearranging and adding/removing elements from a report can be done within a reasonably short amount of time.

#### 4.3.1. General Report Structure

The reports all have a rather similar structure. That is, there is a master page which contains the logo, the name of the report and page numbers to be printed to every page of a report except the title page. This title page as well contains the logo, name of the report and additionally a date of creation in order to identify and trace back the data that underlies the report.

Both reports make use of two data sources in the wider sense, that is the metric source and the actual content data source, ignoring the fact that they lie within the same database. Whereas the metric source delivers aggregate figures along various dimensions such as time, country, license, etc. which are used to populate charts, the actual content data source holds the metadata and is used for the tables of data catalogue and dataset metadata, respectively.

#### 4.3.2. European Dashboard Report

The European dashboard contains all information that is available so far on the over-the-aggregate metric-level (see deliverable D2.3 section 3.1) and a list of the data catalogues. Hence, the overall aggregate figures like the amount of data catalogue or age information are presented in a list, while all other metrics are represented as graphs.

Accordingly, the amount of new data catalogues (*catnewmonthfreq*) is presented as line chart with an additional accumulated indication. The amount of datasets per data catalogue (*catdatasetsfreq*) is presented as ordered bar chart, the amount of catalogues per geographic region as bar (*catgeofreq*) and pie chart (*catgeoprop*). The last chart is the proportion of catalogues using specific software platforms as a pie (*catsoftfreq*).

The last section of the report is a list of all data catalogues currently indexed, showing the most important metadata fields.



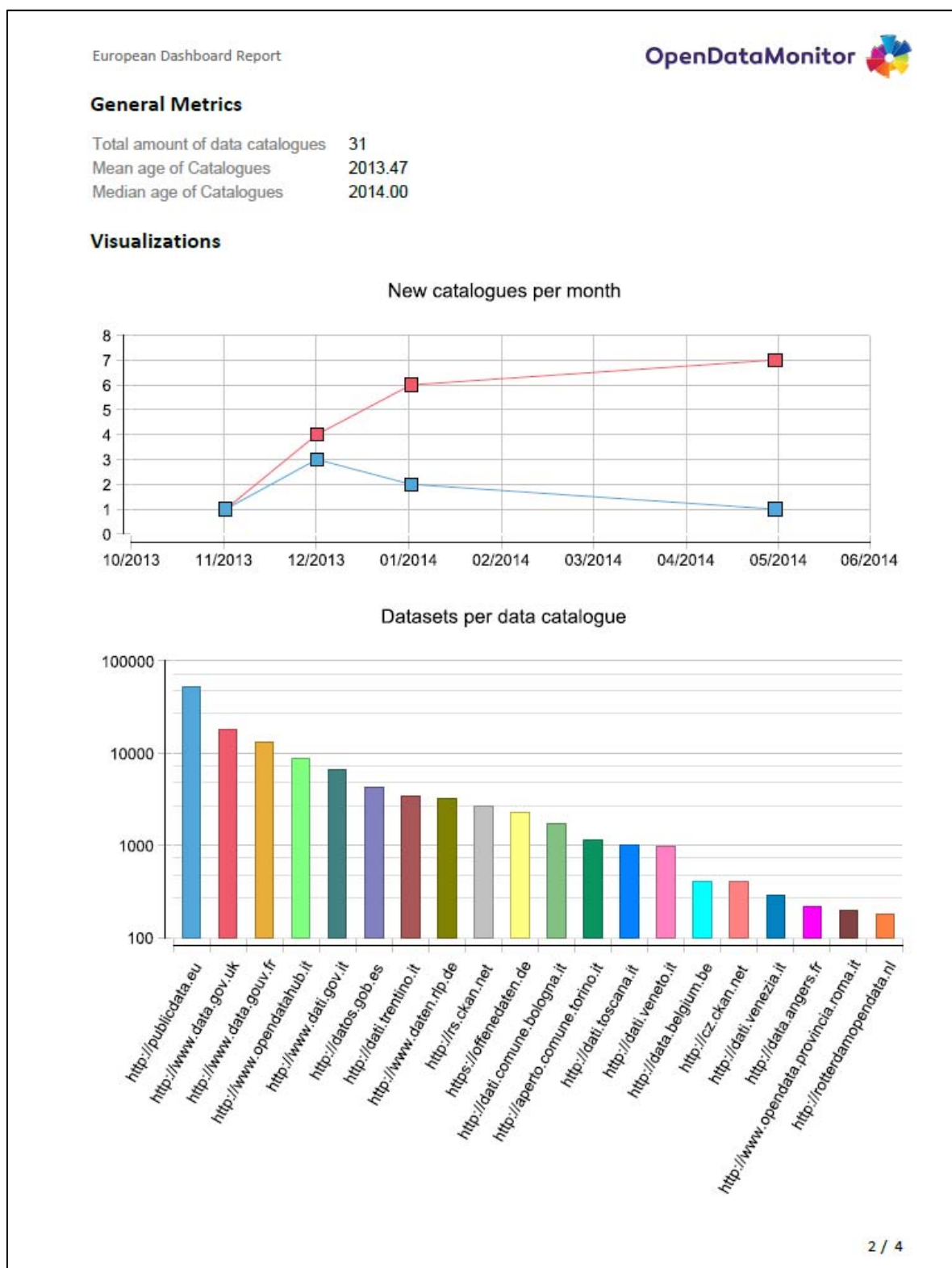


Figure 6: Page two of the European dashboard report<sup>7</sup>

<sup>7</sup> showing general aggregate figures and two charts

### 4.3.3. Data Catalogue Report

In general, the data catalogue report is rather similar to the European dashboard one. However, there is one major technical difference – the report has a parameter as input value. This parameter is used to pass to the report the data catalogue that the report is to be generated for. From a technical perspective, this parameter is passed on to all data source queries in order to obtain only data that is relevant. By doing so, the same report definition can be reused for all the data catalogues.

From the content perspective, an additional metadata section is put right to the beginning of the report. Subsequently, again single figures are put into a list, followed by five charts showing the following metrics: *catfileformatfreq*, *catmimetypefreq*, *catdsbylicenseprop*, *catmachineread-formatprop* and *catopenlicfreq*. Eventually, a table is appended that lists all the datasets of a catalogue, together with its URL and tags.

## 5. Graphic User Interface Status Report

The following section shows and describes the current status of the OpenDataMonitor user interface. While in most cases there is already real harvested and harmonized data shown some screens still hold visualisation examples which are already working but not connected with real data. Also the current filter opportunities will be adjusted according further insights. During the second project year all visualisations will be steadily tested and adapted to maximise the usability for the end-users.

### 5.1. Data Catalogues Dashboard

The Data Catalogues Dashboard holds several opportunities to learn more about the open data landscape. The filters on the left side of the screen will allow the user to learn more about data catalogues that are of relevance. The user can filter the aggregated data e.g. by country or by source type and choose between different visualisation opportunities. Also information on the growth of data catalogues over time will be presented. The force layout graph it is shown in the interface but not connected with real data, in order to inspire the data analysts of this project to combine the metrics defined so far for an efficient data interpretation.

