DIGIT
Unit B1

FOSSA WP3 - Deliverable n. 2

Proposition of tools to perform periodic inter-institutional inventories of software assets and standards

Date: 07/03/2016
Doc. Version: Final

TABLE OF CONTENTS

TABLE OF FIGURES

TABLE OF TABLES

1. Deliverable Overview

The main aim of this deliverable is to accomplish the objective of *"Task 2: Propose tools to perform periodic inter-institutional inventories of software assets and standards*" of the FOSSA Pilot Project, which is to prepare a list, together with necessary justifications, of tools which can be used to keep and consolidate an inventory of software assets and standards, targeting regular automatic collection of data from systems existing in the European Commission and the European Parliament.

The list also contains information necessary to support the subsequent selection of inventory tools by the European Commission and the European Parliament.

This study briefly recalls (Section 2) the main features and constraints of the Pilot Scenario, comparing it with a Target Scenario, as already described in Deliverable 1 of Work Package 3 (WP3-DLV1) of the FOSSA Pilot Project. This is done to clarify how the features and constraints of the Pilot Scenario impact on the choice of the families of tools for the inventory process, and how easing some of such constraints in a Target Scenario may lead to a different approach to the selection of tools.

Subsequently (Section 3), the general architecture of the inventory process and its layers are shown, together with its successive steps (the inventory of software components and standards, the collection of pertinent metadata, the filtering and ranking of the data obtained in the previous two steps). In particular, the architecture is put in relation with the Target Data Model (TDM) described in WP3-DLV1. This will help to recommend the manual or the automatic management of information for each entity of the TDM, and therefore to identify where and how to use pertinent families of tools.

In Section 4, for each of the three inventory steps mentioned above, the applicable approaches are described and evaluated vis-à-vis the recommendations provided in Section 3. This in order to identify the most appropriate approach to executing each of such steps, including the applicable families of tools.

Section 5 describes how, for each family of tools to be used in the various layers of the architecture and steps of the inventory process, recommended tools are identified. This starts from a long list of potential candidate tools, filtered through appropriate selection criteria in order to obtain a shortlist that is submitted to a detailed scoring and ranking based on further specific criteria.

Section 6 deals in fact with such scoring and ranking. The output of this section is therefore the ranking of tools for each step and layer of the inventory process and architecture. As an output, it provides the European Institutions with a sound recommendation for the selection of tools to execute of the software and standard inventories in WP4 and WP5 of the FOSSA Project.

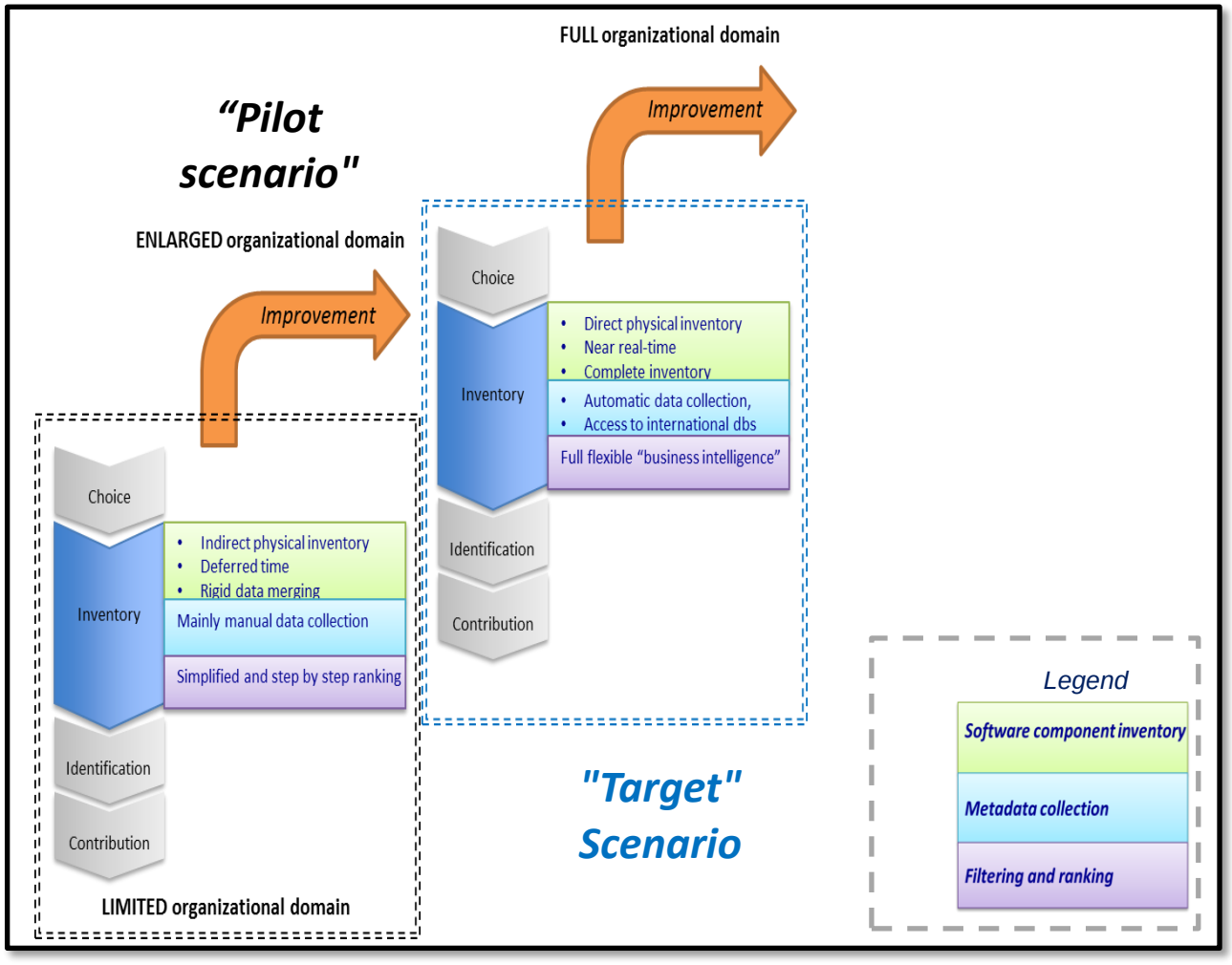
Finally, Section 7 provides a perspective view to this task, with a highlight of how the Target Scenario described in section 2 may impact on the choice of a wider range of tools for the execution of the inventory. A list of tools that may be applicable to such Target Scenario is submitted to a first tentative process of selection, scoring and ranking as described in Sections 5 and 6.

2. OSS Pilot Inventory Scenarios

As already described in Work Package 3, Deliverable 1 (WP3-DLV1) of the FOSSA Pilot Project, based on the information collected during the interviews and the assessment phase, the execution of the OSS Inventory should be analysed under two different scenarios, represented in Figure 1 and explained in the next paragraphs.

Each of the two scenarios shortly describes the features and constraints of the inventory process for its three main steps: Software components and Standards inventory, Metadata collection, Filtering and ranking.

Figure 1 - Pilot and target scenario



2.1. Pilot Scenario - Main features and constraints

The Pilot Scenario described in WP3-DLV1, "Open Source Software Inventory Methodology", and highlighted in the picture above, is briefly recalled here in order to point out the impact that its features and constraints have on the choice of the inventory tools, as it will be detailed in section 3.

One major constraint, in the framework of the Pilot project, is the lack of authorization to install any new agent to autonomously retrieve the data needed for the Inventory.
Consequently, the data must be collected by requesting (and obtaining) flat files (.csv) from the data sources identified during the interviews (e.g. AppV, Landesk, Satellite…). All such data sources have different Data Models, with information on the same domain that could be fragmented through different data sources; this determines a strong need to properly elaborate and integrate the different collected files.
Such files are provided by the data owner by an on-demand ("pull") approach starting from requests set by the coordinator of the inventory process (or “Inventory Manager”). However, the non-automatic, voluntary nature of such flow does not provide any guarantee on the exact timing and final format of data that would be effectively provided.
An additional relevant data quality issue is that the information currently made available by the European Commission and the European Parliament only partially covers the minimum set of information identified by Target Data Model (TDM). In the Pilot Scenario, therefore, the information content of the TDM will be partial, due to the lack of data sources. In particular, the information on the "Standard" entity, identified as core (see section 3.3 below), is limited. It may be therefore necessary to cross-check such information with external sources (e.g. through web crawling).

