D2.1 Data Warehouse (DWH) for agriculture policy impact assessment data management

agricore

Deliverable Number	D2.1
Lead Beneficiary	AAT
Authors	Juan Carlos Castillo Alcántara (AAT), Rebeca Gutiérrez Salgado (AAT), Ángel Javier Jiménez Pérez (AAT), Massimo Gioia (AAT), Mercedes Pichardo Cayón (AAT), Fernando Dorado Rueda (IDE)
Work package	WP2
Delivery Date	M23
Dissemination Level	Public

www.agricore-project.eu

The Agricore project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement No. 816078

Document Information

Project title	Agent-based support tool for the development of agriculture policies
Project acronym	AGRICORE
Project call	H2020-RUR-04-2018-2019
Grant number	816078
Project duration	1.09.2019-31.8.2023 (48 months)

Version History

Version	Description	Organisation	Date
0.1	Deliverable Template (TOC)	AAT	14-jun-2021
0.2	Development of sections	AAT	24-jun-2021
0.3	Feedback from IDENER resolved	AAT	27-jul-2021
1.0	Final version	AAT	29-jul-2021

Executive Summary

This document provides a detailed description of the role and functioning of the DWH within the AGRICORE project architecture, as a fundamental component for the storage, processing and exchange of data. By providing an extensive list of requirements necessary to fulfil the modularity and distributed architecture principles of the project, a complete guide to the architecture defined as the solution and to the tools and technologies selected to build the DWH, and descriptions of how the DWH and the ARDIT indexer will be connected for data ingestion. Finally, a deployment plan is defined and divided into several separate phases, increasing in each of them the features, functionalities and capabilities of the DWH.

Abbreviations

Abbreviation	Full name
ABM	Agent-Based Model
ARDIT	Agricultural Research Data Index Tool
CPU	Central Processing Unit
CSV	Comma-separated values
DCAT-AP	Data Catalogue Vocabulary Application Profile
DWH	Data Warehouse
ETL	Extract-Transform-Load
EU	European Union
FR	Functional Requirement
GML	Geography Markup Language
HTTP	Hypertext Transfer Protocol
HDFS	Hadoop Distributed File System
HTTPS	Hypertext Transfer Protocol Secure
ICT	Information and Communications Technology
IT	Information Technology
JDBC	Java Database Connectivity
JSON	JavaScript Object Notation
KML	Keyhole Markup Language
KPI	Key Performance Indicator
NFR	Non-functional Requirement
ODBC	Open Database Connectivity
OLAP	On-Line Analytical Processing
OLTP	On-Line Transaction Processing
RAM	Random Access Memory
RSS	Really Simple Syndication
SHP	Shapefile
SQL	Structured Query Language
YARN	Yet Another Resource Negotiator

List of Figures

List of Tables

Table of Contents

1 Introduction

The main purpose of the presented document, D2.1 Data Warehouse (DWH) for agriculture policy impact assessment data management, is to explain the AGRICORE's DWH architecture design and its implementation, describing the elements, tools, technologies, protocols and processes involved. In addition, its requirements, uses and applications within the project, identifying those responsible for its development, deployment and maintenance.

The document focuses mainly on the connection between the Agricultural Research Data Index Tool (ARDIT) and the data warehouse, as they provide storage of potentially useful data sources for the project. It begins by outlining the requirements set for the DWH. Then, details the architecture designed, describing how it can be implemented according to a set of technical requirements, information about the DWH deployment plan, how will the architecture's objectives be ensured, the steps taken at this point, as well as, new requirements and changes detected during development.

1.1 Intended Audience

This document is primarily intended for all the partners that are involved in the consortium to understand how the DWH has been designed and implemented, helping the integration of other project modules into it.

Developers, researchers, other stakeholders and, ultimately, anyone interested in this project who wants to make use of the DWH, can consult this document to learn about the design process, the technologies that compose the architecture, the aims and objectives of the DWH, and how to operate it.

The European Commission is also in the scope of the intended audience to report on the progress of the AGRICORE project and meet the project's milestones.

1.2 What is a Data Warehouse?

A first definition of what a data warehouse is would come from one of its main fathers and creators, **Bill Inmon**, who defined it [1] as:

"*...a subject-oriented, integrated, time-variant and non-volatile collection of data in support of management's decision-making process*"

- **Subject-oriented:** the data warehouse can analyse data on a particular topic or subject area.
- **Integrated:** a data warehouse can integrate different types of data from multiple sources.
- **Time-variant:** changes in the data over time are registered. So, access to historical information is kept.
- **Non-volatile:** when new data is stored, old data is kept in order to have access to historical data.

Another alternative definition comes from **Ralph Kimball**, an important author in the data warehousing field, who said [2]:

"*A data warehouse is a copy of transaction data specifically structured for query and analysis*"

Both authors, Inmon and Kimball, created two completely different models and architectures in which they focused their views on how a data warehouse should work. Inmon's approach seeks information integration and is used to store large volumes of data in large-scale projects. In addition, the internal structure must be normalised, the data must undergo a pre-processing phase to eliminate inconsistencies or errors, avoiding data redundancy. On the other hand, Kimball's approach is oriented towards information queries and more simple projects. The internal structure proposed by Kimball is not normalised, which makes the DWH more exploitable, adding more different possibilities when working with the data.

However, none of the solutions proposed by then is definitive in a design, both have advantages and disadvantages. Furthermore, there are other alternative solutions that even can modify or combine both [3]. In the end, a data warehouse is a typical data management system that supports reporting and analytical tasks in a project, business or company. It serves as a large repository that contains integrated and cleansed data from multiple different sources. These data can be queried later in order to produce analytical reports or to assist in decision making.

1.2.1 ETL processes

In order for the stored information to be useful, it must be accurate and high-quality so, it needs to be processed before it is stored. ETL processes allow data, from multiples sources, to be moved, reformatted, cleansed and loaded into the DWH. An ETL process responds to Extract, Transform and Load, y is divided into three phases.

- **Extraction phase**: includes the extraction of data from their origin. These data may have different structures and formats, so it is necessary to verify that they conform to the expected structure.
- **Transformation phase:** in this phase, the data received are processed, filtered and cleaned. For example, by deleting unnecessary records, merging values, creating new ones or anonymising the data, masking or replacing personal data that could identify individuals, groups or organisations.
- **Loading phase:** the data is loaded into the DWH. By maintaining a loading history, it allows having a complete historical record over time.

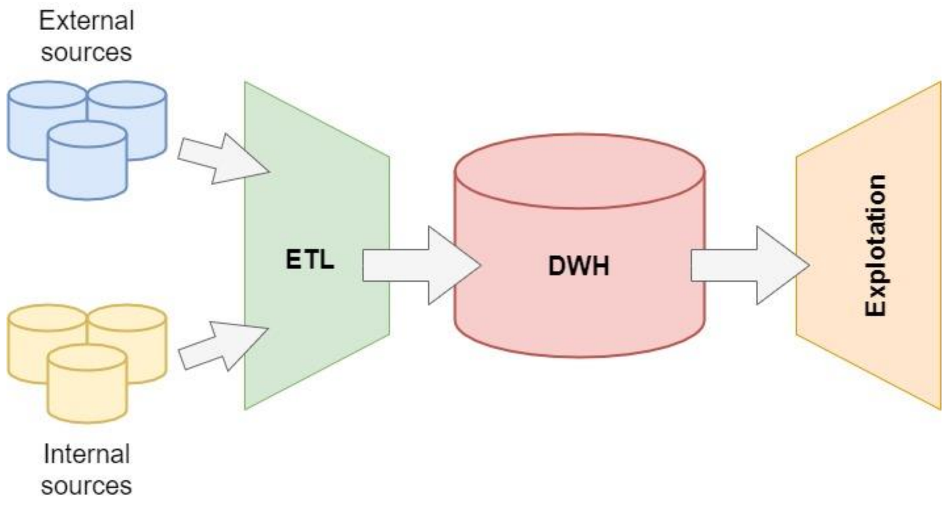

Figure 1 DWH concept

1.2.2 Data warehouses, data lakes and databases

In the data storage field, doubts or confusion may arise between concepts such as data warehouse, data lake or database. A database is used to register related and organised data, they are useful for small and atomic transactions, and contain only the most up-to-date information. Databases also make use of OLTP, a transaction-oriented type of processing, where access to data is optimised for reading and writing tasks and where insert, modify and delete operations are performed. Whereas, a DWH is used as a complete information system that stores historical and commutative data from different sources to be analysed. Data warehouses make use of OLAP, another type of processing oriented towards analysis, and are optimised for reading or querying. Insertions, modifications or deletions are less frequent and less efficient. Usually, the most common operation executed on a DWH will be to rapidly run a number of complex queries on large multi-dimensional databases. So, a data warehouse may contain multiple databases to store all the data.

On the other hand, the data lakes, which, unlike the data warehouses, are used to store raw information, exactly as it is received from its origin, without processing or filtering it, so the data may not be curated. Data lakes can be an alternative to data warehouses depending on their intended use. Both have certain advantages and disadvantages. Data lakes, for example, since they could be not structured, provide more flexibility and possibilities for working with them, but require more experienced users who know how to structure, classify, combine and analyse the data afterwards. In the data warehouses, once data is extracted from its source, it passes through a processing layer that makes it ready to use. In addition, access is simpler and faster and can be used by a wider audience, due to a better ability to understand data that has already been processed, cleaned, filtered and has some intelligible structure [4].

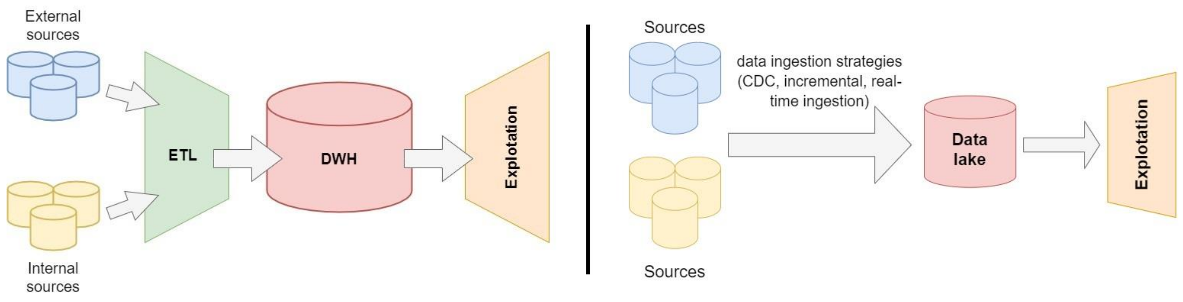

Figure 2 DWH and Data Lake schemes comparison

In certain projects, data warehouses, data lakes and databases can be used in combination for the whole data storage process. The image below (extracted from [5]) includes an example schema of how the three components are integrated into a complete storage system. In this case, data lakes are used to store structured, semi-structured or unstructured data extracted from their sources. Then, the ETL processes prepare the data before loading it into the data warehouse, where one or more databases could be used to store it [6].