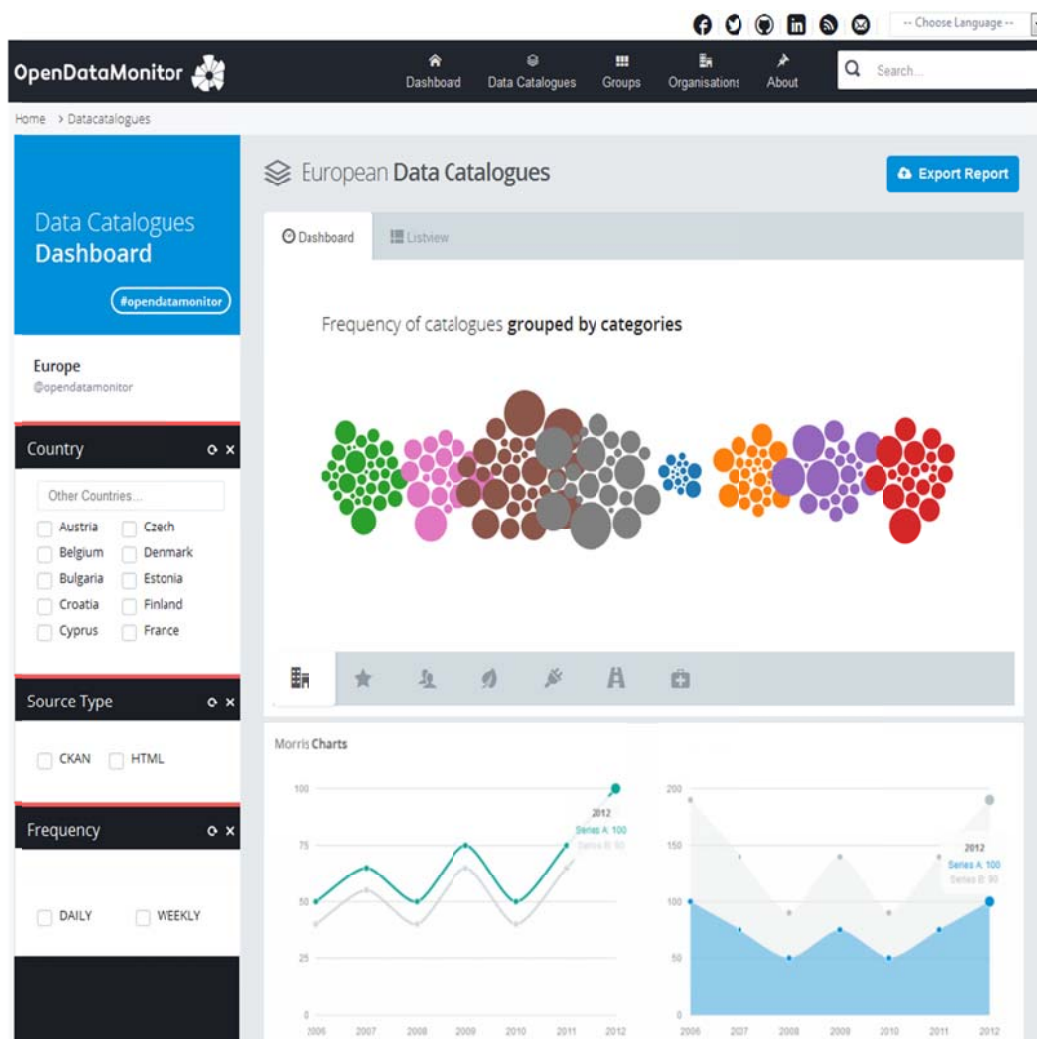
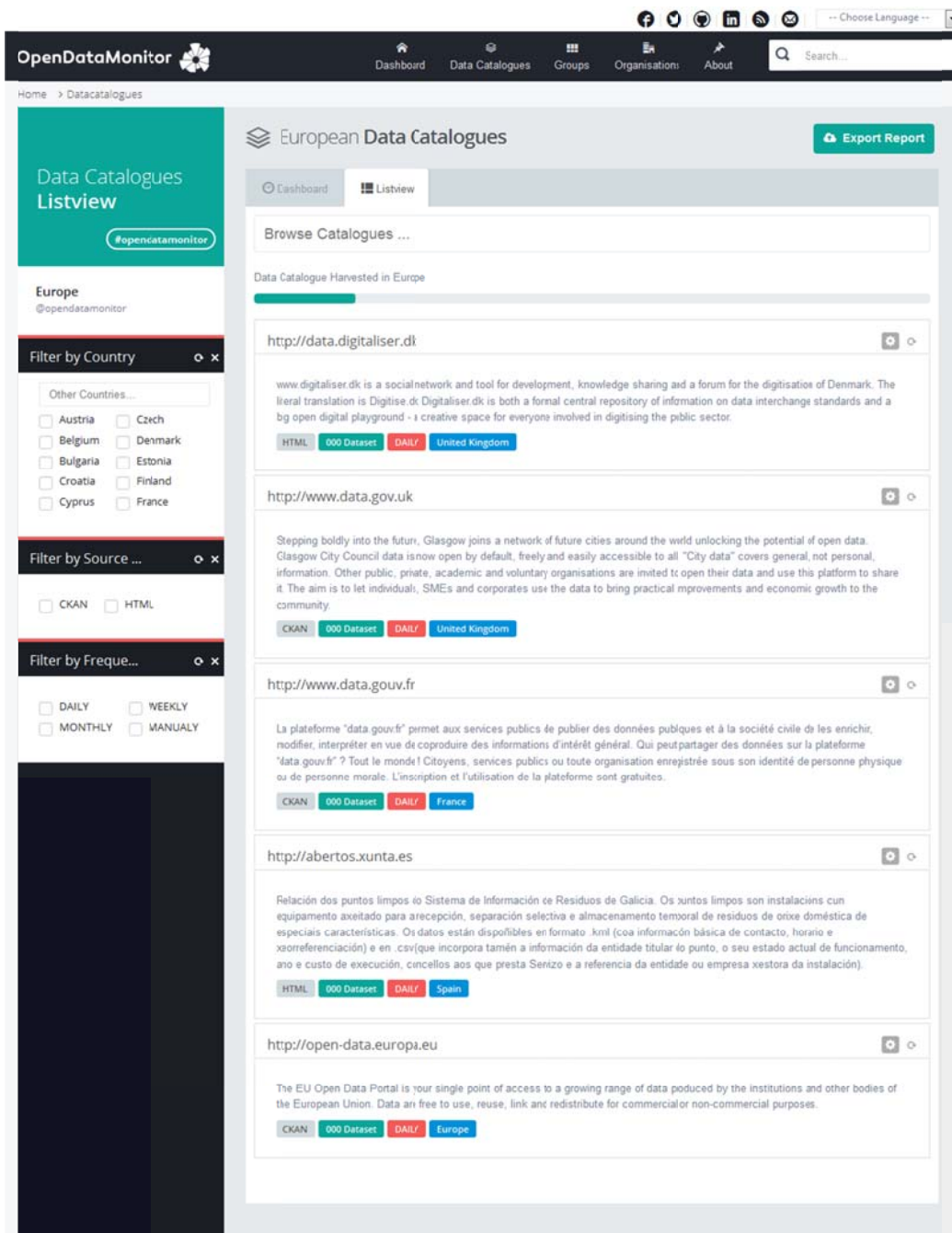


Figure 7: Data Catalogues Dashboard

## 5.2. Data Catalogues List View

The Data Catalogues list view allows the user to easily get an overview on existing data catalogues. On the left side there are several pre-defined filter options, such as country or source type that guarantee the user to quickly find the relevant data catalogues. For each of the found data catalogues a description, the source type (e.g. CKAN, HTML), the number of datasets, the update frequency and the location is provided. The data catalogues information is being retrieved from predefined APIs written from the backend of the application. Most of the fields of the data catalogues are being entered manually once in the first harvesting job. The fields entered manually from OpenDataMonitor and that don't guarantee further updates are: catalogue name, description.



The screenshot displays the 'Data Catalogues Listview' interface. On the left sidebar, there are three filter sections: 'Filter by Country' (listing countries like Austria, Czech, Belgium, Denmark, Bulgaria, Estonia, Croatia, Finland, Cyprus, France), 'Filter by Source' (listing CKAN, HTML), and 'Filter by Frequency' (listing DAILY, WEEKLY, MONTHLY, MANUALLY). The main content area is titled 'European Data Catalogues' and shows a list of catalogues harvested in Europe. Each catalogue entry includes its URL, a description, and a row of metadata: source type (CKAN or HTML), number of datasets (000 Dataset), update frequency (DAILY), and location (United Kingdom or France). The catalogues listed are:   
1. <http://data.digitaliser.dk>: Digitaliser.dk is a social network and tool for development, knowledge sharing and a forum for the digitisation of Denmark.   
2. <http://www.data.gov.uk>: Glasgow City Council data is now open by default, freely and easily accessible to all.   
3. <http://www.data.gouv.fr>: La plateforme "data.gouv.fr" permet aux services publics de publier des données publiques et à la société civile de les enrichir.   
4. <http://abertos.xunta.es>: Relación dos puntos limpos do Sistema de Información de Resíduos de Galicia.   
5. <http://open-data.europa.eu>: The EU Open Data Portal is your single point of access to a growing range of data produced by the institutions and other bodies of the European Union.