2.2. Target Scenario – main features

As for the study on the inventory methodology, a Target Scenario has been considered, in order to identify guidelines for the possible evolution of the OSS inventory activities to a more streamlined process, which may also impact on the choice of tools to perform it. This would be obtained mainly by easing some of the constraints pointed out for the Pilot scenario.

First of all, the access to wider, more complete information sources, in line with the requirements set by the Inventory Manager, must be granted to industrialise and automatize the Inventory process, so to ensure the completeness of the inventory and Target Data Model feeding.
In particular, the information must be complete with regard to the criticality assessment criteria and to the core entities of the TDM defined for the inventory.
Additionally, the required information must be easily and quickly accessible, and a scheduled ("push") approach must be enabled; this approach should be automated to grant persistent efficiency.

It must be underlined that, even in this Target Scenario, the constraints of the Pilot Scenario in terms of access to servers have been still considered applicable. However, the scenario and the architecture resulting from it may change if such constraints are removed, allowing the use of further fully integrated tools, that shall be shortly described at the end of the present study.

3. General inventory process architecture (Pilot Scenario)

In the following paragraphs we are going to describe in detail:

- The various technical components that will implement the solution, organized in layers;
- The data management patterns, describing how the various entities in the Target Data Model will be managed and fed with the pertinent information.

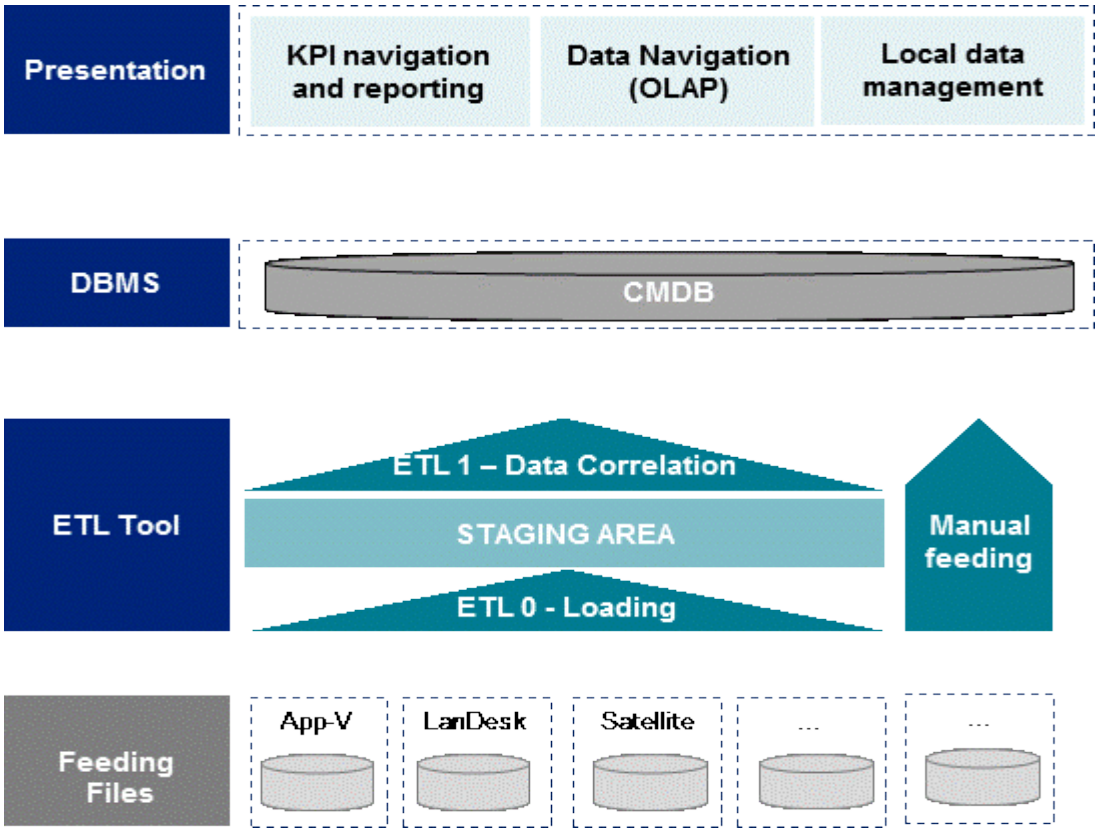
The architecture described below relies on families of tools, within which the candidate tools for the execution of the Pilot inventory will be selected.

3.1. Solution architecture

Based on the features and constraints of the pilot scenario, the technical solution for a federated CDMB is composed by the following layers, from the upper to the lower:

- **Presentation layer**: the user-facing part of the solution, composed of the following main modules:
    - Reporting;
    - Data Navigation (OLAP);
    - Local Data Management;
- **DBMS**: the Data Base engine that will store the data of interest, either coming from external systems or locally managed by the user;
- **ETL Tool**: the system or component that will perform the filtering, transformation and loading in the target data model of the data loaded in the staging area; this system should provide facilities for data lookup, encoding and data processing;
- **Feeding files**: these files, coming from the various asset inventory tools currently in use, will provide the information about the installed software base in the system in use. All files will be stored in the Staging area of the Target Data Base for further processing. The originating systems are not in scope and no further software components are needed.

Figure 2 - Solution Architecture



3.2. Data management patterns

In the architecture described above, two distinct feeding patterns of the mapped data into the repository are possible:

- **Manual – Locally managed** by the end users: the data will be locally managed by an ad-hoc user interface (local data management module) mapping the data content of the various entities;
- **Automatic – Feeding files** loaded from external systems/sources using an ETL Tool; the required data will be provided in the form of file extractions (.csv or other).

The feeding pattern of a certain entity may change with time, depending on the maturity of the solution. For example, the information on "Software Rating" may be fed manually in the framework of the Pilot Project, while at a later stage it could be integrated automatically with a software quality inspection tool.

For each entity of the data model, the feeding pattern is defined by applying the following criteria:

1. **Estimated effort** to manually populate the entity; this parameter is function of:
   - the frequency of update of the related information;
   - The volume of input data.
2. **Type of data source (e.g. structured, unstructured).**

In order to determine the best feeding pattern of each entity of the TDM, an opportunity assessment is made by using the following two variables:

A. **Estimated effort** to manually populate the entity. This estimation is based on two more variables:

- the **frequency of update**, with two possible values:
  - **Low**: the information is stable over a long time span (monthly or higher): e.g. list of licenses, list of standards;
  - **High**: the frequency with which the information changes is high (weekly/daily): e.g. list of installed software;
- The **volume of input data** to load, with two possible values:
  - **Low**: less than 100 instances;
  - **High**: more than 100 instances.

This variable is attributed the following range of values, computed by combining the sub-variables described above in the **scoring table** below:

- **Low effort**: less than a man-day, for low frequency / low volume data
- **Medium effort**: from one to five man-days, for low frequency / high volume or high volume / low frequency data
- **High effort**: more than five man-days, for high volume / high frequency data

Table 1 - Manual effort estimation

	Low Volume	High Volume
Low Update Frequency	Low	Medium
High Update Frequency	Medium	High

B. **Type of data source**, with three possible values:

- **Unstructured**: the information in the source is not structured, so it is not possible to build a parser to create a flat file to feed the CMDB: e.g. Software documentation, Standards;
- **Structured, easy to get**: the data is available in a structured format, either from internal EC systems (e.g. list of installed Software), or from external systems/repositories (xml metadata for Software, when available);
- **Structured, hard to get**: data that are structured but require building a custom tool in order to produce an input file, (e.g. Community support), or from a structured file, like Software dependencies.

Once the two above variables (Estimated Effort and Data Source Type) are computed for each entity of the TDM, the choice of feeding pattern is made by crossing them in a table, which provides the following values:

- **Manual**: when the data source is unstructured or data is hard to get and effort is low;
- **Automatic**: when data is structured;
- **By opportunity**: if the data is hard to get and estimated effort to collect them is medium to high, a cost/benefit analysis should be performed, in order to evaluate the complexity of building feeding tools and/or acquiring the proper data from external sources. Such analysis, based on the possible data sources, may be performed through a list of possible extraction tools, presented in Table 3, which can be built to support the extraction.

Table 2 - Feeding pattern evaluation variables

	Low Effort	Medium Effort	High Effort
Structured, easy to get	Automatic	Automatic	Automatic
Structured, hard to get	Manual	By opportunity	By opportunity
Unstructured	Manual	Manual	Manual

The result of the evaluation made by applying the above variables to each entity of the TDM is presented in Table 3 below. Such table also provides a list of possible tools that can be built to extract data from external sources. Further detail on the rationales used to attribute a certain value to the variables described above is provided in Appendix 1. .