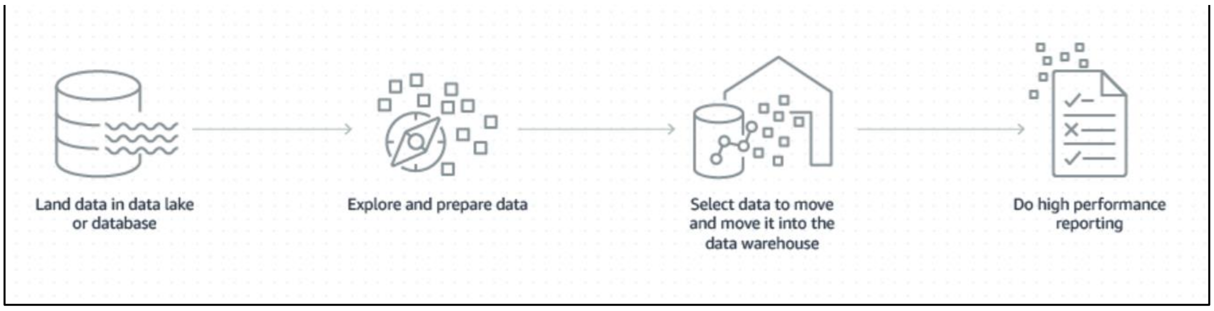
Land data in data lake or database
Explore and prepare data
Select data to move and move it into the data warehouse
Do high performance reporting

Figure 3 Amazon's example of a complete storage system

The AGRICORE project relies on the exclusive choice of a data warehouse as the tool for ingesting, transporting, storing and processing the data. The project requires the data to be processed and stored in a structured and accurate way, for later use. In addition, as the tool will be used by EU policymakers and stakeholders, the choice of a DWH allows for a wider range of possible users to operate with it.

1.3 AGRICORE Project Data Warehouse

According to AGRICORE's General Agreement document, "*The AGRICORE project proposes a novel tool for improving the current capacity to model policies dealing with agriculture by taking advantage of the latest progress in modelling approaches and ICT. Specifically, the AGRICORE tool will be built as an agent-based approach where each farm is to be modelled as an autonomous decision-making entity that individually assesses its own context and makes decisions on the basis of its current situation and expectations. This modelling approach will allow simulating the interaction between farms and their context (which will account for environment, rural integration, ecosystem services, land use and markets) at various geographic scales – from regional to global. To do so, advances in big data, artificial intelligence algorithms, mathematical solvers and cloud computing services will be applied to optimise the extremely-long parameterisation and calibration phase required by current agent-based tools, to better mimic the modelling of farmers' behaviour and interactions, to credibly assess the local effects of global events and EU policies, and in general to improve policy design, impact assessments and monitoring*".

In AGRICORE, the DWH takes a very important role in the whole project architecture, centralising the exchange of data between the different modules. The main functionality of the AGRICORE data warehouse will be to serve as a large data repository allowing certain components or modules of the architecture to be supplied with information. It will also be used to ingest, transport, process and store data generated by the same or other components, generating a cyclical process of data storage and supply. An example of this would be the simulations launched by the tool itself. The simulation processes need synthetic populations, among other components, to be executed. These synthetic populations need, in turn, to be fed with accurate data that would come from the Agricultural Research Data Index Tool, so there will be a direct connection between ARDIT and the DWH to store potentially useful datasets. Once the synthetic population is generated, it will be also stored in the DWH, as well as, other elements that compose the simulation process and the results obtained from it.

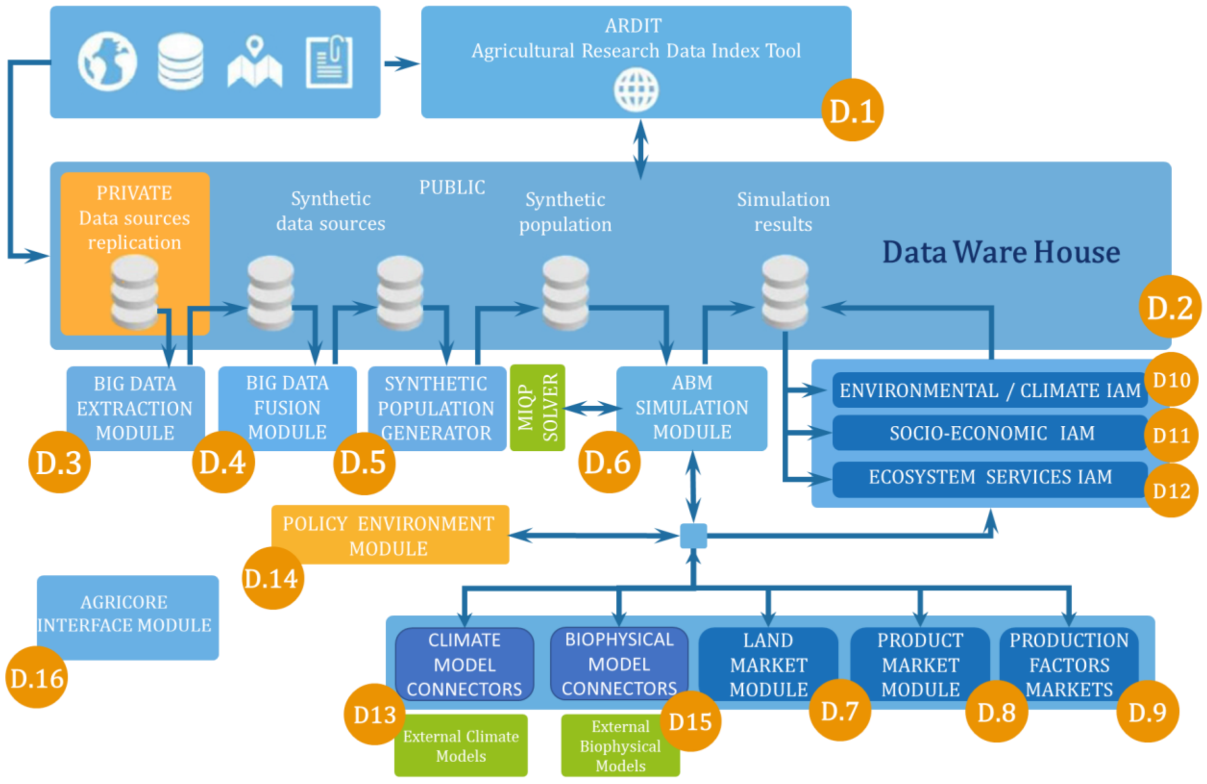

Figure 4 AGRICORE Modular Architecture

- D1 - ARDIT (Agricultural Research Data Index Tool, previously referred to as European Datasource index tool). The tool allows users to search for different sources of publicly available data on the Web, categorised by the methodology implemented according to the ontology AGRICORE DCAT-AP 2.0 extension.
- D2 - DWH: Data Warehouse tool suitable for supporting the analyses contemplated within the AGRICORE project.
- D3 - Data extraction Module: Module that extracts all the data of interest from multiple datasets considered in the project. Data extraction encompasses the capabilities for accessing different datasets, selecting the necessary data and formatting it for further processing.
- D4 - Data fusion module: Combine the individualised data with the probability distributions of the variables to generate the joint probability distributions.
- D5 - Synthetic populations generator: Module aimed to obtain realistic synthetic population making use of the Synthetic Reconstruction method.
- D6 - ABM simulation engine: Instantiate agents for each farmer generated, evaluating its situation and making decisions based on its preferences.
- D7, D8, D9 - Multi-Market Modules: They allow the interaction of the agents present in the ABM-e through the main markets that affect the agricultural sector: the land market (D7), the market for products derived from agricultural and livestock activity (D8) and the markets for agricultural production factors (D9) such as labour, fertilisers and pesticides, or machinery.
- D10, D11, D12 - Impact Assessment Modules: Provides different modules used to evaluate the KPI's (Key Performance Indicators) related to specific topics (e.g. Environmental / Climate KPI's).

- D13 - Climate Model Connectors: They allow connecting ABM-e and other modules with pre-existing weather and climate models. Thus, these models can provide the AGRICORE suite with microservices for the generation of climate conditions for simulation, as well as for the prediction of climate effects based on the decisions of the agents.
- D14 - Policy environment module: Define different policies and translate them into an input for the simulation engine.
- D15 - Biophysical Model Connectors: They allow the ABM-e and other modules to be connected to pre-existing biophysical models. Thus, these models can provide the AGRICORE suite with microservices for generating biological, edaphological and agronomic conditions for simulation, as well as for predicting the effects on the biotic and abiotic factors of the ecosystem as a consequence of the actions of the agents.
- D16 Agricore interface module: Centralise all the interactions of the user with the AGRICORE tool, allowing to see the results of the simulations defined and performed along the simulation process.

The project partners involved in the task of designing and building the DWH, in development, support or supervisory role, are listed below:

Partner abbreviation	Partner full name
AAT	Ayesa Advanced Technologies
AKD	Akdeniz University
AUTH	Aristotle University of Thessaloniki
IAPAS	Institute of Agrophysics Polish Academy of Sciences
IDE	IDENER
STAM	STAM
UNIPR	The Università degli Studi di Parma
UTP	Bydgoszcz University of Science and Technology

Table 1 List of partners involved in DWH development

1.4 Architecture Requirements

This chapter reviews all the pre-established requirements which were necessary for the DWH architecture development, dividing them according to their type. During the requirements gathering phase, it was defined the need to download and install ARDIT and the DWH locally on the user's own equipment and machines. As a result, the possibility of having two versions of ARDIT and the DWH was proposed. The first version of both could be downloaded and installed on personal machines by everyone. The second one would consist of a fully publicly available version of ARDIT, accessible through the internet, with connections with a DWH exploited in the cloud. Thus, it was considered to have a Global and Local Indexer, the global one would be the public one, connected with the DWH exploited in the cloud. And the local would be the one available to be downloaded, installed and executed locally. The chapter describing the designed architecture provides more details about the two existing indexer and DWH versions.

1.4.1 Functional requirements

Functional requirements (FR) define the system behaviour under working conditions. FRs are product features that developers must implement in order to achieve the users' goals [7].

- **AG.D1.FR.007-1. DWH connection.** An administrator can configure the DWH connection parameters in ARDIT.
- **AG.D1.FR.007-2. ETL Execution in the DWH.** A user can execute an ETL in their local DWH.
    - **AG.D1.FR.007-2-4. Launch an ETL in the job queue launcher:** A user can execute an ETL added to a job queue for the DWH.
    - **AG.D1.FR.007-2-5. ETL execution feedback:** A user could see the feedback of the ETL execution process.
    - **AG.D1.FR.007-2-6. ETL launched feedback:** A user can check if a ETL has been executed and stored in the DWH.
    - **AG.D1.FR.007-2-7. ETL execution only if it is correct:** A user can execute an ETL in their local DWH.
    - **AG.D1.FR.007-2-8 ETL isolated tracked environment:** The application must provide an isolated environment for its execution. An ETL should have tracked its links as well as its names, its output table name to avoid collisions between its data, folder names and common files.
- **AG.D2.FR.001. Centralise the information exchange within the AGRICORE IT architecture:** The system should centralise all information exchange within all AGRICORE modules. DWH could be deployed in a cloud, in local architecture or both.
- **AG.D2.FR.002. Provide high-performance analysis capabilities to the DWH:** DWH must provide technical capabilities to launch high-performance operations such as Spark. These operations could have different purposes such as data extraction, processing and generation, among others.
- **AG.D2.FR.003. Easy-to-manage access permissions:** the DWH must provide access permission using the permission mechanism provided by Hadoop.
    - Each file and directory in the DWH must be associated with an owner and a group, having separate permissions for the user that is the owner, for the users that are members of the group, and for all the other users.
- **AG.D2.FR.004. Separate critical/private information from information suited to be made public:** The DWH must be able to separate different information sections as private and public. The ETL developers should have administrator access to the system but not for ETL launchers.
- **AG.D2.FR.005. Support both private and public cloud infrastructure deployment.**
- **AG.D2.FR.006. The DWH must support a sandbox deployment.** A virtual environment will be provided to allow the end-user to build, test and deploy the complete DWH.
- **AG.D2.FR.007. Data output stored in DWH.** The data fusion module must provide its output storing the results in the DWH.
- **AG.D2.FR.008. The DWH will need a direct connection to the ARDIT indexer**: in order to store potentially useful data sources in the DWH, both modules need to be connected.
- **AG.D2.FR.009. Feedback on the deletion of data sources:** users will receive feedback from the indexer when a data source is removed from the DWH.