Figure 8: Data Catalogues List View

### 5.3. Data Catalogue Profile Dashboard

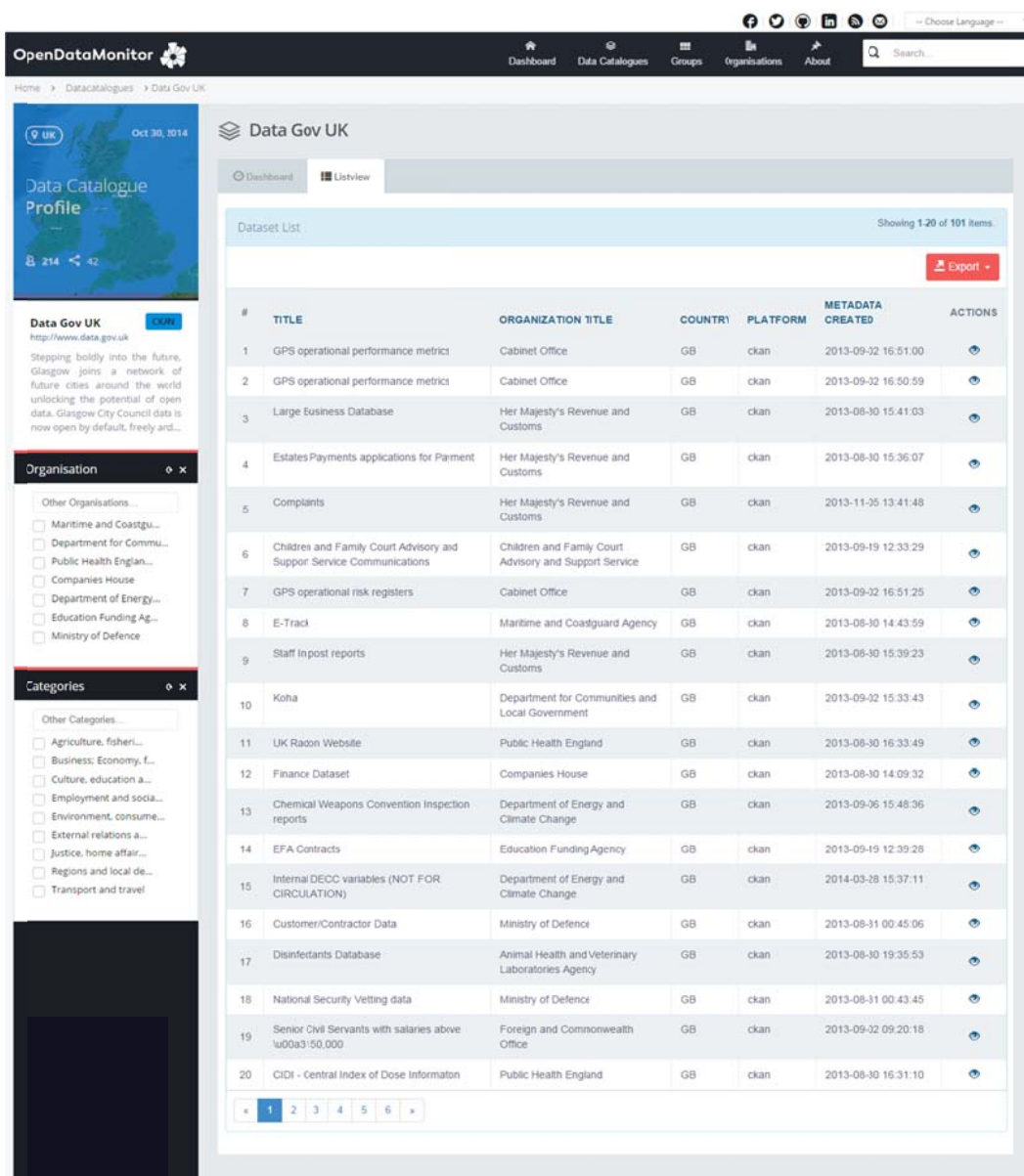
The Data Catalogues Profile has three different views. The first one is the Dashboard view which can be seen below. It holds several visualisations that provide a quick overview on the current status of the catalogue. This can be for instance the total number of provided datasets in the catalogue or how many new datasets are added over a period. In addition charts will be shown that group or compare categories. Using the filters on the left side of the screen will directly provide changes to the shown visualisations. Additional filters will be added in the next months of development.



Figure 9: Data Catalogues Profile Dashboard

## 5.4. Data Catalogue Profile List View

The second view of the Data Catalogue Profile is the List View. While the Dashboard provides an aggregated overview for a specific catalogue, the List View shows the available datasets in a structured table. For example the user can quickly identify which datasets are provided by which organisation and from which country. Also further data such as the platform type and the metadata timestamp can be shown. As some data catalogues provide a huge number of datasets the user will have the option to reduce the shown list by using the available filter options. This can be for instance organisations or topic categories. The table categories and filter options will be adapted during the second project year based on further insights and user tests.



The screenshot displays the OpenDataMonitor interface for the Data Gov UK profile. The top navigation bar includes links to Dashboard, Data Catalogues, Groups, Organisations, and About. The main content area is titled 'Data Gov UK' and shows a 'Dataset List' with 120 of 101 items. The table lists datasets with the following columns: #, TITLE, ORGANIZATION TITLE, COUNTRY, PLATFORM, METADATA CREATED, and ACTIONS. The table is filtered to show 20 items, with a pagination bar at the bottom indicating 1 to 6 items.

#	TITLE	ORGANIZATION TITLE	COUNTRY	PLATFORM	METADATA CREATED	ACTIONS
1	GPS operational performance metrics	Cabinet Office	GB	ckan	2013-09-02 16:51:00	
2	GPS operational performance metrics	Cabinet Office	GB	ckan	2013-09-02 16:50:59	
3	Large Business Database	Her Majesty's Revenue and Customs	GB	ckan	2013-08-30 15:41:03	
4	Estates Payments applications for Payment	Her Majesty's Revenue and Customs	GB	ckan	2013-08-30 15:36:07	
5	Complaints	Her Majesty's Revenue and Customs	GB	ckan	2013-11-05 13:41:48	
6	Children and Family Court Advisory and Support Service Communications	Children and Family Court Advisory and Support Service	GB	ckan	2013-09-19 12:33:29	
7	GPS operational risk registers	Cabinet Office	GB	ckan	2013-09-02 16:51:25	
8	E-Track	Maritime and Coastguard Agency	GB	ckan	2013-08-30 14:43:59	
9	Staff in post reports	Her Majesty's Revenue and Customs	GB	ckan	2013-08-30 15:39:23	
10	Koha	Department for Communities and Local Government	GB	ckan	2013-09-02 15:33:43	
11	UK Razon Website	Public Health England	GB	ckan	2013-08-30 16:33:49	
12	Finance Dataset	Companies House	GB	ckan	2013-08-30 14:09:32	
13	Chemical Weapons Convention Inspection reports	Department of Energy and Climate Change	GB	ckan	2013-09-06 15:48:36	
14	EFA Contracts	Education Funding Agency	GB	ckan	2013-09-19 12:39:28	
15	Internal DECC variables (NOT FOR CIRCULATION)	Department of Energy and Climate Change	GB	ckan	2014-03-28 15:37:11	
16	Customer/Contractor Data	Ministry of Defence	GB	ckan	2013-08-31 00:45:06	
17	Disinfectants Database	Animal Health and Veterinary Laboratories Agency	GB	ckan	2013-08-30 19:35:53	
18	National Security Vetting data	Ministry of Defence	GB	ckan	2013-08-31 00:43:45	
19	Senior Civil Servants with salaries above £0003:50,000	Foreign and Commonwealth Office	GB	ckan	2013-09-02 09:20:18	
20	CIDI - Central Index of Dose Information	Public Health England	GB	ckan	2013-08-30 16:31:10	

Figure 10: Data Catalogues Profile List View



## 5.5. Dataset Profile

When an interesting dataset was identified in the Data Catalogues List View that was described above the user only has to click on it to see the Dataset Profile. The page holds all gathered data on the dataset and will reveal which metadata is existent and where are gaps that have to be filled.

