Big Data to Enable Global Disruption of the Grapevine-powered Industries

D2.3 - BigDataGrapes Software Stack Design

DELIVERABLE NUMBER	D2.3
DELIVERABLE TITLE	BigDataGrapes Software Stack Design
RESPONSIBLE AUTHOR	Nikos Manouselis (Agroknow)

Co-funded by the Horizon 2020
Framework Programme of the European Union

GRANT AGREEMENT N.	780751
PROJECT ACRONYM	BigDataGrapes
PROJECT FULL NAME	Big Data to Enable Global Disruption of the Grapevine-powered industries
STARTING DATE (DUR.)	01/01/2018 (36 months)
ENDING DATE	31/12/2020
PROJECT WEBSITE	http://www.bigdatagrapes.eu/
COORDINATOR	Nikos Manouselis
ADDRESS	110 Pentelis Str., Marousi, GR15126, Greece
REPLY TO	nikosm@agroknow.com
PHONE	+30 210 6897 905
EU PROJECT OFFICER	Ms. Annamária Nagy
WORKPACKAGE N. | TITLE	WP2 | Grapevine-powered Industries Big Data Challenges
WORKPACKAGE LEADER	Agroknow
DELIVERABLE N. | TITLE	D2.3 | BigDataGrapes Software Stack Design
RESPONSIBLE AUTHOR	Panagis Katsivelis (Agroknow)
REPLY TO	katsivelis.panagis@agroknow.com
DOCUMENT URL	http://www.bigdatagrapes.eu/
DATE OF DELIVERY (CONTRACTUAL)	30 June 2018 (M6), 30 June 2019 (M18, Updated Version), 31 December 2019 (M24, Updated Version)
DATE OF DELIVERY (SUBMITTED)	30 June 2018 (M6), 28 June 2019 (M18, Updated Version), 03 January 2020 (M25, Updated Version)
VERSION | STATUS	3.0 | Final
NATURE	R (Report)
DISSEMINATION LEVEL	PU (public)
AUTHORS (PARTNER)	Pythagoras Karampiperis (Agroknow), Antonis Koukourikos (Agroknow), Raffaele Perego (CNR), Franco Maria Nardini (CNR), Nicola Tonellotto (CNR), Milena Yankova (ONTOTEXT), Mihalis Papakonstadinou (Agroknow), Panagiotis Zervas (Agroknow), Panagis Katsivelis (Agroknow)
CONTRIBUTORS	Sotiris Konstantinidis (Agroknow), Vinicius Monteiro de Lira (CNR), Vladimir Alexiev (ONTOTEXT), Mele Ida (CNR)
REVIEWER	Franco Maria Nardini (CNR)

VERSION	MODIFICATION(S)	DATE	AUTHOR(S)
0.1	User Roles Identification	13/04/2018	Raffaele Perego (CNR), Franco Maria Nardini (CNR), Nicola Tonellotto (CNR), Pythagoras Karampiperis (Agroknow), Panagiotis Zervas (Agroknow)
0.2	Functional & non-functional requirements specification	27/04/2018	Raffaele Perego (CNR), Franco Maria Nardini (CNR), Nicola Tonellotto (CNR), Antonis Koukourikos (Agroknow), Pythagoras Karampiperis (Agroknow), Panagiotis Zervas (Agroknow), Milena Yankova (ONTOTEXT)
0.3	Information Flow Design	11/05/2018	Antonis Koukourikos (Agroknow), Pythagoras Karampiperis (Agroknow), Raffaele Perego (CNR), Franco Maria Nardini (CNR), Nicola Tonellotto (CNR)
0.4	Architecture Layer Definition	18/05/2018	Raffaele Perego (CNR), Franco Maria Nardini (CNR), Nicola Tonellotto (CNR), Pythagoras Karampiperis (Agroknow), AntonisKoukourikos (Agroknow), Sotiris Konstantinidis (Agroknow), Panagiotis Zervas (Agroknow), Milena Yankova (ONTOTEXT)
0.5	Report Document Setup	01/06/2018	Pythagoras Karampiperis (Agroknow), AntonisKoukourikos (Agroknow), Panagiotis Zervas (Agroknow)
0.6	Architecture Design Refinements	08/06/2018	Raffaele Perego (CNR), Franco Maria Nardini (CNR), Nicola Tonellotto (CNR), Milena Yankova (ONTOTEXT)
0.7	Identification of Relevant Technologies	15/06/2018	Raffaele Perego (CNR), Franco Maria Nardini (CNR), Nicola Tonellotto (CNR), Mele Ida (CNR), Vinicius Monteiro de Lira (CNR), Antonis Koukourikos (Agroknow),Sotiris Konstantinidis (Agroknow), Milena Yankova (ONTOTEXT), Vladimir Alexiev (ONTOTEXT)
0.8	Final Report Draft	22/06/2018	Pythagoras Karampiperis (Agroknow), Antonis Koukourikos (Agroknow)
0.9	Internal Review	27/06/2018	Nicola Tonellotto (CNR)
1.0	Deliverable Finalisation	29/06/2018	Pythagoras Karampiperis (Agroknow), Antonis Koukourikos (Agroknow)
2.0	Updated version	28/6/2019	Panagiotis Zervas (Agroknow), Mihalis Papakonstadinou (Agroknow),
3.0	Updated version	03/01/2020	Panagis Katsivelis (Agroknow)

PARTICIPANTS		CONTACT
Agroknow IKE (Agroknow, Greece)	Agroknow	Nikos Manouselis Email: nikosm@agroknow.com
Ontotext AD (ONTOTEXT, Bulgaria)		Todor Primov Email: todor.primov@ontotext.com
Consiglio Nazionale Delle Richerche (CNR, Italy)		Raffaele Perego Email: raffaele.perego@isti.cnr.it
Katholieke Universiteit Leuven (KULeuven, Belgium)	KATHOLIEKE UNIVERSITEIT LEUVEN	KatrienVerbert Email: katrien.verbert@cs.kuleuven.be
Geocledian GmbH (GEOCLEDIAN Germany)	geocledian	Stefan Scherer Email: stefan.scherer@geocledian.com
Institut National de la Recherché Agronomique (INRA, France)	INRA SCIENCE & IMPACT	Pascal Neveu Email: pascal.neveu@inra.fr
Agricultural University of Athens (AUA, Greece)		Katerina Biniari Email: kbiniari@aua.gr
Abaco SpA (ABACO, Italy)		Simone Parisi Email: s.parisi@abacogroup.eu
SYMBEEOSIS LONG LIVE LIFE S.A. (Symbeeosis, Greece)	Symbeeosis	Kostas Gardikis Email: c.gardikis@gmail.com