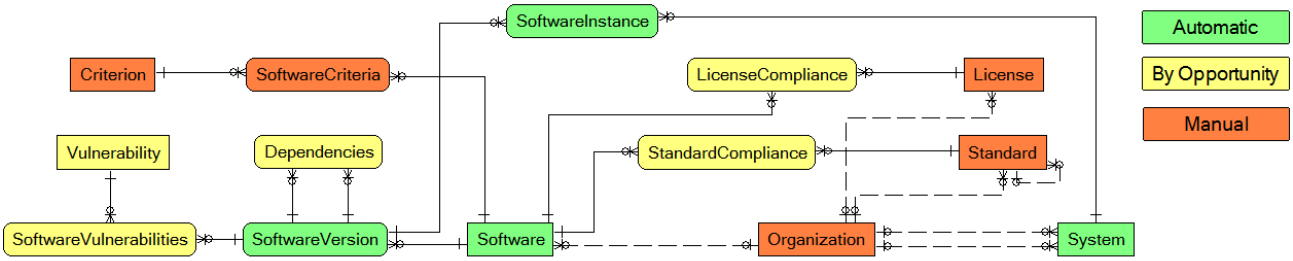
Table 3 - Analysis of possible feeding patterns by TDM entity

Entity	Available data sources	Transformation / mapping	Update Frequency	Data Volume	Effort	Data Source type	Feeding pattern	Custom built Extraction Tool
Software	List of installed software from EC CMDB systems	Names need to be normalized in order to match incoming metadata from external sources (e.g. from package name to project / software name)	Low	High	Medium	Structured, easy to get	Automatic	n/a
Software Version	List of installed software from EC CMDB systems	Versions need to be normalized in order to match incoming metadata from external sources (e.g. from package version to project / software version)	High	High	High	Structured, easy to get	Automatic	n/a
Standard	• DIGIT reference list of standards refreshed • Specialized sites (e.g. ISO, W3C, ANSI, OMG etc.)	Organized into a semantic tree	Low	Low	Low	Unstructured	Manual	n/a
System	List of installed software from EC CMDB systems (if available)	Straight loading	Low	High	Medium	Structured, easy to get	Automatic	n/a

Entity	Available data sources	Transformation / mapping	Update Frequency	Data Volume	Effort	Data Source type	Feeding pattern	Custom built Extraction Tool
Organization	• List of how producers from CMDB system for HW • Organizations managing software from external metadata • EC organization managing systems from Org chart	• Normalized data for HW producers • Normalized data for software development entities • Straight loading for EC management organizations	Low	Low	Low	Unstructured	Manual	n/a
License	Specialized sites (i.e. OpenHub)	none	Low	Low	Low	Unstructured	Manual	n/a
Vulnerability	Publicly available vulnerability sources (e.g. NVD)	Conversion from source message	High	High	High	Structured, difficult to get	By opportunity	Integration with mail box
Standard Compliance	Specialized sites (e.g. OpenHub)	Mapping from specialized web sites	Low	High	Medium	Structured, difficult to get	By opportunity	Web page scraping tool partial coverage by standard type
Licence compliance	Specialized sites (e.g. OpenHub)	Mapping from SW to list of standards	Low	High	Medium	Structured, difficult to get	By opportunity	Web page scraping tool
Criterion	Defined by the methodology	None	Low	Low	Low	Unstructured	Manual	n/a
Software Instance	List of installed software from EC CMDB systems	Mapping from SW version to list of systems	High	High	High	Structured, easy to get	Automatic	n/a
Software Criteria	List of installed software from EC CMDB systems	Mapping from SW version to list of systems	Low	Low	Low	Unstructured	Manual	n/a
Software Vulnerabilities	Publicly available vulnerability sources (e.g.NVD)	None	High	High	High	Structured, difficult to get	By opportunity	Integration with mail box + ETL tool
Dependencies	Package dependencies from software distributions	Mapping from packages version to software versions	Low	High	Medium	Structured, difficult to get	By opportunity	Parsers for package dependencies

Based on Table 3, the following image shows in a synthetic way the feeding approach for the various entities of the TDM.

Figure 3 - Synthetic representation of TDM entities feeding patterns



This fragmented scenario for the data collection can be enhanced by leveraging on two drivers:

1) **Improve the quality of data sources**, moving to structured and easy-to-access data sources (i.e. pay a data provider or buy commercial solutions).
2) **Build / Acquire** data collection tools for the different data sources required for the completion of the TDM.

4. Open Source Inventory Approaches in the Pilot Scenario

In the framework of this study, several approaches to the collection of data for the core entities of the TDM have been considered and evaluated as per the criteria set forth in section 3 above. The approaches considered are only those realistically applicable to the Pilot Scenario, i.e. under the constraints described in paragraph 2.1.

In the following paragraphs, we describe the approaches identified for the three steps of the inventory process described in Section 2 (Software component inventory and Standard Inventory, Metadata Collection, Filtering and Ranking).

4.1. Step 1 - Software component inventory

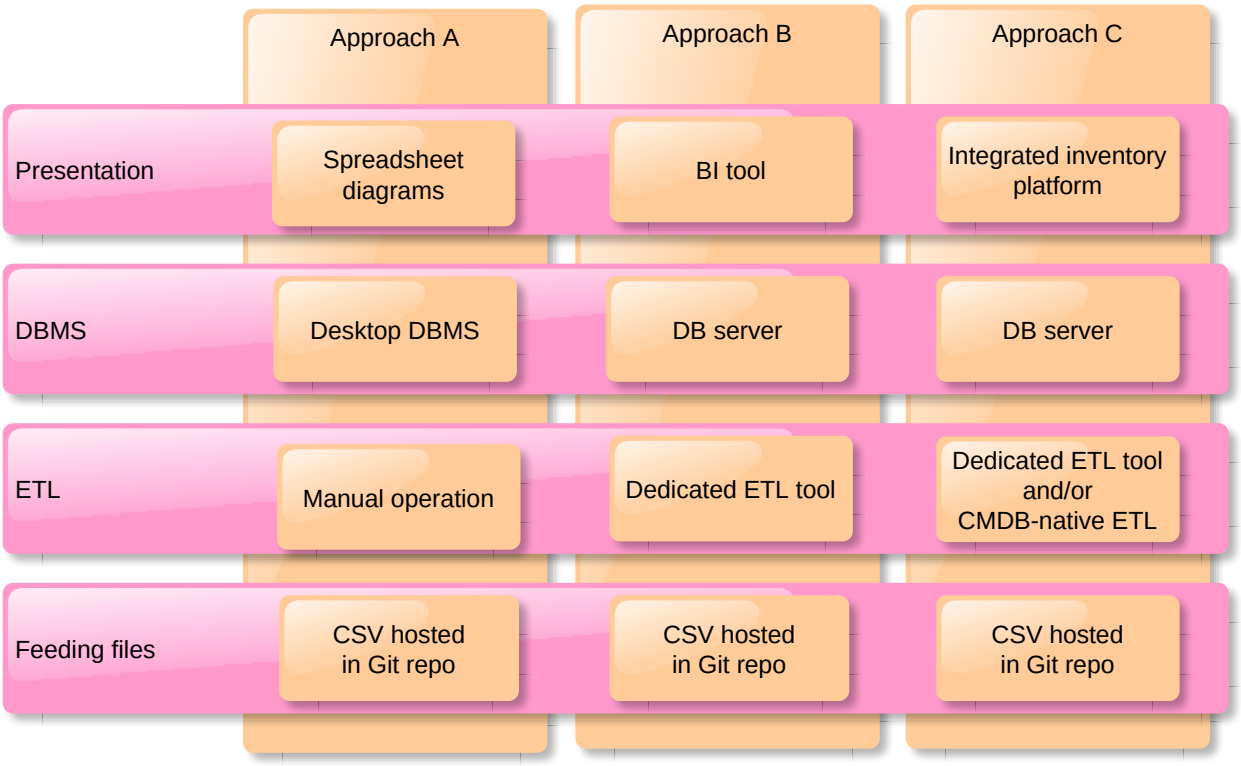
Upon the outcome of the interviews and the assessment phase, the following possible approaches to the execution of Step 1 (Software component Inventory) of the OSS Inventory approach have been identified.

