European Commission

DIGIT
Unit B1

Deliverable n. 1
Open Source Software Inventory Methodology
FOSSA WP3

Date:	04/04/2016
Doc. Version:	Final

TABLE OF CONTENTS

TABLE OF FIGURES

TABLE OF TABLES

1. Deliverable Overview

1.1. Executive Summary and Purpose

The aim of the present document is to describe a replicable methodology for building and maintaining an inventory of free and open source software and open standards. Moreover, the proposed approach is based on the consolidation of existing processes and tools.

Section 2 of this document shortly describes what is meant by ”Open Source Software”, how the information relevant for this work has been collected, the overall distinction between the pilot approach to be applied in the short term and a longer term approach, the requirements and the criteria set for the inventory methodology.

Section 3 outlines the information sources, collected as per the previous section, used for the development of the inventory methodology sources identified during the workshops with the DIGIT and the EP; and how they are grouped to provide a consistent view.

Section 4 describes the limitations set to the present study, either due to the lack of sufficient information at the issue date of the present release of the document (v 1.6) or because of non-applicability of the methodology to some parts of the European Commission / European Parliament ecosystems. In particular, at the above-mentioned issue date, only insufficient information was available on the ecosystem of the European Parliament. Consequently, the applicability of this study on such ecosystem will be further assessed at a later stage on the basis of the additional information gathered, unless otherwise specified in the present document.

Section 5 describes the data structure of the forecasted inventory, according to the available information and the customer’s requirements.

Section 6 defines a replicable methodology to perform periodic inter-institutional inventories of software assets and standards, starting from a preliminary assessment of the existing processes, tools and repository/inventories and a collection of requirements. This section aims at proposing an inventory methodology that shall be applicable in the short run with limited impact on the current practices of the Institutions. Such aim shall be achieved by:

- leveraging on the existing inventory management processes
- taking into account the complexity of the ecosystems of the European Commission and the European Parliament, and the scarcity of information related to some parts of such ecosystems at the moment of the analysis.

Section 7 highlights some longer-term scenarios and proposes additional improvements that may be implemented to further enhance the effectiveness of the inventory process in the future.

2. Approach

To thoroughly understand this work, it is necessary to make some initial clarifications on:

- the type / types of software this project targets;
- the distance between this "pilot" project and a "target" scenario;
- the differences between the processes and the IT solutions supporting the "target" scenario and the "pilot" scenario.

2.1. Open source software

Open-source software (OSS) is computer software with its source code made available with a license in which the copyright holder provides the rights to study, change, and distribute the software to anyone and for any purpose.

Therefore, according to the global objectives of the FOSSA pilot project, the development approach of this methodology focuses on software components used by the European Commission and the European Parliament for which source code is readily available under an appropriate license.

To ensure the appropriate management of open-source software, the governance process has to include:

- Choice - Proactive choice of secure and supported open source;
- Inventory - Maintenance of an accurate list of open source components throughout the system/software development lifecycle;
- Identification - Identification of vulnerabilities during the development;
- Remediation/Contribution - Fast remediation in presence of new vulnerabilities on the "in production" OSS.

2.2. Target scenario and pilot scenario

The approach of this methodology is to set the basis for a future improvement of the European Institutions' practice along two directions: the security of the internal information systems and the possible contribution to European open source communities.

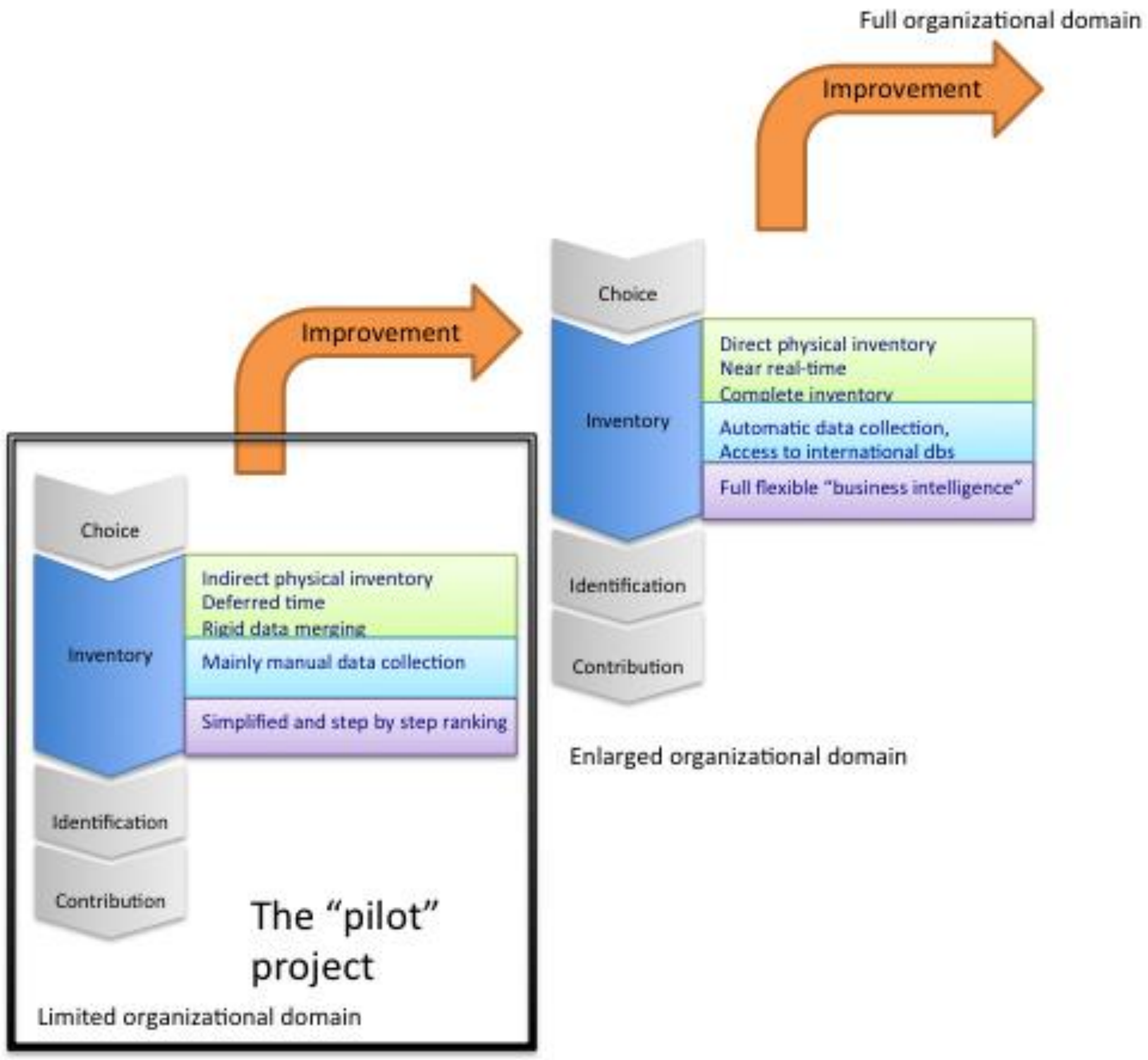
In addition, the methodology developed in this pilot project is designed to be used both in this first experience (pilot scenario) and in with the improvements tested in the pilot scenario, in a future larger scale context (target scenario).

The three core aspects whose maturity can change between the pilot experience and the recurring process are:

- Software components data collection: Processes, tools and techniques to collect the component inventory and the performance then can provide in terms of speed and accuracy;
- Metadata collection: Processes, tools and techniques to collect “sustainability” data (about communities behind the used OSSs), “vulnerability data” (about the known defects of the software) and “business criticality” data (about the relevance of software into the DIGIT environment);
- Filtering and ranking: Processes, tools and techniques to interactively filter/prioritize the inventory based on a set of criteria/thresholds.

The picture below summarizes the progressive maturity improvements of the core aspects:

Figure 1- Evolution of inventory process from pilot to target scenario


Full organizational domain
Improvement
Choice
Inventory
Direct physical inventory
Near real-time
Complete inventory
Automatic data collection,
Access to international dbs
Full flexible “business intelligence”
Identification
Contribution
Enlarged organizational domain
Improvement
Choice
Inventory
Indirect physical inventory
Deferred time
Rigid data merging
Mainly manual data collection
Simplified and step by step ranking
Identification
Contribution
The “pilot” project
Limited organizational domain

The target scenario is described in paragraph 7.2, “The target scenario – first step”.

On the other side, this pilot project implements only some significant actions, performed in a non-industrialised way, on a reduced scale or with a simplified IT solution. In this regard, the overall features of the pilot processes/solution may be as follows:

1. A simplified version of DIGIT-C2 CMDB scheme enriched with new attributes needed to identify OSS components and OSS products (hierarchy);
2. Installation and population of the above scheme with one-shot extractions from existing systems: a manual / semi-automatic loading of data (the aim is to identify the ETL rules); the input format of extraction files is defined by the project manager – in this pilot scenario – and by the inventory manager – in the target scenario. The data-owner organization is responsible to provide the data in the required format.
3. Ranking criteria design and dashboard implementation;
4. OSS ranking and selection of candidates OSS for code review;
5. “Pilot” inquiry of large internet knowledge bases (allowed by temporary free/low cost licenses[1]) to collect additional metadata;
6. “Pilot” list of standards;
7. Standard-to-Product mapping of candidate OSS.

2.3. Data collection

The data collection will cover the following high level scope: software installed at the datacentre(s), software installed on PCs (desktops and laptops) and mobile devices. This scope was based on the available sources of information identified during the interviews of the different responsible stakeholders identified by DIGIT.

During the analysis performed in this pilot project, a significant amount of information has been collected from a wide range of organisational units in DIGIT to address the European Commission’s requirements. However, this was not equally possible on the European Parliament’s side due to organisational difficulties in the data collection process, so that a significantly smaller amount of information was made available on its ecosystem.

The set of information collected during the interview process includes:

- Organisational unit to which the interviewees belong;
- General information on the activities performed by the organisational unit;
- Processes performed and role;
- Possible data sources candidate for the inventory;
- Possible attention points.

1 Not needing call for tender or heavy acquisition procedures and included in the total cost of the project.

2.4. Requirement collection

The requirements below have been drawn:

- based on the FOSSA Project Charter (task description, success criteria and user needs), as confirmed in WP3 Quality Framework document;
- from the interviews to the stakeholders as a deduction made upon the shortcomings of the current process described by the stakeholders.

