DIGIT
Unit B1

FOSSA WP4 - Deliverable n. 6

The inventory of OSS in use in the European Institutions

Date: 15/11/2016
Doc. Version: Final

TABLE OF CONTENTS

TABLE OF FIGURES

TABLE OF TABLES

1. DELIVERABLE OVERVIEW

The present deliverable contains both the description of the process applied to define the inventory of open source software in use at and – as annexes – the extractions from the pertinent database through several Pentaho dashboards, showing the main data of interest as agreed with DIGIT.

The scope of the inventory has been determined by the availability of data sources identified in the initial phase of the project, and therefore limited to the European Commission. However, as already stated in WP3 DLV1, the inventory process remains applicable to the European Parliament, provided that input data are supplied in the proper format.

This study describes (Section 2) the various steps through which the inventory has been performed, from the treatment of the input data to the detailed analysis of software information, including the software rating (both under a business criticality perspective, linked to the use done of the inventoried software in the European Commission, and under a sustainability perspective, determined according to the metrics developed by WP1 in its Deliverable 6) and the analysis of dependencies.

The following chapter (Section 3) describes in particular how the Pentaho dashboards, by which the inventory data are navigated through and analysed, have been created and can be modified and integrated. Furthermore, it lists the reports created via such dashboards to display the inventory results.

Section 4 briefly highlights how the inventory process applied in the FOSSA pilot project can be repeated in future exercises, identifying the different entry point of the process in such cases and connecting it to the process described in Section 2 above.

Finally, Section 5 analyses the inventory results providing an overview, elaborated on the raw inventory data, of the weight, distribution and typology of the inventoried OSS. Furthermore, this section analyses the OSS from the point of view of its sustainability, with particular focus on the critical software shortlist defined in Section 2.

2. SOFTWARE INVENTORY PROCESS

2.1. Treatment of input data

The input for the inventory process have been, as per WP3 DLV1[1], the exports coming from the various sub-inventory systems, provided in CSV format by the respective system owners in the initial source collection phase, in January 2016.

The inputs from Satellite, LANDesk and the CMDB have not been submitted to any processing, and have been loaded inside the inventory database by the ETL jobs.

For AppV, the input file received was not a CSV file but an Excel table. Consequently, it was transformed in CSV with the format "license type;software name;version;DG;number of users;number of unique users".

Once in the inventory database, the software list is sorted by descending order of occurrences. All software items from the DB have been listed by SystemType (workstations - source: LANDesk & AppV, or servers - source: satellite & CMDB).

2.2. Open source screening

Open source items have been manually screened in the full list of software, resulting from the previous phase. The screening has been based on the operator's knowledge and on Google searches.

The output of this step is the manually generated 'license-override.csv' file, which flags the open source items.

2.3. Software Type filtering

From the open source only list of software items, a manual categorisation of the items is applied according to the following categories:

Table 1 - Software clusters

AppSoftware
CustomSoftware
MobileSoftware
RuntimeSoftwarePlatform
OperatingSystem
DevelopmentPlatform

[1] WP3 DLV1, "Open Source Software Inventory Methodology", page 33

DataCenterResources
Library-Utility

The categories above have been defined in WP3 DLV1 (Inventory methodology), except for the Library-Utility type. This type was added during this software type categorisation process to filter some system files, libraries, utilities and small pieces of software.

The output of this step goes into the "opensource-softwaretype.csv" file.

All workstation software items (from LANDesk and AppV) have been categorized as above. On the other hand, due to the high manual workload of this step, several files from the data centre (about one third of the total, belonging to the SystemType "Server") were not categorised. The categorization may be completed once the ongoing inventory process is in place.

2.4. Evaluation of software through criticality criteria and definition of the critical open source software shortlist

The metrics applied for the evaluation of the inventoried OSS are mostly meant to identify and rank the software that is potentially critical due to its presence and use at the European Institutions (what we could call "business criticality"). A shortlist of the software items with the highest occurrences in the inventory has been identified and analysed based on such criteria.

In a subsequent step, the software items identified as critical according to this shortlist have been further evaluated and ranked on the basis of sustainability criteria defined by WP1.

The criticality criteria applied are:

1. Number of instances:
    - Rationale: the more a software is deployed, the more it impacts the infrastructure and/or the user base, and the more damage a vulnerability could cause;

    - Rating: this metric divides the number of instances of an item by the number of instances of the most common item in the list, thus providing a weighted view of the frequency of a software compared to the others of the list; its possible values range from above 0 to 1.

2. Exposure to users
    - Rationale: a vulnerability in a component exposed to the end user (i.e. that offers an interface to the end users) increases the risk of an exploit attacking the software. This criteria only applies to data centre infrastructure, since workstation users have a direct login to their machines;

    - Rating: a binary rating whose values can be either "1" (exposed to users) or "0" (not exposed to users).

3. Relation with security:

- Rationale: a vulnerability in component related to a security aspect may increase the damage due to an exploit. Examples of security-related software are the solutions meant to secure communication, to manage authentication, to manage processes and permissions...

- Rating: a binary rating whose values can be either “0,5” (security-related) or “0” (not security-related).

The total score provided by the sum of the three above scores is then normalized by dividing it by 2,5 (the sum of the highest values of the three criteria), so that the applicable values for the final score (the “Business Criticality Index”) range from 0 to 1.

The table below shows the calculation of the Business Criticality Index of the top critical items. Such items are divided in different groups depending on their sources (App-V, LANDesk and Datacentre) so that it is possible to identify the components easily.