The goal of this project is to not be used a main tool for finding datasets and their resources, therefore the interface contains the link that will forward the end users in order for him to find further information about the selected dataset.

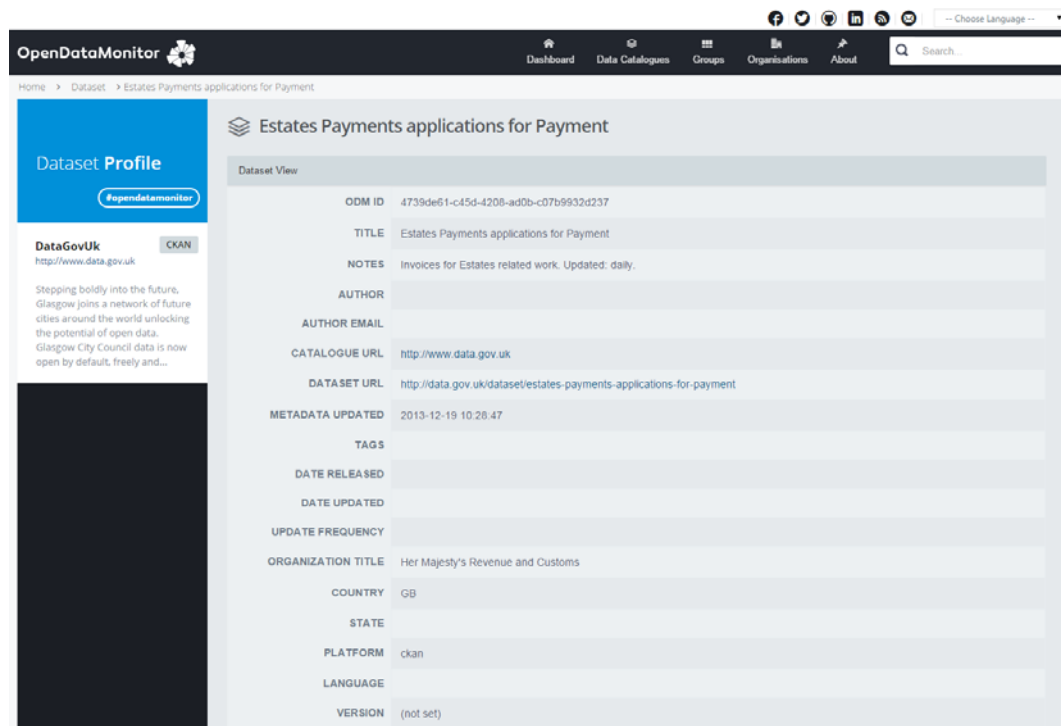


Figure 11: Dataset Profile

## 5.6. Groups

In only 31 data catalogues are being encountered 672 distinct categories. This categories pool of data will result into a complex information visualization with hard decision making and reuse of the data interpretation. Without the intention of creating a new vocabulary or mapping schema in the Open Data landscape, this project will also group the categories to show more concrete information. The mapping schema is using the policies defined from the European Commission<sup>8</sup> : (1) Agriculture, fisheries and food; Energy and natural resources, (2) Business; Economy, finance and tax, (3) Culture, education and youth; Science and technology, (4) Employment and social rights/affairs, (5) Environment, consumers and health, (6) External relations and foreign affairs, (7) Justice, home affairs and citizens' rights, (8) Regions and local development, (9) Transport and travel.

Except for the below gallery view interface there will be included other ones, like group profile and group dashboard. Additionally the group mapping will be integrated in other related interfaces with the role as filters.

<sup>8</sup> [http://ec.europa.eu/policies/index\\_en.htm](http://ec.europa.eu/policies/index_en.htm)

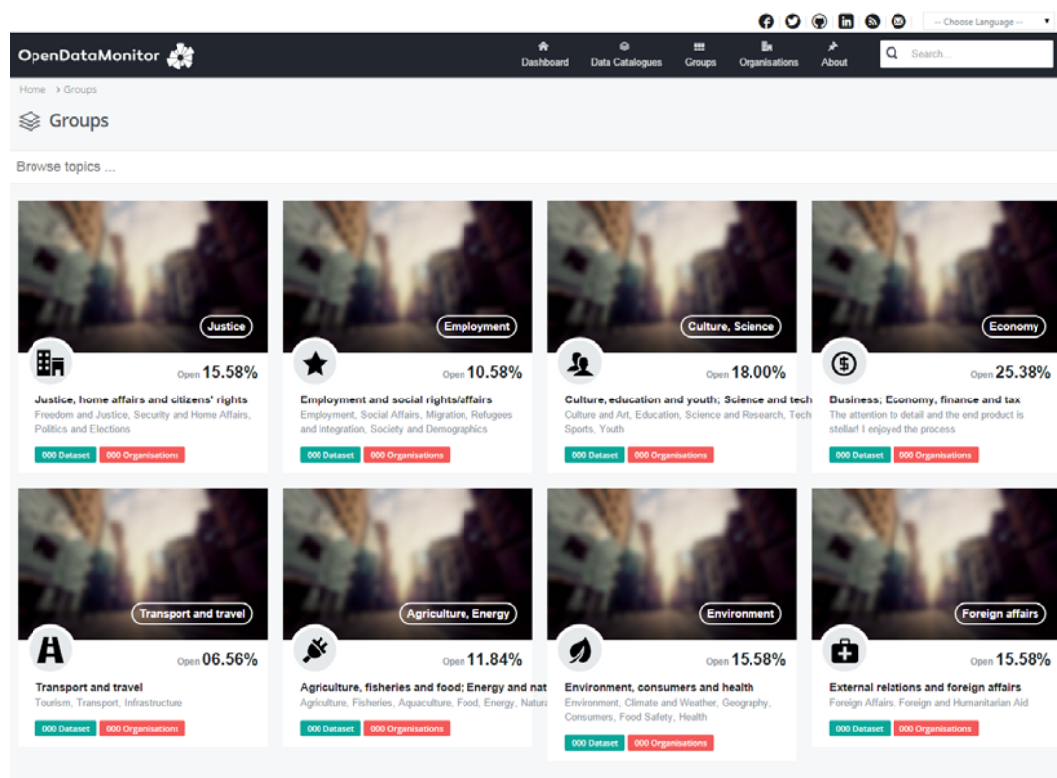


Figure 12: Groups

## 5.7. Internationalization (i18n)

The ODM platform will feature multilingual interfaces in order to address the language barriers which are present within the target audience of the project. Due to this approach it is possible to locate the different end users and automatically provide the content of the platform in the desired language. In addition users are able to manually choose the language in which the platform should be displayed. Figure 13 illustrates the implemented language selector. At the moment the following languages are supported which represent the languages spoken within the ODM consortium: English, German, Greek and Spanish.

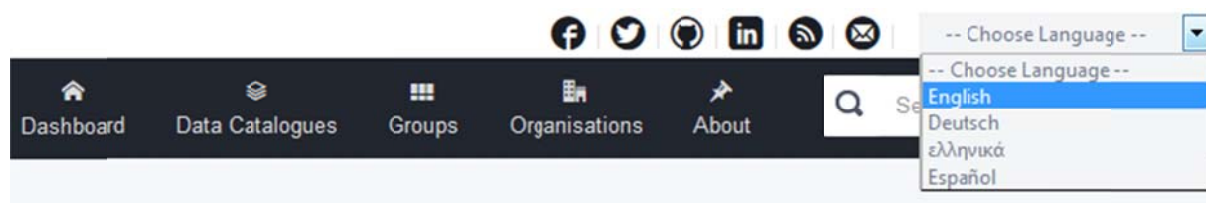


Figure 13: Language selector of the ODM platform

Basically it is necessary to differentiate between static and dynamic content when dealing with the translation of a website. In the next sections both types of content will be investigated in more detail.