- **AG.D2.FR.010. Users will have administrator permissions on their DWH workspaces:** in the Local Indexer, users will have the necessary permissions to operate freely on their workspaces.
- **AG.D2.FR.011. The Global Indexer will have a common workspace:** unlike the Local Indexer, the Global will not have individual workspaces for each user, but only one common workspace.

1.4.2 Non-functional requirements

Non-functional requirements (NFRs) are requirements that specify criteria that can be used to judge the operation of a system, rather than specific behaviours. NFRs are not related to the functionality of the system but define how the system should perform under operational conditions. The non-functional requirements related to the DWH within the AGRICORE project are detailed in the next points[8].

1.4.2.1 Processing requirements

- **AG.D2.NFR.001. Use massive parallel programming (MMP) technologies and/or Hadoop/Spark:** the system must provide high computing technologies using Hadoop/Spark to execute high-demand operations. These processing tasks will be performed by different workers in a distributed way.
- **AG.D2.NFR.002. Provide high-performance analysis capabilities to the DWH:** the DWH must provide technical capabilities to launch high-performance operations through Spark or Hive. These operations could have different purposes such as data extraction, processing and generation, among others.
- **AG.D2.NFR.003. The DWH system must provide vertical and horizontal scalability:** the DWH must provide capabilities to increase the resources on each node or to add more nodes, if the processing, storage and performance needs require it, using technologies such as Docker Swarm or Kubernetes.
- **AG.D2.NFR.004. The DWH must provide a distributed data processing:** the processing must be carried out in a distributed way across different worker nodes to support fault tolerance.
- **AG.D2.NFR.005. It must be possible to stop, pause or prioritise loading processes on the data warehouse:** users with the appropriate permissions/roles shall be able to stop, prioritise and, in general, manage the loading processes.

1.4.2.2 Deployment requirements

- **AG.D2.NFR.006. Support both private and public cloud infrastructure deployment**: the DWH system must be able to be deployed in public (Amazon Web Services, Azure, Google Cloud) and private (VMWare, OpenStack, HyperV) architectures.
- **AG.D2.NFR.007. The Local Indexer and the DWH associated with it must be independent and can be deployed separately:** both tools will be stored in separate repositories and their installations will not be independent of each other.
- **AG.D2.NFR.008. A sandbox deployment based on microservices must be provided:** Containers will be used to encapsulated microservices in an isolated environment, that allows the users to test, build and deploy the desired software. Docker, Docker Swarm or Kubernetes could be used as container platforms.

1.4.2.3 Connectivity requirements

- **AG.D2.NFR.009. ETL processes must be used in the data exchange between the DWH and the data sources indexer**: ETL processes will assist in the data sources extraction from their origins and loading them into the data warehouse.
    - **AG.D2.NFR.009-1. The ETL processes must support data sources from HTTP/HTTPS or JDBC connections:** to enable the ingestion of data sources in different formats or from existing databases.
    - **AG.D2.NFR.009-2. The data warehouse and ETL processes must support the ingestion of data sources in different formats:** CSV, Excel, JDBC and JSON formats must be supported in the ingestion of data sources**.**
    - **AG.D2.NFR.009-3. The data warehouse and ETL processes must support the ingestion of geo-referenced data sources:** the DWH must support the ingestion of data sources in GeoJSON, GML, SHP y KML formats.
    - **AG.D2.NFR.009-4. Formats used to store data sources in the DWH must also be used as ingesting formats:** data sources in formats such as Avro or Parquet can be ingested into the DWH.

1.4.2.4 Storage requirements

- **AG.D2.NFR.010. Data is stored in different nodes across the DWH:** data will be partition into blocks of a specific size that guarantee the distributed storage.
    - **AG.D2.NFR.010-1.** The size of each block shall be appropriate, allowing for a reduction in seek time.
    - **AG.D2.NFR.010-2.** The metadata information of each block has to be stored.
- **AG.D2.NFR.011. Use a combination of SQL and non-SQL databases:** a combination of SQL and non-SQL databases with Hadoop/Spark would be necessary to execute high-demand operations
- **AG.D2.NFR.012. Users should be able to select where they want to store the data sources:** when loading new data sources, users should be able to select whether they want to store them in Hadoop's HDFS, Hive or any other technology used.
- **AG.D2.NFR.013. The DWH must allow storing data sources in different formats:** CSV, Avro, Parquet or structured plain text must be supported as formats for storing data-source in the DWH.
- **AG.D2.NFR.014. The DWH must run periodic processes to delete data sources loaded from ARDIT:** in order to avoid disk space overflow, deletion processes should be executed to remove data sources stored in the DWH.
- **AG.D2.NFR.015. The DWH system will allow the user to select how long data sources should be stored before deletion:** when loading new data sources from ARDIT, users should be able to select, from a menu of pre-set options, how long the data sources should be kept in DWH.

1.4.2.5 Performance requirements

- **AG.D2.NFR.016. The DWH shall make efficient usage of the available memory.**
- **AG.D2.NFR.017. The DWH must have low response times.**
- **AG.D2.NFR.018. The DWH will be able to recover data in the case of a node failure.**
- **AG.D2.NFR.019. The system shall support data replicability to be tolerant to human and computer failures:** each block has to be replicated in other nodes to avoid data losses.

- **AG.D2.NFR.020. The system should provide an organised shutdown:** the DWH must provide an organised system shutdown functionality to avoid the loss of launched and running processes or jobs.

1.4.2.6 Veracity requirements

- **AG.D2.NFR.021. The data sources stored must be complete and accurate:** the DWH system should not store data sources for which the loading process has failed.
- **AG.D2.NFR.022. Data sources whose loading processes fail should be discarded:** if a data source loading process fails, the system must notify the error to the user and the data source loading process must be discarded completely.

1.4.2.7 Variety requirements

- **AG.D2.NFR.023. ETL help guidelines:** the instructions to extract the information from the official source and load it into a DWH should be stored in the ARDIT database allowing for the easy population of the DWH by people who are not familiar with the technology
- **AG.D2.NFR.024. The DWH system must take into account the last update date of the data sources, particularly for those that are generated periodically:** the last update date will allow to notify outdated data sources or to check which version is stored in the DWH.
- **AG.D2.NFR.025. The loading tools available in the Local Indexer will keep track of the data sources stored by the user locally.**
- **AG.D2.NFR.026. ETL architecture version:** an ETL must have stored the architecture version where it has been executed to track which ETLs are supported for each architecture.

1.4.2.8 Accessibility requirements

- **AG.D2.NFR.027. Data in the DWH can be accessible for ad-hoc querying:** the accessibility of each file depends on the permissions given to the user who wants to query it.
- **AG.D2.NFR.028. Separate critical/private information from information suited to be made public:** the DWH must be able to separate different information sections as private and public. The ETL developers should have administrator access to the system but not for ETL launchers.
- **AG.D2.NFR.029. Data sources created within the DWH must be accessible only by the users who create them:** new data sources can be generated in the data warehouse from others already stored. These new data sources must be saved in private sections within the DWH, exclusive to their creators, and they will never be visible or accessible from ARDIT.
- **AG.D2.NFR.030. The DWH system must provide different user roles or profiles:** it must be defined who can monitor and/or control the process running in the DWH or who can manage the stored ETL scripts.
- **AG.D1.NFR.031. Authentication information:** the Global Indexer will not have authentication information of 3rd sites access. But, on the other hand, Local Indexer will have them.

2 Architecture Approach

This chapter describes and details the architecture designed as a solution. Given the importance of ARDIT in it, it is necessary to explain its purpose and use within the AGRICORE project.

The Agricultural Research Data Index Tool (ARDIT) provides a publicly accessible index of data sources available for policy assessment. It makes use of the characterisation methodology and the ontology defined in the project. Based on the DCAT-AP standard, a specification of W3C's Data Catalogue Vocabulary (DCAT) used by the European Union, a set of common attributes and properties in data sources were defined, in order to homogenise them. This standard has been modified for AGRICORE, adding new attributes to the data sources to meet the needs of the project.

ARDIT is a browser-based web application, built using Spring framework for the server-side code and a REST API in Java, and Angular framework to build a web client using TypeScript, HTML and CSS. All this, using PostgreSQL as the database manager and Docker to automate the deployment of the application in all the different software environments.

2.1 Architecture Scheme

Based on the requirement that all developments in the AGRICORE project must follow a distributed microservice-based architecture, and given the need to be able to deploy and use the tools in private environments, the designed architecture proposes the holding of two different approaches, a Global architecture and a Local architecture. This will be combined with the use of containerisation technologies, such as Docker, to facilitate deployment in different software environments, and to ensure distributed deployment and scalability of operations.

The following table illustrates the main differences between the global and local architectures:

	ARDIT	DWH	Description
Global architecture	Global indexer. Publicly available indexer on the web	Cloud-based DWH with a restricted access	The solution used to register new data sources in the indexer and to load potentially useful data sources for the project objectives into the data warehouse
Local architecture	Local indexer. Available for download and installation by any user in private environments	Available for download and installation by any user in private environments	The solution that will allow researchers, developers and anyone interested in the project to download and install both tools for testing and use in private single or multi-node environments

Table 2 Differences between global and local architectures

There will be a publicly accessible indexer (**Global Indexer**), which will be used as an open data portal, allowing the search of data sources according to multiple parameters and the access or download from their origins. Advanced functionalities, such as ETL processes to load data into the DWH, will be restricted to a limited number of users.

On the other hand, there will be a **Local Indexer**, whose main objective is to enable the possibility of exploiting the DWH locally (on-premise), in private environments, with limited resources that depend on each machine where it is used. Users will be able to download the complete tool, including the data sources indexer and the DWH, and will be able to install them in a simple way on their computers. In this case, the Local Indexer will not have any restrictions for launching ETL processes on the DWH, as all the processing and storage capabilities will depend exclusively on the available hardware from the user's computer or cloud instances and will not have any impact on other Local Indexers.

Meanwhile, the Global Indexer will have a direct connection with a DWH exploited in the cloud. This data warehouse will be used as a basis by European Union policymakers to run simulations and to assist in decision making. It will also have increased processing, storage and velocity capabilities, among others. For this reason, the access and use of the DWH exploited in the cloud will be restricted for most users.