Table 2 - Shortlist of critical items ranked by Business Criticality Index

Software Name	# Instances of the most common of the list	# Instances	Relative # of instances	Exposure to users	Relation with security	Score	Normalized score
Datacenter							
Apache http	6087	2859	0,4697	1	0	1,4697	**0,5879**
openssl & pyOpenSSL	6087	4918	0,8080	0	0,5	1,3080	**0,5232**
nspr & nss	6087	4108	0,6749	0	0,5	1,1749	**0,4700**
shadow-utils	6087	4104	0,6742	0	0,5	1,1742	**0,4697**
Openldap	6087	4103	0,6741	0	0,5	1,1741	**0,4696**
Libselinux	6087	4103	0,6741	0	0,5	1,1741	**0,4696**
Openssh	6087	4103	0,6741	0	0,5	1,1741	**0,4696**
Iptables	6087	4103	0,6741	0	0,5	1,1741	**0,4696**
m2crypto	6087	4103	0,6741	0	0,5	1,1741	**0,4696**
linux kernel	6087	4087	0,6714	0	0,5	1,1714	**0,4686**
Apache Tomcat	6087	603	0,0991	1	0	1,0991	**0,4396**
Glibc	6087	4103	0,6741	0	0	0,6741	**0,2696**
AppV							
Filezilla	2749	2749	1,0000		0,5	1,5000	**0,6000**
Putty	2749	1606	0,5842		0,5	1,0842	**0,4337**
Winscp	2749	1006	0,3660		0,5	0,8660	**0,3464**
Eclipse	2749	442	0,1608		0	0,1608	**0,0643**
Workstations							
firefox	43324	43324	1,0000		0,5	1,5000	**0,6000**
VLC	43324	40434	0,9333		0	0,9333	**0,3733**
tortoise	43324	1144	0,0264		0,5	0,5264	**0,2106**
keepass	43324	487	0,0112		0,5	0,5112	**0,2045**
7-Zip	43324	5143	0,1187		0	0,1187	**0,0475**
netbeans	43324	1787	0,0412		0	0,0412	**0,0165**

The “number of instances” is calculated based on the number of deployments; as for App-V, it must be understood as "the number of users".

The figure below shows the ranking of shortlisted OSS items based on the Business Criticality Index, grouped by system type, by descending order of criticality.

Figure 1 - OSS shortlist criticality ranking

As a result of the above evaluation, open source items with most occurrences, possibly security related (openssl...) and possibly facing a large user base (Apache, Tomcat) are considered critical.

The output of this phase goes into the “criteria-Datacenter.csv”, criteria-Landesk.csv and criteria-AppV.csv files, manually generated based on information coming from the analysis of the software items.

Note: the inventory sources on whose basis the critical software shortlist had been drawn, according to the project plan, have been integrated at a later stage. The Business Criticality Index (BCI) for the shortlisted items has been recalculated accordingly. Furthermore, additional items have been identified, having a BCI above 0.4, as shown in the table below:

Table 3 - Additional critical items

Software Name	# Instances of the most common of the list	# Instances	Relative # of instances	Exposure to users	Relation with security	Score	Normalized score
bash	6087	4624	0,7597	0	0,5	1,2597	**0,5039**
yum	6087	4103	0,6741	0	0,5	1,1741	**0,4696**
pam	6087	4103	0,6741	0	0,5	1,1741	**0,4696**
rpm	6087	4103	0,6741	0	0,5	1,1741	**0,4696**
selinux-policy	6087	4103	0,6741	0	0,5	1,1741	**0,4696**
sqlite	6087	4103	0,6741	0	0,5	1,1741	**0,4696**
sudo	6087	4098	0,6732	0	0,5	1,1732	**0,4693**
sync	6087	4085	0,6711	0	0,5	1,1711	**0,4684**
Apache Tomcat Server	6087	603	0,0991	1	0	1,0991	**0,4396**
subversion	6087	3585	0,5890	0	0,5	1,0890	**0,4356**
redhat-lsb	6087	3573	0,5870	0	0,5	1,0870	**0,4348**
bzip2	6087	6087	1	0	0	1,0000	**0,4000**

All of the newly identified critical items are installed on servers and have (apart from Bash) a BCI which is equal or lower than the already shortlisted items, so it can be assumed that the value of the subsequent analysis, although done on a restricted set of items, has not been significantly affected.

2.5. Evaluation of the critical software through WP1 (sustainability) criteria

The shortlisted items resulting from the analysis described in paragraph 2.4 have been submitted to manual investigation to be rated against the WP1 sustainability metrics[2].

Such metrics have been calculated automatically through an Excel file in which, once entered the parameters used to define the metrics, these are automatically calculated according to the methodology provided in WP1 DLV6, see Annex 10.

The parameters used for the calculation of sustainability metrics have been looked for in openhub.net, Wikipedia.org, github.com, and the specific community websites of each software items, their documentation, and their bugtrackers.

The output of this phase goes into the same files “criteria-Datacenter.csv”, “criteria-Landesk.csv” and “criteria-AppV.csv” containing the criticality criteria evaluated as per paragraph 2.4 above, manually integrated with the information on the WP metrics calculated through the file above.

Due to the length of the full names of the metrics, their IDs (M1 to M34) have been used instead in the database.

[2] See WP1, DLV6, ”Final metrics definition”.

2.6. License identification for critical software items

For critical software items, manual research has been performed to identify the applied licenses. Output of this step also goes in the "license-override.csv" file, generated as per paragraph 2.3 above. In the file, each row corresponds to a pair of "software item/ respective licence". For software with multiple licenses, several rows for the same software item have been created.

After this step, the shortlisted critical software items present in the file are mentioned with the exact names and versions of their licenses, whereas the other open source items are simply labelled as 'open source'.

2.7. Identification of dependencies

Within the critical software shortlist, the items from the datacentre servers have undergone a dependency identification process. Actually, the information of the dependencies can only be found in Linux systems. In Linux, when a software package is installed, the package manager resolves a list of dependencies that are downloaded and installed on the fly, whereas on Windows, the installed software packages contain all their dependencies.

Finding the dependencies of a Linux package can be done, on any RedHat-like machine, using this command:

```
yum deplist httpd | grep provider
```

where "httpd" is the name of the package to get dependencies from.