- **Static Content**

Content which is directly published to regular files on the webserver can be referred to as static content. This content does not change in case somebody views the page at different



times. The labels below the icons in Figure 13 are a good example for such content. Yii2 provides two different ways to handle the translation of static content:

- Message Translation

This type of translation is based on the `\Yii::t()`<sup>9</sup> function of the framework. In essence this function takes two parameters whereas the first parameter defines the message category and the second one the actual message. The following code snippet shows a simple example of the usage of this function. It outputs the message “I am a message” in the desired target language.

```
echo \Yii::t('app', 'I am a message!');
```

- View Translation

Another way to translate static content is to directly provide translations of certain files. This way no specific function of the framework must be used. Instead the files are stored directly within a subfolder which references the language the files are translated in. For example with the following folder structure the framework will use the German translation of the file “index.php” in case the target language is defined as German.

```
views/site/de-DE/index.php
```

- **Dynamic Content**

In contrast to static content, dynamic content is created with each request of the specific web page. This means that the same view can contain different content for different users. Such dynamic content therefore is not stored within regular view files, but it is retrieved directly from the database.

There are different possibilities to store translated content within a database. For the ODM platform we created an extra table which contains all translated data. Figure 14 displays the schema of this table.

#	Name	Typ	Kollation	Attribute	Null	Standard	Extra
1	<u>id</u>	int(11)			Nein	kein(e)	AUTO_INCREMENT
2	table_name	varchar(100)	utf8_general_ci		Nein	kein(e)	
3	row_id	int(11)			Nein	kein(e)	
4	field_name	varchar(100)	utf8_general_ci		Nein	kein(e)	
5	display_value	text	utf8_general_ci		Nein	kein(e)	
6	lang	varchar(5)	utf8_general_ci		Nein	kein(e)	

Figure 14: Table schema of the i18n table

<sup>9</sup> <http://www.yiiframework.com/doc-2.0/yii-baseyii.html#t%28%29-detail>

A possible entry of this table is displayed in Figure 15. In this case the record with the unique “row\_id” 47 from the table “metric\_info” is selected. In addition the name (e.g. label) of the target column respectively field is stored in the table. The content of the translation is saved in the “display\_value” column. The language of the translation is stated in the column “lang”. Due to this approach the translation can be matched exactly to the desired content.

id	table_name	row_id	field_name	display_value	lang
1	metric_info	47	label	Gesamte Anzahl an Katalogen	de_DE

Figure 15: Possible entry in i18n table

When dealing with translations it is important to focus on the formatting of the data in different languages in order to provide the correct presentation of the data. Perfect examples for this issue are dates. Date and time is displayed differently when changing the target language of the content. The message translation capabilities of Yii help to overcome this problem. In detail Yii provides a helper class to address this issue. This class is registered as an application component and can therefore be called in a more convenient way. The following listing taken from the official Yii documentation<sup>10</sup> shows the usage of the date formatter in combination with the translation of the given date.

```
Yii::$app->formatter->locale = 'en-US';
echo Yii::$app->formatter->asDate('2014-01-01'); // output: January 1, 2014
Yii::$app->formatter->locale = 'de-DE';
echo Yii::$app->formatter->asDate('2014-01-01'); // output: 1. Januar 2014
Yii::$app->formatter->locale = 'ru-RU';
echo Yii::$app->formatter->asDate('2014-01-01'); // output: 1 января 2014 г.
```

<sup>10</sup> <http://www.yiiframework.com/doc-2.0/guide-output-formatter.html>

## 6. Know-How

The OpenDataMonitor project does not only offer the community to make use of the developed parts and the Demonstration Site but also wants to share important insights that were made during particular project steps. Therefore a Knowledge Base was established that is constantly updated with new insights. To make the access to this knowledge as easy as possible for the community the Knowledge Base does not only provide a link to the public deliverables that were created but breaks them down into smaller parts and articles. This chapter will give a brief overview on the main concept and features of the Knowledge Base that is directly connected with the project website.

### 6.1. Home

The start page is designed very simple and focuses on the three main parts of the site (Knowledge Base, Forum and FAQ). In addition basic visualisations will be shown that refer to current insights provided by the OpenDataMonitor Demonstration Site. This can be basic metrics or analysis visualisation (The shown visualisations in the screenshot indicates an example). At the right bottom of the page further links to relevant social media channels (e.g. Facebook, Twitter etc.) are shown.

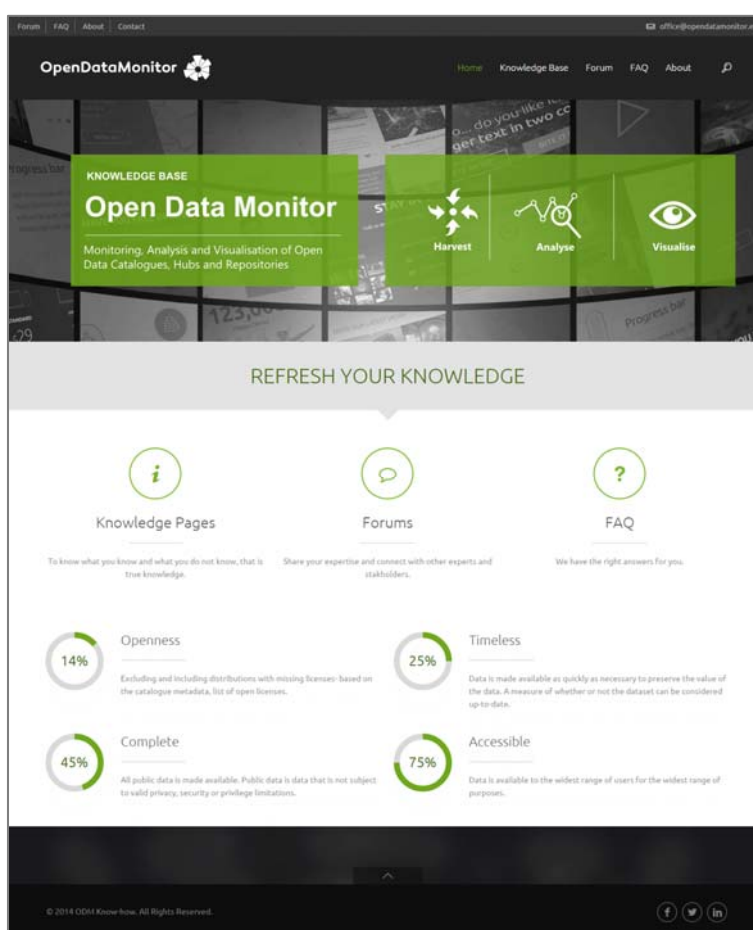


Figure 16: Knowledge Base Home

## 6.2. Knowledge Base

The Knowledge Base holds a broad range of relevant insights that were made during the ODM project. Each piece of information is shown as an article. The main site holds an overview of available articles indicating the author and date of creation. Each article is allocated to a category (e.g. Open Data Landscape, Metadata Processing, Reporting Tools etc.). If a category is selected only the connected articles are displayed. On the right side of the screen a search bar and the number of articles in each category is shown. In addition a list indicates recent articles that were posted.