ACRONYMS LIST

API	Application Programming Interface
NOSQL	Not-only SQL
RDF	Resource Description Framework
OWL	Web Ontology Language
SPARQL	SPARQL Protocol and RDF Query Language
GIS	Geographic Information System
JSON	JavaScript Object Notation

EXECUTIVE SUMMARY

Deliverable D2.3 is the first submitted iteration of a living document that will describe the components and architecture of the BigDataGrapes software stack, providing design principles and technical guidelines in order for the platform's functionalities to be aligned with the requirements of the end users represented in the project.

The current version of the specification is based on the preliminary analysis of the use cases provided by the three BigDataGrapes industrial partners, covering the basic functional and non-functional expectations from the platform in order for the latter to conform to the requirements of those use cases.

It is expected that, as the use cases are refined and executed, the overall specification and the corresponding architecture will evolve towards the new requirements that arise. To this end, the specifications document is treated as a living document, with regular submission to the EC of versions that report on significant changes in design and functionality.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

The purpose of this document is to drive research and development work in the context of the BigDataGrapes project. It will balance long-term strategic issues with immediate requirements of the project's current phase technical development.

At this first version of the deliverable, the architectural design of the BigDataGrapes software stack is based on the preliminary analysis of the use cases and datasets identified by the time of the deliverable's submission and the subsequent elicitation of functional and non-functional requirements for the platform.
While technical details are not thoroughly analysed at the present stage, the architecture clearly identifies candidate technologies, tools and frameworks to be used for the development of the platform. The selection is based on the proven performance and wide adoption of the technologies, ensuring the sustainability and efficiency of the envisioned solution.

The report is organised as follows: Section 2 presents the main user roles to be supported by the platform along with their main operations within the platform. Section 3 briefly presents the main categories of data expected to be processed within BigDataGrapes. Section 4 describes the non-functional requirements from the software stack, while Section 5 deals with the functional requirements.

Finally, Section 6 analyses the main information flows that will be executed over the platform, proceeds to the presentation of the architectural design of BigDataGrapes, placing the components participating in the flows to distinct, clearly defined layers and concludes with a summary of the technologies to be used for implementing the core components of the software stack.

2 BIGDATAGRAPES PLATFORM OVERVIEW AND SUPPORTED ACTORS

BigDataGrapes targets the technology challenges of the grapevine powered data economy as its business problems and decisions. Such challenges involve processing, analysis and visualisation of data with rapidly increasing volume, velocity and variety: satellite and weather data, environmental and geological data, phenotypic and genetic plant data, food supply chain data, economic and financial data and more.

The Platform aspires to advance the Big Data technical landscape by defining novel methods for distributed indexing, processing and inference, to produce consolidation and harmonization techniques that can efficiently operate over large data collections, and to develop scalable and efficient machine learning methodologies for generating predictive analytics over disparate collections of extremely large sizes.

At this stage, we identify four (4) main actors: Administrator, Developer, Data Scientist and Decision Maker. The core activities for each actor are summarised as follows:

- **Administrator**: User management; Access management; Applications and Components configuration;
- **Developer**: Component integration and deployment;
- **Data Scientist**: Configuration of the machine learning components integrated in the platform;
- **Analyst** / **Decision Maker**: Invocation and usage of the analytics and visualization components of the platform.