The output of this step goes into the "dependencies.csv" file, generated from the above command through the transformation of the result into CSV format.

2.8. Input of the analytical files in the database

Once those additional input files (criteria-AppV.csv, criteria-Datacenter.csv, criteria-Landesk.csv, dependencies.csv, license-override.csv, opensource-softwaretype.csv) were generated, new Talend jobs were designed to process them and inject them into the inventory database.

The whole docker-compose stack could then be redeployed.

3. CREATION OF PENTAHO DASHBOARDS AND EXTRACTION OF INVENTORY REPORTS

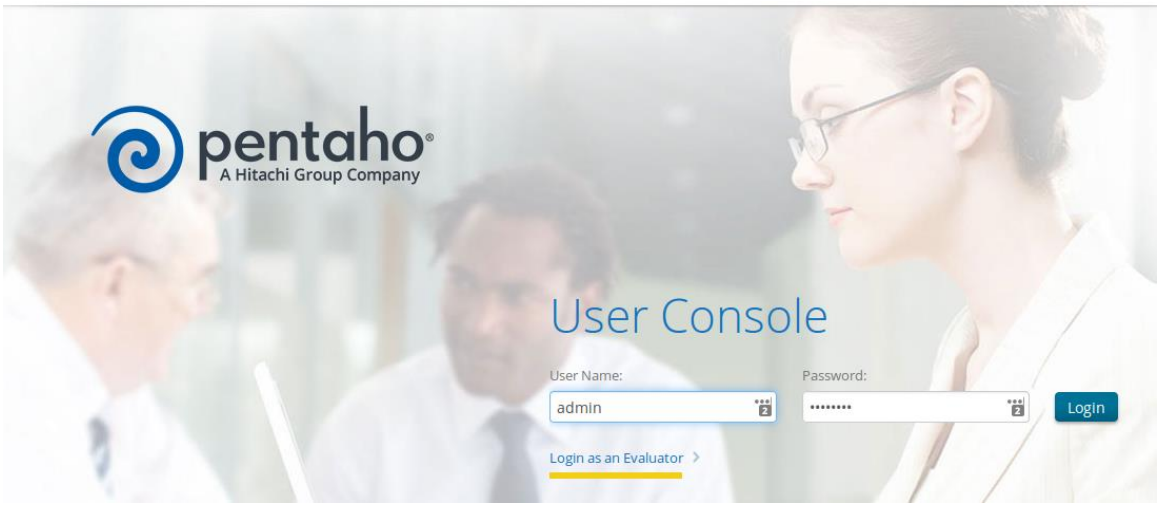
As per DLV3 [3], inventory data can be navigated and analysed through Pentaho dashboards. These are collections of other content components displayed together with the goal of providing a centralized view of key performance indicators (KPI)s and other business data movements.

This section gives an overview of the structure of the Pentaho dashboarding feature, to help DIGIT manage and edit the dashboards.

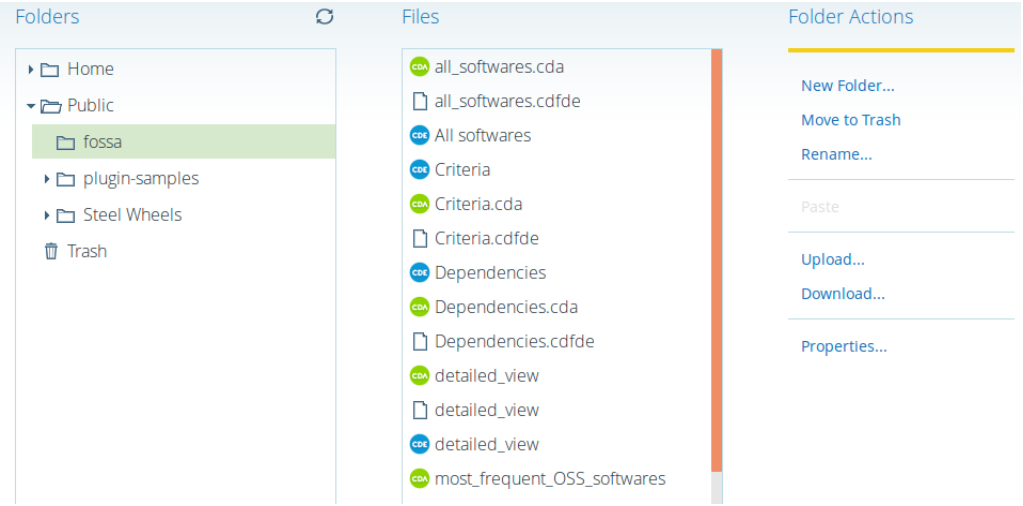
[3] WP3 DLV3, "Selection of tools to perform periodic inter-institutional inventories of software assets and standards"

1. Connect to the Pentaho User Console (on the docker host, url would be http://localhost:8080/pentaho).

Figure 2 - Pentaho User Console

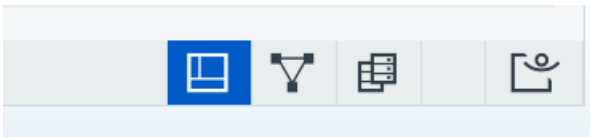


To edit an existing dashboard, click on 'Browse files' and go to the folder Public -> fossa



Select the CDE file from the dashboard to edit, and click edit in the right panel. This opens the CDE (Community Dashboard Editor). The editor is split in 3 tabs (on the right hand): layout, components, and datasources.