Table 1 - Requirements

REQUIREMENT	PRIORITY	SOURCE
The inventory process shall be preferably based on existing customer's processes	1	FOSSA Project Charter
The inventory process shall allow efficient analysis of installed software notwithstanding the quantity of data to be browsed in order to identify OSS	1	Interviews with stakeholders
The inventory process shall target automated collection of data	1	FOSSA Project Charter
The specific requirements that must accomplish the methodologies and tools to be used in the creation of the inventory of assets and standards shall be proposed.	1	FOSSA Project Charter, Key success factor 2
The methodology to obtain the software inventory shall take into account libraries, versions and dependencies between components.	1	FOSSA Project Charter, Key success factor 3
The methodology to obtain the software inventory shall let European Commission and European Parliament categorize the components by several criteria: criticality, existing support, areas where the components are used.	1	FOSSA Project Charter, Key success factor 4
Each component shall be accompanied with information necessary to assess its sustainability, according to the metrics defined in Work Package 1 of FOSSA. Some examples known at the time of writing of this Technical Annex: development process, automatic regression testing, vulnerability reporting process, size of team supporting the project, financing of the team.	1	FOSSA Project Charter, Key success factor 4

REQUIREMENT	PRIORITY	SOURCE
Extended information about each OSS component shall be obtained (e.g. community that is behind, type of licence).	1	FOSSA Project Charter, Key success factor 7
The list of free and open technical specifications and open standards used by European Institutions is elaborated.	1	FOSSA Project Charter, Key success factor 9
The inventory methodology and execution shall be developed along the following main dimensions: 1.Physical (see Offer) 2.Logical (see Offer) 3.Organisational: (see Offer and Quality Management Framework) 4.Legal (see Offer – definitions of open source software and open standards)	1	Contractor's Offer Project Quality Management Framework
The initiative will deliver a replicable methodology for building and maintaining an inventory of free and open source software and open standards. Moreover, the proposed approach is based on the consolidation of existing processes and tools.	1	Contractor's Offer
The methodology will be replicable and easily extendable to the other stakeholders, outside of this initiative.	1	Contractor's Offer
The deliverable presents a definition of a target data model	1	Contractor's Offer
FOSSA needs to contribute towards assessing current security of OSS components used at the European Commission and the European Parliament	1	FOSSA Project Charter, User needs
FOSSA needs to propose tools and procedures to assess the risk of new components and applications…	1	Contractor's Offer
The inventory process shall identify housed software installed by users possessing admin rights	2	Interviews with stakeholders
The process shall ensure the tracking of releases of the inventoried software	3	Interviews with stakeholders
The inventory process shall be repeatable according to a standardised sequence of activities	1	Interviews with stakeholders

REQUIREMENT	PRIORITY	SOURCE
The inventory process shall run at regular intervals of time	2	Interviews with stakeholders
Authorised users shall be able to run the inventory process upon request	1	Interviews with stakeholders
The inventory process shall avoid double inputs and needs for data reconciliation	1	Interviews with stakeholders

Priority ranks from "1" (high) to "3" (low)

2.5. Criteria definition

The core aspects of this pilot project are:

- The collection of data about the software present and in use in the IT infrastructure of the European Commission and Parliament;
- The selection (from the large set of present software) of the best candidate for a code review (Work Package 6 of the FOSSA project).

For the second point, three categories of criteria are defined in this project:

- "sustainability" criteria (applied to open source communities and projects);
- "vulnerability" criteria (applied to software components and applications);
- "business criticality" criteria (applied to software components and applications).

Sustainability criteria rank communities and projects through a set of metrics based on those defined at this aim by WP1 of the FOSSA pilot project, appropriately shared and reviewed. Vulnerability and business criticality criteria are specifically defined in this WP.

As introduced in the section on data collection, to reach the goal of this pilot project, it is reasonable to have two levels of filtering on the software inventory:

- A first level based on quantitative criteria:
  - The number of known vulnerabilities;
  - The internal occurrences;
  - The volume;
  - The external occurrence;
- A second level based on qualitative criteria (expressed as an high/medium/low value):
  - Contribution to the ranking from the "sustainability" criteria[2];
  - License type (openness);
  - Scalability;
  - Technical adaptability (compatibility);
  - Security;

[2] If the examined software is produced/owned by a project/community with a high ranking ("highly sustainable"), then the software is also ranked high in terms of eligibility for the code review.

- Roadmap;
- Documentation;
- Compliance with standards (checking or unchecking branches and leaves in a tree representation of standards).

Quantitative criteria can be applied for an initial restriction of the assets to be inventoried, while qualitative criteria can be applied to rank them and finally identify the best candidates for the inventory. The application of quantitative criteria can also help reducing the amount of computational cost.

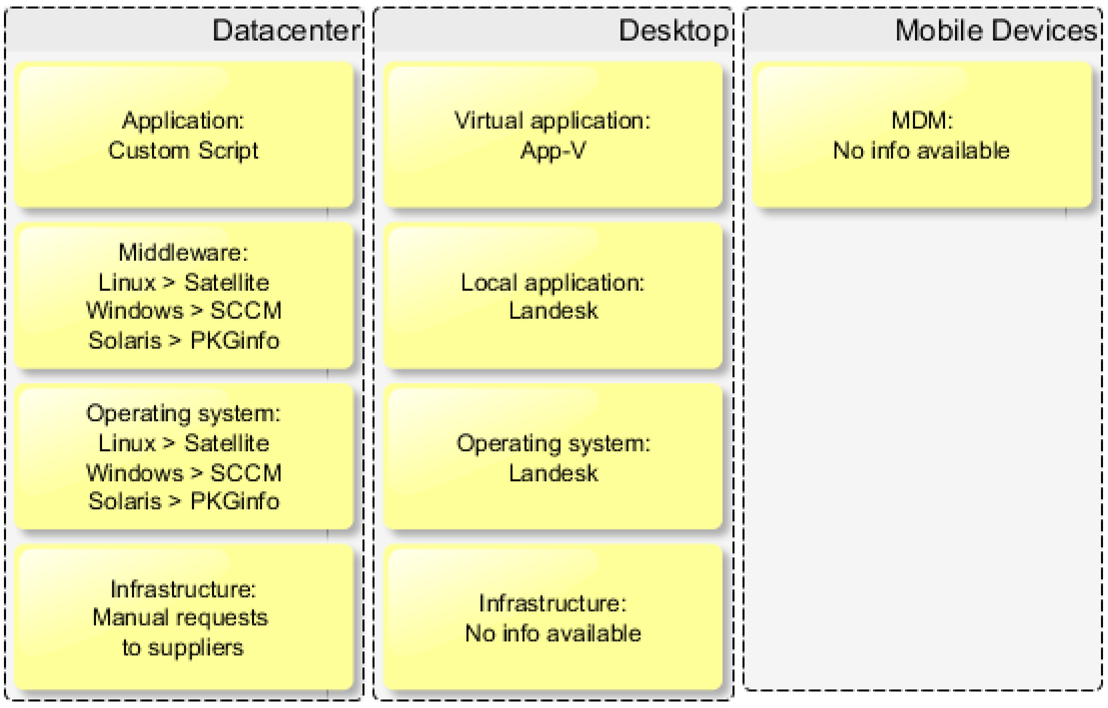
3. AVAILABLE INFORMATION SOURCES

Some generic sources of information used in the EC / EP will not be relevant in the scope of this study.

As an example of this, ABAC / ABAC Asset, the financial ordering system & financial asset tool used throughout the EC, has a business rule stating that contracts and assets below €420 are not mandatorily recorded in the tool. Based on that, the OSS references will most probably be very limited or inexistent, and thus the input not really useful for our mission.

During the interview phase, different information sources were identified. They are schematically represented as follows:

Figure 2 – Information sources (European Commission)


Datacenter
Application: Custom Script
Middleware: Linux > Satellite Windows > SCCM Solaris > PKGinfo
Operating system: Linux > Satellite Windows > SCCM Solaris > PKGinfo
Infrastructure: Manual requests to suppliers
Desktop
Virtual application: App-V
Local application: Landesk
Operating system: Landesk
Infrastructure: No info available
Mobile Devices
MDM: No info available

A list of the information sources with the form under which they have been collected (.csv / .xls files, .doc files, .htm files etc.) is provided in Annex 1.

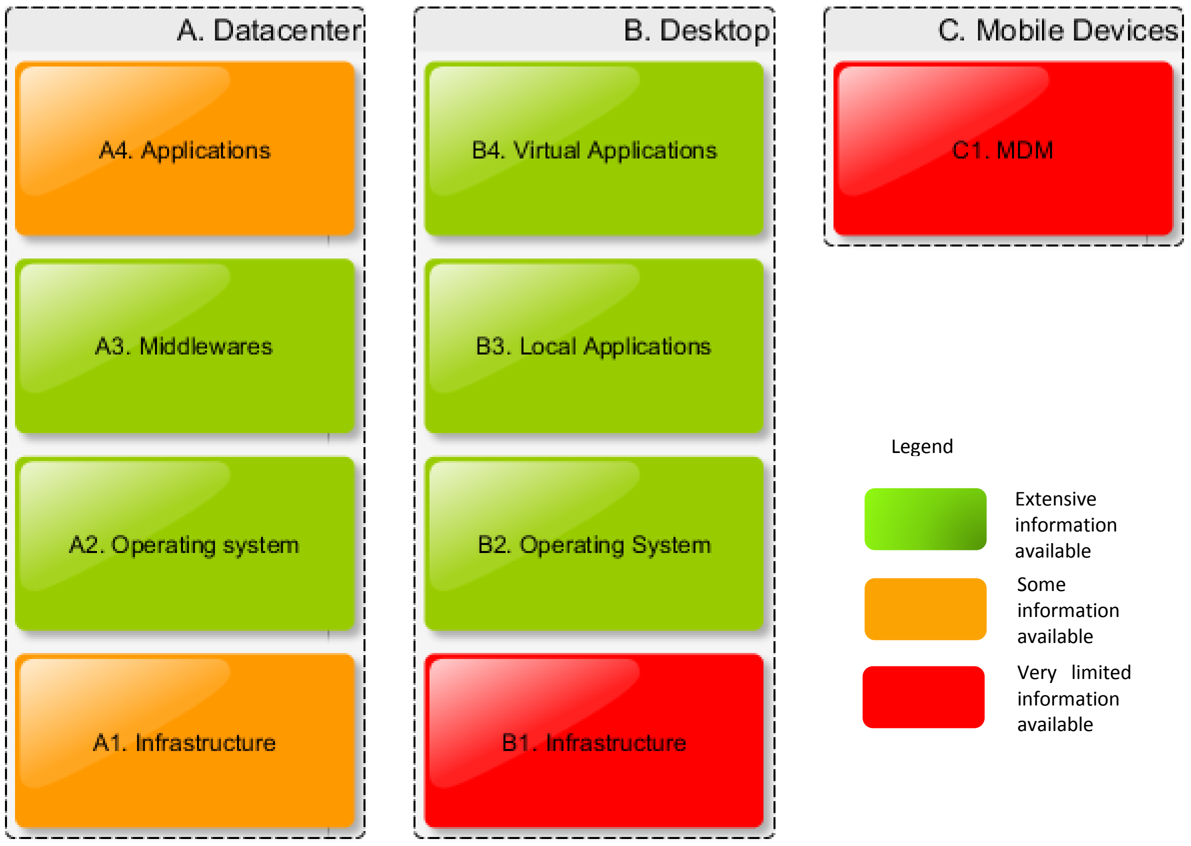
A detailed view of the currently available inventory exports from DIGIT is addressed in section 6.1 Data collection and transformation.

4. Limitations

In the interviews conducted during the first phase of this project, we observed that some areas of the initial scope of the study will be limited due to not only the lack of access to systems, but also the lack of available information, particularly concerning the European Parliament ecosystem. Some of these limitations and constraints may also be caused by organisational or security European Commission/Parliament restrictions.

The high level scheme presented below outlines the main areas of the study, where the different colours represent the different levels of detail achieved for each domain.

Figure 3 – Level of detail of available sources (European Commission)



In the methodology chapter, each domain will be further detailed with the different sources of information available.

This chapter will focus on limitations and constraints established during the study.

4.1. A - Datacentre

DIGIT Datacentre team does not directly control all machines under its responsibility (for example, DIGIT B uses physical / virtual machines not entirely controlled by the DIGIT Datacentre team). Due to the lack of information on the machines (physical or virtual) out of such control, such machines will not fall in the scope of the present study.

4.1.1. A4- Applications

The applications (hosted or housed) running on the servers present in the DIGIT Datacentre are mostly not controlled by DIGIT. DIGIT handles the requests to make available to the users a specific environment (Infra/OS/middleware), but has no specific rights to consolidate and manage the applications running over these environments.

In order to build and consolidate such inventory, custom scripts may be developed to identify the applications and the specific libraries installed on these servers, at least for hosted servers. This solution will be further detailed in Section 6.

At a first stage, a simple script may explore recursively some of the usual standard installation paths to build an initial inventory. At a later stage, the standard installation paths shall be defined.

4.1.2. A1- Infrastructure

This layer groups all the possible open source software embedded inside physical devices such as routers, load balancers, SANs, switches, firewalls…

To build an OSS inventory for such devices, manual requests will need to be addressed to manufacturers of these devices. In order to optimize the timeframe, only a shortlist of main devices and appliances will be subject to these manual requests.

4.2. B - Desktop

Only Standard workstations & laptops provided by DIGIT will be considered here. The BYOD will remain out-of-scope. Similarly, some specific workstations are also excluded as OLAF (Anti-Fraud Office) and JRC (Joint Research Centre).