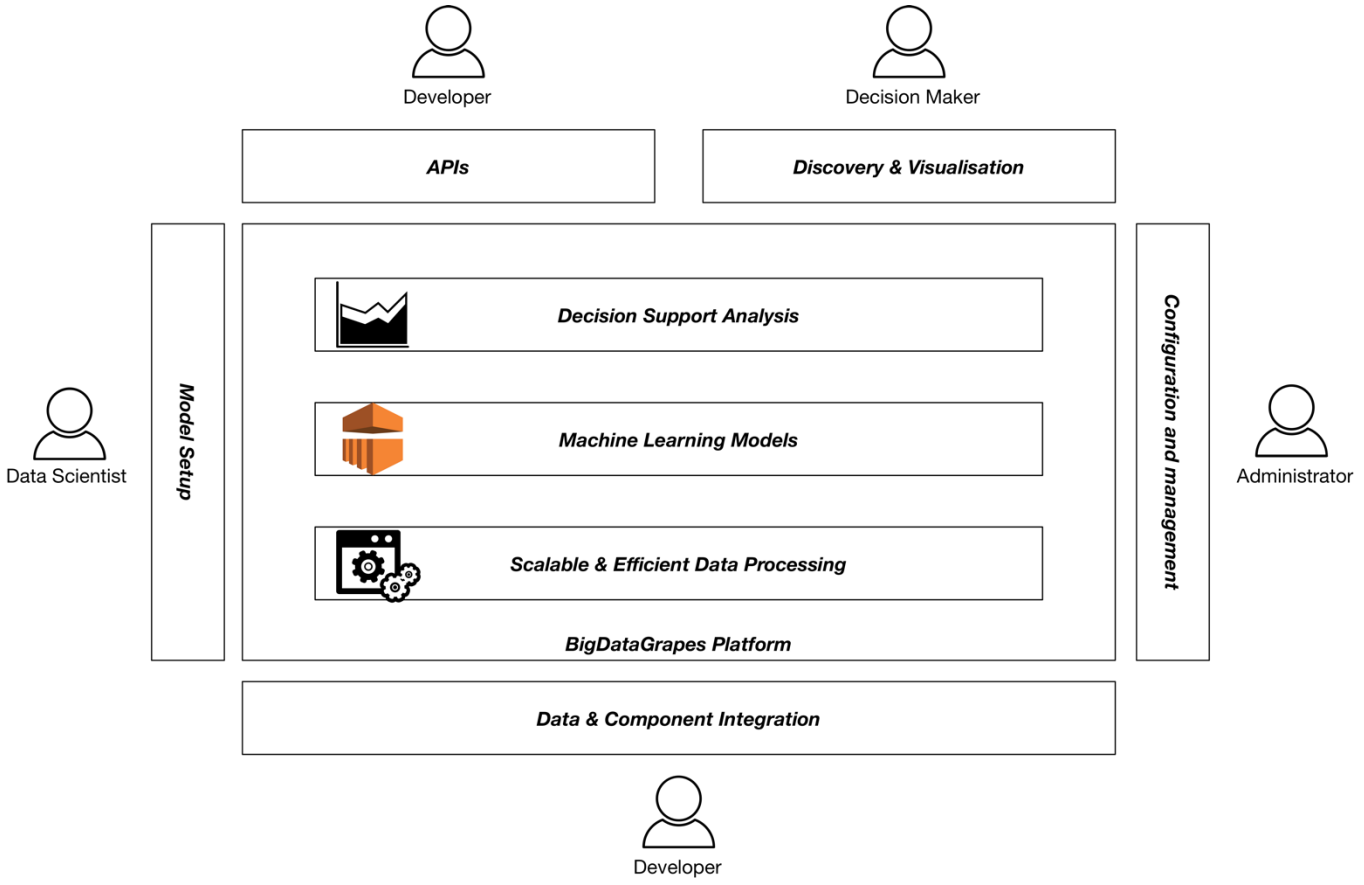

Figure 1: Logical Organisation of BDG Platform

Figure 1 illustrates the logical organization of the big data software stack of the BDG platform. The platform comprises components that implement novel methodologies for efficient large-scale data processing and prediction analysis to support business decisions in the wine grapes context. The components depicted in the figure will be exposed via clearly defined and thoroughly documented APIs and deployed on an integrated Big Data platform that will serve the BigDataGrapes use cases.

3 BIGDATAGRAPES DATA SOURCES AND TYPES

Figure 2 depicts a preliminary taxonomy of the data sources foreseen to be used by the project's use cases over the BigDataGrapes platform. This overview enables the analysis and understanding of the variety and characteristics of the different data to be processed and analysed via the platform's components.

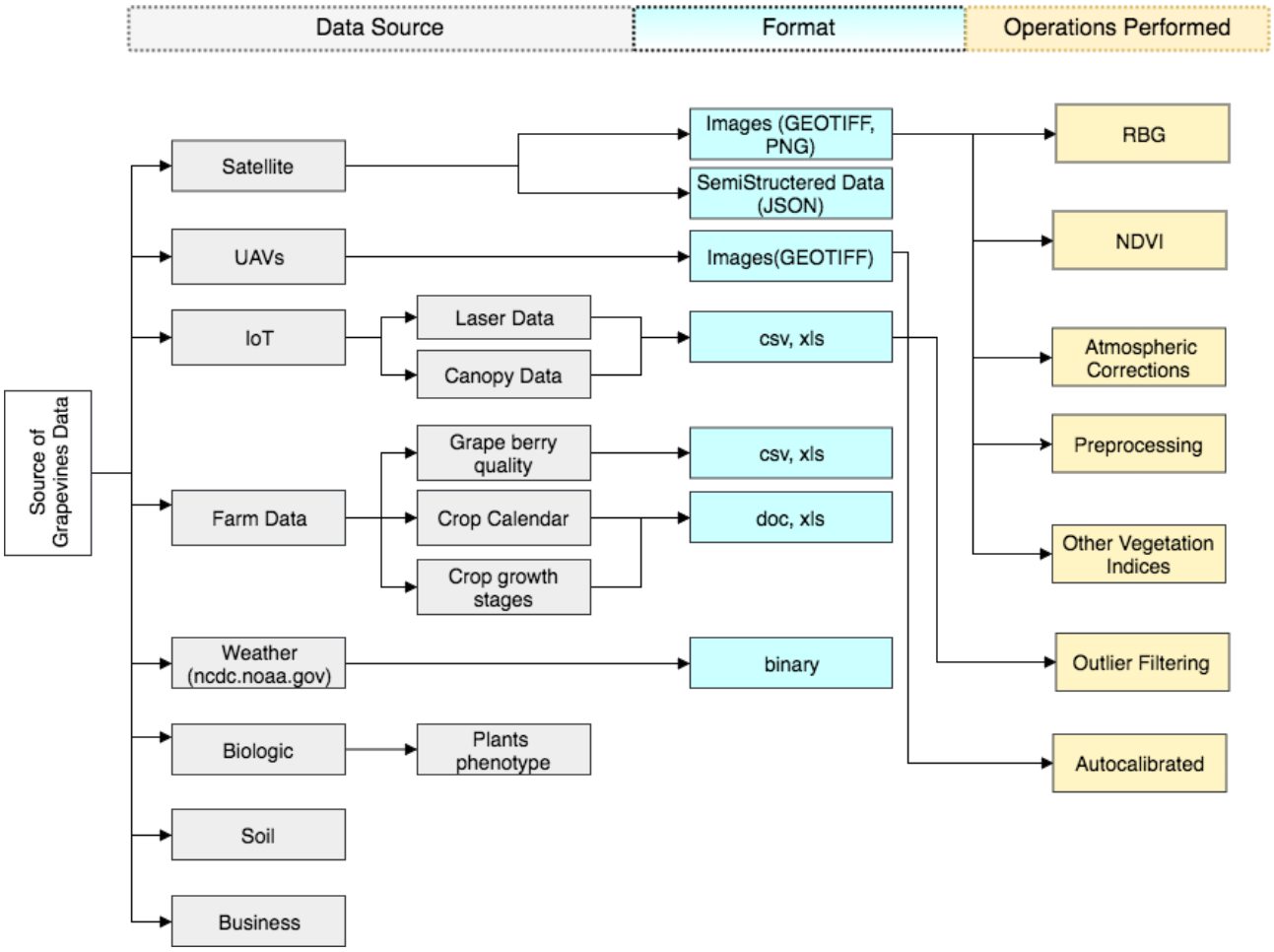

Figure 2: BigDataGrapes Data Sources

4 NON-FUNCTIONAL REQUIREMENTS

The section summarises the non-functional requirements for the BigDataGrapes platform.