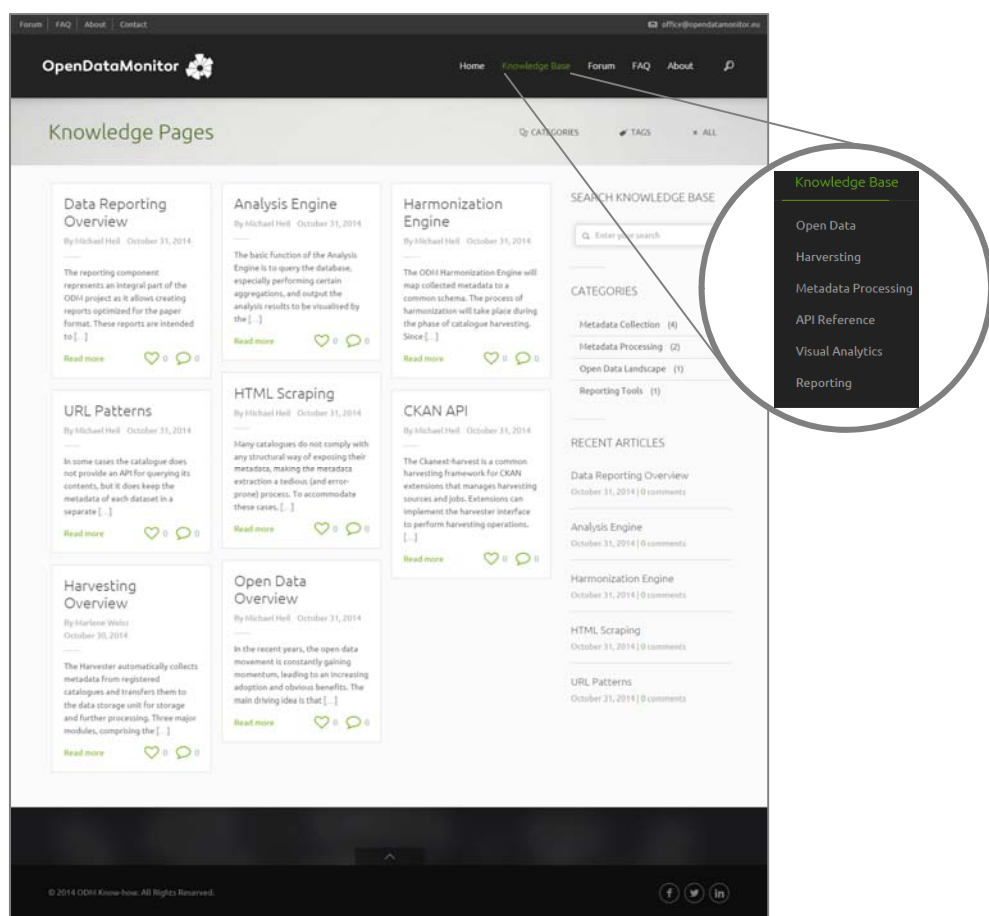
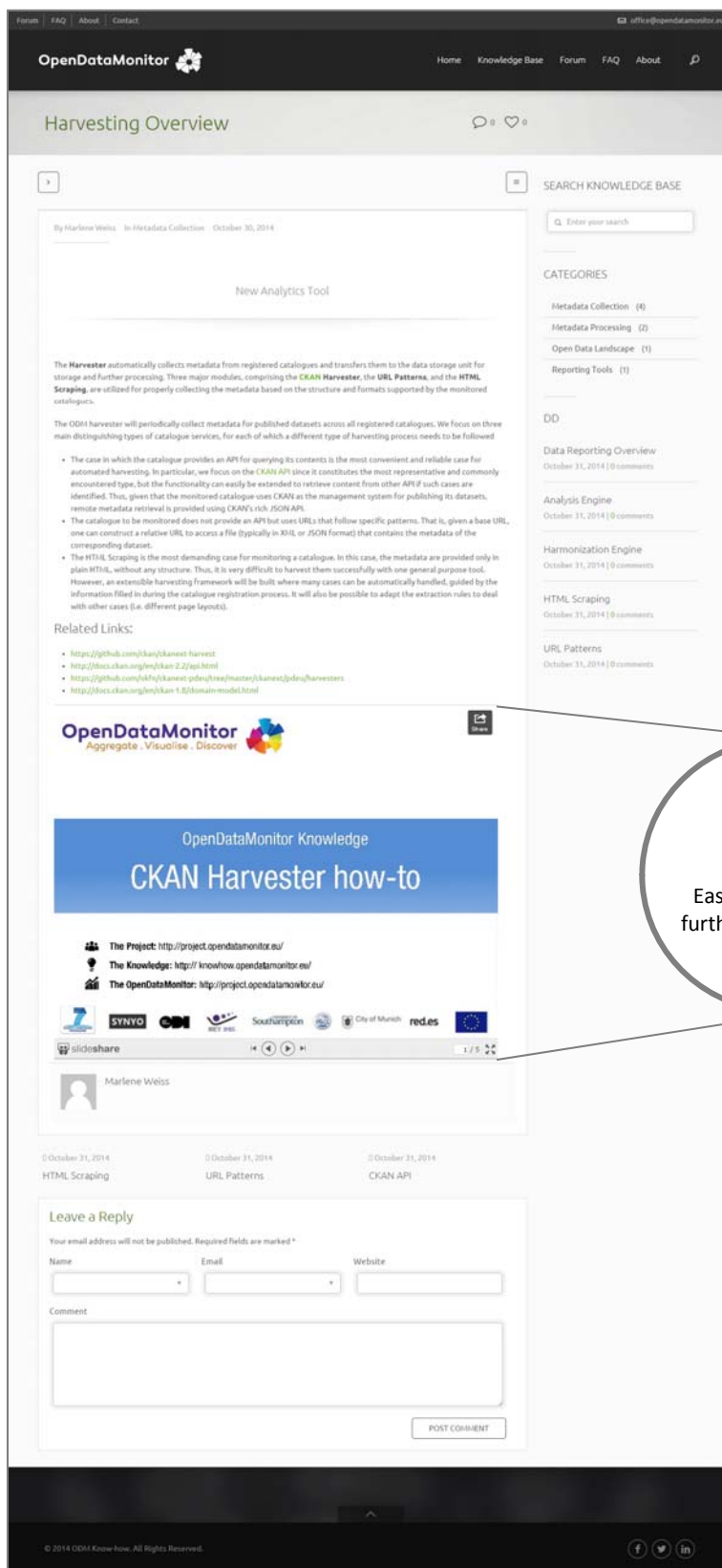



Figure 17: Knowledge Base

Each article can be viewed in detail and will allow the author to easily include visualisations and presentations (e.g. Slideshare) to support the open data community with relevant knowledge.



OpenDataMonitor 

Home Knowledge Base Forum FAQ About

## Harvesting Overview

By Marlene Weiss In Metadata Collection October 30, 2014

### New Analytics Tool


The **Harvester** automatically collects metadata from registered catalogues and transfers them to the data storage unit for storage and further processing. Three major modules, comprising the **CKAN Harvester**, the **URL Patterns**, and the **HTML Scrapping**, are utilized for properly collecting the metadata based on the structure and formats supported by the monitored catalogues.

The ODH harvester will periodically collect metadata for published datasets across all registered catalogues. We focus on three main distinguishing types of catalogue services, for each of which a different type of harvesting process needs to be followed:

- The case in which the catalogue provides an API for querying its contents is the most convenient and reliable case for automated harvesting. In particular, we focus on the CKAN API since it constitutes the most representative and commonly encountered type, but the functionality can easily be extended to retrieve content from other API if such cases are identified. Thus, given that the monitored catalogue uses CKAN as the management system for publishing its datasets, remote metadata retrieval is provided using CKAN's rich JSON API.
- The catalogue to be monitored does not provide an API but uses URLs that follow specific patterns. That is, given a base URL, one can construct a relative URL to access a file (typically in XML or JSON format) that contains the metadata of the corresponding dataset.
- The HTML Scrapping is the most demanding case for monitoring a catalogue. In this case, the metadata are provided only in plain HTML, without any structure. Thus, it is very difficult to harvest them successfully with one general purpose tool. However, an extendible harvesting framework will be built, where many cases can be automatically handled, guided by the information filled in during the catalogue registration process. It will also be possible to adapt the extraction rules to deal with other cases (i.e. different page layouts).

Related Links:


- <https://github.com/ckan/ckanext-harvest>
- <http://docs.ckan.org/en/ckan-2.2/api.html>
- <https://github.com/ckan/ckanext-pdpyftree/master/ckanext/pdpyftree/harvesters>
- <http://docs.ckan.org/en/ckan-1.8/domain-model.html>

OpenDataMonitor   
Aggregate · Visualise · Discover

### OpenDataMonitor Knowledge

## CKAN Harvester how-to

The Project: <http://project.opendatamonitor.eu/>  
The Knowledge: <http://knowhow.opendatamonitor.eu/>  
The OpenDataMonitor: <http://project.opendatamonitor.eu/>



slideshare 1 / 5

Marlene Weiss

October 31, 2014 HTML Scrapping    October 31, 2014 URL Patterns    October 31, 2014 CKAN API

Leave a Reply

Your email address will not be published. Required fields are marked \*

Name  Email  Website

Comment

POST COMMENT


© 2014 ODM Know-how. All Rights Reserved. 

Figure 18: Article View

### 6.3. Forum

An integrated forum will give every user the opportunity to get in contact with the OpenDataMonitor project team and the whole open data community to ask specific questions. The forum holds typical features such as posting new questions, sending replies and mark important questions. On the right side of the screen the forum categories are shown as well as the number of posted articles. Below the user will find the latest topics as well as a list of topics that are marked as most popular. The forum can be easily filtered by the existing categories.