Both, Local and Global Indexer have their own independent databases. To enable data synchronisation between them, all Local Indexers subscribe to the Global Indexer in order to receive notifications when new data sources are added to the global. RSS will be used to allow the Global Indexer to notify new changes to the Local Indexers, enabling the possibility to update the set of referenced data sources locally. In addition, both indexers will have utilities that will help or assist the user when loading data sources into the DWH. The loader tools will provide templates for the ETL processes depending on the type or format of the data to be loaded into the data warehouse. These templates, as well as the ETL scripts themselves, will be stored in each indexer's database.

A catalogue with instruction and/or operation guides for loading data into the DWH will be also available in the indexers to help non-experienced or non-advanced users on how to extract the data sources from its sources, how to generate and launch ETL scripts and where to store it in the DWH. In addition, to manage the access to the indexers, both the global and local, make use of their respective and separated Lightweight Directory Access Protocol (LDAP) databases, where user credentials are stored for authentication purposes.

The following figure shows the complete architecture of the DWH with the connection to ARDIT, also the separation of the Global and Local indexers, all the existing components and the communication flow between them. It is important to clarify that, although it does not appear in this first figure, the Global indexer has the same DWH system scheme as the local indexer.

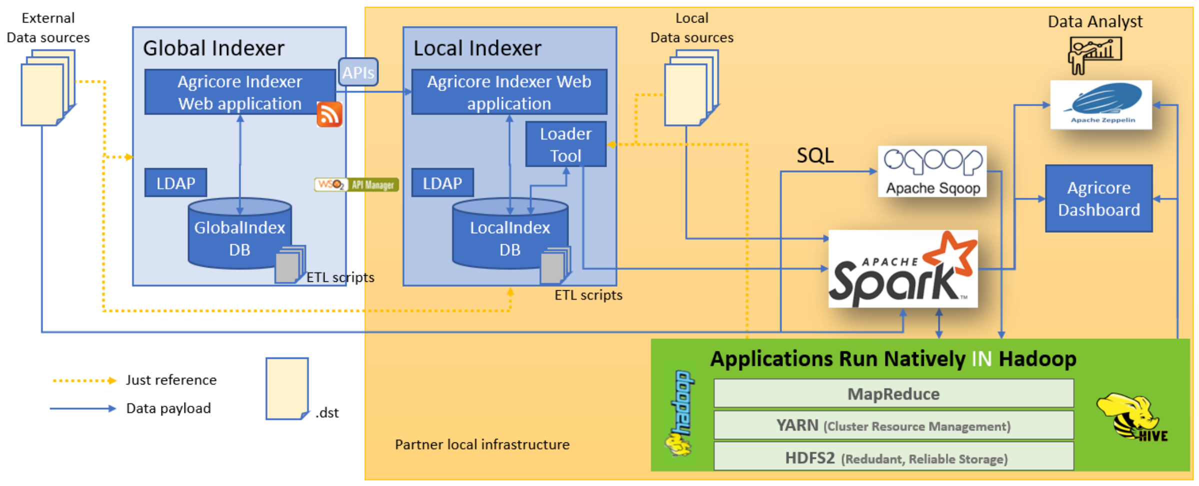

Figure 5 AGRICORE DWH complete architecture

2.1.1 Global architecture

The following figure shows the Global Indexer architecture. This version focuses on the fully publicly data sources indexer, which will be available on the web. The Global Indexer will have its own database loader tools, ETL scripts, and LDAP tool to manage the access control to the platform. This indexer will have full connection to a more restricted DWH, used for analysis and simulations by project personnel, authorised personnel and policymakers, and unlike the Local Indexer, it will be the only version where users, with specific roles, will be able to characterise new data sources and add them so that they can be indexed. This DWH will use the same tools and technologies whether it is deployed on Amazon or Azure cloud-based solutions, or cloud-independent. Users will be able to use the ARDIT tool to search and download data sources based on multiple attributes and properties, and the possibility to load data sources to the DWH, as mentioned above, will be severely restricted. Authorised users will be able to load them from different external sources, such as existing databases, by retrieving them through HTTP/HTTPS requests to their sources, or even retrieving data sources stored in other existing cloud-based solutions.

When loading new data sources into the DWH, users will use a set of ETL templates through the loading tools, which will facilitate the storage process, according to the data source origin or format. In addition, users will be able to select how long they want to keep the data sources stored or whether they prefer to store them in Hadoop's HDFS or in Apache Hive. Once the ETL script has been generated, and after checking that it is correct, the user will be able to launch the loading process into the DWH or download the ETL script generated, which will be stored in the database too. After the loading process has been launched, the user will receive feedback on it.

Given the higher resource cost of using cloud-based solutions to exploit the DWH, all data sources will be stored in a single communal workspace. In this way, if a data source is stored in the DWH, it cannot be stored again.

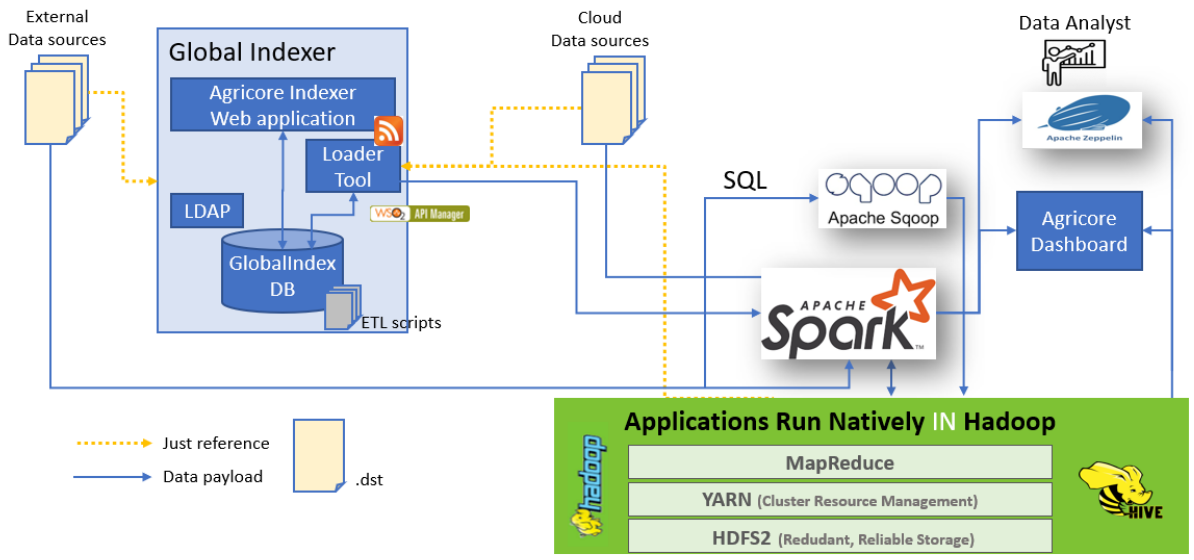

Figure 6 AGRICORE DWH global architecture

2.1.2 Local architecture

Finally, the Local Indexer focuses on the possibility of using the ARDIT together with the DWH on personal computers and machines. In contrast to the Global Indexer, this DWH version is installed locally. Therefore, for the Local Indexer architecture, both the ARDIT and the DWH will be available for download and installation by users, and this will be done separately as both components will be independent. The Local Indexer will be crucial for users to be able to test and work freely with the tools without affecting the Global Indexer and the DWH in the cloud, which will be vital to achieving the AGRICORE project objectives.

The Local Indexer works very similarly to the Global one, but with some differences. First of all, the Local Indexer will also have its own database, loader tools and ETL scripts. It will be possible to store data sources from available external sources, but also it will be possible to load data sources that are stored locally on the user's computer.

In addition, users will be able to update their indexer databases to synchronise them with the Global Indexer database when changes to the latter occur. Users will also be able to subscribe via RSS to the Global Indexer to receive notifications when new data sources have been registered, modified or deleted. This will allow updating the collection of characterised datasets in the local indexers, but will never include these datasets in the local DWH, this process will have to be done manually by the user through the loading tools supplied.

Unlike the data warehouse exploited in the cloud, the DWH for local use will store the data sources in separate and private sections for each user. The owner of each section will have full control over it.

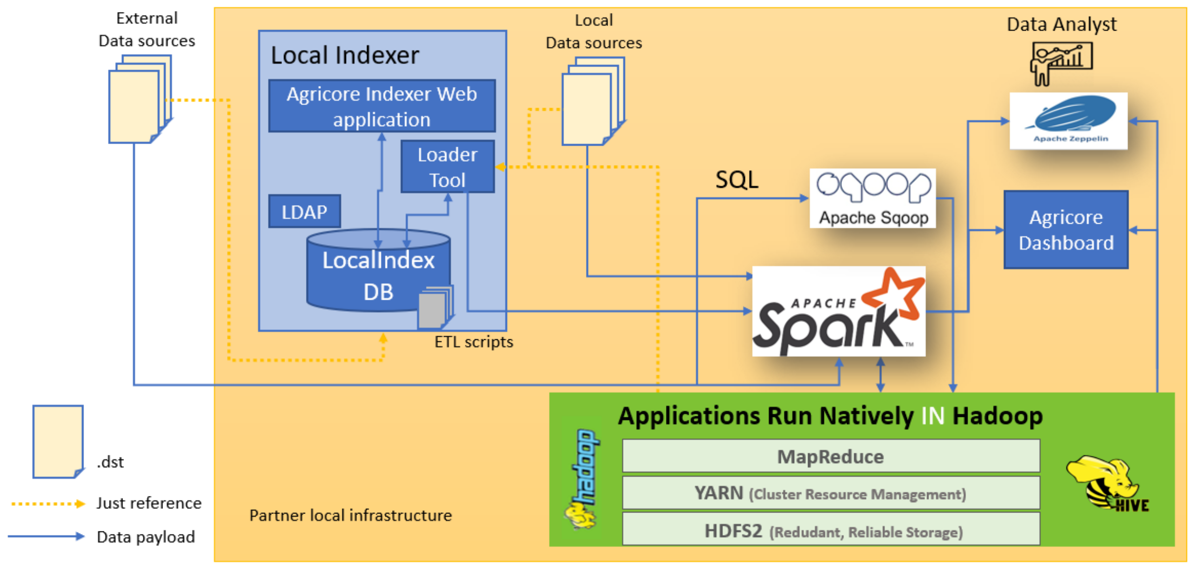

Figure 7 AGRICORE DWH local architecture

2.1.3 DWH Architecture

2.1.3.1 Functional Description

DWH is responsible for ingestion, processing and storing data in a distributed way. To ensure the correct functioning of these processes, DWH must contain different functional elements:

- File distributed system to store the data in a distributed way. This prevents data loss by using data replication in case of hardware or humans failures occurs.
- Software component to coordinating and managing the resources of the entire cluster (e.g. CPU, Memory).
- Connectors or software elements that allow ingesting data from different data sources (e.g. import data from MySQL database).
- Batch processing component or components to process data from the file distributed system, being able to read and combine data stored in different nodes. Processing is done in parallel by distributing the workload among different workers.
- A programming environment that enables the user to make different operations related to the DWH (e.g. a data processing pipeline, manage files in the file distributed system).

These five functional blocks are combined and form the Data Warehouse. Once implemented, a data analyst can interact with the platform to access the required data. Then, the analyst can create his own data processing for the specific use case and store the resulting file back in the DWH so that only this file can be accessed by this user or group of users in the analyst's organisation.