The list of orders for approved software is stored in the ABAC database, but it is not in an exploitable state, as it is composed of scanned orders in landscape view.

4.2.1. B1 - Desktop infrastructure

In the scope of OSS study, no relevant information can be provided even if some infrastructure information is available through LanDesk inventory tool.

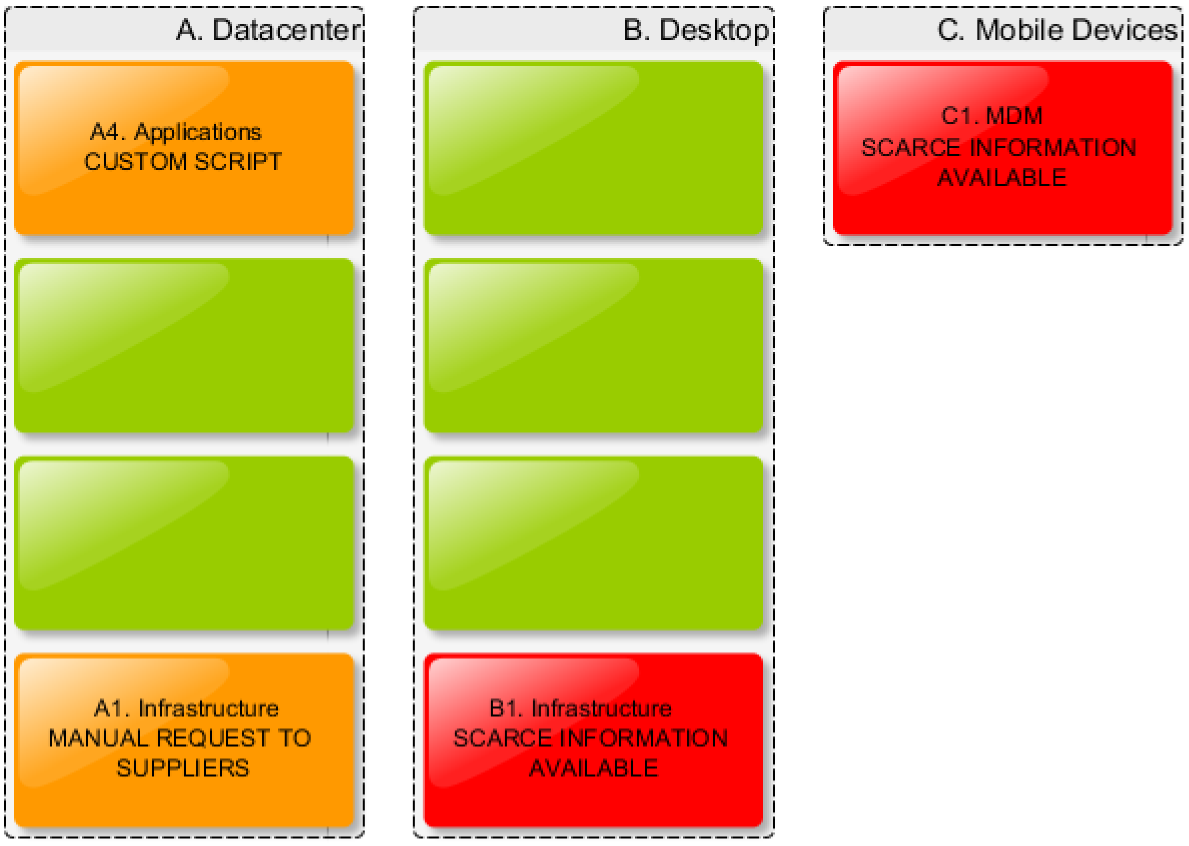
4.3. C - Mobile Devices

On mobile devices under provided by DIGIT, only the “MobileIron” agent is installed through MDM channel. This platform, in the configuration purchased by DIGIT, does not include any OSS software. No inventory tool is currently implemented/activated.

As DIGIT does not manage the installed Apps on Mobile devices, this domain will temporarily remain out-of-scope.

The figure below summarizes the approach adopted to manage the limitations to the various areas mentioned above:

Figure 4 – High-level approach to manage limitations (European Commission)



5. Target data model definition

This chapter details the Target Data Model that has been defined for this project phase.

The Target Data Model is a conceptual, object-oriented model. Therefore it is technology-independent and is not intended to be an image of the database the inventory tool will use. Hence, entities are not mapped in one-on-one to database tables.

The model has been defined on the provided data basis, to encompass all the information that has been collected so far. It can be further extended to include any relevant information that was not available or assessed as relevant in this project phase.

The model describes:

- **Entities**: coherent aggregates of information, related to real-world objects, ideas or contexts, which are commonly stored into database tables;
- **Attributes**: simple pieces of information (text, numbers, lists, etc.) belonging to an Entity, which are commonly stored into database tables' columns;
- **Relationships**: connections that represent some kind of hierarchy or interaction between entities.

Each entity has the following properties:

- **Name**: a sequence of words that identifies the entity;
- **Description**: a short phrase that explains the role and information content of the entity;
- **Requirements**: a list of the project requirements that led to the definition of the entity;
- **Sources**: a list of the information sources from which the entities' information are gathered (e.g. Landesk, App-V, Satellite);
- **Type**: if the Entity is a specialisation of another entity, the value is "Dependent"; else, the value is "Independent".

Attributes are organised by Entities. Each Attribute has the following properties:

- **Name**: a sequence of words that identifies the Attribute;
- **Definition**: a short phrase that explains the role and information content of the Attribute;
- **Required:** is the field required or not**;**
- **Is PK**: tells if the Attribute is used to identify the entity it belongs to;
- **Is FK**: tells if the Attribute references an external entity.

The model is built around the *Software* and *System* core entities. *Software* aggregates all the information required to perform the software inventory, software attributes and meta-data, while *System* contains the information related to the systems, physical or virtual, where the software is deployed. The data for the *Software* entity are manually and locally managed by inventory managers, while the ones that belong to the *System* entity are automatically loaded from external systems (Landesk, App-V, Satellite and other CMDBs). A third Entity, *SoftwareInstance*, represents the software that has been actually deployed, and works as a bridge between the two.

The *Software* Entity is related to versions and licenses. Each software version is tied to its evaluation criteria, which are evaluated to assess if the software must be included in the Critical Software

Shortlist. The software classes that were declared as in-scope in the requirements are also modelled as specialisations of the *Software* entity.

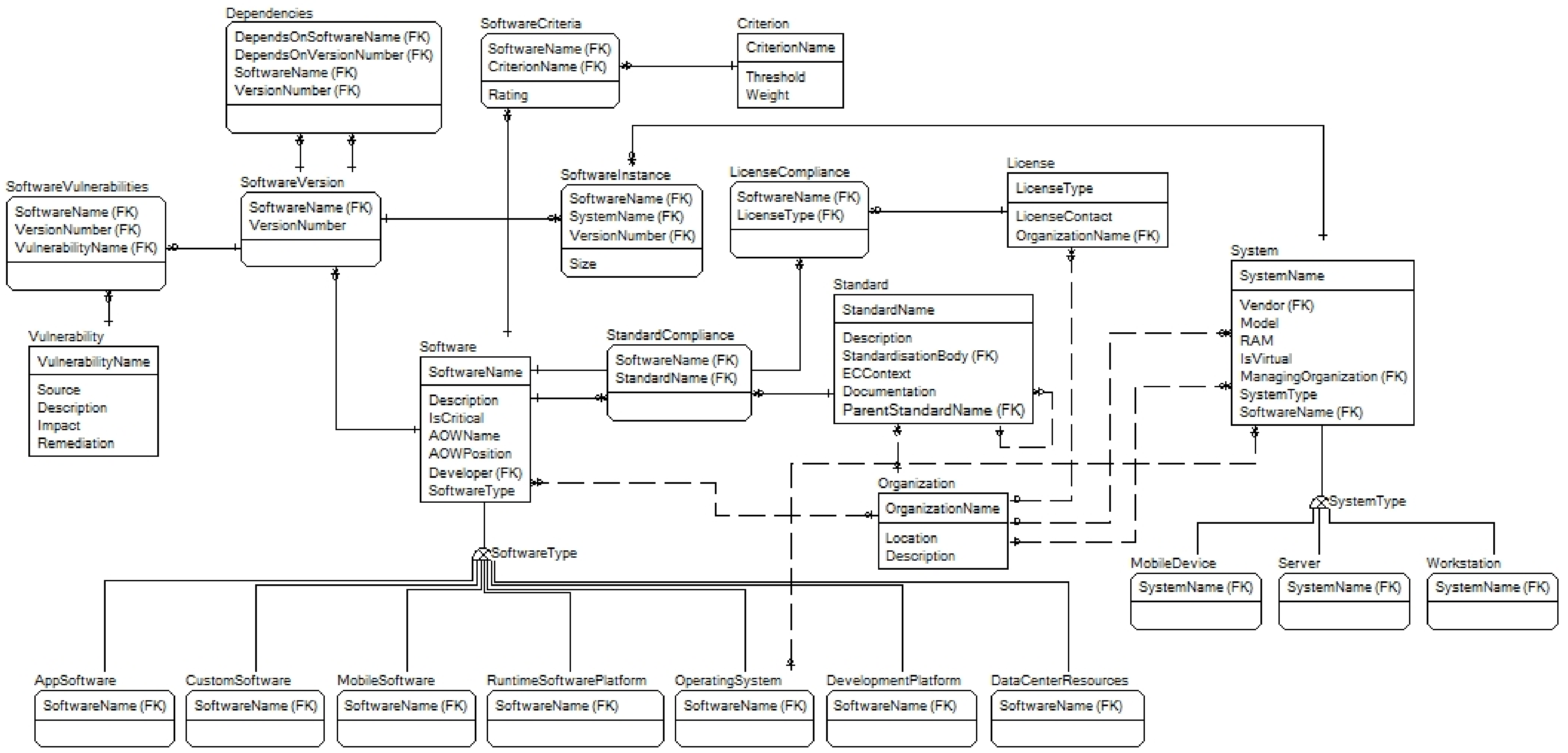
The *System* Entity is subdivided into workstations, servers and mobile devices. The first two system types are in-scope, while the last one is currently under evaluation. It will be excluded from the Data Model if definitively assessed as out-of-scope.

Both *Software* and *System* are related to the standards they comply with. As the standard inventory is a project requirement, the *Standard* Entity contains all the information gathered from the information sources, and can be considered as a third core entity.

Organisations that own or produce software, standards and systems are also related to the three main Entities and have been modelled.

Details about project requirements have been mapped onto the Entities that answer to those requirements. The same operation has been performed for the data sources that have been currently identified as available.

Figure 5 - Target Data Model diagram



More detailed information about the model is provided in Annex 4, where each entity is associated to the pertinent data sources and to the requirements listed in section 2.4. Requirements are shown also for the attributes of each entity. Some attributes may not relate to specific requirements, but to generally accepted best practices (e.g. the presence of the entity name within the attribute of each name), in which case no specific requirement is indicated.

6. Methodology

6.1. Data collection and transformation

6.1.1. *Inventory of software*

The data collection will be a “pull” process starting with a periodic reminder (for example an e-mail) to the interested counterparties (the European Commission and European Parliament units owning the relevant data) sent by the process owner, or inventory manager (to be properly identified and appointed). The reminder message shall indicate a due date and a set of instructions for operators on how to execute the data extraction and allocation into the repository.

Once the inventory manager has received confirmation from all data providers, (s)he will start the ETL sub-process to populate the inventory database.

As for the Data Centre, the hypothesis underlying this pilot project is that all the collected data are about known software. This means that all items treated in the inventory must have previously been recognised as software components or software products bearing some brand name (including in-house codes) and that can be associated with an external manufacturer (or an organisational unit) or with a community. As for desktops, on the other hand, it is expected that the full list of installed software is made available for the inventory.

When the inventory database is populated, the inventory manager can launch the ranking-dashboard to manually adjust the ranking criteria based on a first set of quantitative criteria (possibly excluding some criteria and/or fixing thresholds) and to interactively select the most relevant set of software applications/components.