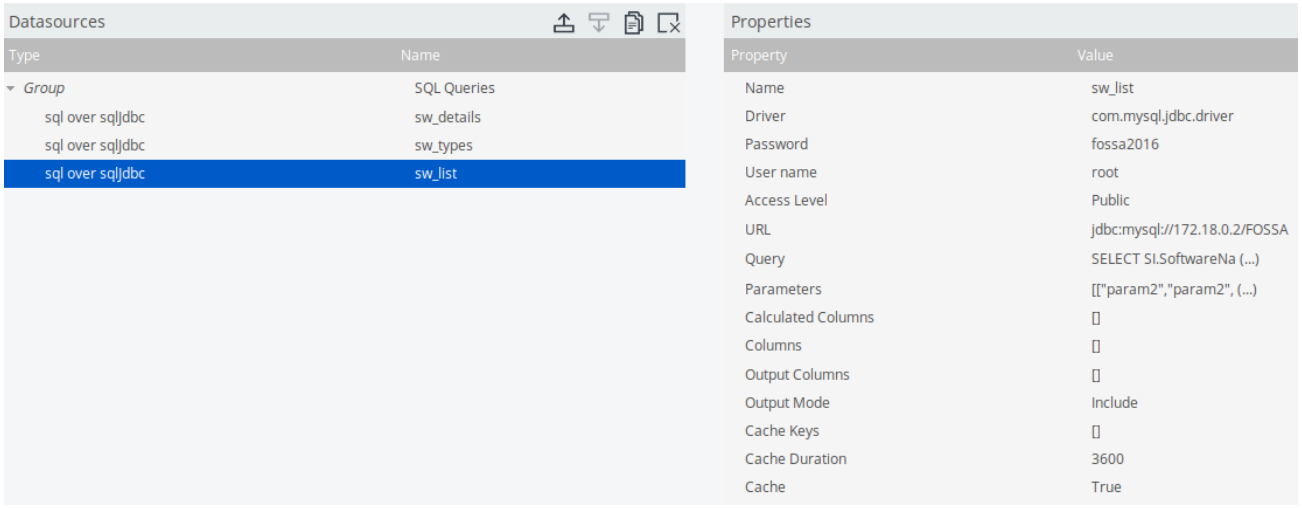
Figure 3 - Community Dashboard Editor



The layout tab is meant to design the look and feel of the dashboard (the columns, the rows, the title...). The component tab allows to add the type of component the dashboard will contains (charts, tables, dropdown list). Those components will be filled with the data coming from the database. The datasource tab is used to configure the access to the database.

The datasource tab appears as follows:

Figure 4 - Pentaho CDE Datasource tab



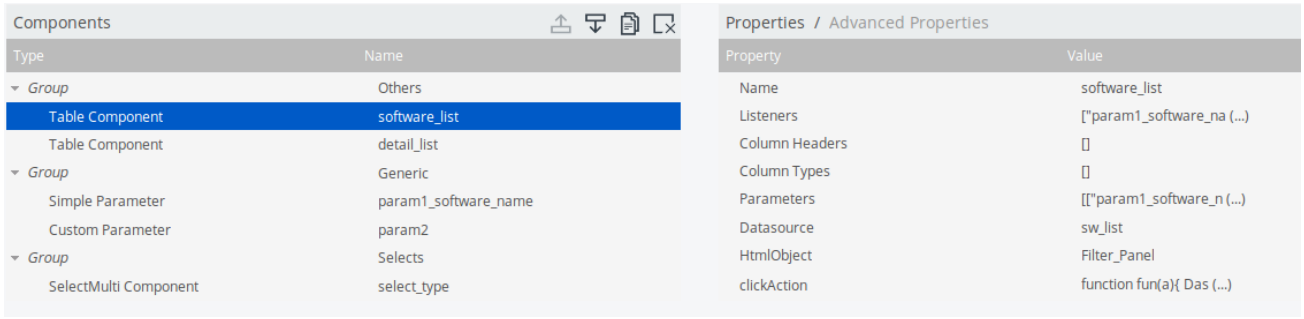
As MariaDB is used to store the data, the datasource type is SQL requests over SQLJdbc.

The settings to access the database are shown in the capture above. The IP may vary, depending on the docker-compose deployment (docker-compose creates a new virtual network for each deployment). This can be checked by using "docker inspect #mariadb_container" on the docker host.

The 'Query' setting handles the SQL request to access the data.

Once the datasources are configured for every SQL request needed for each component of the dashboard, the components can then be configured. The component tab looks as follows:

Figure 5 - Pentaho CDE Component tab



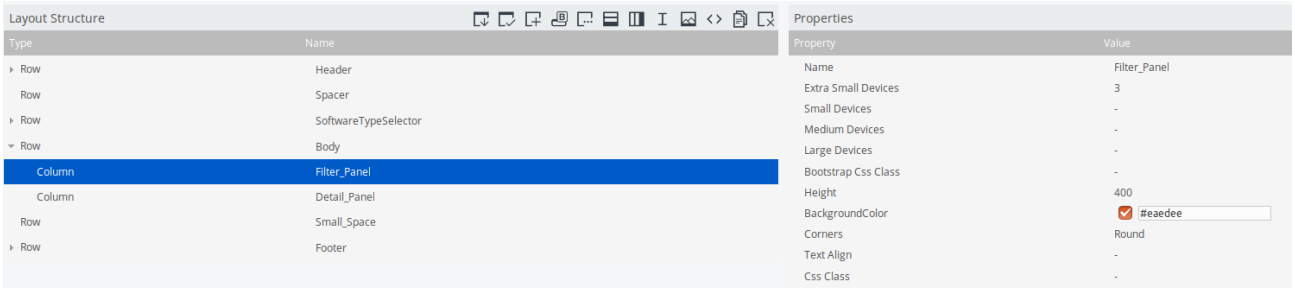
This dashboard includes three types of components: tables, parameters, and selection components.

The tables display the list of software items and the details on a specific software. They are linked to a specific 'Datasource' (designed in the Datasource tab) and to a specific HtmlObject (defined in the Layout tab).

To impact the behaviour of the 'detail_list' table based on a clickAction on the 'software_list' table, parameters and clickAction are defined.