4.1 STREAMLINED, EFFICIENT DEPLOYMENT STRATEGY

Identifier	Description
RNF/Deployment-01	The platform components must be designed in order to enable their deployment in both cloud infrastructures and local servers.
RNF/Deployment-02	The platforms components will be virtualised as docker containers. For this purpose, the docker image of each component must be self-contained, including the necessary library dependencies.
RNF/Deployment-03	Complex platform components will be defined as multi-container Docker applications to facilitate their setup and deployment
RNF/Deployment-04	Each platform component that composes a Docker image must provide two bash scripts to facilitate its building and running: *build.sh* and *run.sh* respectively.
RNF/Deployment-05	Each platform component must be capable of being deployed independently from all other components.
RNF/Deployment-06	The production environment must contain only published components. For a component to be considered formally published, it must complete all procedures defined within the development and testing phases of its implementation.

4.2 DEVELOPMENT

Identifier	Description
RNF/Development-01	The source code for each component will be versioned and monitored using Git repositories. GitHub will be used for setting up and managing these repositories. Each platform component will be associated with its own GitHub project.
RNF/Development-02	A testing environment will be setup and used to perform tests during the development of the BigDataGrapes components
RNF/Development-03	The platform components must be highly modular.

4.3 DOCUMENTATION

Identifier	Description
RNF/Documentation-01	The platform components will be individually documented. The documentation must include at least the following aspects: - Setup instructions; - Technologies / frameworks used; - Input / Output specification;

	- Data and Process Flow description and diagram; - Use Cases relevant to the component.
RNF/Documentation-02	Interfacing components must carry a specification of their API and an exemplary API testing interface.

4.4 FAULT TOLERANCE

Identifier	Description
RNF/FaultTolerance-01	Each component must be designed to allow multiple servers to handle incoming tasks.

4.5 CONTINUOUS INTEGRATION

Identifier	Description
RNF/Integration-01	The platform components must be continuously integrated. The tool GitLab-CI will be used for carrying out the integration task.
RNF/Integration-02	The Docker Swarm Clustering Manager will be used to orchestrate the Docker containers of the platform components.

4.6 INTEROPERABILITY

Identifier	Description
RNF/Interoperability-01	Interfacing components must expose their functionality via RESTful APIs.

4.7 LICENSING

Identifier	Description
RNF/Licensing-01	When applicable, platform components will be released under a license that permits their reuse and redeployment. In the cases where proprietary rights are established, components and technologies will be made available under the restrictions defined in the Consortium Agreement of the project.

4.8 SCALABILITY

Identifier	Description
RNF/Scalability-01	Although the platform aims to cover a specific set of use cases, it must be designed to serve a high number of applications/systems.
RNF/Scalability-02	The platform must enable the individual applications and systems to expand their processing capabilities upward and outward to support scaling up.

4.9 OPEN-SOURCE

Identifier	Description
RNF/OpenSource-01	Any code accompanying research results achieved within the project will be open-sourced and accompany relevant publications / demonstrations.
RNF/OpenSource-02	When open-source third-party libraries can be used for the implementation of the platform, they will be preferred to proprietary /closed-source libraries.

4.10 PERFORMANCE AND RESPONSIVENESS

Identifier	Description
RNF/Performance-01	The Platform must ensure acceptable response times for all envisioned use cases.

4.11 TESTABILITY

Identifier	Description
RNF/Testability-01	Platform components will be tested via the appropriate scenarios both in isolation and within an integrated setting.
RNF/Testability-02	Tests must be replicable and must be provided along with the component's distribution.

4.12 USABILITY

Identifier	Description
RNF/Usability-01	Users with an internet connection and proper credentials may access the platform.
RNF/Usability-02	An access token must be provided when consuming the interfaces exposed by the platform. Each application supported by the platform must have its own access token.

5 FUNCTIONAL REQUIREMENTS

The present section identifies the main functional requirements from the platform by each of the foreseen BigDataGrapes use cases.

5.1 USE CASE A: DATA ANOMALY DETECTION

RF-A.01	EO Data Anomaly Detection & Classification
Description	In order to make efficient use of Earth Observation (EO) data for Farm Management applications, it is crucial to be able to differentiate between data issues and anomalies.
Objectives	- Develop a data pre-processing component able to classify between EO data issues and anomalies. This component will rely on machine learning/deep learning libraries in order to perform the classification task. - Develop a data pre-processing component able to automatically send warning message to the specialist actors in response to certain events such that data anomaly detection.

5.2 USE CASE B: PREDICTION

RF-B.01	Yield Prediction
Description	Provide decision makers with yield estimations that may affect variable rate applications, the timing of harvest operations, as well as storage and shipping of production.
Objectives	- Develop a data analytics component performing yield estimations, leveraging existing wealth of geospatial, weather, soil data and vegetation indices.

RF-B.02	Predicting Biological Efficacy
Description	There is a need in extracting the most out of pharmaceutical plants for both economic and environmental reasons. A real challenge is to add high value to by-products. The goal of the pilot is to prove the correlation between data from other scenarios to the quality of extracts from vine materials developed (grape seed oil, vine leaf extracts etc)
Objectives	- Develop a data analytics component able to support decision making

RF-B.03	Crop Quality Prediction for Optimizing Post Harvest Treatments of Table Grapes
Description	Table grapes quality affects selling price, storability duration and post-harvest treatments. Moreover, studies indicate that consumers from different countries have different preferences on consuming agricultural products. Crop quality also determines the table grapes harvest. Therefore, the need for crop quality

	prediction is of great significance.
Objectives	- Develop a data analytics component able to support crop quality prediction

RF-B.04	Crop Quality Prediction for Optimising Wine Making
Description	Winemaking needs knowledge of the grape's quality at harvest. Different sugar content in wine grapes can produce wine with different characteristics. As a result, winemakers have no idea on the quality of wine grapes that they buy from growers and adapt their vinification processes accordingly while the last cannot achieve higher selling prices for their products due to better quality.
Objectives	- Develop a data analytics component able to support selling price decisions.

5.3 USE CASE C: FARM MANAGEMENT

RF-C.01	Optimisation of Farm Practices in the Vineyard
Description	Use soil moisture and temperature profiling satellite data to define management practices, such as irrigation, fertilization, and application of phytochemicals, in order to improve grape quality and quantity.
Objectives	- Develop a data analytics component able to support management decisions on the field.