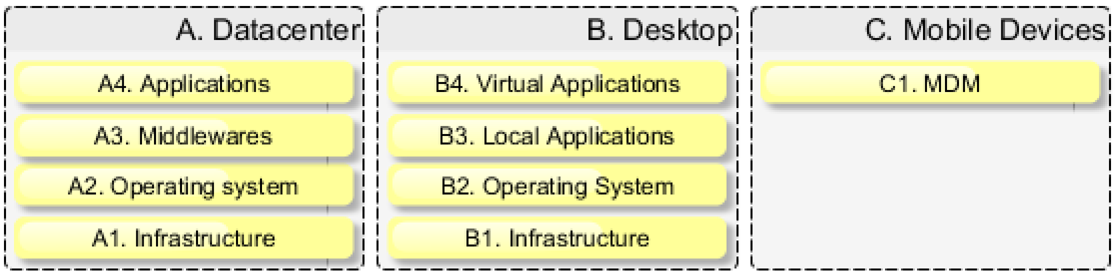
Finally, the set of selected software (component) can be enriched with metadata such as licensing type, known vulnerabilities etc. and prepared for the final ranking.

The “unknown” software, i.e. the software not having been associated to a community, to an organisational unit of the EU Institutions or to another identifiable manufacturer, is the first candidate to inspection but is out of the scope of this pilot project (see above and section 7.2, “Next Steps”). However, some additional processing can be done on the software recognised as open source to decide how to contribute to the OSS communities.

One of the use-cases of the inventory could be to get a shortlist of critical software components, by applying criteria to the inventory items in order to rank the by criticality. The final ranking is performed by the inventory manager, using the dashboard again, adjusting the previous ranking based on a second set of qualitative criteria.

At the end, the final ranking identifies the software to be submitted to code review as a priority.

As seen in Chapter 4, the data collection covers the following high level scope:



The sources of the inventory will therefore cover three major areas: datacentres, desktops and mobile devices.

In the next paragraphs, this figure will be further detailed with the quality of the coverage for each area, indicated by the colours used to represent it:

🟩 Extensive information available

🟧 Some information available

🟥 Very limited information available

6.1.2. *Inventory of standards*

As a product of this pilot project, the inventory of standards is produced integrating the existing list of standards (dating back to 2011) provided by DIGIT and enriching it with publicly available information (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, updated versions or possible replacement standards will be looked for on relevant external data sources, thus obtaining an updated list;
- 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 properly identified in the updated list, it will be looked for in the main libraries of Standards (ISO standards, WSC etc.) to get a full description and identification thereof;
- Once defined the shortlist of critical 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);
- In some cases, standard compliance (e.g. file format support) may be identified by analysing the dependencies.

The inventory will be proposed as a browsable semantic tree according to the tools available by the EuroVoc project (http://eurovoc.europa.eu/drupal/?q=en).

In a target scenario, the inventory of standards may be maintained on a regular basis possibly using an internet-based semantic search engine, in order to select additional standards and to add them to the inventory.

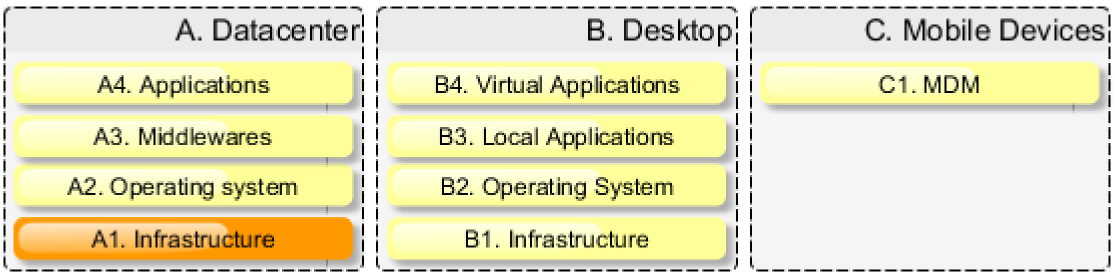
6.1.3. A - Datacenter resources

6.1.3.1. A1 - Infrastructure

This layer groups all the possible open source software embedded inside physical devices such as routers, load balancers, SANs, switches, firewalls…

Currently there are no inventories of the software components (firmware) of those devices.

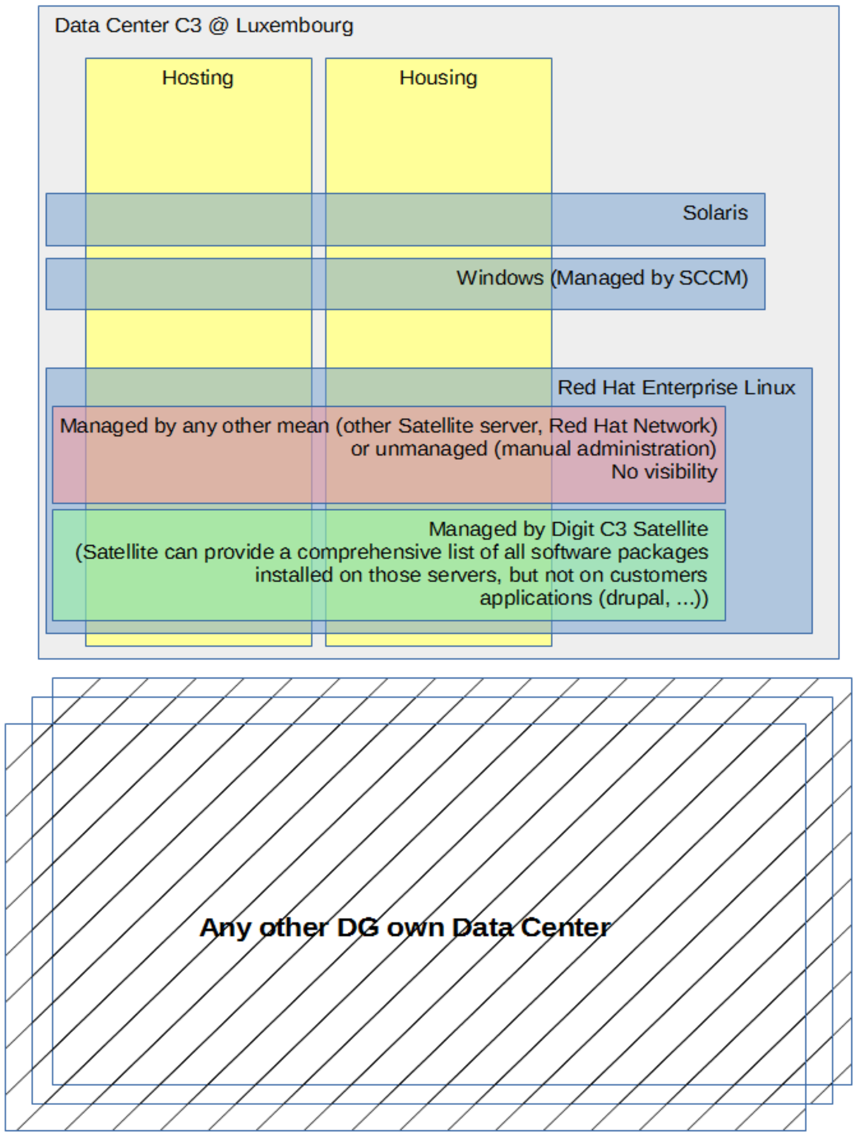
The recommended methodology is to start such an inventory from the list of devices and to contact the vendors in order to get information about the software they run. As this is a long and manual process, it is suggested to perform it based on a very limited set of devices (2 or 3). Even if the output of such a limited sample won't be exploitable as is, the benefit will be that the structure and the process of collecting the information will be in place, and the exercise could be continued later on.



6.1.3.2. A2 - Operating systems

The following picture describes the situation of the operating systems managed by DIGIT.

Figure 6 – Outline of DIGIT-operating systems


Data Center C3 @ Luxembourg
Hosting
Housing
Solaris
Windows (Managed by SCCM)
Red Hat Enterprise Linux
Managed by any other mean (other Satellite server, Red Hat Network) or unmanaged (manual administration) No visibility
Managed by Digit C3 Satellite (Satellite can provide a comprehensive list of all software packages installed on those servers, but not on customers applications (drupal, ...))
Any other DG own Data Center

DIGIT C3 manages a datacentre in Luxemburg. This datacentre provides hosting and housing services. Among the servers, either in the housed or hosted part, three major operating systems are supported: Red Hat Enterprise Linux, Solaris and Windows:

- Windows servers are managed by Microsoft System Centre Configuration Manager (SCCM);
- Solaris servers are manually managed by the team (i.e. no centralised configuration tool used);
- Linux servers are either managed by the Red Hat Satellite tool from DIGIT C3 (green box in the figure), or are managed by any other means (pink box on the figure), such as:
  - By another Satellite server operated by the customer;
  - Directly connected to the Red Hat Network;
  - Or unmanaged (manual administration).

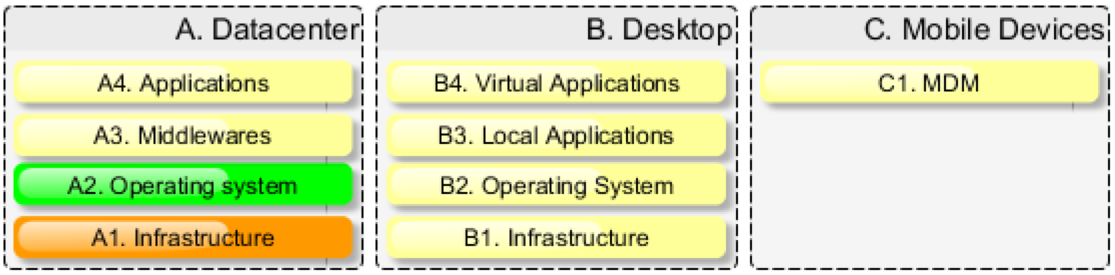
Additionally, other Directorates General also manage their own infrastructure (represented with the hatched boxes).

DIGIT has no visibility on the servers represented by the pink and hatched boxes in Figure 8, as they are not under its control. For these reasons, this methodology will focus on the other areas:

- Windows systems, expected to run little to no open source software, from SCCM exports;
- Solaris systems, from manual export (pkginfo command);
- Linux systems managed by DIGIT C3 Satellite server, from the following commands: spacewalk-report inventory and spacewalk-report system-packages-installed. The latter command outputs the list of all packages, and of their versions, installed on all the systems managed by the satellite server. This includes the libraries installed on the systems.

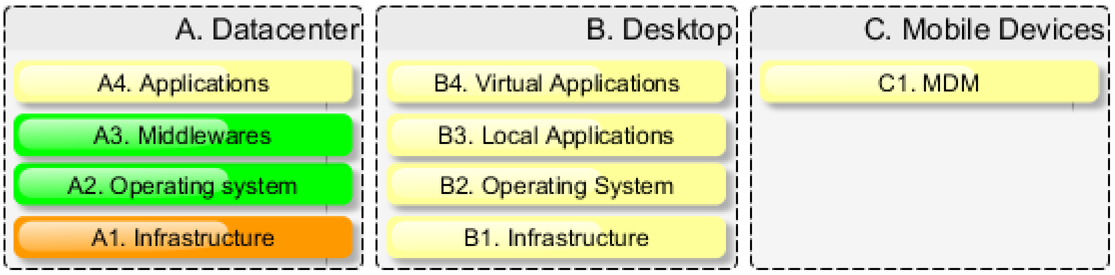
However, only the software installed using the respective software management tools from each OS will be collected (i.e. package manager for Linux and Solaris, and Add/Remove software for Windows). This means that any application added to the system through any other way will not be reported through these methods. This can include:

- Source code compiled on the system;
- Executable copied on the system;
- Applications downloaded from a git/svn repository;
- Webapps for Apache, Tomcat, Weblogic, etc. provided by the users.



6.1.3.3. A3 - Middleware

The middleware layer includes the application servers or database servers. As this software is installed through the usual package manager of the distribution, the scope and limitations of the previous section 6.1.3.2, “Operating systems”, apply to the present section as well.