Once all the components are configured, they should be associated to various layout elements, from the 'Layout' tab.

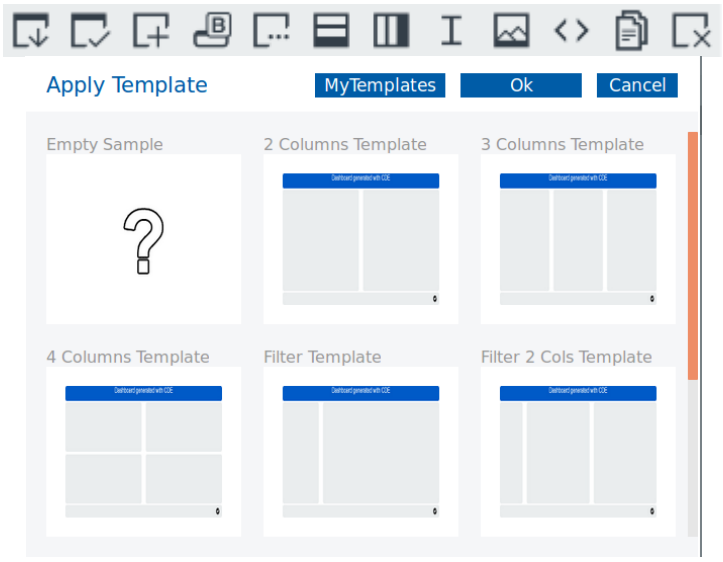
Figure 6 - Pentaho CDE Layout tab



Those elements are usually rows and columns. Their names should be invoked as the 'HtmlObject' of the components settings.

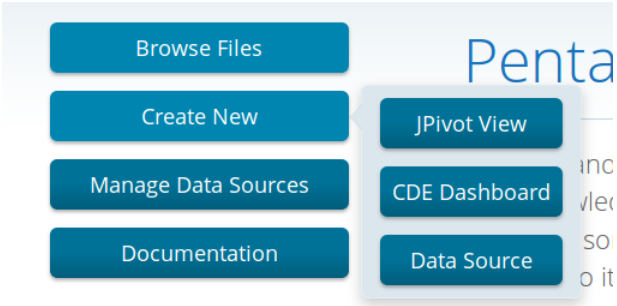
Some pre-existing layout templates are available in the menu bar of the Layout tab.

Figure 7 - Pentaho layout templates



Now that all the layout elements, components and datasources are configured, the dashboard can be tested using the preview button on the right hand on the screen

Should it be necessary to create a new template, there would be no need to browse files as mentioned in step 2. It would be actually sufficient to go directly to Create New -> CDE Dashboard from the Pentaho User Console.



The Pentaho dashboards implemented for the FOSSA pilot project are:

1. Summary of items: it shows the total amount of:
   - Number of systems;
   - Number of total software items;
   - Number of Open Source Software items;
   - Number of OSS items installed by system type;
   - Number of OSS items installed by software type;
2. Summary of instances: it shows the total amount of:
   - Number of total software instances;
   - Number of Open Source Software instances;
   - Number of OSS instances installed by system type;
   - Number of OSS items installed by software type;
3. Detailed view: in this dashboard the user can select any given software and get information about its versions, the systems it is installed on, and other available data for that software;
4. List of all software by descending order of instances;
5. List of OSS by descending order of instances;
6. List of OSS by system type. This dashboard brings the possibility to only count the items which are installed at least X times (X being a parameter), in order to focus the attention only on the most relevant items;
7. List of proprietary software by system type (with the same possibility to cut the less relevant items);
8. List of OSS by software type, with the possibility to select development or non-development oriented software;
9. List of critical OSS and its rating against the criticality and sustainability criteria;
10. List of critical OSS with pertinent dependencies.

All of these dashboards show the raw data provided by the inventory sources.

An extra dashboard presents an aggregated view of the most frequently installed open source software of software type 'AppSoftware' and system type 'Workstation'. This dashboard eliminates redundant items, part of the same application, in order to show a more high level view. It is provided as a showcase of what can be produced out of Pentaho (given the approved data model) to present a consolidated list. However, due to the need for further manual elaboration to obtain appropriately consolidated data, this dashboard has not been taken as reference for the analysis of the inventory content. This analysis has been performed manually and is presented in Section 5 below.

The extraction files in annex (corresponding to dashboards 1 to 10 above, with summary data split in Dashboards 1 and 2 – "items" and "instances" – to avoid runtime errors due to the high number

of simultaneous requests triggered, but merged into Annex 1, and excluding Dashboard 3 that is meant for a dynamic view) provide the raw inventory data for the EU-FOSSA project.

4. THE ONGOING INVENTORY PROCESS

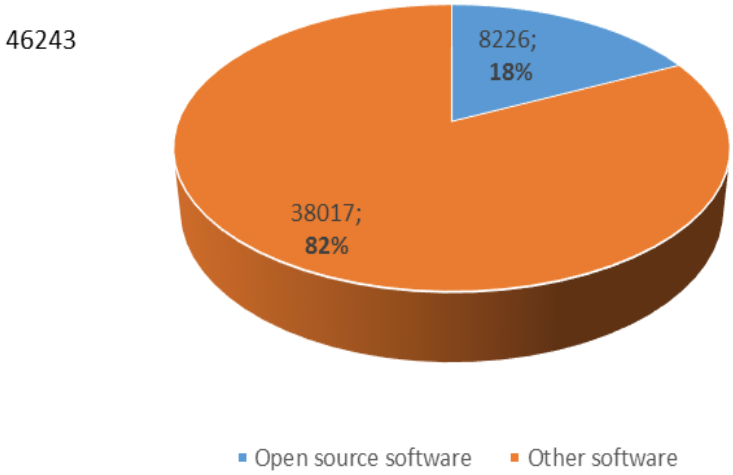
The ongoing inventory process is basically identical with the process applied for the FOSSA inventory described in section 2 above. The main difference consists in the fact that, as described in WP3 DLV1[4] the input files shall be obtained by the inventory manager triggering a request to the various owners of the respective sub-inventory systems to push their export in CSV format in a Git repository.