RF-C.02	Management Zones Delineation for Vineyards
Description	Delineation of management zones has provided many advantages to the wine grape growers. A holistic approach that includes imagery from UAVs and satellites, weather data from infield weather stations and open source weather along with in field non-destructive measurements will support to delineate better management zones and take full benefit of his crop production
Objectives	- Develop a data analytics component able to support management decisions concerning the delineation of management zones.

5.4 OTHER USAGE SCENARIOS

RF-O.01	Geospatial and Image Raster Processing
Description	Uses of spatial data types (i.e. geometry and rasters) and make use of the spatial processing and analytics functions and indexes.
Objectives	- Develop a data ingestion component able to support such spatial and raster operations in parallel.

RF-O.02	Storage Scalability
Description	The system is currently facing issues with data storage scalability. It is limiting the scalability of processing nodes and connected DBs. The system is quickly reaching the practical limits of DB sizes (on physical machines about 1 TB due to the necessity to create backups) - on VM it is 200GB - 500 GB due to the size of storage available on standard VMs.
Objectives	- Develop a persistent layer able to provide scalable storage and fault tolerance to the data access.

6 ARCHITECTURE OVERVIEW

6.1 DATA AND INFORMATION FLOW

The conceptual operational entities described in section 7 will exchange information in ways defined by specific workflows as derived by the general use case requirements from the platform. Figure 3 depicts the information exchange foreseen for the different components comprising the BigDataGrapes architecture.

The information entry point for the system is the *Data Ingestion Layer*, responsible for accessing the relevant data sources for a given operation. Towards this, the access mechanism for each of the sources should be identified or implemented if it does not already exist.

The next step is the possible *Pre-processing* of the ingested data. This includes statistical analysis of the targeted content towards anomaly and outlier detection, as well as, linguistic analysis (POS-tagging, stemming, named entity recognition, etc.) towards the next step which is the semantic enrichment of the data. The *Semantic Enrichment Components* carrying out the latter will comprise the required modules for entity matching and classification, ontology population, relation extraction, etc. For the tasks, the components will make use of the BigDataGrapes GraphDB instance. The pool will provide the BigDataGrapes model that will incorporate the conceptualisation of all entities foreseen in the available data, including mappings with existing external ontologies.

The pre-processed datasets will be stored in the persistence infrastructure of the project. Depending on the nature of the data, they will be stored in distributed storage systems as graphs, traditional databases or spatial databases. To tackle the challenges derived from the amount of data and performance requirements of the envisioned use cases, the data will be distributed in multiple instantiations of the storage systems relevant to each data type. These instances will be accessed and managed by the appropriate *Federators*, which will be the components accessible to the rest of the components and will hide the complexities of the storage mechanisms, providing a single access point for each federation.

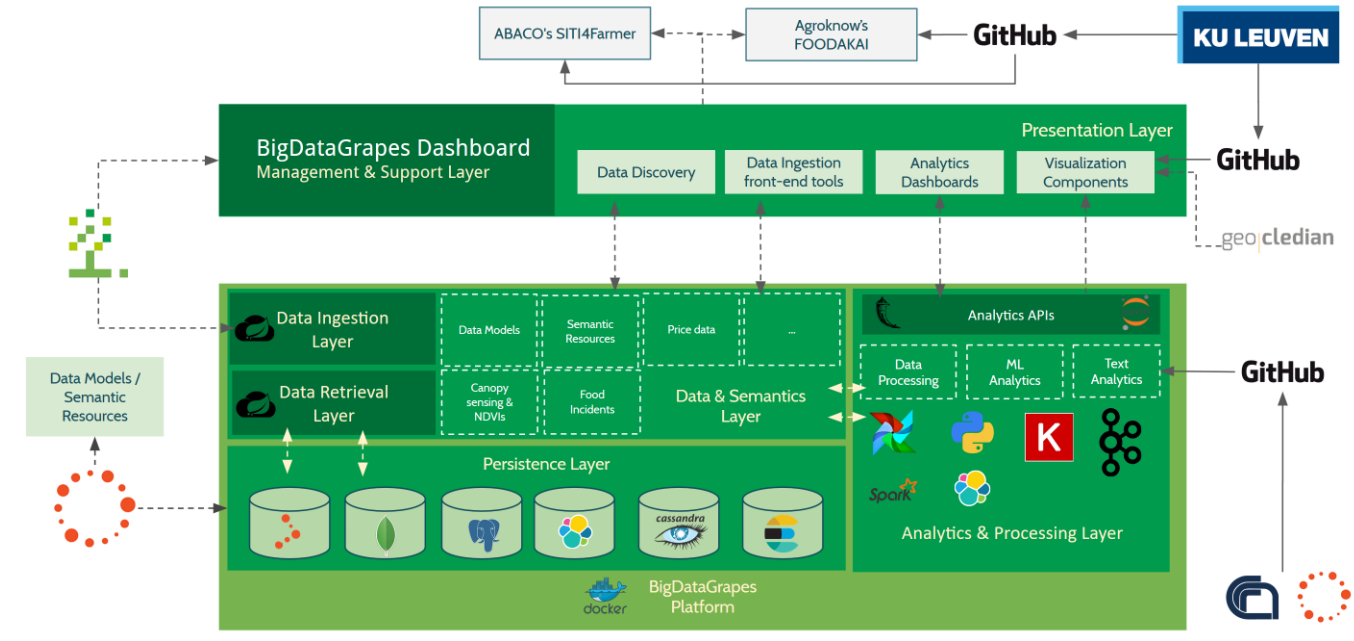

Figure 3: BigDataGrapes Top-level architecture

Following the completion of the pre-processing, enrichment and storage procedures for the ingested datasets, the data will be made available to the various analytics components included in the BigDataGrapes platform. Each of the analytics components pose different requirements for its input, thus the architecture foresees the appropriate *Data Preparation routines*, in order to aggregate and format the data to be

processed in a way suitable for a given component. The produced analytics are also persistently stored in the underlying storage infrastructure.

Following the analysis of the data, the final stage of the foreseen information flow is the presentation of the analysis results to the relevant end user. This is accomplished by the BigDataGrapes *Presentation Layer* that comprises all analytics dashboards and visualization components.

6.2 ARCHITECTURE LAYERS

All the platform architectural layers mentioned above are further analysed in Figure 4.