Table 4 - Analysis of possible Software components inventory approaches

APPROACH	DESCRIPTION	INTEGRATION, REUSABILITY AND EFFORT ANALYSIS
A – Manual loading	Get the various information/data sources and load them manually into a single Personal Productivity System file. Each step is local in relation to the Inventory tool: 1. Get input files from Landesk, App-V, Satellite …; 2. Manually integrate the files; 3. Store information into a Personal Productivity System file (e.g. OpenOffice); 4. Visualise through reports or via spreadsheets graphics.	Level of integration: Low. Only desktop tools and manual tasks. Level of reusability: Low. Several manual steps to repeat or adapt Implementation effort: Low. No development. Operations effort: Very High. Crunching the data manually would be very time consuming
B – Integrate into a local database	Integrate the available information/data sources into a Local database: 1. Get input files from Landesk, App-V, Satellite… 2. Integrate the files through an ETL tool, working as a bridge between sources and the Inventory tool; 3. Store information into a database (MySQL, PostgreSQL, etc.) local to the Inventory tool; 4. Visualise via Business Intelligence tools local to the Inventory tool.	Level of integration: Medium. Server based DBMS is suggested, along with Business Intelligence and ETL tools. However, the inventory platform does not rely on a CMDB. Level of reusability: Medium. The inventory platform can be migrated towards a CMDB in the future using an ETL tool. Implementation effort: Medium. The number of tools involved in this stack is moderate. Operations effort: Low. Building the inventory once the platform is ready will be mostly automated.
C – Integrate directly on a CMDB tool	Integrate the available information/data sources directly on a CMDB tool: 1. Directly integrate input files from Landesk, App-V, Satellite and other data sources into a target CMDB tool: a. Through a CMDB-native ETL tool, or b. Through an external ETL tool 2. Visualise data directly from inside the CMDB tool.	Level of integration: High. Specific solutions dedicated to IT asset management are used. Level of reusability: High. A CMDB tool already implements standards and best practices in the asset management field. Implementation effort: High. A CMDB tool can be complicated to set up and manage. Operation effort: Low. These tools (ETL, CMDB) would allow maximum level of automation.

The following figure describes the possible families of tools to implement each layer of the solution architecture, comparing the three approaches described above, for the execution of the Software Component Inventory.

Figure 4 - Families of tools for the execution of software components inventory in the various approaches



Based on the considerations of Section 3 (see Table 3, entities “Software”, “Software Version” and “Software Instance”), and on the constraints of the Pilot scenario, Approach “B” of Table 4 is recommended for the execution of this step of the Pilot inventory. Approach “C” may be considered in a target perspective. The pertinent families of tools shown in Figure 4 will therefore be analysed more in detail in the following sections 5 and 6.

4.2. Step 1 - Standards inventory

The interviews and the assessment phase have led to the following considerations related to the nature of the data for the collection of Standards:

- data sources pertinent to standards are heterogeneous and currently only partially known;
- data are mostly unstructured.

These assumptions have led to the conclusion of the necessity to manually collect and feed into the Solution the information pertinent to standards, integrating the information available from sources of the European Institutions with further publicly available information. A detailed analysis of the rationale for this approach is shown in Table 5 below.

Table 5 - Analysis of possible Standards inventory approaches

APPROACH	DESCRIPTION	INTEGRATION, REUSABILITY AND EFFORT ANALYSIS
A – Manual collection	Get the various information/data and load them manually into the related entity in the TDM. Integrate and update the data sources provided in the Methodology phase with the cross-checking with external data sources (i.e. information from software producers, list of standards such as ISO standards, W3C, OMG, NIST, British Computer Standards, ANSI, OASIS…). In particular: • For each standard of the list, look on the external data sources for an updated version or a replacement standard • Such updated list of standards shall then be cross-checked with the software shortlist coming out of the software inventory • If a certain standard is not present in the updated list, it will be looked from in the main libraries of Standards (ISO standards, WSC etc.) to get an exact description • The updated list will be completed with the additional standards discovered as above • For all shortlisted software, the standards it complies with will be identified by checking the information provided by the producer, cross-checked with information provided by third party sources (e.g. Sonatype Nexus) • For some standards compliance, like file format support, may be identified by looking to dependencies	Level of integration: Low. Many manual tasks to be executed. Level of reusability: Low. Several manual steps to repeat or adapt. Implementation effort: Medium. Limited development for web crawling. Operations effort: High / Very high. Crunching the data manually would be very time consuming (depending on the amount).

4.3. Step 2 - Metadata collection

The interviews and the assessment phase have led to the following considerations related to the nature of the data for the Metadata collection:

- data sources of the standard are heterogeneous and currently partially known; moreover they can be difficult to acquire in a structured format;
- data are mostly unstructured.

These assumptions have led to the following proposed approaches for this step:

Table 6 - Analysis of possible Metadata collection approaches

APPROACH	DESCRIPTION	INTEGRATION, REUSABILITY AND EFFORT ANALYSIS
A – Metadata collection on reference sources	Get the various information/data and load them manually into the related entity in the TDM. An example of the main sources for Metadata are: 1. Vulnerability assessment through matching inventoried software on the web [software vulnerability assessment webpage] 2. Data on communities gathered through desk research Web-based sources can be inspected by using ad-hoc web agents This approach can grant the full coverage of the Metadata needed in the TDM.	Level of integration: Low. Many manual tasks to be executed. Level of reusability: Low. Several manual steps to repeat or adapt. Implementation effort: Low. No development. Operations effort: Very high. Crunching the data manually would be very time consuming.
B – Use of trial licenses for COTS tool	Gather the Metadata using COTS tool for software inspections. This approach only ensures a partial coverage of the Metadata needed (e.g. software vulnerability…)	Level of integration: Medium. Partial coverage of the Metadata needed. Level of reusability: High. Implementation effort: Medium. Integration needed. Operations effort: Medium. Automation of the collection process.

On the basis of the above analysis and of the estimated variables described in section 3, the approach for the execution of this step of the inventory should be assessed by opportunity, considering on one side the relatively high effort to manage the pertinent data, on the other the possibility to access structured data sources to get the necessary information.

4.4. Step 3 - Data filtering and ranking

In order to get the monitored software pool and considering the "Pilot project" scenario, constraints, overall timeline and context identified through the interviews and the assessment of the and interviews the following proposed approaches for this step have been developed.

Table 7 - Analysis of possible data filtering and ranking approaches

APPROACH	DESCRIPTION	INTEGRATION, REUSABILITY AND EFFORT ANALYSIS
A – Manual elaboration	Off-line data processing (e.g. manually elaborate spreadsheets…)	Level of integration: Low. Many manual tasks to be executed. Level of reusability: Medium. Implementation effort: Low. No development. Operations effort: Very High.
B – Business Intelligence tools	Support the analysis with an OLAP (On-Line Analytical Processing) tool.	Level of integration: High. But only if we're using a CMBD tool. Level of reusability: High. Implementation effort: High. Operations effort: Medium. Only for the report configuration step.
C – Visualization on CMDB tool	Visualise Reports, diagrams etc. offered directly from the selected CMDB tool (if this has been the choice for the previous Step 1).	Level of integration: High. But only if we're using a CMBD tool. Level of reusability: High. Implementation effort: N/A, native if using a CMDB tool. Operations effort: Low. Predefined report on the CMDB tool.

In consideration of the relevant effort to manage the analysis of significant volumes of data, the manual approach cannot be recommended, and approaches “B” or “C” should be preferred. The latter, in particular, may be particularly efficient if a CMDB tool has been chosen for the execution of Step 1 of the inventory process.

5. TOOLS SELECTION IN THE PILOT SCENARIO

5.1. Approach and selection criteria

This section describes the process through which the tools, needed to implement the recommended approaches for the Inventory process as described in the previous section, are selected, shortlisted and rated.

Such process passes through the following steps:

1 **Definition of Longlists -** For the execution of the activities of each layer of the architecture described in Section 3, and for the families of tools associated to the approaches described in Section 4, this study has respectively identified three longlists of potential candidate tools;
2 **Definition of Selection Criteria –** in order to filter the Longlists and identify candidate tools to be submitted to a detailed scoring and ranking, a list of Selection Criteria has been defined, allowing to identify the consistency and relevance of the tools in the Longlists within the "Pilot project" scenario (in terms of scope, constraints, customer's preferences etc.);
3 **Selection of Shortlists –** The application of the Selection Criteria to each Longlist determines a Shortlist of tools that have then been submitted to a subsequent evaluation through scoring and ranking.

Section 6 will then identify the scoring criteria to apply to the selected tools, compare and rank them based on the aforementioned criteria.

5.1.1. Selection criteria

The Selection Criteria (SC) listed below are based on requirements expressed in the tender, in the offer and in the initial assessment phase of the project.