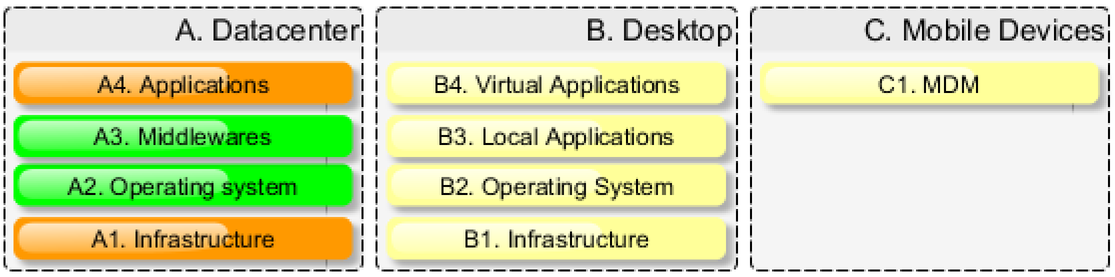
6.1.3.4. A4 - Applications

The applications, in Figure 6, are the software hosted by the application server (Tomcat, Weblogic, Apache and Coldfusion). Those applications are provided by the users, and DIGIT has no visibility on them. No inventory currently exists listing the various applications the application servers run. Thus, the only way to keep this layer in the scope cannot be, as for the other layers, to rely on existing tools or inventories, but to develop a script that shall discover the applications inside the application servers.

Based on information gathered from DIGIT C2 technical teams on the standard configuration of various application server types, the script will establish a list of files, looking in specific paths (/var/lib/tomcat…). The collected information may include the file name, the libraries, the version…

However, it is acknowledged that:

- The configuration of application servers may vary from one to another, thus the script may not see the webapp files if they are stored in a non-standard path;
- The quality of the script result may not provide the requested information on the application (licence type, version, etc.). This will be clarified at the early stages of the testing of the script.

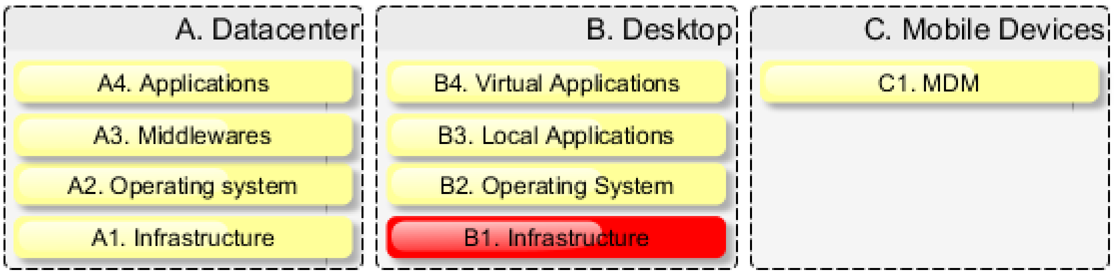


6.1.4. <u>B - Desktop</u>

This section covers the workstation and laptop software.

6.1.4.1. B1 - Infrastructure

In this section, “infrastructure” includes landline phones, printers, copiers, video conferencing devices and similar items. The firmware of those devices has not been listed and no inventory is currently available to rely on, in order to select the open source components. For this reason, this layer is not covered by the methodology.



6.1.4.2. B2 & B3 - Operating systems & local applications

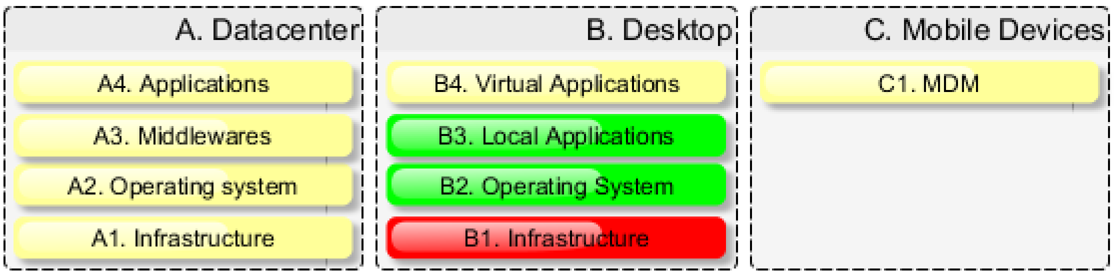
The information on the operating systems and local applications installed on the workstations is managed by Landesk, a tool operated by DIGIT A2.

In the case of typical workstation users not having administrative rights on his computer, there is no risk that a software component not managed by Landesk be installed on the machines.

However, roughly 10% of users do have administrative rights, and so, can install any software on their machine. If they do so, Landesk will discover it and it will appear in a daily report.

Should the admin user decide to disable Landesk on his computer, the system would be automatically banned from Active Directory.

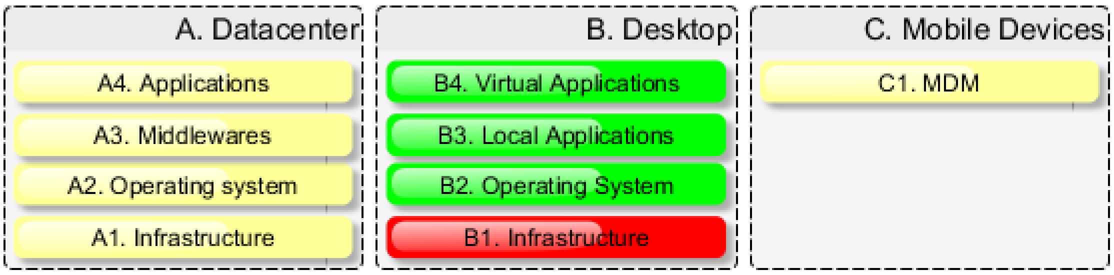
For all those reasons, Landesk is considered a reliable source of information on all the applications installed on the workstations managed by the DIGIT.



6.1.4.3. B4 - Virtual Applications

Besides the local applications installed on the workstations, DIGIT A2 also provides virtual applications through the Microsoft App-V technology.

The App-V service already can export the catalogue of virtual applications and their usage.

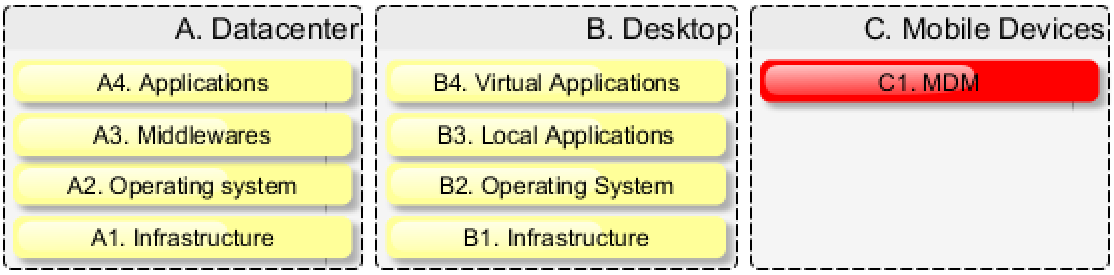


6.1.5. C - Mobile devices

6.1.5.1. C1 - MDM

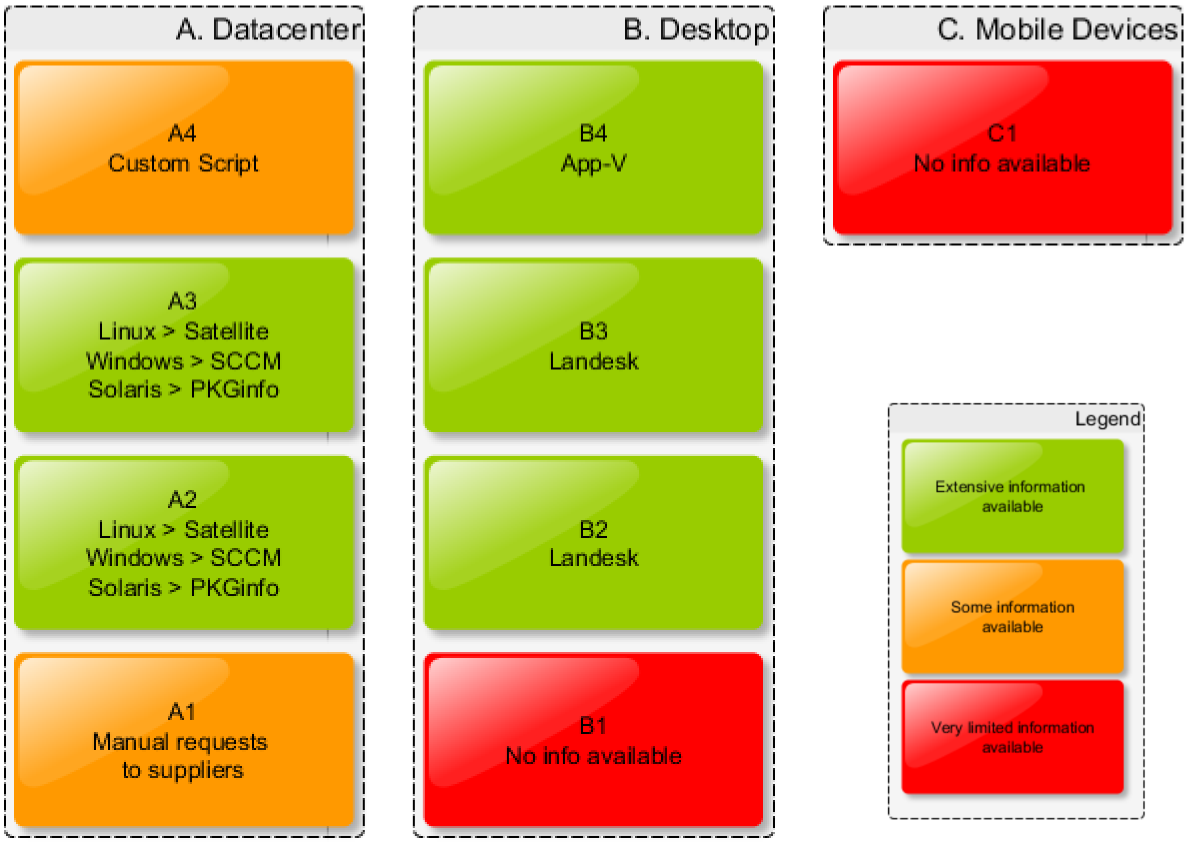
The mobile devices are managed by the MDM system. However, the MDM tool cannot collect all the applications installed the mobile devices. Hence, there is no current inventory, nor any current tool in place that would build such an inventory of open source mobile device applications. Moreover, as far as the MDM security layer is concerned (for instance, securing e-mail application), and from the customer's understanding, no substantial open source software is installed.

Eventually, even if the methodology described in the present chapter could very well cover the mobile devices, such devices will remain out of scope in the pilot scenario due to the lack of information available at the issue of this release of the document.