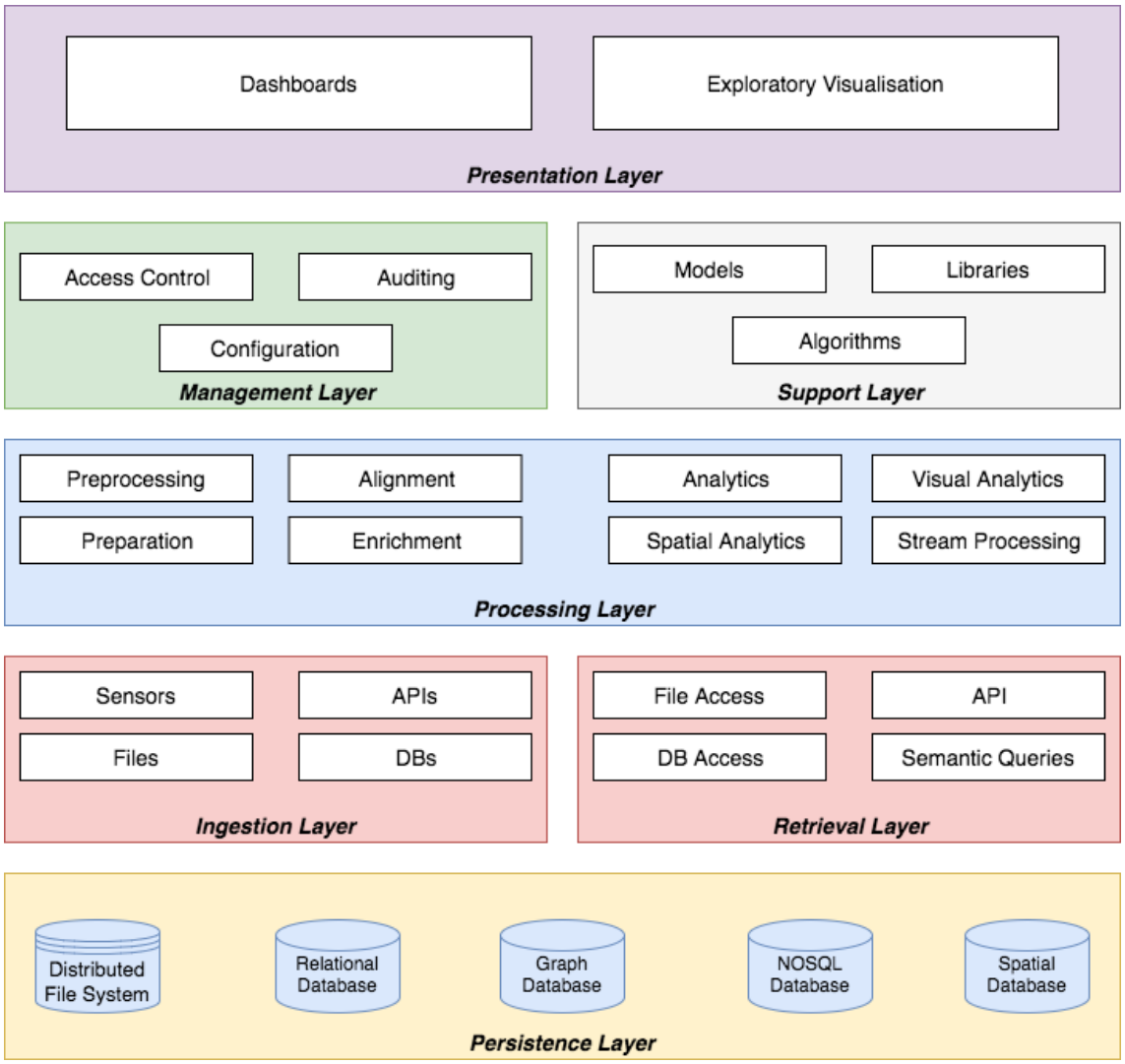

Figure 4: BigDataGrapes Architecture Layers

The purpose and scope of each of the seven core layers are summarised in the following subsections.

6.2.1 Persistence Layer

The layer deals with the long-term storage and management of data handled by the platform. Its purpose is to consistently and reliably make the data available to the processing layer. The layer incorporates persistence technologies in accordance with the data sources summarised in section 3 of the document, organised in appropriate federations in order to ensure performance, redundancy and uptime.

6.2.2 Data Ingestion Layer

The Ingestion Layer comprises the components responsible for moving external data assets to the persistence layer of the platform. The ingestion layer will incorporate all the connectors required for accessing the sources to be used in the context of the BigDataGrapes use cases, as well as, the necessary logic to transform the input data into the formats and schemas agreed upon for the relevant component of the persistence layer where a given resource will be stored.

6.2.3 Data Retrieval Layer

In similar fashion, components of the Data Retrieval layer are responsible for exposing the stored data to the Processing and Presentation layers. Depending on the needs of each processing component, data can be retrieved via:

- Access to files lying on a distributed file system;
- Direct access to relational or NOSQL databases, via the execution of custom or pre-defined queries (depending on the use case and the degree of control that the end-users should have);
- Direct access to Graph databases and triple stores, via the execution of custom or pre-defined semantic queries (depending on the use case and the degree of control that the end-users should have);
- Calls on Application Programming Interfaces (APIs) that expose the underlying stored data in a controlled fashion.