At that point, the Inventory manager may proceed by following up the steps from paragraph 2.2 on.

5. THE INVENTORY RESULTS

The inventory has identified 46243 software items (see Annex 1), out of which 8226 (about 18%) are OSS (see Figure 8 below, Dashboard 3[5] and Annex 2).

Figure 8 - Inventoried software items

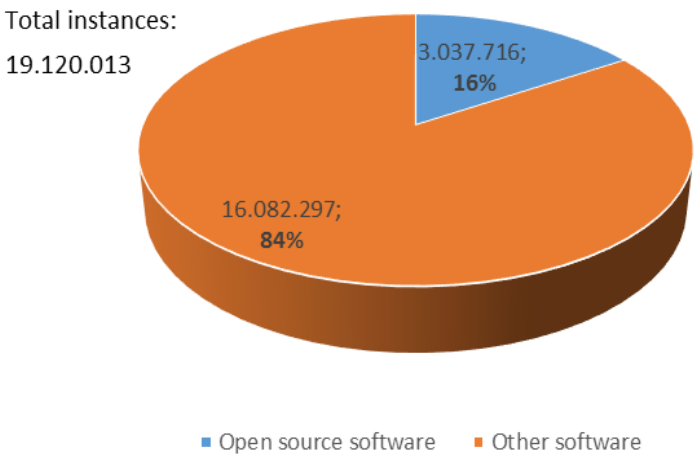


In terms of number of installations and deployments, OSS counts on 3.037.716 instances, out of 19.120.013 instances for all software installed or deployed, representing 16% of all the instances (see Figure 9 below).

[4] WP3 DLV1, "Open Source Software Inventory Methodology", p. 33

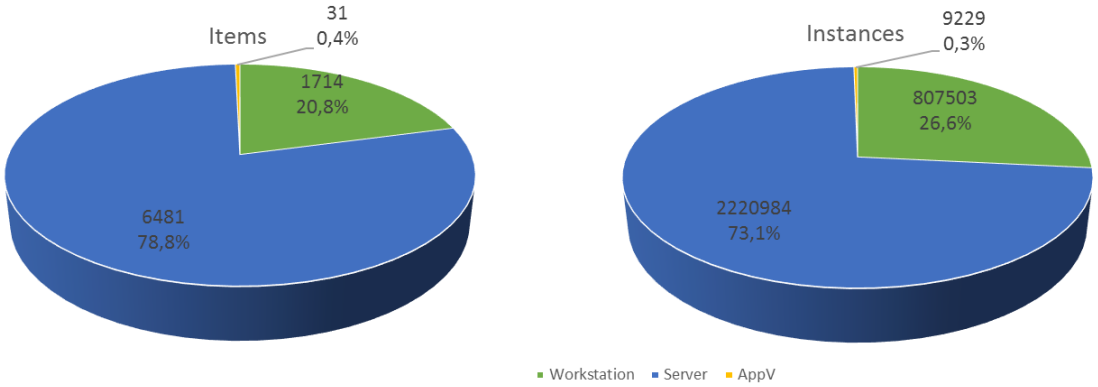
[5] The dashboard numbers mentioned in this section refer to the list of Section 3, p. 17.

Figure 9 - Inventoried software instances



The majority of Open source software items (6481) is installed on Datacenter servers, while 1714 are installed on workstations and only 31 managed by AppV. In terms of instances, around 73.1% of the OSS instances are installed on Datacenter servers (see Figure 10 below, Dashboard 5 and Annex 3).

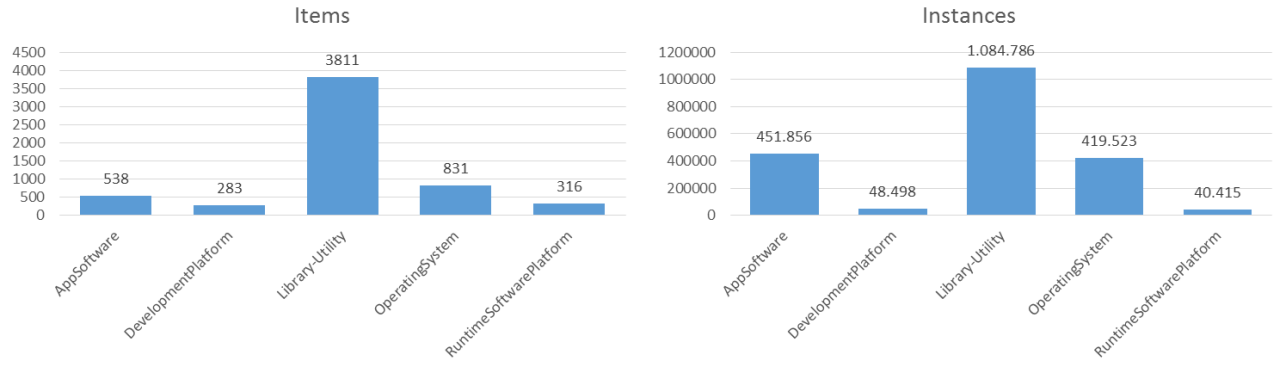
Figure 10 - OSS by system type



The analysis of OSS by software types, described in Paragraph 2.3 above, covering the inventoried items with the highest number of instances, has shown that the vast majority of OSS, both in terms of items and in terms of instances, belongs to the "Library-utility" type, followed by the "Operating system" type, as per Figure 10 below (see Dashboard 7 and Annex 4). No software has been identified as belonging to the types "CustomSoftware", "DatacenterResources", nor "MobileSoftware"; as for the latter, this is due to the fact that mobile applications are out of the scope of the present inventory, as discussed in WP3 DLV1[6].