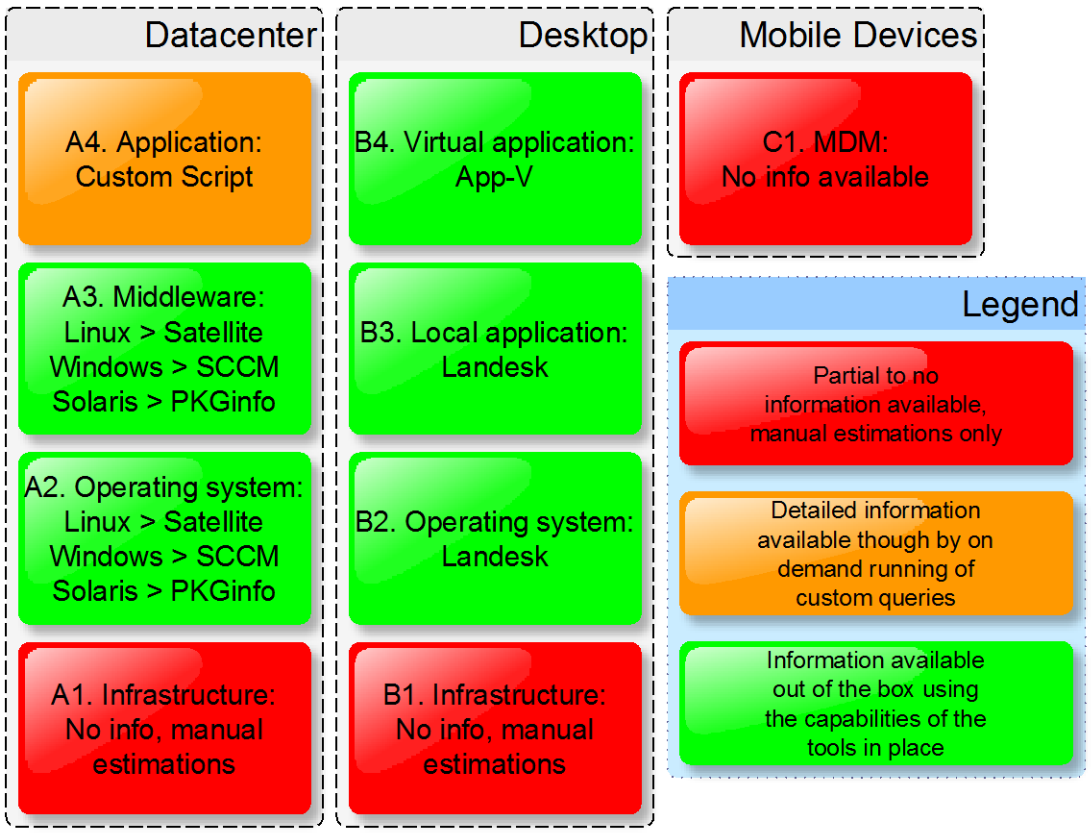
Finally, based on the various sources of information that will be used to build the inventory, the general figure can now be instantiated as follows.

Figure 7 - Coverage of inventory with information sources (EU Commission)


A. Datacenter
A4 Custom Script
A3 Linux > Satellite Windows > SCCM Solaris > PKGinfo
A2 Linux > Satellite Windows > SCCM Solaris > PKGinfo
A1 Manual requests to suppliers
B. Desktop
B4 App-V
B3 Landesk
B2 Landesk
B1 No info available
C. Mobile Devices
C1 No info available
Legend
Extensive information available
Some information available
Very limited information available

Another way to qualify the information sources is to rate to what extent the information can be accessed. The following figure gives an overview of this situation.

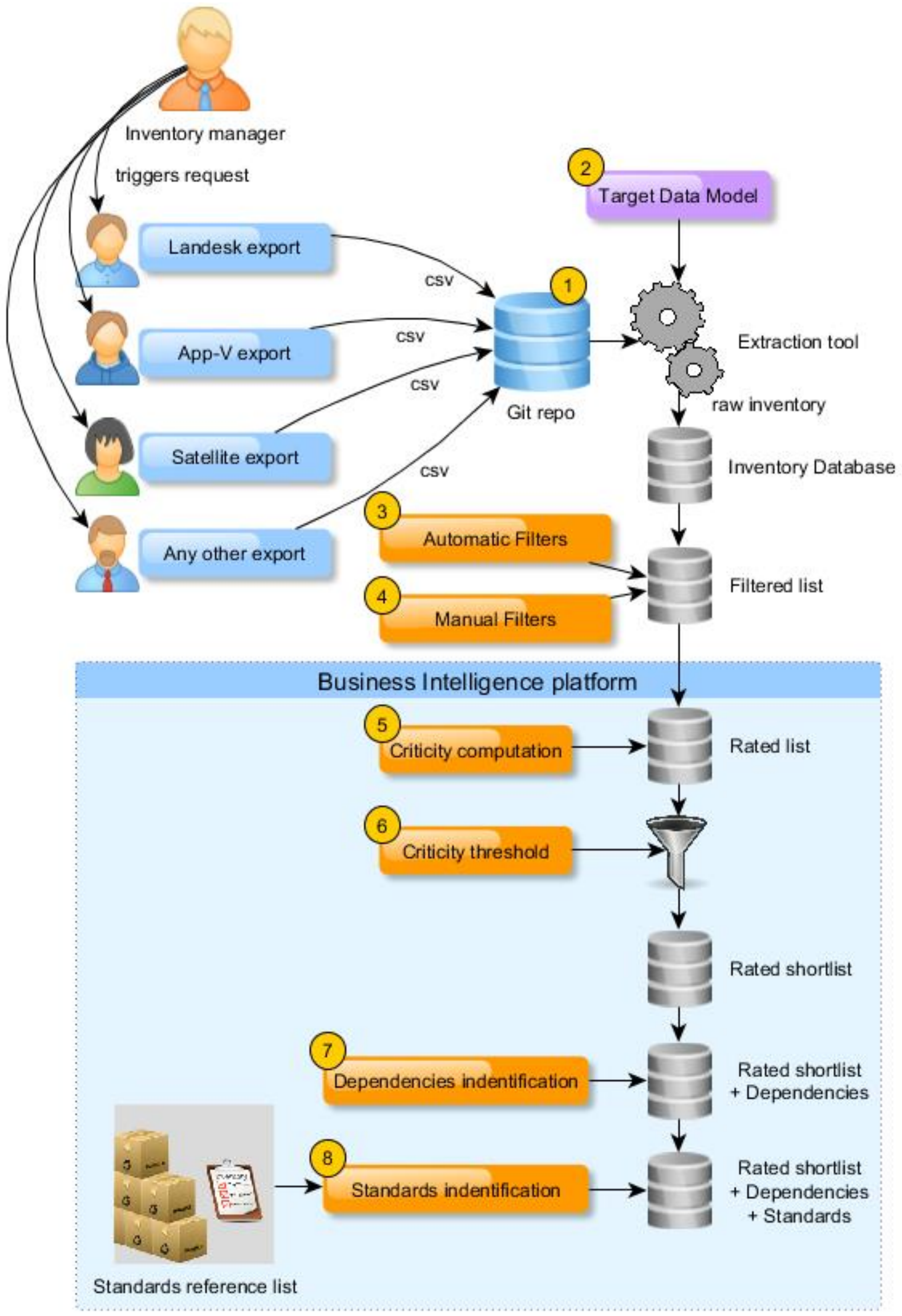
Figure 8 – Readiness of the information sources (EU Commission)


Datacenter
A4. Application: Custom Script
A3. Middleware: Linux > Satellite Windows > SCCM Solaris > PKGinfo
A2. Operating system: Linux > Satellite Windows > SCCM Solaris > PKGinfo
A1. Infrastructure: No info, manual estimations
Desktop
B4. Virtual application: App-V
B3. Local application: Landesk
B2. Operating system: Landesk
B1. Infrastructure: No info, manual estimations
Mobile Devices
C1. MDM: No info available
Legend
Partial to no information available, manual estimations only
Detailed information available though by on demand running of custom queries
Information available out of the box using the capabilities of the tools in place

6.2. Pilot inventory process

The process to collect the information and build the inventory may be summarised as follows.

Figure 9 – Pilot process overview



(1) [Manual step] The inventory manager triggers a request to the various owners of the respective sub-inventory systems to push their export in CSV format in a Git repository. Guidelines about how to perform such exports must have been documented beforehand.

(2) [Manual step] After the Target Data Model is designed to define the structure under which the inventory will be stored, a routine is developed to process the raw data and generate the initial Inventory Database. This step is in a different colour since it is only performed on the first iteration of the inventory process.

Note: the Extraction tool is an automated processing service that will convert the data from the various CSV export into a format that will populate the Inventory Database. This Extraction tool will be composed of a main process, and various plugins for each type of export to handle.

(3) [Automatic step] The raw inventory is filtered according to several automatic filters (known OSS list, amount of installation instances…). Such filters can be applied to the list of items to be inventoried via an automated routine. The filtering criteria are applied in a specific sequence in order to optimize the filtering process.

[RAW info] ➔ [Auto OSS filter] ➔ [Auto SLA filter] ➔ [Auto #instances filter] ➔ …(4)

(4) [Manual step] After this first automatic filtering, a manual filtering is applied on the remaining result set. Among the manual filters are the SLA type, the Specific OSS list, etc.

(3) … ➔ [OSS manual filter] ➔ […]

Note: at the time of writing, some export samples from various information sources are still missing, which prevents from giving an accurate list of the various filters that will be applied.

The inventory can be used for several purposes. At this stage, the inventory is built and usable. However, our consortium would like to provide an added value with extra processing of the data using a business intelligence platform. The steps below are optional.

(5) [Automatic step] A use case of this inventory is to get a visual overview of the critical software components. The inventory items could be ranked by criticality with the application of weighted criteria. This ranking is called the Global Criticality Index.

(6) At this stage, a validation meeting is organised with DIGIT stakeholders to validate the results and define the threshold above which the OSS will be further analysed.

(7) [Manual step] The critical software shortlist resulting from step (6) is submitted to an in depth analysis to identify the dependencies among software components. Due to the variety of software in the critical shortlist (either running on Linux, Windows, Solaris, on a server or a workstation, installed through a package manager or manually deployed, …), no generic automatic method can be implemented to identify the dependencies, and this will be a case-by-case operation.

(8) [Manual step] The critical software shortlist is subject to the identification of the standards they implement, based on a reference list of standards. This reference list will be consolidated out of various sources (customer's existing material, standardisation bodies…).

As the level of details available for each item of this shortlist might vary, the identification of standards will be a manual operation.

From steps (5) to (8), filtering, applying criteria, weighting and rating operations will take place into a business intelligence tool that will provide extensive features in that area. This will allow easy screening and sorting of the items.

6.3. Interoperability & extensibility

This methodology, although originally designed on the basis of the information available on the ecosystem of the European Commission, is also meant to be applied to different organisations, including the European Parliament (on which no relevant detailed information has been collected at the issue date of the present document).

Extending the methodology to the European Parliament, or any other additional organisations, would require supplying the data into the Git repository (step 1) and implementing a new plugin in the Extraction tool that would process this further set of data.

7. Recommendations and Next Steps

7.1. Recommendations

Based on the first analysis performed in the framework of the present study, some recommendations are hereby provided on future actions that the European Commission and the European Parliament may implement to enhance the efficiency and effectiveness of the inventory process defined in this pilot project:

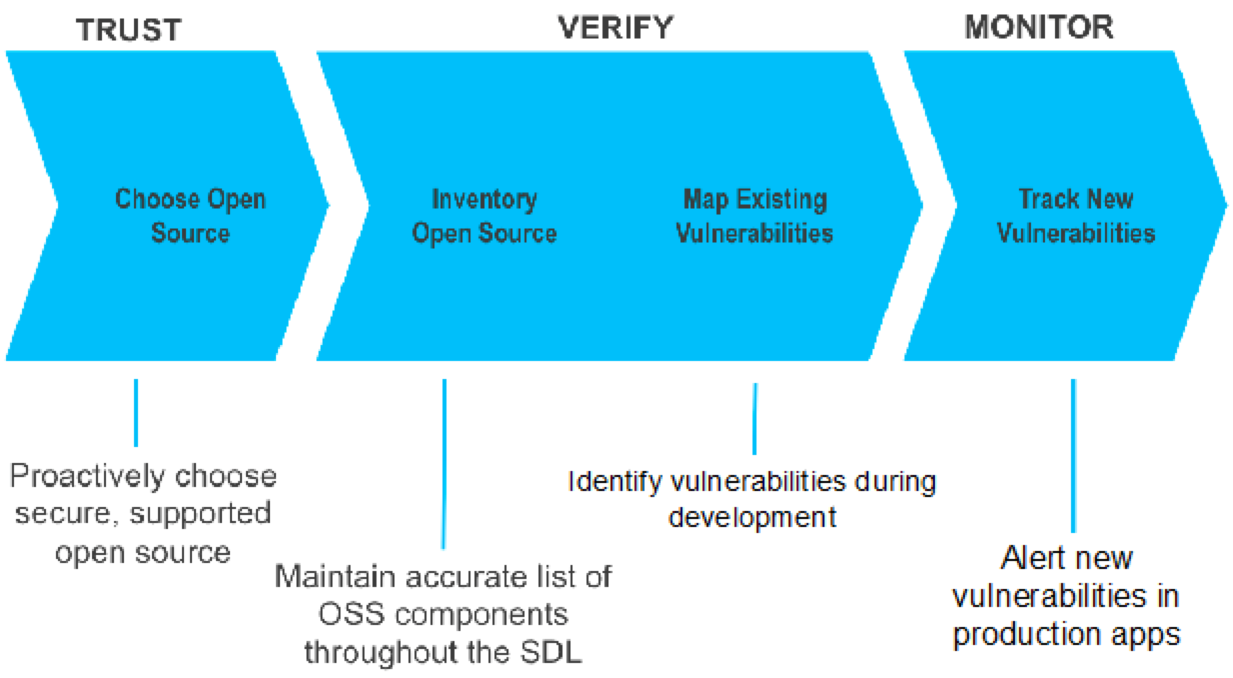
- To continue along the guidelines set by this pilot project, enlarging its scope and consolidating processes and IT systems:
    - Involving the owners of information systems that are not under the responsibility of DIGIT and promoting the federation of their CMDBs;
    - Industrialising the methodology described in these pages;
    - Industrialising the processes and information system elements introduced in this pilot experience, transforming them in an "industrial" solution (see the following section)