2.1.3.2 Proposed Software Elements

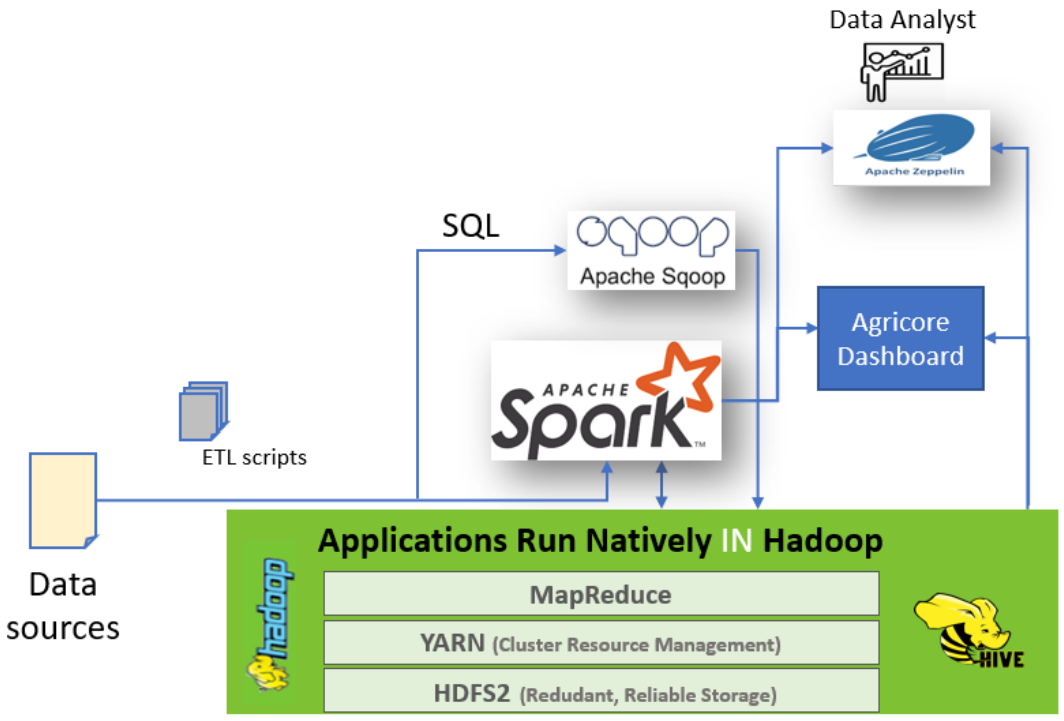

Figure 8 AGRICORE DWH system scheme

The DWH is composed of different software elements working in a coordinated way that allows us to interact with it in a simple and efficient manner. For the choice of these software elements, it has been decided to use different pieces that compose the Apache Hadoop ecosystem. Apache Hadoop is an open-source framework to develop Big Data applications in a reliable, scalable and distributed way. It is designed to scale up from single servers to thousands of machines, each of them with its own hardware. The chosen modules or software pieces to implement the DWH are explained as follows:

- **Hadoop Distributed File System (HDFS)** for storing large data sets using the MapReduce paradigm [9]. By distributing the storage and computation resources across different nodes, the resource can grow with demand. This property allows the DWH to scale as the size of data increases. HDFS stores the data in blocks, which is the minimum amount of data that it can read or write. This fact enables ingesting files larger than any individual disk connected in the HDFS cluster. In order to minimise the cost of seeks, HDFS blocks are large compared to disk blocks in the traditional system. As can be seen in the next image [10], there are two main components in HDFS [11][12].
    - **Namenode** plays the role of the master node in HDFS, containing all the metadata information about the system: owner of the information, the permission of files, mapping between blocks of data and the data nodes. A secondary namenode can be used to achieve high availability, avoiding a single point of failure in the system.
    - **Datanodes** store and retrieve data in the form of blocks. Datanodes report the status and health of the data blocks located on that node, and based on this information, the namenode could request to the Datanode the creation of additional replicas, remove them, or decrease the number of data blocks present on the node.

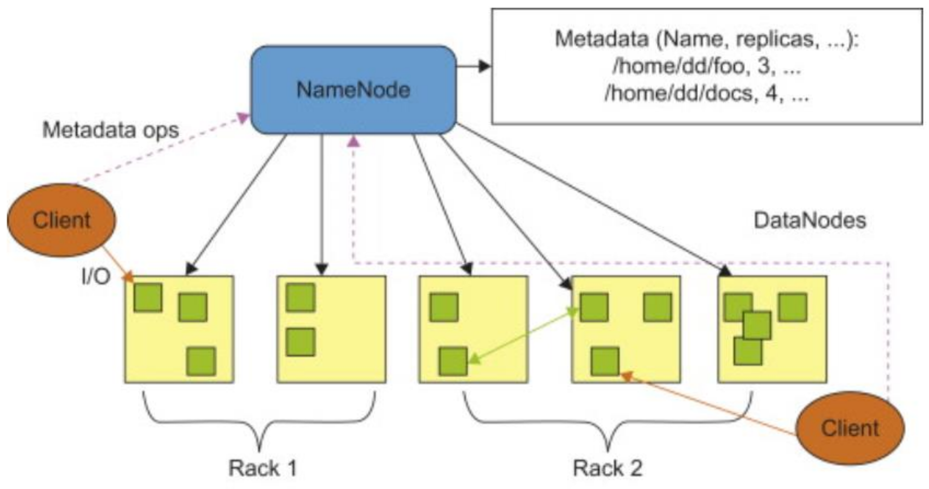

Figure 9 High-level architecture of the HDFS

- **Yet Another Resource Negotiator (YARN)** is Hadoop's cluster resource management system [13]. It provides APIs for requesting and working with the available cluster resources. YARNs is composed of two major components:
    - **Resource Manager** to manage the available resources in the cluster.
    - **Nodes Manager** running on all the nodes in the cluster to launch and monitor containers. A container executes an application-specific process with a constrained set of resources (e.g. CPU, Memory usage) and reports metrics to the Resource Manager.

YARN scheme representing how the data is distributed in the nodes through MapReduce paradigm [14]:

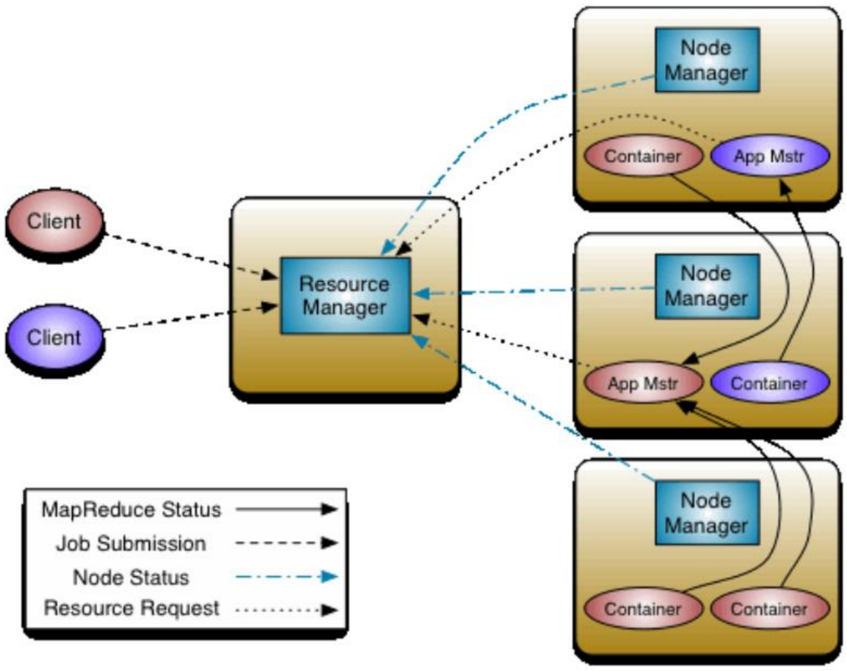

Figure 10 YARN scheme for the distribution of the data to different nodes using MapReduce

Batch processing engines. The implementation of 3 different ways to access data stored in a distributed way in HDFS is proposed:

- **MapReduce** is a Hadoop framework in which it is possible to write applications that can process large data stored in different nodes. There are two main operations in this process: (1) Map, sorting and filtering the data and thereby organising them in form of a group. The output of this operation is a pair of <key,value>. Then, (2) Reduce, is in charge of processed the output from the map operation, performing the summarisation by aggregating the mapped data. An example of the MapReduce pipeline is presented in the diagram below [15]:

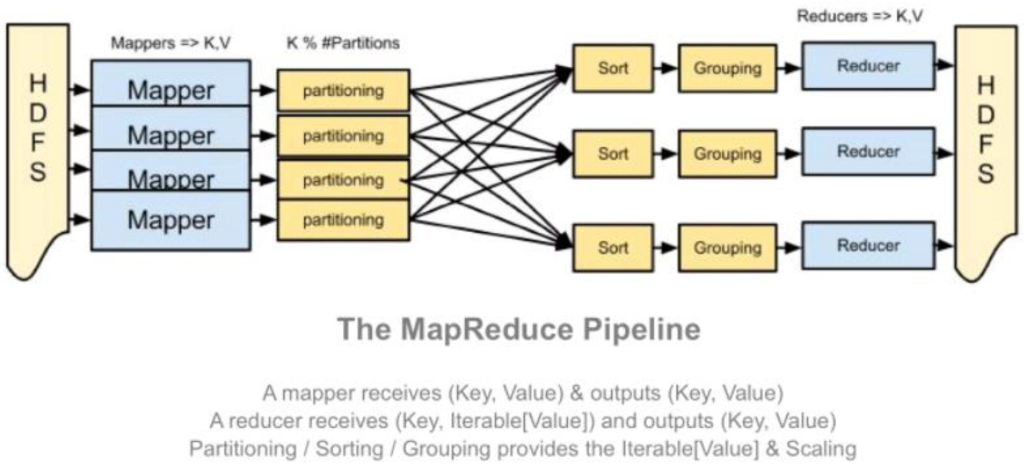

Figure 11 MapReduce pipeline scheme

- **Apache Hive** is a data warehouse software project built on top of Apache Hadoop for providing data queries and analysis[16]. Hive gives an SQL-like interface to query data stored in various databases and file systems that integrate with Hadoop. Traditional SQL queries must be implemented in the MapReduce Java API to execute SQL applications and queries over distributed data. Hive provides the necessary SQL abstraction to integrate SQL-like queries (HiveQL) into the underlying Java without the need to implement queries in the low-level Java API. Since most data warehousing applications work with SQL-based querying languages, Hive aids the portability of SQL-based applications to Hadoop. The architecture of Hive is comprised of different components [17][18].
    - **Hive Client** is used to creating Hive applications in a wide range of programming languages. There are different drivers (e.g., Thrift, JDBC, ODBC) that could be chosen in the case of the need to connect an external application to the Hive cluster.
    - **Hive Services** to use and manage the Hive Cluster:
        - Hive CLI to execute commands in the console.
        - Hive Web User Interface to execute operations in a Web Browser.
        - Hive Metastore is responsible to store all the metadata of the tables and partitions in the Hive warehouse.
        - Hive Server is used to accepts requesting from different Hive Client.
        - Hive Driver is in charge of transfer the external queries to the compiler.
        - Hive compiler creates MapReduce jobs from Hive queries.
        - Hive Execution Engine executes the plan created by the Hive compiler on top of the Hadoop ecosystem.