- **SC1** - Only Open Source tools have been considered;
- **SC2** - Only locally deployable platforms have been considered;
- **SC3** - Only widely used tools (solutions with an high number of downloads in the last year) have been selected;
- **SC4 –** Only tools providing an API have been considered;
- **SC5** - (applicable to ETL tools only) **–** the selected tools must provide functionalities for the management of CSV and other text input files.

Selection Criteria are evaluated on a binary logic: the feature described by each criteria is either present or not, and only tools satisfying all of the four Selection Criteria have been included in the Shortlist.

5.2. Longlists of tools

For each inventory layer, the Longlist has been defined considering software tools analyses by Gartner Inc. and Forrester Inc.[1] (“magic quadrants” and “waves”), focused on the appropriate

[1] https://www.forrester.com/report/Vendor+Landscape+Software+Composition+Analysis/-/E-RES122796#figure4

https://www.gartner.com/doc/2980720?ref=SiteSearch&sthkw=Black%20Duck&fnl=search&srcId=1-3478922254

software segments (i.e. data management tools, business intelligence tools and software composition management tools).

The respective Longlists are:

1 **ETLs**: Informatica_PowerCenter, Analytics_Canvas, IBM-DataStage, Talend, ORACLE-Data_Integrator, Pentaho_Data_Integration (Kettle), Microsoft-SSIS, Jaspersoft_ETL, Clover ETL, Apatar, KNIME, OpenRefine, Rinho_ETL, SAS-Data_Integration_Studio, ORACLE-Warehouse_builder

2 **Asset management and DMBS tools:**
- **CMDB tools:** GLPI, OCS Inventory, Itop, CMDBuild, I-doIT, BMC Remedy, ServiceNOW ITSM
- **Relational database management systems**: IBM-DB2, Sybase-ADS, Apple-FileMaker, MariaDB_Community-MariaDB, Microsoft-SQL_Server, ORACLE-MySQL, ORACLE-RDB, SAP_HANA-SAP_AG, PGDG-PostgreSQL, Sybase-SQL_Anyware, Teradata-Teradata

3 **Business intelligence tools:** Eclipse BIRT, Pentaho-BI Suite, TACTIC, Splunk, ActiveReport, IBM-Cognos, Halo, Microsoft-SQL_Server_Reporting, ORACLE-Hyperion, SAP-NetWeaver, Sybase-Sybase_IQ, Zoho-Zoho_Report, RapidMiner, Jasperreport

5.3. Shortlists of tools

Applying the Selection Criteria on the Longlists, Shortlists for the three layers have been defined, as per paragraph 5.1.1. The resulting shortlist of rthe various

1. **ETLs**: Talend Open Studio, Pentaho_Data_Integration (Kettle), Clover ETL, Apatar, Jaspersoft_ETL
2. **Asset management tools:**
   - CMDB tools: GLPI, OCS Inventory, Itop, CMDBuild, I-doIT
3. **Business intelligence tools**: BIRT, Pentaho-BI Suite, Halo, RapidMiner, Jasperreport

For Relational Database Management Systems, please refer to paragraph 6.4.

6. TOOLS SCORING AND RANKING IN THE PILOT SCENARIO

Once defined the list of tools selected as per the previous section, Scoring / Ranking Criteria (SRC) are identified and applied to such list, in order to rate and rank the tools.

Scoring and Ranking Criteria are meant to measure the ability of the selected tools to meet the requirements analyzed in the assessment phase of the project, by assigning them appropriate weights.

6.1. Scoring / ranking criteria

SRC are listed in the following table, along with their respective references to the business requirements. Criteria may have three applicable values: Low (L), Medium (M), High (H), or just two (L and H), for Criteria responding to a binary logic.

Table 8 - Scoring and Ranking Criteria (SRC)

CRITERIA	EVALUATION	
SRC1 – Community Activity	Age of the project: Age < 2 years → L Age between 2 and 8 years → M Age > 8 years → H	Contributors (active contributors in the last year): Less than 100 contributors → L Between 100 and 1000 contributors → M More than 1000 contributors → H
SRC2 – Support	Support from few niche players → L Support from many International players → H	Support in extra-European countries → L Support from players with presence in Belgium & Luxembourg → H
SRC3 – Customizable data model (if applicable)	The measure in which the tool allows may adapt its default data model to the Target Data Model suggested in DLV1 Rigid / Unadaptable → L Flexible / Adaptable → H	
SRC4 – API	The use of more than one API model allows more adaptability of the tool to the context of use One API model used → L Two API models used → M Three or more API models used → H	
SRC5 - Security	Security (Known open vulnerabilities or defects / LOC): More than 3/1000 (or not published) → L Between 3/1000 and 1/10.000 → M Less than 1/10.000 → H	
SRC6 – Technology	Proprietary technologies → L Widespread technologies → H	Several heterogeneous technologies→ L Few homogeneous technologies → H

As shown in the table, some SRC are functions of couples of sub-criteria. The combined rating of the sub-criteria determines the rating of the corresponding SRC. The concerned SRC are:

A. Community Activity (SRC1) is function of two dimensions: the age of the community (in years) and the number of active contributors in the last year:

Table 9 - Community Activity sub-criteria and rating

Community activity		Age		
		≤2 y	>2 y and ≤8 y	>8 y
Contributors	≤100	Low	Low	Medium
	>100 and ≤1000	Low	Medium	High
	>1000	Medium	High	High

B. Support (SRC2) is function of two dimensions: local presence or not of support offices (which would lower the cost in case of need of interventions on spot) and nature of the support provider (small, niche players vs. international players):

Table 10 - Support sub-criteria and rating

Support		Presence of local / European support	
		Abroad	Local (BE & Lux)
Nature of player providing support	Niche	Low	Medium
	International	Medium	High

C. Finally, Technology (SRC6) is also function of two dimensions: number and homogeneity of the languages used (use of many languages for the programming of various parts of the tool vs. the use of homogeneous technologies / languages) and ownership of the technology (proprietary / low spread vs. open / widespread):

Table 11 - Technology sub-criteria and rating

Technologies		Number and homogeneity	
		Several and heterogeneous	Few and homogeneous
Ownership / spread of language / technology	Proprietary/ low spread	Low	Medium
	Open / widespread	Medium	High

The scoring is then calculated associating points to each parameter (**L = 1, M = 2, H =3**). All SRC are attributed equal weight in the scoring and ranking.

6.2. Scoring and ranking of ETL tools

For each ETL tool included in the Shortlist of paragraph 5.3, the respective features corresponding to each SRC have been identified as follows, based on the sources listed in Appendix 2:

Table 12 – Features of Shortlisted ETL tools

Criteria / Solution	Talend Open Studio	Pentaho[2] Data Integration (Kettle)	Apatar	KNIME	Jaspersoft[3] ETL
SRC1 - Community Activity	10 years 130 Contributors	10 years 120 Contributors	8 years 0 Contributors	5 years 25 Contributors	7 years Not Published
SRC2 - Support	• CGI (Benelux), • iAdvise, Progaja,	• CSC (Benelux) • KNOW.BI (Benelux)	• (none) • XeoKido (DE),	• (none) • Cloudera (GB),	• UNISYS (Benelux) • JSE (IRL), Column
SRC3 - Customizable data model	n.a.	n.a.	n.a.	n.a.	n.a.
SRC4 - API	REST, JAX-WS 2.2, JSR-224, SAAJ-SOAP, XA,	REST, Java	SOAP	Java, REST	REST, SOAP
SRC5 - Security	Avg. 0.6/10.000 (700K LOC)	Avg. 0.2/10.000 (1.5M LOC)	Not published (20K LOC)	Not Published (460K LOC)	Avg. 15/Not Published (Not Published)
SRC6 - Language / Technology	Java, XML	Java, XML, SQL, JavaScripts	Java, ActionScript, XML, HTML	Java, XML, HTML	Perl, Java, SQL

The scoring of the above features as per the scoring model described at the end of paragraph 6.1 provides the following results:

[2] Although recently acquired by Hitachi Data Group, it is still (and reportedly will be, according to the engagement taken by the new ownership) managed as Open Source

[3] Although recently acquired by TIBCO, it is still managed (and reportedly will be, according to the engagement taken by the new ownership) managed as Open Source

Table 13 - Scoring of shortlisted ETL tools

Criteria / Solution	Talend Open Studio		Pentaho Data Integration		Apatar		KNIME		JasperSoft ETL	
SRC1 - Community Activity	High	3	High	3	Low	1	Low	1	Low	1
SRC2 - Support	Medium	2	Medium	2	Low	1	Medium	2	High	3
SRC3 - Customizable data model	n.a.	0	n.a.	0	n.a.	0	n.a.	0	n.a.	0
SRC4 - API	High	3	Medium	2	Medium	2	Medium	2	High	3
SRC5 - Security	High	3	High	3	Low	1	Medium	2	Low	1
SRC6 - Language / Technology	High	3	High	3	Medium	2	High	3	High	3
TOTAL		**14**		**13**		**7**		**10**		**11**

As it can be noted, the scores of Pentaho Data Integration and Talend Open Studio are very close. In principle they are substantially interchangeable depending on DIGIT's preferences and policies. However, the point of advantage for Talend compared to Pentaho is essentially due to the fact that Talend's API set looks more complete and powerful than that of Pentaho.
While Pentaho is an absolutely "Java-oriented" platform, Talend appears to be more flexible to be integrated with web services, especially by the compliance with JSR-224/ JAX –WS standard, enabling the integration with XML Web services via Java API.
In addition, although the acquisition of Pentaho by Hitachi Data Group did not change the "open" nature of the product, its future road-map is not yet available.