- To adopt security practices into the software development/adoption lifecycle:
    - To select and install only secure, supported open source;
    - To actively maintain accurate list of OSS components and applications;
    - To identify vulnerabilities during the development;
    - To alert product/solution managers of potentially vulnerable applications basing on the track of new vulnerabilities

- To foster the adoption of a common CMDB consolidating all the different inventories;

- To focus on the internal software development/acquisition processes adopting best practices and solid solutions.

Figure 10 - Proposed high level to-be approach


TRUST
VERIFY
MONITOR
Choose Open Source
Inventory Open Source
Map Existing Vulnerabilities
Track New Vulnerabilities
Proactively choose secure, supported open source
Maintain accurate list of OSS components throughout the SDL
Identify vulnerabilities during development
Alert new vulnerabilities in production apps

A best-practices solution would combine elements of TRUST, VERIFICATION, and MONITORING:

1 –TRUST means providing developers and architects with a way to choose open source components that are free of known vulnerabilities, and have active community support. This is a proactive step that reduces risk downstream in the software development process, and is the most cost-effective means of risk reduction.

2 – VERIFICATION means maintaining an accurate inventory of open source software and being able to map all its known vulnerabilities, in any and all applications, at any point in the SDL.

3 – MONITOR means being able to monitor the released code for newly discovered vulnerabilities and alert the right people for remediation. With over 4,000 new vulnerabilities each year, a comprehensive solution should be to continuously monitor the constant stream of new vulnerabilities, and automatically notify the administrator of any new vulnerabilities in the open source used in deployed applications, including which applications use the code, how critical the vulnerability is, and how to remediate it.

An additional very important aspect on which the added value of the “target” scenario can be significantly higher than the “pilot” scenario is the enlargement of the software component scope to “unknown” software.

As described in the data collection section, this pilot project is based on the hypothesis that only “known” software components/application will be dealt with. This is a strong constraint but it is absolutely reasonable for a first experience. The “unknown” software management is indeed quite complex. It requires the management of large amounts of raw inventory data and that the “unrecognised objects” have to be collected to match them with some “known” data patterns, in order to understand their nature: source code, executable, scripts etc. and professional tools to scan and recognize them.

Despite the complexity of the abovementioned process, from a security point of view, the most interesting elements are the “unknown” software components, which is why we strongly recommend considering this aspect as a priority in the future developments of this project.

7.2. The target scenario – first step

As per paragraph 2.2, the first step after the conclusion of this pilot project should be to start a program to reach the “target” scenario, with robust and agreed processes and an industrial-grade IT support solution.

The suggested “target” scenario is:

- DIGIT makes recurring automatic inventories to collect the software components that are in place (development and production);
- DIGIT has a consolidated CMDB and this CMDB is regularly enriched with inventory data;
- DIGIT has a consolidated repository where it stores a “referring” copy of any in-house developed or downloaded/used software (source, executable, data etc.);
- On a regular basis, DIGIT makes automatic verifications that the code present on the systems corresponds to the referring copy;
- DIGIT has a policy to apply a form of licensing to its in-house developed software and has a policy to evaluate whether to submit this software to a public community or to contribute to an OSS initiative;
- DIGIT has a policy to foster employees’ contribution to open software communities with the products of their work;
- On a regular basis, DIGIT scans the code repository with appropriate tools to find any possible “alien” or “unlicensed” software component.

A detailed discussion about the tools that can be used to support the open source inventory and the following ranking is the objective of a specific deliverable. Here, we can synthetize the overall features of the "target" processes/solution as follows:

1. Industrial automatic discovery and inventory tool, able to collect all the information about software components
2. Automatic inquiry of large internet databases to find additional metadata (licensing form, community dimension, vulnerabilities etc.)
3. Semi-automatic semantic web engine capable to enrich an initial list of standards
4. Graphic editing of the standard taxonomy
5. "business intelligence" dashboard with customisable ranking criteria/rules
6. Automatic publishing of the inventory and ranking as open-data on http://open-data.europa.eu/.

The "target" recurring processes are therefore the following:

1. Automatic and semi-transparent open source software component inventory and classification
2. Automatic inquiry of internet databases
3. Semi-automatic ranking
4. Selection of candidates for the code review

This ideal situation will be enriched and detailed as the project progresses, and will eventually provide a set of pragmatic recommendations to improve procedures, tools and data quality of the European Institutions.

8. Appendix 1: References and Related Documents

ID	Reference or Related Document	Source or Link/Location
1	*Project charter*	*https://webgate.ec.europa.eu/CITnet/confluence/display/FOSSA/Project+Charter*
2	*WP3/4/5 Contractor's offer*	*https://webgate.ec.europa.eu/CITnet/confluence/pages/viewpage.action?pageId=497944869*
3	*WP3 Quality Management Framework*	https://webgate.ec.europa.eu/CITnet/confluence/pages/viewpage.action?pageId=497944874
4	*WP1-WP3 metrics alignment meeeting*	https://webgate.ec.europa.eu/CITnet/confluence/pages/viewpage.action?pageId=497944902
5	*WP1 draft list of sustainability metrics*	https://webgate.ec.europa.eu/CITnet/confluence/pages/viewpage.action?pageId=497944432

9. Appendix 2: Abbreviations and Acronyms

ABAC / ABAC Asset	Corporate Ordering and Asset management system
BYOD	Bring Your Own Device
CMDB	Configuration Management Data Base
ETL	Extract, Transform and Load
FOSSA	Free & Open Source Software Application
MDM	Mobile Device Management
NIST	National Institute of Standards and Technology
OS	Operating System
OSS	Open Source Software
OSVDB	Open Source Vulnerability Database
RHEL	Red Hat Enterprise Linux
SAN	Storage Area Network
SDL	Software Development Library
SLA	Service Level Agreement
Svn	Subversion

10. Appendix 3: Annexes

Annex 1 – Target Data Model Description

Entity				
Name	**Definition**	**Requirement**	**Source**	**Type**
AppSoftware	The entity describes application / app software.	#4: Middleware in scope (e.g. application servers), application out of scope (e.g. webapps) #8: * All application developed and running in EP data center are listed in ADA	- CMDB (BMC remedy, BMC Atrium console, BMC proactive net) - service catalogue - ADA (Annuaire Des Applications): around 300 to 500 items - DTA (Document technique d'Architecture) - CARAP (CARtographie des Applications)"	Dependent
Criterion	The entity describes a quality criterium used to assess if related software belong to the Critical Software Shortlist.	The methodology to obtain the software inventory shall let European Commission and European Parliament categorize the components by several criteria: criticality, existing support, areas where the components are used. FOSSA needs to propose tools and procedures to assess the risk of new components and applications...	Defined by the methodology.	Independent
CustomSoftware	The entity describes software defined ad-hoc.	Custom developed code and base software platform customisations (i.e. shell scripts, SAP ABAP code, RDBMSs Stored procedures views, and triggers, operating systems configuration files, Javascript components etc.) is in scope.		Dependent
DataCenterResources	This layer groups all the possible open source software embedded inside physical devices such as routers, load balancers, SANs, switches, firewalls...	Data center resources are in scope.		Dependent
Dependencies	The entity lists all the software on which a software depends on.	The methodology to obtain the software inventory shall take into account libraries, versions and dependencies between components.	Package dependencies from software distributions.	Dependent
DevelopmentPlatform	The entity describes a software development platform or tool.	Development tools and platforms are in scope.		Dependent
License	The entity describes a software license and its terms.	Extended information about each OSS component shall be obtained (e.g. community that is behind, type of licence).	Specialized sites (i.e. OpenHub).	Independent
LicenseCompliance	The materialised relationship connects a software to the licenses it complies with.	Extended information about each OSS component shall be obtained (e.g. community that is behind, type of licence).	Specialized sites (i.e. OpenHub).	Dependent
MobileDevice	The entity describes a portable device (smartphone, tablet, etc.).	#2: Only mobile devices (phones & tablets) provided by digit, and managed by MDM, are in scope.	- MDM export	Dependent
MobileSoftware	The entity describes software that has been developed for mobile devices.	Mobile software could be in scope.		Dependent

Entity				
Name	**Definition**	**Requirement**	**Source**	**Type**
OperatingSystem	The entity describes an operating system.	Operating systems are in scope.		Dependent
Organization		Extended information about each OSS component shall be obtained (e.g. community that is behind, type of licence).	The entity details the Organisation that released the Software, Standard or System.	Independent
RuntimeSoftwarePlatform	The entity describes a web server, DBMS, application server or any kind of runtime/middleware;	Base software platforms (web servers, firewalls, DBMSs, application servers, and any kind of middleware) are in scope.		Dependent
Server	The entity describes a computer used for hosting purposes.	#3: Hosted servers in scope. Housed servers out of scope. #7: - Only machines under control of DIGIT C3 OCP • About 2500 OS installed • Physical / VM (mainly Virtual) - EG : Outside their control : Digit B uses machines / virtual machines outside their control Responsible for a certain set of products, linked to the OS layer (RedHat/Windows/Solaris) Boundaries * Only Red Hat servers managed by Satellite by Digit C3 OCP * Windows servers are managed by SCCM but don't run OSS (or very few) * Solaris servers run OSS but are not managed, and getting an inventory of those would require manual operations on the servers * development libraries are not inventoried by satellite * no visibility on pieces of OSS running in storage, network dedicated devices (SAN, NAS, Routers, Switches, load balancers, firewalls, ...). Need to reach out to vendor to get info regarding this.	- CMDB (BMC remedy, BMC Atrium console, BMC proactive net) - service catalogue - Satellite Server (Management & deployment server for RedHat)	Dependent
Software	The entity describes software.	The specific requirements that must accomplish the methodologies and tools to be used in the creation of the inventory of assets and standards shall be proposed. #4: Middleware in scope (e.g. application servers), application out of scope (e.g. webapps) #6: - This Unit limits its action to ONLY the OS layer, not application layer	List of installed software from EC CMDB systems.	Independent
SoftwareCriteria	The materialised relationship connects softwares with their related quality criteria.	- Each component shall be accompanied with information necessary to assess its sustainability, according to the metrics defined in Work Package 1 of FOSSA. Some examples known at the time of writing of this Technical Annex: development process, automatic regression testing, vulnerability reporting process, size of team supporting the project, financing of the team. - FOSSA needs to propose tools and procedures to assess the risk of new components and applications...	List of installed software from EC CMDB systems.	Dependent