The following image illustrates an example of Hive's architecture [19]:

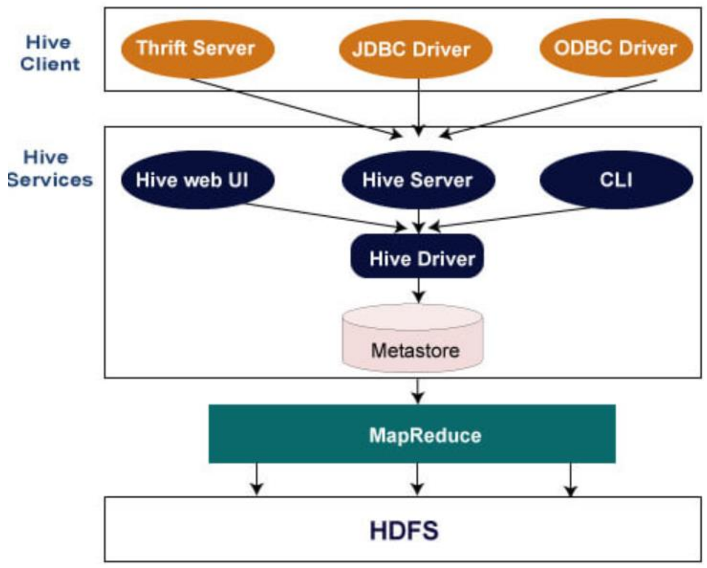

Figure 12 Apache Hive architecture

- **Apache Spark** is an open-source distributed general-purpose cluster-computing framework [20]. Spark provides an interface for programming entire clusters with implicit data parallelism and fault tolerance. Spark facilitates the implementation of both iterative algorithms, which visit their data set multiple times in a loop and interactive/exploratory data analysis, i.e. the repeated database-style querying of data. The latency of such applications may be reduced by several orders of magnitude compared to Apache Hadoop MapReduce implementation. Spark is designed to be highly accessible, offering different APIs in Python, Java, Scala and SQL. It translates the instructions in these languages to perform data transformation over data stored in HDFS in a fast way in comparison with Hive or MapReduce. Spark is not only a processing engine, it consists of a stack of different tools to develop different applications related to the field of Big Data and Artificial Intelligence[21][22].
    - **Spark Core** contains the components related to task scheduling, memory management, fault recovery, interaction with storage systems (e.g. HDFS), building resilient distributed datasets (RDDs), which are a collection of items distributed across different nodes in parallel.
    - **Spark SQL** is a package for querying data via SQL.
    - **Spark Streaming** component enables micro-batch processing of live streams of data.
    - **Spark MLib** module contains different Machine-Learning algorithms to implement ML approaches directly in Spark.
    - **GraphX** is included to perform graph-parallel computations,
    - **Cluster Managers** (YARN, Mesos or Spark Standalone) to manage the task with one central coordinator (Driver node) and some distributed workers. This driver node converts the user program into tasks and passed this task to the workers.

Spark ecosystem schema and cluster components [23][24]:

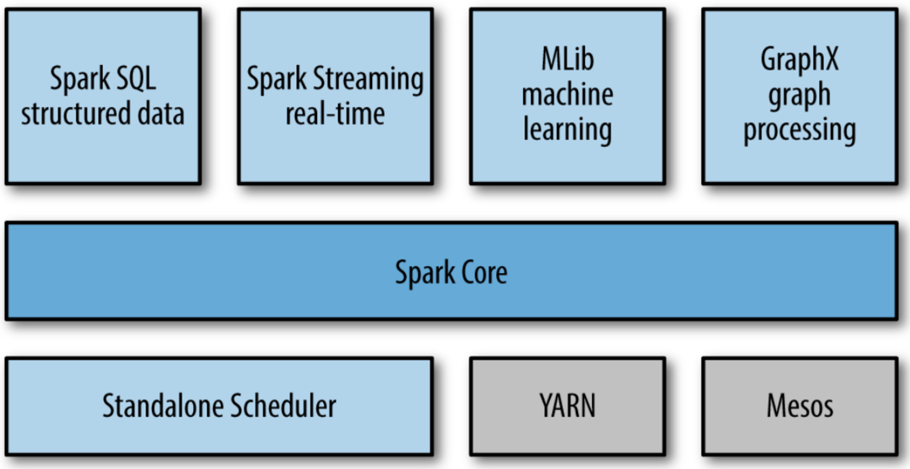

Figure 13 Spark ecosystem

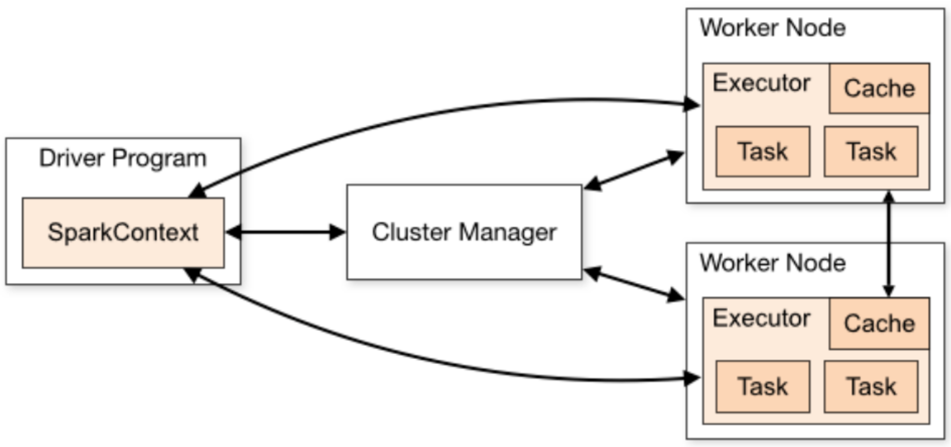

Figure 14 Spark cluster components

- **Apache Sqoop** is a tool designed to transfer data between the Hadoop framework and relational databases or mainframes [25]. You can use Sqoop to import data from a relational database management system (RDBMS) such as MySQL or Oracle or a mainframe into the Hadoop Distributed File System (HDFS), transform the data in Hadoop MapReduce, and then export the data back into an RDBMS. Sqoop automates most of this process, relying on the database to describe the schema for the data to be imported. Sqoop uses MapReduce to import and export the data, which provides parallel operation as well as fault tolerance [26][27].

The following figure depicts a diagram of the data import and export processes in Sqoop [28]:

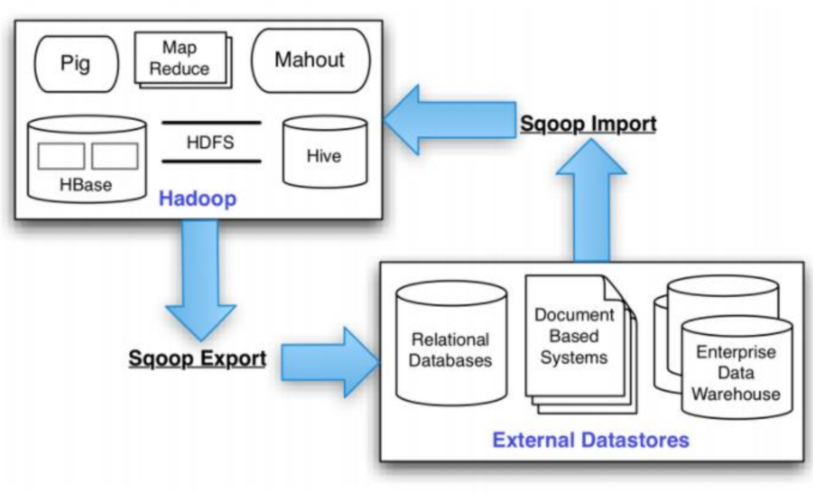

Figure 15 Sqoop import/export process scheme

- **Apache Zeppelin** is a web-based notebook that enables data-driven interactive analytics with a wide range of programming languages and tools[29]. It allows to mix languages in the same notebook, easily converting the notebook into a presentation style to report Bussiness Intelligence analysis[30].

Referring to AGRICORE DWH system scheme figure, where the DWH system components are illustrated when new data sources need to be loaded into the DWH, the user has two possibilities. The first one, storing them directly on the HDFS using Spark, which is in charge of partitioning the data to be distributed to the different nodes. The second is through Hive, so the data can be queried through SQL statements afterwards.

In both cases, if the data sources are in an external relational database, Sqoop would be used to store it in the HDFS. Both Sqoop and Hive make use of Hadoop's MapReduce paradigm, which handles data partitioning, so they do not need to go through Spark to be stored in HDFS.

Once the data is stored, the AGRICORE suite provides two different tools to interact with it. The first one is the AGRICORE Dashboard, a multiplatform desktop application that will provide a user-friendly interface to allow users to launch simulations combining different synthetic populations, policy sets, KPIs, etc. Also allowing data query and visualisation in graphs, charts, tables or maps. The second one is Apache Zeppelin, although it provides a more basic and limited set of charts, and being necessary to know a programming language in order to use it, this tool will contribute to the DWH development, use and testing, allowing data ingestion, analytics and visualization, so users familiar with it or with a certain programming language will be able to interact with the DWH while the AGRICORE dashboard is being developed.

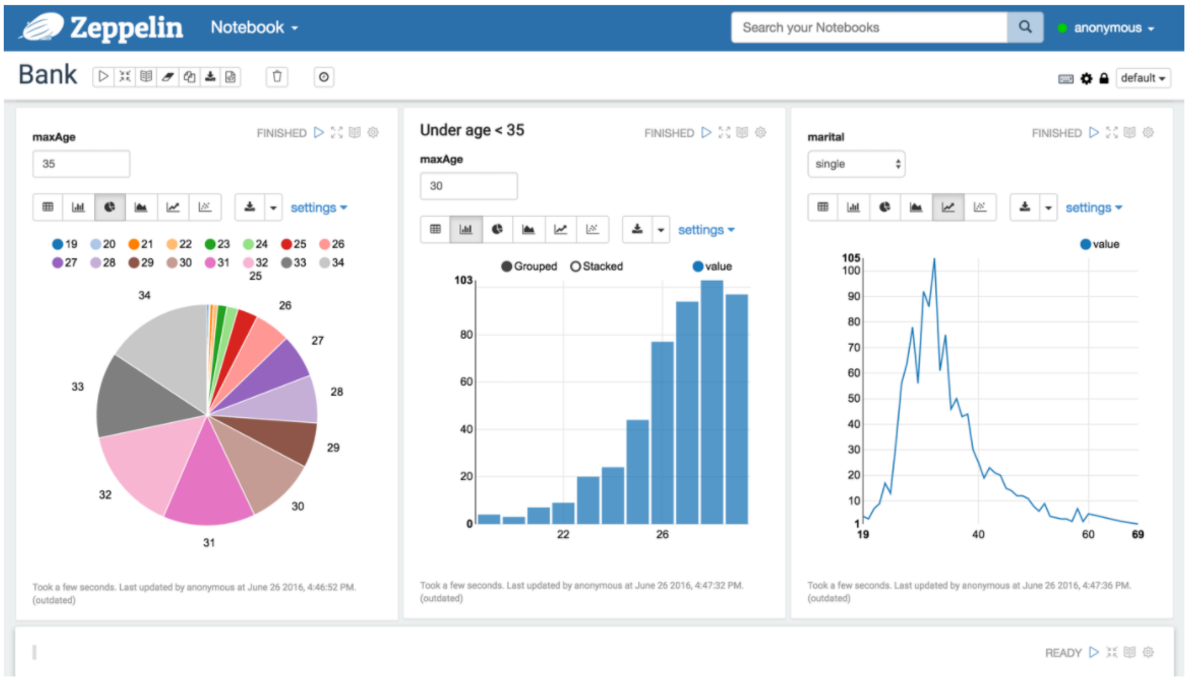
Zeppelin
Notebook
Search your Notebooks
anonymous
Bank
default
maxAge
35
FINISHED
settings
19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34
Under age < 35
maxAge
30
Grouped
Stacked
value
marital
single
value
Took a few seconds. Last updated by anonymous at June 26 2016, 4:46:52 PM. (outdated)
Took a few seconds. Last updated by anonymous at June 26 2016, 4:47:32 PM. (outdated)
Took a few seconds. Last updated by anonymous at June 26 2016, 4:47:36 PM. (outdated)
READY

Figure 16 An example of data visualisation in Zeppelin extracted from its official site

3 Deployment Plan

The deployment plan includes a guide to the steps to be performed in order to make a software operational or functional in a given environment and the processes required to make that asset available to its users. For the DWH deployment plan, an incremental approach will be followed, starting with a simpler version of the DWH with limited functionality and features. In the following phases, the functionalities and features will be increased gradually, and thus, the complexity of the system will increase.