[6] WP3 DLV1, page 31

Figure 11 - OSS by software type



Raw inventory data have been elaborated to group all items pertinent to a certain software package and eliminate multiple items referring to the same application, in order to provide a picture where the extent of use of major software packages can be clearly perceived. The elaboration has been done manually to obtain the top-30 software items by instances for each software type. However, it is still possible to go through the unedited data in the various Pentaho dashboards listed in Section 3. The detail of the elaborations done is provided in Annex 7.

An extraction of the above mentioned elaboration is shown in the tables below.

Table 4 - Top 10 OSS Applications

Software	Number of instances
Firefox	43324
Info-ZIP	42890
VLC	40434
Calibre	25506
7-Zip	5143
FileZilla FTP Client	2749
Greenshot OCR	490
KeePass	487
WinDirStat	400
GIMP	392

As shown in Table 4, the top 4 OSS applications (Firefox, Info-ZIP, VLC and Calibre) account for 1/3 of all OSS applications (with the top 10 accounting only for slightly more than that). The rest of this software type, apart from the following two occurrences in the top list (7-Zip and Filezilla FTP Client) is extremely fragmented among a variety of applications, each of which representing not more than 0,1% of the OSS applications instances (and not more than 0,003% of the total inventoried OSS instances).

Table 5 - Top 10 OSS Development platforms

Software	Number of instances
Notepad++	4953
GCC	3904
subversion	3585
Perl Command Line Interpreter	3419
CVS	2765
NetBeans Platform Launcher	1787
TortoiseSVN client	1144
sqldeveloper	963
Git	867
Eclipse	442

As for the Development platform software type, the top 10 items account for about 50% of the total.

Table 6 - Top 10 OSS Libraries / utilities

Software	Number of instances
Firefox utilities[7]	42033
VLC-cache-gen	40586
Calibre utilities[8]	12067
bzip2	6087

[7] Include Plugin-container and Plugin-hang-ui

[8] Include Calibre-Debug, Calibre-Parallel, Markdown-Calibre, Calibre-Complete, Calibre-Eject, Calibre-Customize, Calibre-Smtp, Calibre-Server, Fetch-Ebook-Metadata

Software	Number of instances
zip	4522
ncurses	4103
libgcc	4103
python-urlgrabber	4103
libuser	4103
elfutils-libelf	4103
bzip2-libs	4103
libxml2-python	4103
keyutils-libs	4103
krb5-libs	4103
python-iniparse	4103
libsemanage	4103
libutempter	4103
m2crypto	4103
libgcrypt	4103
cyrus-sasl-lib	4103
rpm-libs	4103
e2fsprogs-libs	4103

As it could be expected, the top three items in the above list are utilities related with the top three OSS applications of Table 4 (Firefox, VLC and Calibre). The top items from this list account for 15% of the total instances in this software type. The lower incidence of the top items here, compared to those of the other software types, can be linked to the high diversification of the items included in this type.

Table 7 - Top 10 Open Source Operating systems

Software	Number of instances
openssl	4918
bash	4624
curl	4208
gawk	4175
passwd	4171
basesystem	4103
redhat-logos	4103
rhn-setup	4103
iptables	4103
glib2	4103
openssh-server	4103
yum	4103
chkconfig	4103
pam	4103
rpm	4103
glibc	4103
selinux-policy	4103
iputils	4103
openssh	4103

Almost all items included in the “Operating System” software type refer to RedHat Enterprise Linux. The remaining few items not related with that package refer to platforms such as QEMU, MinGW or Cygwin installed on workstations, and emulating a Linux operating system on a Windows desktop.

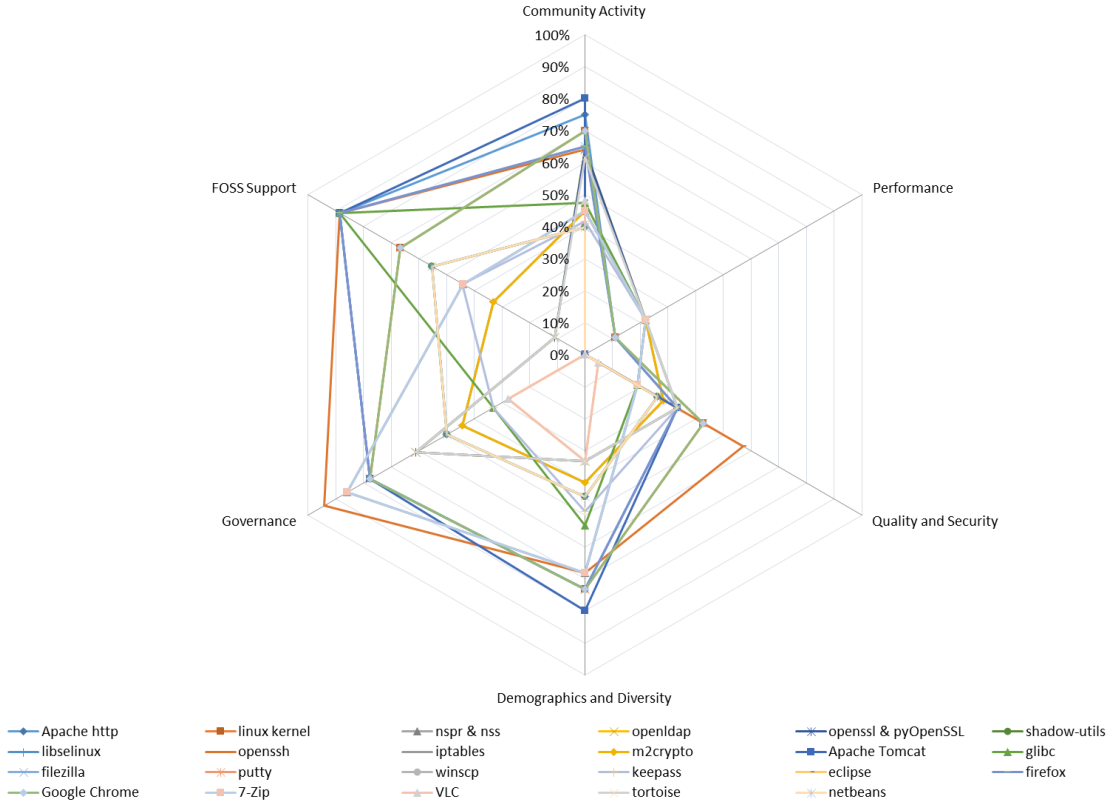
Table 8 - Top 10 Open source Runtime software platforms