Entity				
Name	**Definition**	**Requirement**	**Source**	**Type**
SoftwareInstance	The entity represents a deployed software, hence it relates with one or more hosts.	- The specific requirements that must accomplish the methodologies and tools to be used in the creation of the inventory of assets and standards shall be proposed. - The inventory process shall identify housed software installed by users possessing admin rights. #4: Middleware in scope (e.g. application servers), application out of scope (e.g. webapps) #7: - This Unit limits its action to ONLY the OS layer, not application layer	List of installed software from EC CMDB systems.	Dependent
SoftwareVersion	The entity describes the version of a Software.	- The process shall ensure the tracking of releases of the inventoried software. - The methodology to obtain the software inventory shall take into account libraries, versions and dependencies between components.	List of installed software from EC CMDB systems.	Dependent
SoftwareVulnerabilities	The materialised relationship connects a software version with its related detected vulnerabilities.	FOSSA needs to contribute towards assessing current security of OSS components used at the European Commission and the European Parliament.	Publicly available vulnerability sources (i.e.NVD).	Dependent

Entity				
Name	**Definition**	**Requirement**	**Source**	**Type**
Standard	The entity describes a standard, whose characteristics are: openness, transparency and being based on consensus.	The list of free and open technical specifications and open standards used by European Institutions is elaborated. #5: - the catalogue of standards is not ready yet; the main short-term objective is to define the EU standards adoption process, based on standards such as ISO DTR 28380-1 or CAMSS (defined by ISA) - it is recommended to refer to DG CNECT at least to use their same naming conventions, approaches etc. to define the standards catalogue within the FOSSA project #6: - Some OSS applications are frequently used; they are known but not recommended (certified) Ex : DRUPAL - Some "Reference configurations" are enforced for middleware platform (eg.: Weblogic) - May include in the methodology a business Case / Test Case for including Test and result of spot audit of servers and recommend process update to incorporate this type of audit in Test process for new and updated applications. #10: *Only technical standards recommended and implemented by CEF are considered * CEF has no visibility on the application of standards they recommend as they are implemented by member states directly * EIF : Table of Standards => Website - CEF eDelivery (EBMS AS4 OASIS standard) developed by Digit B4 - By domains - Legal - Organisational - Semantic - Technical * CEF belongs to Technical domain * CEF provides building blocks to member state (NA's) that implements standards they recommend. * Most building block instances run in member states infra, few run in Digit infra. * Recommended building blocks from CEF are: eSignature, eID, eDelivery, eInvocing, Automated Translation)"	- DIGIT reference list of standards refreshed - Specialized sites (i.e. ISO, W3C, ANSI, OMG etc).	Independent
StandardCompliance	The materialised relationship connects a software to the standards it complies with.	The specific requirements that must accomplish the methodologies and tools to be used in the creation of the inventory of assets and standards shall be proposed.	Specialized sites (i.e. OpenHub).	Dependent
System	The entity represents a real machine or device on which software has been installed.	- The specific requirements that must accomplish the methodologies and tools to be used in the creation of the inventory of assets and standards shall be proposed. - The inventory process shall identify housed software installed by users possessing admin rights.	List of installed software from EC CMDB systems (if available).	Independent
Vulnerability	The entity describes a vulnerability which was found on a specific version of a software.	FOSSA needs to contribute towards assessing current security of OSS components used at the European Commission and the European Parliament.	Publicly available vulnerability sources (i.e.NVD).	Independent

Entity				
Name	**Definition**	**Requirement**	**Source**	**Type**
Workstation	The entity describes a desktop or laptop device.	#1: Only workstations & laptops provided by DIGIT are considered here (no BYOD). Also excluded are OLAF (Anti-Fraud Office), JRC (Joint Research Centre), EUROPOL (?) The list of order for approved software is stored in the ABAC database, but it isn't in an exploitable state (scanned orders in landscape view).	- Landesk management suite exports - DIGITline : list of product used	Dependent

Attribute(s) of "AppSoftware" Entity				
Name	**Definition**	**Requirement**	**Is PK**	**Is FK**
SoftwareName	The name that identifies the software.		Yes	Yes

Attribute(s) of "Criterion" Entity				
Name	**Definition**	**Requirement**	**Is PK**	**Is FK**
CriterionName	The name that identifies the criterion.		Yes	No
Threshold	The specific criticality threshold for the criterion.	A threshold is required as part of the critical software assessment methodology.	No	No
Weight	Measures the relevance of the criterion and influences how it is taken into account when assessing software criticality.	A weight is required as part of the critical software assessment methodology.	No	No

Attribute(s) of "CustomSoftware" Entity				
Name	**Definition**	**Requirement**	**Is PK**	**Is FK**
SoftwareName	The name that identifies the software.		Yes	Yes

Attribute(s) of "DataCenterResources" Entity				
Name	**Definition**	**Requirement**	**Is PK**	**Is FK**
SoftwareName	The name that identifies the software.		Yes	Yes

Attribute(s) of "Dependencies" Entity				
Name	**Definition**	**Requirement**	**Is PK**	**Is FK**
DependsOnSoftwareName	The name that identifies a software on which the software under analysis depends on.	Required to relate a software to the one it depends on.	Yes	Yes
DependsOnVersionNumber	The name that identifies the version of a software on which the software under analysis depends on.	Required to relate a software to the one it depends on.	Yes	Yes
SoftwareName	The name that identifies the software.	Required to relate a software to the one it depends on.	Yes	Yes
VersionNumber	Reports the version the software is, or was.	Required to relate a software to the one it depends on.	Yes	Yes

Attribute(s) of "DevelopmentPlatform" Entity				
Name	**Definition**	**Requirement**	**Is PK**	**Is FK**
SoftwareName	The name that identifies the software.		Yes	Yes

Attribute(s) of "License" Entity				
Name	**Definition**	**Requirement**	**Is PK**	**Is FK**
LicenseType	The specific type of the license, which refers to a specific standard.		Yes	No
LicenseContact	The name of the reference person for the license.	Input to support decision process is available.	No	No
OrganizationName	The name that identifies the organisation that defined the license.	Input to support decision process is available.	No	Yes

Attribute(s) of "LicenseCompliance" Entity				
Name	**Definition**	**Requirement**	**Is PK**	**Is FK**
SoftwareName	The name that identifies the software.		Yes	Yes
LicenseType	The specific type of the license, which refers to a specific standard.		Yes	Yes

Attribute(s) of "MobileDevice" Entity				
Name	**Definition**	**Requirement**	**Is PK**	**Is FK**
SystemName	The name that identifies the system.		Yes	Yes

Attribute(s) of "MobileSoftware" Entity				
Name	**Definition**	**Requirement**	**Is PK**	**Is FK**
SoftwareName	The name that identifies the software.		Yes	Yes

Attribute(s) of "OperatingSystem" Entity				
Name	**Definition**	**Requirement**	**Is PK**	**Is FK**
SoftwareName	The name that identifies the software.		Yes	Yes

Attribute(s) of "Organization" Entity				
Name	**Definition**	**Requirement**	**Is PK**	**Is FK**
OrganizationName	The name that identifies the organisation.		Yes	No
Location	The physical location (i.e. place) the headquarters of the organisation is stationed.	Input to support decision process is available.	No	No
Description	Further details on the organisation.		No	No

Attribute(s) of "RuntimeSoftwarePlatform" Entity				
Name	**Definition**	**Requirement**	**Is PK**	**Is FK**
SoftwareName	The name that identifies the software.		Yes	Yes

Attribute(s) of "Server" Entity				
Name	**Definition**	**Requirement**	**Is PK**	**Is FK**
SystemName	The name that identifies the system.		Yes	Yes

Attribute(s) of "Software" Entity				
Name	**Definition**	**Requirement**	**Is PK**	**Is FK**
SoftwareName	The name that identifies the software.		Yes	No
Description	Further details about the software.		No	No
IsCritical	Tells if the software belongs to the Software Critical Shortlist.	Conceived to mark software which belongs to the Critical Software Shortlist, defined in Phase III of the project.	No	No
AOWName	The application owner name.	Input to support decision process is available.	No	No
AOWPosition	The application owner position.	Input to support decision process is available.	No	No
Developer	The development entity that designed the software.	Input to support decision process is available.	No	Yes
SoftwareType	It defines the type of the System: application software, custom software, mobile software, runtime platform, operating system, development platform or data center resources.		No	No

Attribute(s) of "SoftwareCriteria" Entity				
Name	**Definition**	**Requirement**	**Is PK**	**Is FK**
SoftwareName	The name that identifies the software.		Yes	Yes
CriterionName	The name that identifies the criterion.		Yes	Yes
Rating	The value of the criterion for the specific software.	A weight is required as part of the critical software assessment methodology.	No	No

Attribute(s) of "SoftwareInstance" Entity				
Name	**Definition**	**Requirement**	**Is PK**	**Is FK**
SoftwareName	The name that identifies the software.	Required to relate a software instance to its abstract information.	Yes	Yes
SystemName	The name that identifies the system.	Required to relate a software instance to its abstract information.	Yes	Yes
VersionNumber	Reports the version the software is, or was.	Required to relate a software instance to its abstract information.	Yes	Yes
Size	The memory space (in MB) the instance needs.	Input to support decision process is available.	No	No

Attribute(s) of "SoftwareVersion" Entity

Name	Definition	Requirement	Is PK	Is FK
SoftwareName	The name that identifies the software.		Yes	Yes
VersionNumber	Reports the version the software is, or was.		Yes	No

Attribute(s) of "SoftwareVulnerabilities" Entity

Name	Definition	Requirement	Is PK	Is FK
SoftwareName	The name that identifies the software.		Yes	Yes
VersionNumber	Reports the version the software is, or was.		Yes	Yes
VulnerabilityName	The name that identifies the vulnerability type.		Yes	Yes

Attribute(s) of "Standard" Entity

Name	Definition	Requirement	Is PK	Is FK
StandardisationBody	The organisation that defined the standard.		No	No
StandardName	The name that identifies the standard.		Yes	No
Description	Further details about the standard.		No	No
StandardisationBody	A reference to the Standard content.	Input to support decision process is available.	No	Yes
ECContext	The European Community Context the standard is related to.	Input to support decision process is available.	No	No
Documentation	The documentation that the standard have, in text format.	Input to support decision process is available.	No	No
ParentStandardName	The name that identifies the standard which references or contains this standard.	Required to define a data structure to model the hierarchy of standards.	No	Yes

Attribute(s) of "StandardCompliance" Entity

Name	Definition	Requirement	Is PK	Is FK
SoftwareName	The name that identifies the software.		Yes	Yes
StandardName	The name that identifies the standard.		Yes	Yes

Attribute(s) of "System" Entity

Name	Definition	Requirement	Is PK	Is FK
SystemName	The name that identifies the system.		Yes	No
Vendor	The Organization that produces the system.	Input to support decision process is available.	No	Yes
Model	The specific model of the machine, comprehensive of producer and version.	Input to support decision process is available.	No	No
RAM	It measures the Random Access Memory size of the machine.	Input to support decision process is available.	No	No
IsVirtual	It tells if the machine is a Virtual Machine.	Input to support decision process is available.	No	No
ManagingOrganization	The name that identifies the organisation that manages the system.	Input to support decision process is available.	No	Yes
SystemType	It defines the type of the System: mobile device, server or workstation.		No	No
SoftwareName	The name that identifies the software.		No	Yes

Attribute(s) of "Vulnerability" Entity

Name	Definition	Requirement	Is PK	Is FK
VulnerabilityName	The name that identifies the vulnerability type.		Yes	No
Source	The affected software's source code.	Generally available basic information on vulnerability, required to model it properly.	No	No
Description	Further details on the vulnerability.	Generally available basic information on vulnerability, required to model it properly.	No	No
Impact	An indicator of the expected harm received if the vulnerability is actually exploited.	Generally available basic information on vulnerability, required to model it properly.	No	No
Remediation	The description of the required actions to resolve the vulnerability.	Generally available basic information on vulnerability, required to model it properly.	No	No

Attribute(s) of "Workstation" Entity

Name	Definition	Requirement	Is PK	Is FK

SystemName	The name that identifies the system.		Yes	Yes