This type of incremental approach will allow us to analyse the foundations on which the DWH will be built, enabling testing and collaboration with the project partners involved in all the deployment phases, but especially in the initial phases, in case new requirements arise or if it is necessary to change those already defined.

3.1 Architecture Delivery Strategy

The deployment plan is divided into the following phases:

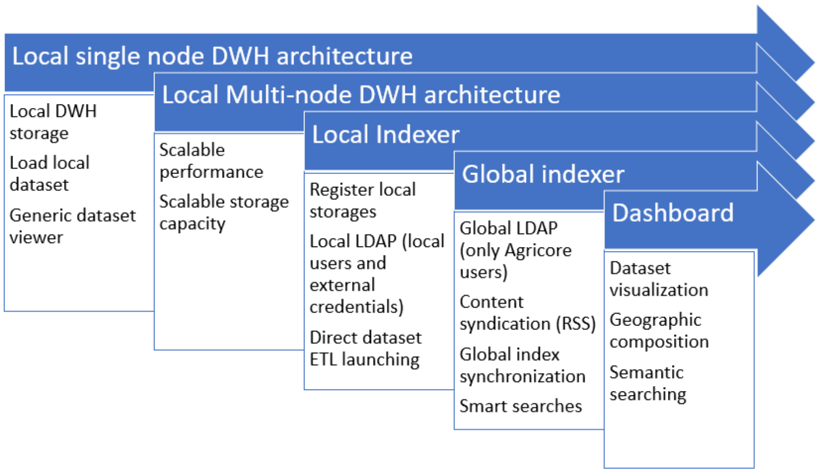
Local single node DWH architecture
Local DWH storage
Load local dataset
Generic dataset viewer
Local Multi-node DWH architecture
Scalable performance
Scalable storage capacity
Local Indexer
Register local storages
Local LDAP (local users and external credentials)
Direct dataset ETL launching
Global indexer
Global LDAP (only Agricore users)
Content syndication (RSS)
Global index synchronization
Smart searches
Dashboard
Dataset visualization
Geographic composition
Semantic searching

Figure 17 DWH architecture delivery strategy

3.1.1 Local single node DWH architecture

The objective in this first phase would be to deploy a preliminary version of the local DWH architecture. This version includes the possibility of deploying a very simple version of the DWH on users' computers, using all the technologies proposed in the DWH architecture (HDFS, YARN, Hive, Spark, Sqoop and Zeppelin), but with no scalability and processing limited to a single node, with no data distribution. This version will be used to test the functionality and performance of the tools and technologies used and the loading of locally stored data sources in different formats using a single computer. In addition, this DWH version will be limited to the exclusive use of AGRICORE project staff and specifically to the partners involved in the task, so its distribution will not be public during the first deployment phases. For downloading, the code will be uploaded to the AGRICORE's private GitLab repositories, along with a user manual for its deployment and use on local computers.

The following are the minimum hardware requirements to be able to deploy this version of the data warehouse on a local computer:

- 8 GB RAM memory.
- 4-8 CPU cores with 2-2.5GHz at least of CPU speed.
- For the storage of the data contained in the DWH, the system will need at least 256GB.
- Only a simple internet connection would be required to download and install the software.

3.1.2 Local multi-node DWH architecture

This second phase aims to extend the performance of the DWH version delivered in the first phase, enabling scalability and distributed processing using a multi-node strategy, and therefore allowing the DWH to be deployed on multiple machines. This will allow to increase horizontal and vertical scalability and hence the available resources for each node or the number of worker nodes, depending on the processing and storage needs. As well as the first version, this second one will be also used for the testing of locally stored data sources and its use will also be restricted to the partners involved in the project.

The minimum hardware requirements required to deploy a multi-node version of the data warehouse is as follows:

- 32 GB RAM memory.
- 8-16 CPU cores with 2-2.5GHz at least of CPU speed.
- The system required at least 512GB of disk storage.
- A network bandwidth (ethernet card) with at least 1 Gbps is required.

3.1.3 Local Indexer

In this third phase, the connection with the local version of the ARDIT indexer (Local Indexer) would be added to the existing version of the DWH, enabling the possibility to access and use ARDIT to launch ETL processes to load data sources stored locally in the DWH. This new step will make the complete local DWH architecture available. Both, the Local Indexer and the already existing and updated version of the multi-node DWH will be available separately for download from the GitLab repositories, so they will be downloadable independently and their installations will have no dependencies between them.

In addition, during this phase, once the sandbox with the local tool suite is available for installation, it could be released to new project stakeholders in order to receive increased testing of the tools and feedback.

The minimum requirements needed to deploy both data warehouse and local indexer in a multi-node approach are detailed in the next points:

- 32 GB RAM memory.
- 8-16 CPU cores with 2-2.5GHz at least of CPU speed.
- 1 TB of disk capacity is required to store the data in the DWH.
- A network bandwidth (ethernet card) with at least 1 Gbps is required.

3.1.4 Global Indexer

The fourth phase of the deployment comprises the inclusion of the Global Indexer, the fully publicly accessible version of ARDIT and the deployment of the DWH in the cloud. This second version will enable the use of ARDIT to store data sources, potentially useful for the AGRICORE project purposes, within the DWH exploited on cloud, being this functionality limited to a smaller set of users. Besides, the connection between the Global Indexer and the locally installed indexers will also be enabled. In this way, the latter will be able to subscribe to the Global Indexer using RSS to obtain notifications about possible updates in the characterised dataset collections, allowing users to update them in their local versions and synchronise the databases with the Global Indexer.

It is important to clarify that in this case, the Global Indexer will be uploaded to private GitLab repositories due to its internal use for the AGRICORE project. During this phase, considering the synchronisation between the Global and the local indexers, it could be discussed the possibility of making public the complete DWH local architecture following the open-source guidelines and objectives of the project.

The minimum requirements to deploy the whole system in a cloud-based solution are [31]:

- 48 GB RAM memory.
- 8-16 CPU cores with 2-2.5GHz at least of CPU speed.
- 1 TB of disk capacity is required to store all the data in the DWH.
- A network bandwidth (ethernet card) with at least 1 Gbps is required.

3.1.5 Dashboard

The last phase would be related to the AGRICORE dashboard deployment, a desktop application that would play an essential role in the simulation execution and the visualisation and graphical representation of the results obtained, for their analysis. The dashboard would replace Apache Zeppelin as the main tool for data visualisation. Until this phase, Zeppelin was the only way for the users to interact with the DWH directly. The AGRICORE dashboard would facilitate this process for non-experienced users, as Zeppelin requires the use of a programming language in order to be used, and would mean the inclusion of the last remaining component in the complete DWH architecture.

4 Governance Implementation

The concept of governance in software development can be defined as "*mechanisms to enable software developers, project managers and others within a software development organization to carry out their roles and responsibilities*" [32], being a key point that allows that the software development aligns with the objectives, requirements and goals defined for the project. Sometimes, software development is not a definite process whose phases have well-defined deadlines. The development may change thought the project life cycle, for example, introducing possible improvements, solutions to early-stage errors or problems, changes in a module, component, tool, plugin or library that became obsolete or needs to be updated, or simply because new requirements arise. Therefore, it is very important to define who manages or analyses the possible changes that may occur in the software development, facing an evolving set of requirements, and also, to define how these changes should be implemented [33][34][35][36]. The following figure (extracted from [37]) illustrates the software development life cycle:

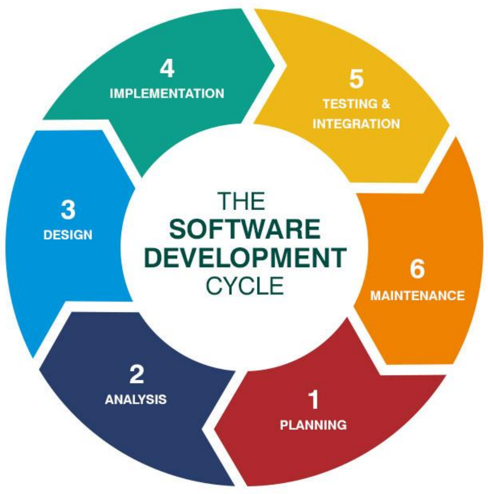

Figure 18 Software development life cycle

In addition, during each phase of the software project life cycle, it is necessary to review the work done to check that the development team has achieved the defined goals and objectives, or, for example, if the defined processes, quality and best practices standards have been applied correctly. Consequently, it is necessary for the governance plan to include clear guidance on roles and responsibilities, defining who does what and how they should do it, and ensuring that all personnel involved in the development follow the agreed plan.