Software	Number of instances
Calibre DB	10855
python	4584
perl	4169
sqlite	4103
openldap	4103
Apache HTTP Server (httpd)	2859
openldap-clients	2634
Tomcat Server	603
postfix	541
Node.js	285

As mentioned in paragraphs 2.5, 2.6 and 2.7, a further, in-depth analysis has been performed on the critical software shortlist defined on the basis of the Business Criticality Index.

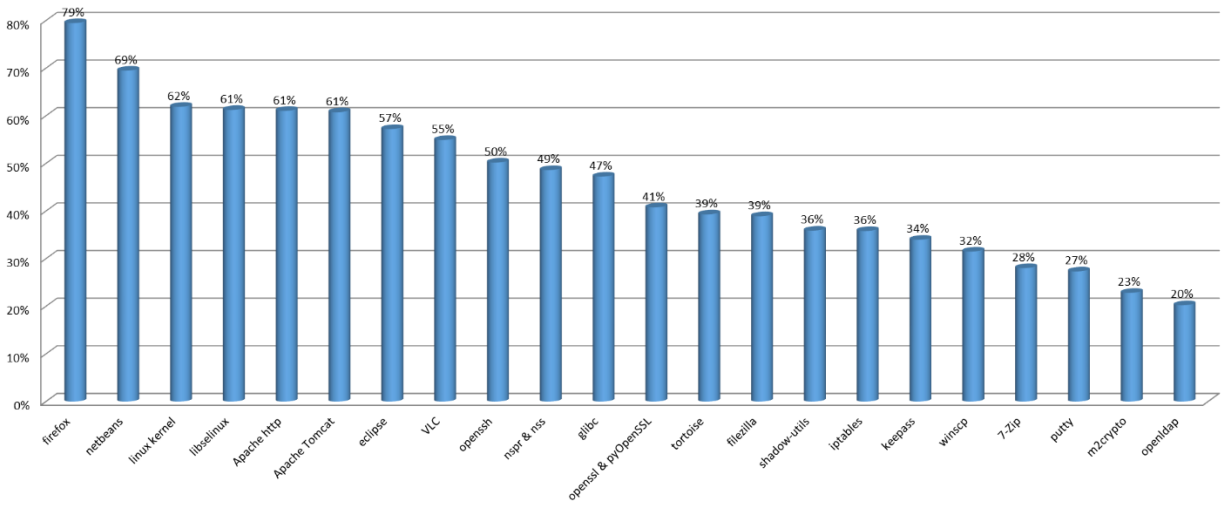
The first analysis, related to the sustainability of OSS projects evaluated based on the 34 metrics defined by WP1 DLV6 (see the file attached under paragraph 2.5), has shown some limits in the actual availability and usability of data related to some of such metrics, particularly concerning the area of Performance (e.g. time spent in code reviews). Such lack of information has lowered the average rating of the projects in that area, as presented in the figure below:

Figure 12 - Comparison of shortlisted OSS based on sustainability ratings by category



The overall Sustainability Index (SI), calculated as the average of the 34 metrics, shows that the critical shortlist includes software with a wide range of SI values, from 20% to almost 80%:

Figure 13 - Average of all metric categories showing overall sustainability of OSS projects



All information on critical OSS criticality and sustainability is summarized in dashboard 9 of the list of Section 3, p. 17 above, and extracted in Annex 5.

The inventory analysis has also covered one area that may amplify significantly the risks occurring in one of the inventoried OSS components: the dependencies analysis performed on the Critical OSS shortlist (see Dashboard and Annex 6).

The components listed in the table below have more than 1 dependency upon the shortlisted items:

Table 9 - Top shortlisted components by number of dependencies

Components	Number of dependencies
Glibc	10 dependencies
Bash	8 dependencies
zlib, coreutils	4 dependencies
shadow-utils, libselinux, openssl-libs, system	3 dependencies
chkconfig, audit-libs, nss-util, krb5-libs, pcre, nspr, nss-softokn-frebl, libcom_err	2 dependencies

This analysis shows a relative fragmentation of the dependencies, apart from Glibc and Bash, as shown in the figure below:

Figure 14 - Dependencies on Glibc and Bash

6. APPENDIX 1 – LIST OF ANNEXES

Annex 1	Inventory summary
Annex 2	The list of top OSS installed at DIGIT with number of instances
Annex 3	List of top OSS by system type
Annex 4	List of top OSS by software type
Annex 5	List of critical OSS and its rating against criticality and sustainability criteria
Annex 6	Analysis of dependencies of the shortlisted OSS
Annex 7	Elaborations on raw inventory data for Top-30 items by software type
Annex 8	Critical shortlist rating

7. APPENDIX 2 – ABBREVIATIONS AND ACRONYMS

FOSSA	Free and Open Source Software Auditing
WP	Work Package
DLV	Deliverable
csv	Comma-Separated Values (file format)
CMDB	Configuration Management Data Base
ETL	Extract, Transform, Load
OSS	Open Source Software
CDE	Community Dashboard Editor
SQL	Structured Query Language