6.2.4 Processing Layer

The Processing Layer implements the core processes for data management and analysis towards serving the analytics and decision support requirements of the BigDataGrapes use cases. These operations are classified under the following main categories:

- **Pre-processing**: processes designed to validate and pre-process incoming datasets. Different data sources evidently will require different pre-processing mechanisms. Exemplary operations carried out by pre-processing components are generation of provenance metadata, validation and anomaly detection, feature extraction, etc.
- **Alignment**: the alignment components are responsible for discovering and proposing links between semantic resources imported into the platform, either at the schema (ontologies, taxonomies, vocabularies) or at the instance level (identity or similarity between datasets or data items).
- **Enrichment**: the enrichment components carry out the automatic annotation of the available data with semantic information, using the semantic resources available to the platform.
- **Preparation**: The structure in which data are stored after the integration may not be suitable to perform the target analysis. The preparation modules adapt the data to match the format expected by the analytics components. Since each analytical model may expect data in a different format, data preparation is specific to each analytic model and the preparation components will be enriched and extended as necessary.
- **Stream Processing:** The stream processing components are responsible for carrying out analysis over live data streams, as opposed to persistent data collections.
- **Data Analytics**: The components receive as input the prepared data from the preparation components and apply statistical and machine learning methods to extract knowledge and make predictions. At this level, descriptive analysis is done providing some statistical insights on the characteristics and behaviour of the variables under study. In turn, the predictive techniques are based on machine learning models used to explain, classify and predict the targeted variables.
- **Visual Analytics**: In similar fashion, the visual analytics components apply analytical algorithms and methods over the appropriately prepared data to generate the augmented visualisations to be presented to the end-user / decision maker via the relevant components in the presentation layer.
- **Spatial Analytics**: The spatial analytics components entail the functionality for geospatial analysis, making use of the relevant geospatial information, as well as, invoking and using directly the other analytics components of the platform.

6.2.5 Management Layer

The layer incorporates the tools for managing and configuring the operation of the BigDataGrapes platform itself. It targets administrators and managers of a deployment and provides the functionalities for user and role definition and access credentials, log monitoring and auditing, component configuration etc.

6.2.6 Support Layer

The support layer incorporates the modules and functions to be used by the processing components of the platform. These tentatively include implementations of machine learning algorithms, analytical models, geospatial operators, transformation libraries, etc.

6.2.7 Presentation Layer

The presentation layer entails all the user-facing components of the platform. All of them are provisioned under a common master Dashboard that will serve as a showcase of the data, analytics and visualization assets of the project, but also as a monitoring tool for the state of the BigDataGrapes platform. On its back-end, the dashboard will serve as a configuration environment, that will manage user access to the underlying assets, but that will also allow users to propose and submit new assets (data, models, algorithms etc) to the platform.

6.3 CANDIDATE TECHNOLOGIES

The layered organisation of the BigDataGrapes platform presented in the previous sections makes evident that the core layers of the architecture where innovative solutions will be implemented are the Persistence, Data Ingestion and Retrieval, and Processing Layer.

Strong candidate technologies to be exploited for the implementation of the BigDataGrapes components belong to the respective layers. These technologies are further presented in the following subsections and summarized in Figure 5.

Figure 5: Candidate Technologies and Frameworks per Layer

6.3.1 Data Persistence Technologies

Storage is challenging for large datasets. The idea of this layer is to have a single store of all data in the enterprise ranging from raw data (which implies exact copy of source system data) to transformed data which is used for various tasks including reporting, visualization, analytics and machine learning. The data stored include structured data from relational databases (rows and columns), semi-structured data (CSV, logs, XML, JSON), unstructured data (documents, PDFs) and even binary data (images, video) thus creating a centralized data store accommodating all forms of data. Several possible solutions can rescue from such problems. Thus, this layer focuses on where to store such large and heterogeneous data efficiently.

- **Hadoop Distributed File System (HDFS),** the commonly known file system of Hadoop and HBase (Hadoop's database) are the most topical and advanced data storage and management systems available in the market. Both HDFS and HBase are capable of processing structured, semi-structured as well as unstructured data.
- **HDFS**. HDFS is fault-tolerant by design and supports rapid data transfer between nodes even during system failures. It provides also redundancy and supports high availability. HDFS is most suitable for performing batch analytics.
- **HBase**. HBase is a non-relational and open source Not-Only-SQL database that runs on top of Hadoop. HBase an handle large data sets and is not appropriate for batch analytics. Instead, it is used to write/read data from Hadoop in real-time.
- **PostgreSQL**. PostgreSQL is a powerful, open source object-relational database system with over 30 years of active development that has earned it a strong reputation for reliability, feature robustness, and performance.
- **Apache Cassandra**, a wide column storage engine, based on the ideas of BigTable and DynamoDB that offers a decentralized architecture based on clusters and nodes and high write-throughput.
- **MongoDB**, a NoSQL document storage engine that offers data model flexibility and a very high ease of usage both for write as well as also for read data.

For the timeseries databases required within the BDG platform, a strong candidate technology is the Elasticsearch engine.

- **Elasticsearch** is a distributed, RESTful search and analytics engine capable of solving a growing number of use cases. It is designed to provide excellent performance in different deployment settings, including an official Hadoop connector that will be integrated in the BDG platform.

Finally, for storing ontological and semantic information in the BigDataGrapes platform, OntoText brings the industry leading GraphDB platform.

- **GraphDB** is a Semantic graph database fully compliant with the relevant W3C standards (RDF, OWL, SPARQL). It is designed to be highly efficient and robust at large-scale use cases. It is one of the few triple store solutions that provide semantic inferencing, enabling its users to derive additional implied facts from the facts already extant in the store.

Table 1: HDFS / HBase comparison

HDFS	HBase
HDFS is a Java-based system utilised for storing large data sets	HBase is a Java-based Not-Only-SQL database
HDFS has a rigid architecture that does not allow changes. It doesn't facilitate dynamic storage	HBase allows for dynamic changes and can be utilised for stand-alone applications
HDFS is ideally suited for write-once and read-many	HBase is ideally suited for random write and read of

use cases	data that is stored in HDFS

6.3.2 Data Ingestion Technologies

Data ingestion is the first step for building data pipeline and also one of the toughest tasks in Big Data processing. Big Data Ingestion involves connecting to several data sources, extracting the data, and detecting the changed data. It's about moving data from where it is originated, into a system where it can be stored, processed and analysed. Furthermore, these several sources exist in different format such as: Images, OLTP data from RDBMS, CSV and JSON files, etc. Therefore, a common challenge faced at this first phase is to ingest data at a reasonable speed and further process it efficiently so that data can be properly analysed to improve business decisions.

Data can be streamed in real time or ingested in batches. When data is ingested in real time then as soon as data arrives it is ingested immediately. When data is ingested in batches, data items are ingested in some chunks at a periodic interval of time. Effective Data Ingestion process begins by prioritizing data sources, validating individual files and routing data items to the correct destination.