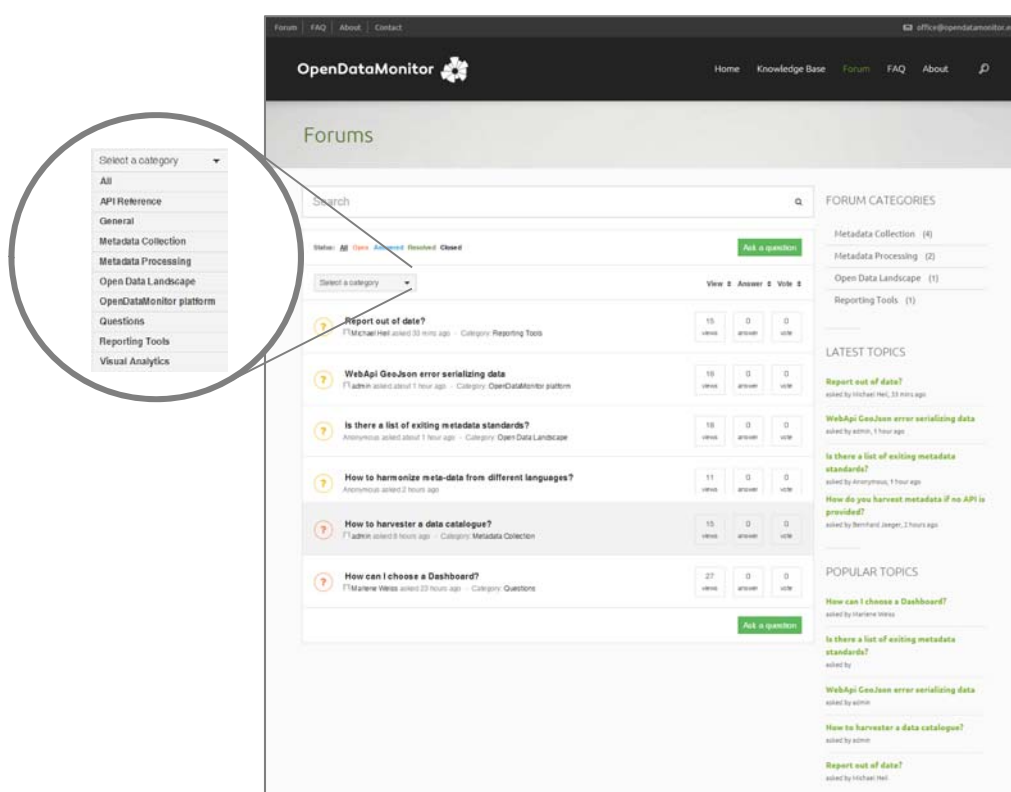


Figure 19: Forum

## 6.4. FAQ

Important questions and such that are often directed to the OpenDataMonitor consortium will be answered in detail in the Frequently Asked Questions section. This will allow the user to quickly get an answer to common questions without searching or posting in the forum.

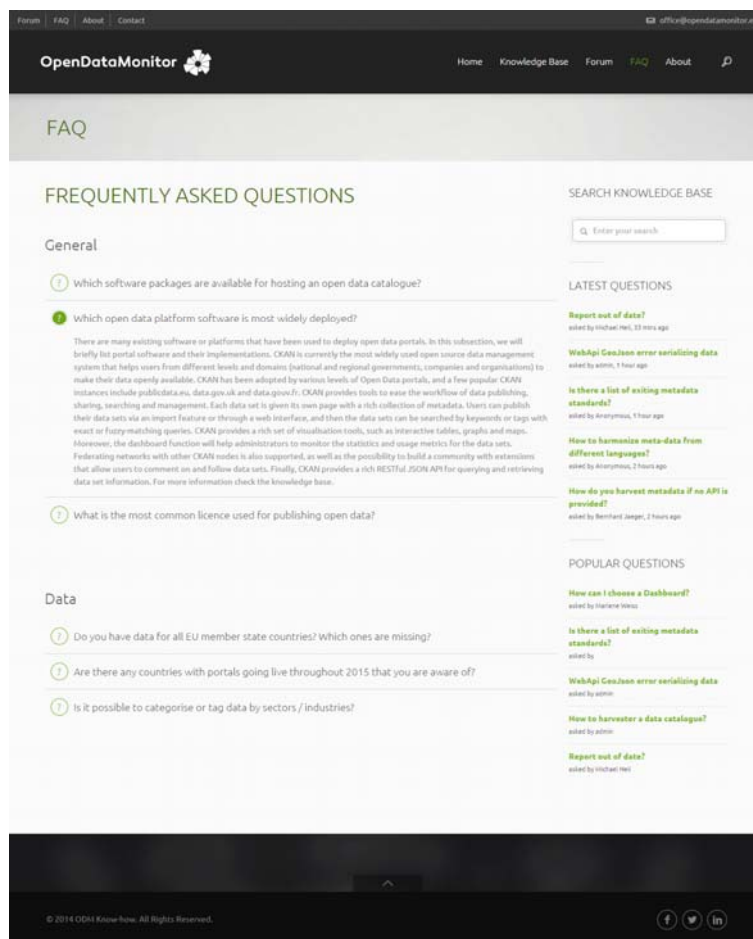


Figure 20: FAQ

The Knowledge Base will be made accessible to the community during the next project months and will be constantly updated and extended based on further knowledge that is created in the consortium. In addition feedback from users of the Knowledge Base can be used to improve the project outcomes and better support the open data community. Providing valuable knowledge and resources will also mean that the OpenDataMonitor is recognised by a higher number of relevant stakeholders which will massively support the dissemination activities of the second project year.



## 7. Conclusion

This deliverable gives a status report on the presentation layer of OpenDataMonitor. The layout of the interfaces developed have a consistency with the designed mockups delivered in D3.2, and the platform is built based on the decisions stated in the early steps of the project. The first nine months of development have been running in parallel with requirements analysis and concept design, using an agile methodology. Mockups, sketches, questionnaires' and open talks are being used to gather all the requirements while applying "Design thinking" process. Design thinking puts the stakeholder in the position of a developer and helps the last one to make the right decisions during the software lifecycle. Not all the interfaces are being programmed completely. The most mature ones are being inserted in this report. The development phase will not stop until month 23<sup>rd</sup> of the project, while addressing all initial requirements, stakeholders' enhancements and community feedback. To open the channels to all the existing and future stakeholders and broad public, the technical team developed an additional product for this project. The additional product represents a simple Knowledge Base with the integration of a Forum module.

Except for the status report described in this report, it's relevant at this stage to mention some crucial tasks that will follow in the second project year.

- Improve and further develop the application backend with administration purposes for different user levels.
- Develop UI2. Country Dashboard: Get an overview about the open data situation within one country.
- Develop UI3. Comparison Dashboard: Compare the open data situation between different entities
- Develop Organisation interface and functionalities and improve groups GUIs.
- Improve the GUI and functionalities of European Data Catalogues dashboard and list view.
- Improve the GUI and functionalities of Individual Data Catalogue Profile dashboard and list view.
- Select the most relevant metrics to be visualised, combine them in a unique graph in order to create the Value added of this project toward information visualisation.
- Improve the general interface aspects like responsible, logic and graphical consistency, usability.
- Create a user registered space with the ability to create specific reports and to offer emailing service for the subscribed users.
- Improve and keep up to date the Knowledge Base while sharing the responsibility for information sharing to the project stakeholders and the broad community.
- House-keeping, performance testing, share on Github, plan for future support of the platform.