In the table below the ranking of the ETL tools is provided along with a synthetic assessment of each of them.

Table 14 - Ranking of shortlisted ETL tools

Tool	Ranking	Comment
Talend Open Studio	**1**	A leader in this segment. Good performance on high volumes. High security, Widespread API
Pentaho Data Integration	**2**	A leader in this segment. Good performance on high volumes.
JasperSoft ETL	**3**	Very good performance on high volumes. Slow learning curve. Free software (source not available)
KNIME	**4**	Medium performance. Not very well documented. Slow learning curve. Niche support
Apatar	**5**	Not well documented; almost abandoned

6.3. Scoring and ranking of CMDB tools

The following tables compare the candidate integrated inventory platforms tools of the pertinent Shortlist based on the SRC.

As for the other families of tools, for each CMDB tool included in the Shortlist of paragraph 5.3, the respective features corresponding to each SRC have been identified as follows, based on the sources listed in Appendix 3:

Table 15 - Features of shortlisted CMDB tools

Criteria / Solution	GLPI	OCS Inventory	iTop	CMDBuild	i-doIT
SRC1 – Community Activity	12 years 7 contributors	10 years 1 contributor	7 years 4 contributors	4 years 14 contributors	10 years 1 contributor
SRC2 - Support	Infotel (France), IWS (Italy), Servicedesk (Brasil)	FactorFX (France)	Infotel (France), Itomig (Germany), qinet (Italy)	Tecnoteca (Italy)	Synetics (Germany)
SRC3 - Customizable data model	yes (via generic objects plugin)	No	Yes	Yes	n.a.
SRC4 - API	SOAP (via webservices plugin)	SOAP	Rest/Json	SOAP, REST	not documented
SRC5 - Security	11,3/10000 467k LOC	0,48/10000 516k LOC	0,21/10000 566k LOC	Not published, 1,05M LOC	Not published 263k LOC
SRC6 - Language / Technology	Php, javascript	C/C++, Java	Php, XML, Javascript	JavaScript, Java	Php, Javascript, XML

As all of those candidates rely on a MySQL/MariaDB DBMS, they can all be fed by an ETL tool such as Talend or Pentaho (see selection proposed in paragraph 6.2).
The scoring of the above features as per the scoring model described at the end of paragraph 6.1 provides the following results:

Table 16 - Scoring of shortlisted CMDB tools

Criteria / Solution	GLPI		OCS Inventory		iTop		CMDBuild		i-doIT	
SRC1 - Community Activity	Low	1	Low	1	Low	1	Low	1	Medium	2
SRC2 - Support	Medium	2	Low	1	Medium	2	Low	1	Low	1
SRC3 - Customizable data model	Medium)	2	Low	1	High	3	High	3	Low	1
SRC4 - API	High)	3	HIgh	3	High	3	High	3	Low	1
SRC5 - Security	Low	1	High	3	High	3	Medium[4]	2	Medium[5]	2
SRC6 - Language / Technology	High	3	Medium	2	High	3	High	3	High	3
TOTAL		**12**		**11**		**15**		**13**		**10**

In the table below the ranking of the CMDB tools is provided along with a synthetic assessment of each of them.

[4] Due to the scarcity of information available, this parameter has been tentatively rated at an average value

[5] Due to the scarcity of information available, this parameter has been tentatively rated at an average value

Table 17 - Ranking of shortlisted CMDB tools

Tool	Ranking	Comment
iTop	**1**	This CMDB tool provides the most interesting features (API, custom data model), has a recent release, and is well documented. Moreover, the company behind this tool has good references[6].
CMDBuild	**2**	This CMDB tool provides less features than iTop (only SOAP API), and the latest release is older.
GLPI	**3**	This inventory tool is well known in its area, and provides multiple plugins to enrich its feature basis. However, this is not a CMDB tool and thus it does not implement ITIL recommendations.
i-doIT	**4**	Even though this tool seems interesting, the community edition seems to be no longer maintained, in favor of the commercial one. Moreover, no documentation was found for this tool.
OCS Inventory	**5**	Poor documentation, poor customization available.

6.4. Relational databases

Due to the fact that there are no qualified or specific selection criteria/user requirements for Relational DataBase Management Systems (RDBMS) category, as they can now rather be considered as a commodity having lost differentiation among the various marketed solutions, this section of the Selection process will be agreed directly with the Customer before the implementation process based on its procurement policies and procedures.

[6] See Appendix 3

6.5. Scoring and ranking of Business Intelligence tools

For each Business Intelligence tool included in the Shortlist of paragraph 5.3, the respective features corresponding to each SRC have been identified as follows, based on the sources listed in Appendix 4:

Table 18 - Features of shortlisted Business Intelligence tools

Criteria / Solution	Eclipse BIRT	Pentaho[7] BI Suite	RapidMiner	JasperSoft[8] Report
SRC1 - Community Activity	11 years 15 Contributors	10 years 110 Contributors	12 years 3 Contributors	7 years Not Published
SRC2 - Support	• Eclipse Foundation Members	• CSC (Benelux) • KNOW.BI (Benelux)	• (none) • Cloudera (GB), DyMatrix (USA), Systek (DK), Avantgarde-Labs (DE), Basis06 (CH)	• UNISYS (Benelux) • JSE (IRL), Column Tech (USA), ProDato (DE)
SRC3 - Customizable data model	n.a.	n.a.	n.a.	n.a.
SRC4 - API	Java, REST	REST, Java	Java, REST	REST, SOAP, Java
SRC5 - Security	Avg. 0.1/10.000 (2.34M LOC)	Avg. 0.2/10.000 (1.3M LOC)	Avg. 2/10.000 (3.53M LOC)	Avg. 15/Not Published (Not Published)
SRC6 - Language / Technology	Java, XML, HTML, CSS	Java, XML	Java, XML	Java, XML

[7] Although recently acquired by Hitachi Data Group, it is still (and reportedly will be, according to the engagement taken by the new ownership) managed as Open Source

[8] Although recently acquired by TIBCO, it is still managed (and reportedly will be, according to the engagement taken by the new ownership) managed as Open Source

The scoring of the above features as per the scoring model described at the end of paragraph 6.1 provides the following results:

Table 19 - Scoring of shortlisted Business Intelligence tools

Criteria / Solution	Eclipse BIRT		Pentaho BI Suite		RapidMiner		JasperSoft Report	
SRC1 - Community Activity	Medium	2	High	3	Low	1	Low	1
SRC2 - Support	High	3	Medium	2	Medium	2	High	3
SRC3 - Customizable data model	n.a.	0	n.a.	0	n.a.	0	n.a.	0
SRC4 - API	Medium	2	Medium	2	Medium	2	High	3
SRC5 - Security	High	3	High	3	Medium	2	Low	1
SRC6 - Language / Technology	Medium	2	High	3	High	3	High	3
TOTAL		**12**		**13**		**10**		**11**

In the table below the ranking of the BI tools is provided along with a synthetic assessment of each of them.

Table 20 - Ranking of shortlisted Business Intelligence tools

Tool	Ranking	Comment
Pentaho BI Suite	**1**	A leader in this segment. Good performance on high data volumes.
Eclipse BIRT	**2**	A leader in this segment. Good performance on high data volumes. Limited scalability
Jasper Report	**3**	Limited community, Freesoftware (Source not available)
Rapid Miner	**4**	Not well documented. Limited Scalability.