Several tools have been developed to move data into Hadoop, among them the BDG platform explore the features of the following technologies:

- **Apache Sqoop**. It is a tool designed to transfer data between Hadoop and relational databases. Sqoop can be used to import data from a relational database management system (RDBMS) such as MySQL or Oracle into the Hadoop Distributed File System (HDFS).
- **Apache Flume**. A Java-based ingestion tool, Flume is used when input data streams-in faster than it can be consumed. Typically, Flume is used to ingest streaming data into HDFS. Multiple Flume agents can also be used collect data from multiple sources into a Flume collector.
- **Apache NiFi**. Apache NiFi is an integrated data logistics platform for automating the movement of data between disparate systems. It provides real-time control that makes it easy to manage the movement of data between any source and any destination.
- **Apache Kafka**. Apache Kafka aims to provide a unified, high-throughput, low-latency platform for handling real-time data feeds. Its storage layer is essentially a "massively scalable pub/sub message queue architected as a distributed transaction log," making it highly valuable for enterprise infrastructures to process streaming data. Additionally, Kafka connects to external systems (for data import/export) via Kafka Connect and provides Kafka Streams, a Java stream processing library.

6.3.3 Data Retrieval Technologies

This layer comprises interfaceable components to be consumed by analytic tools and applications that want to use the power processing of the Platform. This is a field where interactive queries are necessaries, and it’s a zone traditionally dominated by SQL expert developers.

- **Apache Hive**. The Apache Hive data warehouse software facilitates querying and managing large datasets residing in distributed storage. Hive provides a mechanism to project structure onto this data and query the data using a SQL-like language called HiveQL. At the same time this language also allows traditional map/reduce programmers to plug in their custom mappers and reducers when it is inconvenient or inefficient to express this logic in HiveQL.
- **Spark SQL.** Spark SQL lets you query structured data inside Spark programs, using either SQL or a familiar DataFrame API. Usable in Java, Scala, Python and R. Spark SQL supports the HiveQL syntax as well as Hive SerDes and UDFs, allowing you to access existing Hive warehouses.
- **RESTFul APIs**. REST or RESTful API design (Representational State Transfer) is designed to take advantage of existing protocols. While REST can be used over nearly any protocol, it usually takes advantage of HTTP when used for Web APIs.

6.3.4 Data Processing Frameworks

In this layer, the task is to process and transform the data. This layer is composed by components for parallel data processing and machine learning models for generation of analytics information.
In the context of efficient data processing, The Apache Software Foundation has proven to be very helpful in storing and managing vast amounts of data cheaply and efficiently. The Apache Hadoop software library is a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage.
Hadoop is focused on the storage and distributed processing of large data sets across clusters of computers using a MapReduce programming model: Hadoop MapReduce.

- **Hadoop MapReduce**. Hadoop MapReduce is a software framework for easily writing applications which process vast amounts of data (multi-terabyte datasets) in-parallel on large clusters (thousands of nodes) of commodity hardware in a reliable, fault-tolerant manner. As main features there are:
  - Scale-Out Architecture – Add servers to increase capacity
  - High Availability – Serve mission-critical workflows and applications
  - Fault Tolerance – Automatically and seamlessly recover from failures

Other powerful solution for processing big data is the Apache Spark. Spark is a lightning-fast unified analytics engine for big data and machine learning. Compare to Hadoop, both Hadoop and Apache Spark are big-data frameworks, but they serve for different purposes. For example, Spark is a data-processing tool that operates on those distributed data collections; it doesn't do distributed storage.

- **Spark**. It is a fast and general compute engine for Hadoop data. Spark provides a simple and expressive programming model with higher-level libraries, including support for SQL queries, streaming data, machine learning and graph processing.

Machine learning is critical piece in the BDG platform for mining Big Data for actionable insights. Built on top of Spark, MLlib is a scalable machine learning library that delivers both high-quality algorithms and blazing speed (up to 100x faster than MapReduce).

- **SparkMLlib**. It is a scalable machine learning library consisting of common learning algorithms and utilities, including classification, regression, clustering, collaborative filtering, dimensionality reduction, as well as underlying optimization primitives. The library is usable in Java, Scala, and Python as part of Spark applications, so that you can include it in complete workflows.

For the purposes of stream processing, a powerful, well-supported combination of technologies is the usage of Apache Flink for the execution of the processing components over data.

- **ApacheFlink** is an open-source stream processing framework for distributed, high-performing, always-available, and accurate data streaming applications.

Big Data is stimulating new thinking about GIS, pointing to new research directions, and creating exciting opportunities for new tools and products. In this context, for the processing of geospatial information data, the BDG platform uses the GIS Tools for Hadoop and the GeoSpark.

- **GIS Tools for Hadoop.** The GIS Tools for Hadoop are a collection of GIS tools that leverage the Spatial Framework for Hadoop for spatial analysis of big data. The tools make use of the Geoprocessing Tools for Hadoop toolbox, to provide access to the Hadoop system from the ArcGIS Geoprocessing environment.
- **GeoSpark. GeoSpark** is a cluster computing system for processing large-scale spatial data. GeoSpark extends Apache Spark / SparkSQL with a set of out-of-the-box Spatial Resilient Distributed Datasets (SRDDs)/ SpatialSQL that efficiently load, process, and analyse large-scale spatial data across machines.

7 SUMMARY AND CONCLUSIONS

The present document reports on the functional and non-functional requirements posed by the initial specification of the BigDataGrapes use cases, as the latter are detailed in the deliverables of the relevant work packages. Furthermore, an initial set of non-functional requirements has been reported, based on the observations of the user communities' representatives.

Based on the elicited requirements, the first specification for design of the BigDataGrapes software stack and, additionally, the platform's architecture has been produced. The specification aims to cover the requirements by using established standards and components, ensuring that the resulting platform will be maintainable and extensible in the long term.

While the present version of the architecture is complete in the sense that it covers all major operations expected by the platform, it will be naturally refined and extended as work on the project progresses and further details and intricacies of each use case become apparent.

To this end, the specification report will be treated as a live document, with updates applied whenever a major shift on the requirements is identified.

In the forthcoming period, this report will serve as the guide for development and integration work on the project's technical work packages. Initial versions of the tools and components foreseen in the architecture will be provided by the partners involved and will be used in the first deployment of the integrated BigDataGrapes solution to be evaluated by the end-users. Feedback will be subsequently used to possibly update the architectural specification and consequently the entailed components.