## Appendix I. Glossary of acronyms and abbreviations

API	Application Program Interface
BSD	Berkeley Software Distribution
CC	Creative Commons
CKAN	Comprehensive Knowledge Archive Network
CPU	Central Processing Unit
CRUD	Create Read Update Delete
CSS	Cascading Style Sheet
DB	DataBase
GB	Giga Bytes
GPL	General Public License
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
ISO	International Standards Organization
JSON	JavaScript Object Notation
LESS	Leaner CSS
MIT	Massachusetts Institute of Technology license
MVC	Model View Controller
ODM	Open Data Monitor
RAM	Random-Access Memory
SASS	Syntactically Awesome Stylesheets
SOAP	Simple Object Access Protocol
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
UC	Use Case
WSDL	Web Service Definition Language
XD	Experience Design
XML	Extensible Markup Language

## Appendix II. Terminology definition

API (source: <a href="http://howto.gov/api">howto.gov/api</a> )	An Application Programming Interface, or API, is a set of software instructions and standards that allows machine to machine communication-like when a website uses a widget to share a link on Twitter or Facebook.
CKAN (source: <a href="http://ckan.org">ckan.org</a> )	CKAN stands for Comprehensive Knowledge Archive Network, an open source data management system that is the basis of the Open Data Monitor platform and many data hubs around the world.
Cronjob (source: <a href="http://www.howtoforge.com">www.howtoforge.com</a> )	A cron job is a scheduled task that is executed by the system at a specified time/date. 1 crontab. The command to create/edit, list, and remove cron jobs is crontab. If you call it with the -u option, it specifies the name of the user whose crontab is to be tweaked.
CSS (source: <a href="http://en.wikipedia.org">http://en.wikipedia.org</a> )	Cascading Style Sheets (CSS) is a style sheet language used for describing the look and formatting of a document written in a markup language.
Dashboard [5]	An easy to read, often single page, real-time user interface, showing a graphical presentation of the current status (snapshot) and historical trends of an organization's key performance indicators to enable instantaneous and informed decisions to be made at a glance.
Dataset (source: D2.1)	<p>A conceptual entity that “represents a collection of data, published or curated by a single agent, and available for access or download in one or more formats”<sup>14</sup></p> <p>Package (in CKAN) . A data set is usually hosted in an open data repository and can belong to one or more groups.</p>
GET (source: <a href="http://www.qld.gov.au/">http://www.qld.gov.au/</a> )	A HTTP request method. GET is neither an acronym nor an initialise, but is capitalised out of convention. The GET method should be used when requesting a resource.
HTML (source: <a href="http://en.wikipedia.org">http://en.wikipedia.org</a> )	HTML or HyperText Markup Language is the standard markup language used to create web pages.
Kanban (source: <a href="http://en.wikipedia.org/wiki/Kanban_(development)">http://en.wikipedia.org/wiki/Kanban_(development)</a> )	Kanban is a method for managing knowledge work with an emphasis on just-in-time delivery while not overloading the team members. In this approach, the process, from definition of a task to its delivery to the customer, is displayed for participants to see and team members pull work from a queue.
LESS (source: <a href="http://lesscss.org">http://lesscss.org</a> )	Less is a CSS pre-processor, meaning that it extends the CSS language, adding features that allow variables, mixins, functions and many other techniques that allow you to make CSS that is more maintainable, themable and extendable.

<p>Metadata (source: <i>Federal Enterprise Architecture: Data Reference Model</i>)</p>	<p>Metadata describes a number of characteristics or attributes of data; that is, “data that describes data”. (ISO 11179-3). For any particular datum, the metadata may describe how the datum is represented, ranges of acceptable values, its label, and its relationship to other data. Metadata also may provide other relevant information, such as the responsible steward, associated laws and regulations, and the access management policy. The metadata for structured data objects describes the structure, data elements, interrelationships, and other characteristics of information, including its creation, disposition, access and handling controls, formats, content, and context, as well as related audit trails.</p>
<p>MVC (source: <a href="http://en.wikipedia.org">http://en.wikipedia.org</a>)</p>	<p>Model–view–controller (MVC) is a software architectural pattern for implementing user interfaces. It divides a given software application into three interconnected parts, so as to separate internal representations of information from the ways that information is presented to or accepted from the user.</p>
<p>NoSQL (source: <a href="http://en.wikipedia.org">http://en.wikipedia.org</a>)</p>	<p>A NoSQL or Not Only SQL database provides a mechanism for storage and retrieval of data that is modelled in means other than the tabular relations used in relational databases. Motivations for this approach include simplicity of design, horizontal scaling and finer control over availability. The data structure (e.g. key-value, graph, or document) differs from the RDBMS, and therefore some operations are faster in NoSQL and some in RDBMS.</p>
<p>Open Data Catalogue (source: D2.1)</p>	<p>A curated collection of metadata about data sets. Compared with “open data repository”, “open data catalogue” focuses on the organisation of data sets, while “open data repository” refers to the actual data storage. The catalogue would typically be agnostic regarding where the data itself is located: (1) it may all be published on the same Web server as the catalogue, i.e. the catalogue contains a data repository, or (2) may be distributed across the Web, with the catalogue simply pointing to those remote locations, in which case the catalogue is also referred to as a “data aggregator” or “data indexer”.</p>
<p>POST (source: <a href="http://www.qld.gov.au/">http://www.qld.gov.au/</a>)</p>	<p>A HTTP request method. POST is neither an acronym nor an initialise, but is capitalised out of convention. The POST method should be used when sending information to a server or interacting with a web application to change its state.</p>
<p>SASS (source: <a href="http://sass-lang.com">http://sass-lang.com</a>)</p>	<p>Sass is an extension of CSS that adds power and elegance to the basic language. It allows you to use variables, nested rules, mixins, inline imports, and more, all with a fully CSS-compatible syntax. Sass helps keep large stylesheets well-organized, and get small stylesheets up and running quickly, particularly with the help of the Compass style library.</p>

Scrum (source: <a href="http://en.wikipedia.org">http://en.wikipedia.org</a> )	Scrum is an iterative and incremental agile software development framework for managing product development. It defines "a flexible, holistic product development strategy where a development team works as a unit to reach a common goal", challenges assumptions of the "traditional, sequential approach" to product development, and enables teams to self-organize by encouraging physical co-location or close online collaboration of all team members, as well as daily face-to-face communication among all team members and disciplines in the project.
URL	The uniform resource locator, abbreviated as URL, is a specific character string that constitutes a reference to a resource. In most web browsers, the URL of a web page is displayed on top inside an address bar.
XML (source: Wikipedia)	XML (Extensible Markup Language) is a general-purpose specification for creating custom markup languages. It is classified as an extensible language, because it allows the user to define the mark-up elements. XML's purpose is to aid information systems in sharing structured data especially via the Internet, to encode documents, and to serialize data.

### Metadata

(source: Project Open Data <http://project-open-data.github.io>)

Title	Human-readable name of the asset. Should be in plain English and include sufficient detail to facilitate search and discovery.
Description	Human-readable description (e.g., an abstract) with sufficient detail to enable a user to quickly understand whether the asset is of interest.
Tags	Tags (or keywords) help users discover your dataset and should include terms that would be used by technical and non-technical users.
Last Update	Most recent date on which the dataset was changed, updated, or modified.
Publisher	The publishing agency.
Contact Name	Contact person's name for the asset.
Contact Email	Contact person's email address.
Unique Identifier	A unique identifier for the dataset or API as maintained within an Agency catalogue or database.
Public Access Level	The degree to which this dataset could be made publicly available, regardless of whether it has been made available. Choices: Public (is or could be made publicly available), Restricted (available under certain conditions), or Private (never able to be made publicly available).

Data Dictionary	URL to the data dictionary for the dataset or API. Note that documentation other than a data dictionary can be referenced using “related documents” as shown in the expanded fields.
Download URL	URL providing direct access to the downloadable distribution of a dataset.
End Point	Endpoint of the web service to access a dataset.
Format	The file format or API type of the distribution.
License	The license with which the dataset or API is published.
Spatial	The range of spatial applicability of the dataset, which could include a spatial region like a bounding box or a named place.
Temporal	The range of temporal applicability of the dataset (i.e., a start and end date of applicability for the data).
Release Date	Date of formal issuance.
Frequency	Frequency with which the dataset is published.
Language	The language of the dataset.
Granularity	Level of granularity of the dataset.
Data Quality	Whether the dataset meets the agency’s Information Quality Guidelines.
Category	Main thematic category of the dataset.
Related Documents	Related documents, such as technical information about a dataset or developer documentation.
Size	The size of the downloadable dataset.
Homepage URL	An alternative landing page used to redirect a user to a contextual, Agency-hosted “homepage” for the Dataset or API when selecting this resource from the Data.gov user interface.
RSS Feed	URL for an RSS feed that provides access to the dataset.
System of Records	URL to the system of record related to this dataset.