6.6. Summary of tool ranking

The table below provides an overview of the ranking of the shortlisted tools for each layer (starting from the top one) of the Pilot Scenario inventory architecture:

Table 21 - Summary of tool ranking by layer

Layer	Recommended tools
Business Intelligence	1°: Pentaho BI Suite / 2°: Eclipse BIRT / 3°: Jasper Report / 4°: Rapid Miner
DBMS	1°: iTop / 2°: CMDBuild / 3°: GLPI / 4°: i-doIT / 5°: OCS Inventory
ETL	1°: Talend Open Studio / 2°: Pentaho Data Integration / 3°: Jasper Soft ETL / 4°: KNIME / 5: Apatar

No specific ranking is provided for RDBMS tools for the reasons mentioned in paragraph 6.4.

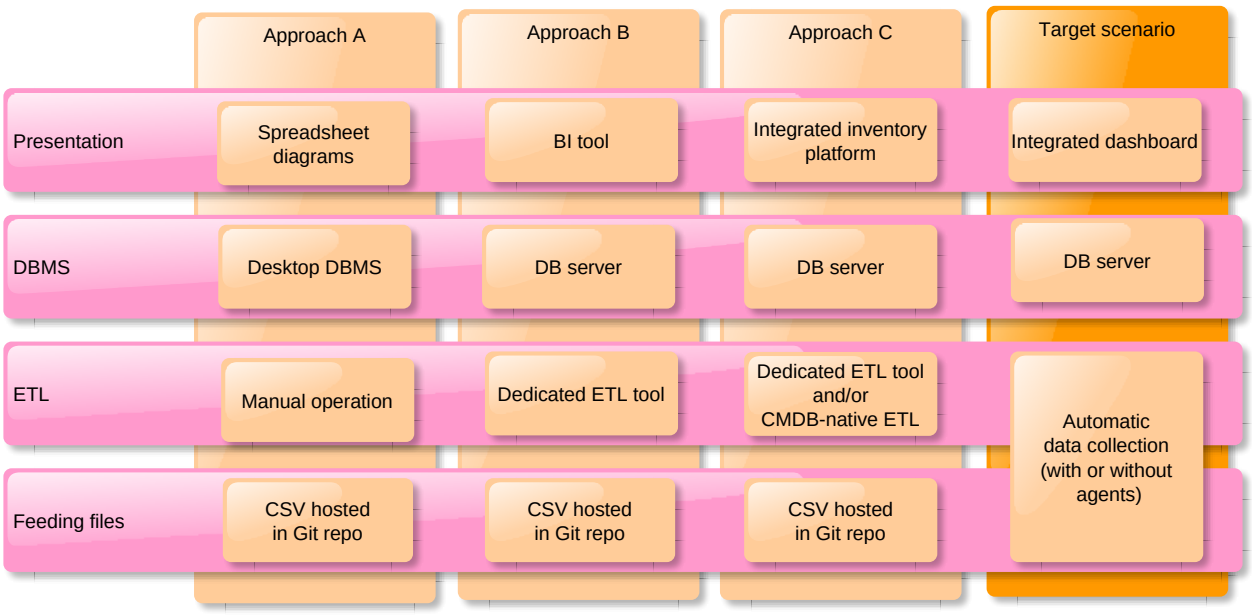
7. TARGET SCENARIO – OVERVIEW OF POSSIBLE ARCHITECTURE AND APPLICABLE TOOLS

7.1. Guidelines for the evolution towards a Target Scenario

Even though this will be the object of further analysis and recommendations at the end of the project, the foreseen Target Scenario may push further the integration, by collecting the data continuously from the agents running on the machines (Landesk, Satellite...) to build a realtime inventory.

In particular, recalling Figure 4, the following figure highlights how a Target Scenario may rely on the use of integrated tools for the collection of data on the Software Component Inventory, and on integrated dashboard for the filtering and ranking of relevant inventory data.

Figure 5 - Families of tools for the execution of software components inventory – possible evolution towards a target scenario



7.2. Overview and first assessment of integrated tools for software portfolio management

In the target scenario, the use of specific tools for the collection of metadata may be considered. Among such tools (none of which is Open Source), originally meant for software composition management and application security management purposes, but that also provide solid asset management and inventory features, there are, for example: Checkmarx, Citigal, Rogue_Wave, HP-Fortify, Covetry, IBM-Security, Pretorian, Sonatype, Security_Compass, Black_Duck_Software, Veracode, Trustwave, Virtual_Forge, Whitesource.

A first selection and evaluation, on the basis of the SRC, of a shortlist of possible tools to be used in this scenario provides the output shown in Table 21 below. It must be underlined, however, that the relatively scarce documentation of most of such tools, recently marketed, does not allow a proper scoring as for the tools listed in the sections above.

Table 22 - First scoring of integrated tools for software portfolio management

Criteria / Solution	Palamida		Rogue Wave		Sonatype		BlackDuck		Veracode		Whitesource	
SRC1 - Community Activity	n.a.	0	n.a.	0	n.a.	0	n.a.	0	n.a.	0	n.a.	0
SRC2 - Support cost	Low	1	Low	1	Medium	2	High	3	Medium	2	Low	1
SRC3 - Customizable data model	n.a.	0	n.a.	0	n.a.	0	n.a.	0	n.a.	0	n.a.	0
SRC4 - API	Rest / WS	2	Not published	0	Not published	0	Rest / Java / WS	2	Not published	0	None	0
SRC5 - Security	Not published	0	Not published	0	Not published	0	Not published	0	Not published	0	Not published	0
SRC6 - Language / Technology	Medium	2	Low	1	Medium	2	High	3	Low	1	Low	1
TOTAL		**5**		**2**		**4**		**8**		**3**		**2**

Based on the above comparison, the table below ranks the above tools, providing a synthetic comment on the rationale for the ranking.

Tool	Ranking	Comment
BlackDuck	**1**	Market leader in this area; fully integrated; oriented to Software portfolio governance
Palamida	**2**	Market leader in this area; fully integrated; oriented to Software portfolio governance
Sonatype	**3**	Market leader in this area; partially integrated; oriented to DevOps management
Veracode	**4**	Young product. Small set of users. Not well documented
Whitesource	**5**	Market leader in this area; partially integrated; oriented to DevOps management
Rogue Wave	**5**	Not well documented; oriented to Development management

8. APPENDIX 1 – RATIONALES FOR THE DEFINITION OF TDM ENTITIES FEEDING PATTERNS

- **Software**
  - **Available data sources:** list of installed software from EC CMDB systems
  - **Transformation / mapping:** names need to be normalised in order to match incoming metadata from external sources (e.g. from package name to project / software name)
  - **Update frequency:** low – new software is not introduced frequently in the inventory
  - **Data volume:** high –the OSS inventory includes the full scope of installed software
  - **Effort:** medium
  - **Data source type:** structured and easy to get – these data come from internal CMDB sources
  - **Feeding pattern:** automatic
  - **Extracting tools:** N/A
- **SoftwareVersion**
  - **Available data sources:** list of installed software from EC CMDB systems
  - **Transformation / mapping:** Versions need to be normalized in order to match incoming metadata from external sources (e.g. from package version to project / software version)
  - **Update frequency:** high – the versioning of inventoried software requires one instance per new version, this implies a high overall frequency
  - **Data volume:** high – high data volumes of inventoried software imply even higher data volume for software versions
  - **Effort:** high
  - **Data source type:** structured and easy to get – these data come from internal CMDB sources
  - **Feeding pattern:** automatic
  - **Extracting tools:** N/A
- **Standard**
  - **Available data sources:** DIGIT reference list of standards refreshed; Specialized sites (i.e. ISO, W3C, ANSI, OMG, etc.)
  - **Transformation / mapping:** Organized into a semantic tree
  - **Update frequency:** low – new standards are published with a low frequency
  - **Data volume:** low – the current standard scope hypothesis, based on the information currently available, is not wide (<50 entries), and there is no foreseeable reason to expect any significant growth in volume
  - **Effort:** low
  - **Data source type:** not structured – information on standards is taken from unstructured sources (mainly text documents)