The following figure (extracted from [38]) illustrates a cyclical governance process showing the different processes described above, for example, the assignment of roles and responsibilities, the decentralisation of control and management, the monitoring or the definition of metrics to measure the progress of the project towards the strategic objectives:

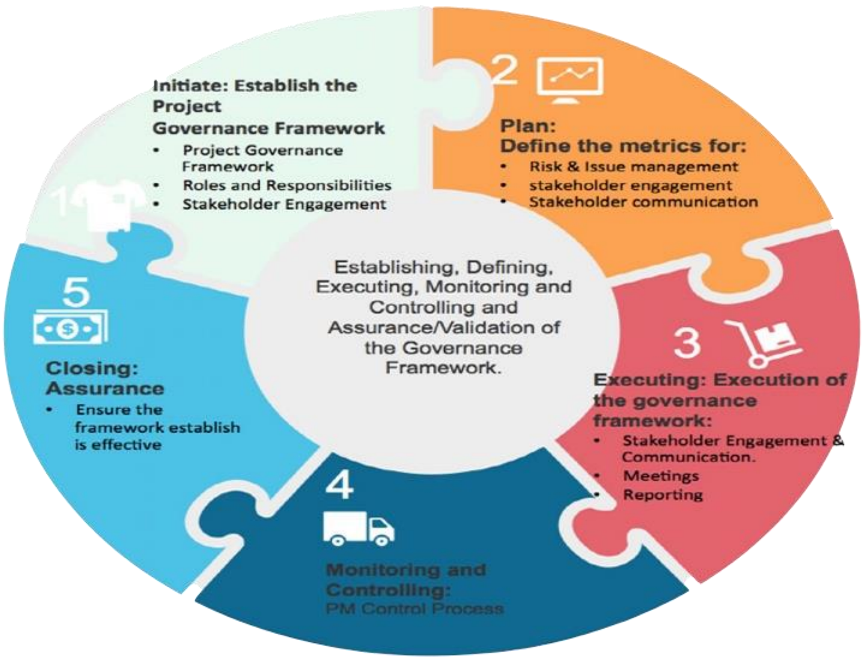
Initiate: Establish the Project Governance Framework
• Project Governance Framework
• Roles and Responsibilities
• Stakeholder Engagement
2
Plan:
Define the metrics for:
• Risk & Issue management
• stakeholder engagement
• Stakeholder communication
Establishing, Defining, Executing, Monitoring and Controlling and Assurance/Validation of the Governance Framework.
3
Executing: Execution of the governance framework:
• Stakeholder Engagement & Communication.
• Meetings
• Reporting
4
Monitoring and Controlling:
PM Control Process
5
Closing: Assurance
• Ensure the framework establish is effective

Figure 19 Project governance life cycle

A good governance plan would also allow decentralized software development, so no single individual or group would be central or decisive in decision-making. A good distribution of responsibilities would encourage creativity and transparency towards the rest of the stakeholders involved, creating a space for collaboration. Although governance plans are generally applied at company or project level, and based on the existence of governance frameworks for different business objectives (ITIL, COBIT, COSO, CMMI, FAIR), it is important to define in the DWH governance the roles and responsibilities of the individuals or groups involved in the development processes of this AGRICORE project module. For this purpose, a set of processes have been defined based on the different stages within the software project life cycle:

4.1 Monitoring

Includes the tasks of supervising the correct development and implementation of the data warehouse within the project objectives. This task is mainly conducted by IDE who, as project coordinating partners, ensure that the AGRICORE projects objectives, in general, and the DWH architecture objectives, in particular, are achieved.

Monitoring is also a task that concerns all partners involved in it, controlling the correct functioning of the DWH and analysing possible new requirements, improvements, errors or risks.

4.2 Architecture

Comprises the architecture design tasks and the selection of technologies, mechanisms and processes necessary to meet the requirements defined for the data warehouse. This task is conducted by AAT, as the leader of the DWH design and development, with the support of the rest of the partners, defining the functionalities necessary to integrate other modules that produce data inputs and outputs in the DWH.

4.3 Development and Deployment

The development and deployment processes refer to the development and implementation tasks of the data warehouse itself, based on the technologies, tools and mechanisms defined in the architecture and also its deployment following the plan defined in the previous chapter. As the last process, it is a task conducted mainly by AAT, as leader of the DWH development, with the support of partners that will produce modules that interact directly with the DWH to transfer, process or store data.

4.4 Maintenance

The maintenance process covers the tasks necessary to ensure the correct functioning of the data warehouse within the AGRICORE suite, managing possible errors, problems, system crashes, upgrades, etc. It is a task performed by AAT, as DWH developers, throughout the project lifecycle with the support of the rest of the partners responsible for monitoring it.

4.5 Retrospective

The retrospective includes the study and analysis of the developed solution to find or detect possible new requirements, standards, technologies or tools to promote a continuous improvement, if possible, of the DWH architecture and system. This task will be mainly conducted by AAT, as DWH developers, and by IDE, as the main project coordinators and supervisors.

4.6 Support

The support tasks comprise the assistance in design, implementation and maintenance of the DWH during the entire project life cycle. All partners involved in this task will provide support to AAT. IDE, AUTH and UNIPR will contribute to the definition of functionalities for the automation of data processing and synthetic population generation. The rest of the partners will provide inputs regarding the requirements for including and processing the information from the data sources analysed in WP1.

5 Conclusions

The deliverable D2.1 Data Warehouse (DWH) for agricultural policy impact assessment data management provides a detailed description of the data warehouse role within the AGRICORE project as a fundamental part of the architecture for the storage and exchange of data. The DWH will act as a key point for the communication with other project modules and will be vital to launch simulations in order to achieve the project's policy assessment objectives. In this way, the following points have been established in the document:

- The definition of an extensive list of functional and non-functional requirements applicable throughout the project lifecycle.
- The definition of an architecture that meets the objectives of the AGRICORE project for building modular software based on distributed architectures, capable of being deployed in public or private cloud infrastructures and using containerisation technologies, to ensure the distribution and scalability of operations. In addition, the connection between the data source index tool, ARDIT, and the data warehouse has been extensively described in order to store data useful for the research and analysis processes of the project.
- An architecture deployment plan divided in several phases, each one of them increasing the features, functionalities and capabilities of the DWH. Furthermore, the hardware requirements necessary to deploy the DWH have been defined.
- The responsibilities and roles of all partners involved in the task of developing the DWH, throughout the project lifecycle, have been defined.
- A reference point to understand how the AGRICORE project data warehouse works, especially for those partners who are going to develop modules connected to it to ingest, exchange, process and store data.

6 References

[1] ^W. H. Inmon, What is a Data Warehouse? Prism, 1995.

[2] ^R. Kimball, The Data Warehouse ETL Toolkit, 4th ed. Wiley, 2005.

[3] ^Astera, "Data Warehouse Concepts: Kimball vs. Inmon Approach." [Online]. Available: https://www.astera.com/type/blog/data-warehouse-concepts

[4] ^Oracle, "What Is a Data Warehouse?" [Online]. Available: https://www.oracle.com/database/what-is-a-data-warehouse

[5] ^Amazon, How do data warehouses, databases, and data lakes work together? [Online]. Available: https://d1.awsstatic.com/diagrams/product-page-diagrams/database-seo-1.270cb06b819915c5f763a0b9f88255e044c4dac5.png

[6] ^Amazon, "Data Warehouse concepts." [Online]. Available: https://aws.amazon.com/data-warehouse/?nc1=h_ls

[7] ^Nuclino, "A Guide to Functional Requirements." [Online]. Available: https://www.nuclino.com/articles/functional-requirements

[8] ^T. R. V. of Wikipedia, "Non-functional requirement." [Online]. Available: https://thereaderwiki.com/en/Non-functional_requirement

[9] ^S. S. R. Chansler H. Kuang, S. Radia, K. Shvanchko, "The Hadoop Distributed File System." [Online]. Available: http://www.aosabook.org/en/hdfs.html

[10] ^Hadoop, Hadoop Distributed File System. [Online]. Available: https://www.sciencedirect.com/topics/computer-science/hadoop-distributed-file-system

[11] ^Apache, "Apache Hadoop." [Online]. Available: https://hadoop.apache.org

[12] ^Apache, "Apache Hadoop documentation." [Online]. Available: https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/SingleCluster.html

[13] ^T. A. S. Foundation, "Apache Hadoop YARN." [Online]. Available: https://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html

[14] ^Apache, Apache Hadoop YARN. [Online]. Available: https://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html

[15] ^stackchief, MapReduce Quick Explanation. [Online]. Available: https://www.stackchief.com/blog/MapReduce%20Quick%20Explanation

[16] ^T. A. S. Foundation, "APACHE HIVE TM." [Online]. Available: https://hive.apache.org/

[17] ^Apache, "Apache Hive." [Online]. Available: https://hive.apache.org

[18] ^Apache, "Apache Hive documentation." [Online]. Available: https://cwiki.apache.org/confluence/display/Hive/GettingStarted

[19] ^J. T. Point, Hive Architecture. [Online]. Available: https://www.javatpoint.com/hive-architecture

[20] ^T. A. S. Foundation, "Apache Spark is a unified analytics engine for large-scale data processing." [Online]. Available: https://spark.apache.org/

[21] ^Apache, "Apache Spark." [Online]. Available: https://spark.apache.org

[22] ^Apache, “Apache Spark documentation.” [Online]. Available: https://spark.apache.org/docs/latest

[23] ^Orelly, Introduction to Data Analysis with Spark. [Online]. Available: https://www.oreilly.com/library/view/learning-spark/9781449359034/ch01.html

[24] ^Apache, Cluster Mode Overview. [Online]. Available: https://spark.apache.org/docs/latest/cluster-overview.html

[25] ^T. A. S. Foundation, “Apache Sqoop.” [Online]. Available: https://sqoop.apache.org/

[26] ^Apache, “Apache Sqoop.” [Online]. Available: https://sqoop.apache.org

[27] ^Apache, “Apache Sqoop documentation.” [Online]. Available: https://sqoop.apache.org/docs/1.4.7/SqoopUserGuide.html

[28] ^Apache, Big Data/Hadoop. [Online]. Available: https://cloudgleebigdata.blogspot.com/2014/02/sqoop-concepts-and-installation.html

[29] ^T. A. S. Foundation, “Apache Zeppelin.” [Online]. Available: https://zeppelin.apache.org/

[30] ^Apache, “Apache zeppelin.” [Online]. Available: https://zeppelin.apache.org

[31] ^Informatica, “Hadoop Cluster Hardware Recommendations.” [Online]. Available: https://docs.informatica.com/data-engineering/data-engineering-integration/h2l/1415-tuning-and-sizing-guidelines-for-data-engineering-integrati/tuning-and-sizing-guidelines-for-data-engineering-integration–1/sizing-recommendations/hadoop-cluster-hardware-recommendations.html

[32] ^S. C. Williams Clay and A. Yaeli, “Software development governance and its concerns,” Jan. 2008, doi: 10.1145/1370720.1370723.

[33] ^V. Hines, “5 Benefits of Good Project Governance.” [Online]. Available: https://wellingtone.co.uk/5-benefits-of-good-project-governance/

[34] ^T. Development, “Importance of Good Governance Processes in Software Development.” [Online]. Available: https://www.tiempodev.com/blog/importance-of-good-governance-processes-in-software-development/

[35] ^V. Georgescu, “What Is IT Governance? Understanding From First Principles.” [Online]. Available: https://www.plutora.com/blog/it-governance

[36] ^A. Finkelstein, “Software Engineering Governance: a briefing.” [Online]. Available: https://www.cs.uoregon.edu/events/icse2009/images/postConf/TB-Governance-ICSE09.pdf

[37] ^BigWater, Software Development Life Cycle (SDLC). [Online]. Available: https://bigwater.consulting/2019/04/08/software-development-life-cycle-sdlc/

[38] ^PMI, Project governance. [Online]. Available: https://www.pmi.org/learning/library/project-governance-critical-success-9945)

Apart from these references, for preparing this report, the following documents have been taken into consideration:

- AGRICORE Proposal: project proposes a novel tool for improving the current capacity to model policies dealing with agriculture by taking advantage of the latest progress in modelling approaches and ICT.

- AGRICORE Grant Agreement ANNEX 1 Part A and B, Research and Innovation action, Number-816078: Official Grant Agreement of the AGRICORE project, which defined the terms and conditions of the project, as well as the main requirements of the project.

Deliverable Number	Deliverable Title	Lead beneficiary	Type	Dissemination Level	Due date
D4.1	AGRICORE requirements and project management platform	AAT	Report	Public	M12 31-ago-2020
D6.1	AGRICORE Architecture and Interfaces	IDE	Report	Public	M24 31-jul-2021