- **Feeding pattern:** manual
- **Extracting tools:** N/A

- **System**
  - **Available data sources:** list of installed software from EC CMDB systems (if available)
  - **Transformation / mapping:** straight loading
  - **Update frequency:** low – new systems are not introduced frequently
  - **Data volume:** high – inventoried systems is high in number
  - **Effort:** medium
  - **Data source type:** structured and easy to get – these data come from internal CMDB sources
  - **Feeding pattern:** automatic
  - **Extracting tools:** N/A
- **Organisation**
  - **Available data sources:** list of HW producers from CMDB systems; organizations managing software from external metadata; EC organization managing systems from Org chart
  - **Transformation / mapping:** normalised data for HW producers; normalised data for software development entities; straight loading for EC management organisation documentation
  - **Update frequency:** low – new organisations are created with low frequency
  - **Data volume:** low – in-scope information on organisation is low, since we only map server managing entities, system producers and software maintainers
  - **Effort:** low
  - **Data source type:** not structured – no specific source exists for organisations, and data surcecs are heterogeneous
  - **Feeding pattern:** manual
  - **Extracting tools:** N/A
- **License**
  - **Available data sources:** specialised sites (i.e. OpenHub)
  - **Transformation / mapping:** none
  - **Update frequency:** low – new license types are not defined frequently
  - **Data volume:** low – in-scope details on licenses are few
  - **Effort:** low
  - **Data source type:** not structured – taken from heterogeneous sources
  - **Feeding pattern:** manual
  - **Extracting tools:** N/A
- **Vulnerability**
  - **Available data sources:** publicly available vulnerability sources (e.g. NVD)

- **Transformation / mapping:** conversion from source message/inventory
- **Update frequency:** high – updates on vulnerability definitions is high
- **Data volume:** high – the inventoried vulnerabilities are high in number: this implies a high data volume
- **Effort:** high
- **Data source type:** structured and difficult to get – updates must be parsed to get the needed data on vulnerability. Also, multiple organisation publish data on vulnerabilities, and this requires reconciliation of data
- **Feeding pattern:** by opportunity
- **Extracting tools:** integration with mail box

- **StandardCompliance**
  - **Available data sources:** specialised sites (e.g. OpenHub)
  - **Transformation / mapping:** mapping from specialised web sites
  - **Update frequency:** low – new standards are not defined frequently, and compliance to licenses rarely changes
  - **Data volume:** high – software is high in number, therefore compliance with standards is high too
  - **Effort:** medium
  - **Data source type:** structured and difficult to get, since information is usually stored either on structured web pages or other sources to be parsed with ad hoc tools, like packages list and dependencies graphs.
  - **Feeding pattern:** by opportunity
  - **Extracting tools:** web page scraping tool; partial coverage by standard type
- **LicenseCompliance**
  - **Available data sources:** specialised sites (e.g. OpenHub)
  - **Transformation / mapping:** mapping from software to list of standards
  - **Update frequency:** low – compliance to license rarely changes
  - **Data volume:** high– software is high in number, therefore compliance with licenses is high too
  - **Effort:** medium
  - **Data source type:** structured and difficult to get
  - **Feeding pattern:** by opportunity
  - **Extracting tools:** web page scraping tool
- **Criterion**
  - **Available data sources:** defined by the methodology
  - **Transformation / mapping:** none

- **Update frequency:** low – criteria are defined during this project by our team
- **Data volume:** low – criteria are not high in number
- **Effort:** low
- **Data source type:** not structured – the team manually defines criteria
- **Feeding pattern:** manual
- **Extracting tools:** N/A

- **SoftwareInstance**
  - **Available data sources:** list of installed software from EC CMDB systems
  - **Transformation / mapping:** mapping from SW version to list of systems
  - **Update frequency:** high – instances of inventoried software are higher in number than software, this means even higher update frequency
  - **Data volume:** high – instances of inventoried software are higher in number than software, this means even higher data volume
  - **Effort:** high
  - **Data source type:** structured and easy to get – this data is coming from internal CMDB sources
  - **Feeding pattern:** automatic
  - **Extracting tools:** N/A
- **SoftwareCriteria**
  - **Available data sources:** list of installed software from EC CMDB systems
  - **Transformation / mapping:** mapping from SW version to list of systems
  - **Update frequency:** low – criteria are applied only to the software shortlist (< 20 entities)
  - **Data volume:** low – criteria are applied only to the software shortlist (< 20 entities)
  - **Effort:** low
  - **Data source type:** not structured – criteria are assigned manually
  - **Feeding pattern:** manual
  - **Extracting tools:** N/A
- **SoftwareVulnerabilities**
  - **Available data sources:** Publicly available vulnerability sources (i.e.NVD)
  - **Transformation / mapping:** none
  - **Update frequency:** high – updates on vulnerability definitions is high
  - **Data volume:** high – the inventoried vulnerabilities are high in number: this implies a high data volume
  - **Effort:** high

- **Data source type:** structured and difficult to get – updates must be parsed to get the needed data on vulnerability. Also, multiple organisation publish data on vulnerabilities, and this requires reconciliation of data
- **Feeding pattern:** by opportunity
- **Extracting tools:** integration with mail box + ETL tool

- **Dependencies**
  - **Available data sources:** package dependencies from software distributions
  - **Transformation / mapping:** mapping from packages version to software versions
  - **Update frequency:** low – software dependencies rarely change
  - **Data volume:** high – dependencies are very high in number, as software is high in number too; hence, data volume is high
  - **Effort:** medium
  - **Data source type:** structured and difficult to get – dependencies must
  - **Feeding pattern:** by opportunity
  - **Extracting tools:** parsers for package dependencies

9. APPENDIX 2 – SOURCES FOR THE IDENTIFICATION OF ETL TOOLS FEATURES

https://www.gartner.com/doc/3102119/magic-quadrant-data-integration-tools
https://www.openhub.net/p?ref=homepage&query=talend; https://nvd.nist.gov/
https://www.talendforge.org/; https://jira.talendforge.org/browse/TDP
https://www.talend.com/products/specifications-application-integration
https://www.talend.com/partners/find-a-partner
http://community.pentaho.com/user-groups/
http://jira.pentaho.com/browse/PDI/?selectedTab=com.atlassian.jira.jira-projects-plugin:summary-panel
http://doc.cloveretl.com/documentation/UserGuide/index.jsp?topic=/com.cloveretl.server.docs/docs/osgi.html
http://apatar.com/roadmap.html
http://apatar.com/partner_directory.html
http://www.enterprisedb.com/
https://www.wrike.com/
https://www.knime.org/
http://anterio.com/index.php?id=112&L=1
http://www.dri-nordic.com/
http://www.sistek.com.tr/tr/
http://www.infocom.co.jp/english/aboutus/index.html
http://community.jaspersoft.com/project/jaspersoft-etl
http://www.jaspersoft.com/sites/default/files/assets/jaspersoft_etl_datasheet_-_eng.pdf;
http://community.jaspersoft.com/download http://community.jaspersoft.com/wiki/getting-started-rest-web-service-api; http://www.jaspersoft.com/partners

10. APPENDIX 3 – SOURCES FOR THE IDENTIFICATION OF CMDB TOOLS FEATURES

http://glpi-project.org/spip.php
http://www.ocsinventory-ng.org/
http://www.combodo.com/itop
http://www.cmdbuild.org
http://www.i-doit.org
www.openhub.net

11. APPENDIX 4 – SOURCES FOR THE IDENTIFICATION OF BI TOOLS FEATURES

http://jira.pentaho.com/browse/PDI/?selectedTab=com.atlassian.jira.jira-projects-plugin:summary-panel
http://doc.cloveretl.com/documentation/UserGuide/index.jsp?topic=/com.cloveretl.server.docs/docs/osgi.html
http://apatar.com/roadmap.html
https://eclipse.org/
http://www.eclipse.org/birt/documentation/integrating/reapi.php
https://rapidminer.com/
http://www.avantgarde-labs.de/
http://clairvoyantlab.com/
http://community.jaspersoft.com/project/jaspersoft-etl
http://www.jaspersoft.com/sites/default/files/assets/jaspersoft_etl_datasheet_-_eng.pdf;
http://community.jaspersoft.com/download http://community.jaspersoft.com/wiki/getting-started-rest-web-service-api; http://www.jaspersoft.com/partners

12. APPENDIX 5 – ABBREVIATIONS AND ACRONYMS

FOSSA	Free and Open Source Software Auditing
WP	Work Package
DLV	Deliverable
TDM	Target Data Model
csv	Comma-Separated Values (file format)
CMDB	Configuration Management Data Base
OLAP	On-Line Analytical Processing
DMBS	Data Base Management System
ETL	Extract, Transform, Load
API	Application Programming Interface
OSS	Open Source Software
RDBMS	Relational Data Base Management System
SC	Selection Criteria
SRC	Scoring and Ranking Criteria