D3.1 FlexiGroBots Platform v1

Document Identification			
Status	Final	**Due Date**	31/12/2021
Version	1.0	**Submission Date**	23/12/2021

Related WP	WP3	**Document Reference**	D3.1
Related Deliverable(s)	D2.2, D3.2, D3.3	**Dissemination Level (*)**	PU
Lead Participant	ATOS	**Lead Author**	Daniel Calvo, ATOS
Contributors	SER, CSIC, WUR, VTT, BIO, ART, LUKE, IDSA	**Reviewers**	Juha-Pekka Soininen, VTT
			Markos Matsas, IDSA

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

Document Information

List of Contributors	
Name	Partner
Daniel Calvo	ATOS
Miguel González	ATOS
José María Miranda	ATOS
Miguel Ángel Esbrí	ATOS
João Valente	WU
Mar Ariza Sentís	WU
Markos Matsas	IDSA
Juha-Pekka Soininen	VTT
Damián Prieto	SER
Sergio Álvarez	SER
Oskar Marko	BIO
Angela Ribeiro	CSIC

Document History			
Version	Date	Change editors	Changes
0.1	05/11/2021	Daniel Calvo (ATOS)	Initial document outline
0.2	30/11/2021	Miguel González (ATOS), José María Miranda (ATOS), Sergio Álvarez (SER), Damián Prieto (SER), João Valente (WU), Mar Ariza (WU), Oskar Marko (BIO)	Section 5
0.3	17/12/2021	Daniel Calvo (ATOS), Markos Matsas (IDSA), Juha-Pekka Soininen (VTT)	Section 3
0.4	18/12/2021	Daniel Calvo (ATOS), Oskar Marko (BIO)	Executive summary, introduction and section 2
0.5	19/12/2021	Daniel Calvo (ATOS), Angela Ribeiro (CSIC)	Section 6
0.6	20/12/2021	Miguel Ángel Esbrí (ATOS)	Section 4 and conclusions

Document History			
Version	Date	Change editors	Changes
0.7	21/12/2021	Daniel Calvo (ATOS), Miguel González (ATOS), Miguel Ángel Esbrí (ATOS)	Address comments from VTT
0.8	22/12/2021	Daniel Calvo (ATOS)	Address comments from IDSA
0.9	22/12/2021	Daniel Calvo (ATOS)	Correct order for references.
1.0	23/12/2021	Daniel Calvo (ATOS)	FINAL VERSION TO BE SUBMITTED

Quality Control		
Role	Who	Approval Date
Deliverable leader	Daniel Calvo	23/12/2021
Quality manager	Ivan Zaldivar Santamaria	23/12/2021
Project Coordinator	Daniel Calvo	23/12/2021

Table of Contents

Document Information 2

Table of Contents 4

List of Figures 7

List of Tables 7

List of Abbreviations 8

Executive Summary 11

1 Introduction 13

1.1 Purpose of the document 13

1.2 Relation to other project activities 14

1.3 Structure of the document 15

2 Artificial Intelligence platform 17

2.1 Implemented functionalities 17

2.2 Requirements 19

2.2.1 Technical requirements 19

2.2.2 Functional requirements 20

2.3 Data models 20

2.4 Application Programming Interfaces (APIs) 20

2.5 Graphical User Interfaces (GUIs) 20

2.6 Installation 21

2.7 Prototype availability within FlexiGroBots 21

2.8 Release planning 21

3 Common data services 23

3.1 Implemented functionalities 23

3.2 Requirements 24

3.2.1 Technical requirements 24

3.2.2 Functional requirements 25

3.3 Data models 25

3.4 Application Programming Interfaces (APIs) 25

3.5 Graphical User Interfaces (GUIs) 26
3.6 Installation 27
3.7 Prototype availability within FlexiGroBots 27
3.8 Release planning 27
4 Geospatial enablers and services 29
4.1 Implemented functionalities 29
4.2 Requirements 31
4.2.1 Technical requirements 31
4.2.2 Functional requirements 32
4.3 Data models 32
4.4 Application Programming Interfaces (APIs) 32
4.5 Graphical User Interfaces (GUIs) 33
4.6 Installation 34
4.7 Prototype availability within FlexiGroBots 36
4.8 Release planning 36
5 Common application services 38
5.1 Situation Awareness 38
5.1.1 SLAM 38
5.1.2 People detection, location and tracking 40
5.1.3 People behaviour estimation 42
5.1.4 Moving objects detection, location and tracking 44
5.2 Utility 46
5.2.1 GIS plug-in 46
5.3 Generalization 49
5.3.1 Disease detection 49
5.3.2 Insect infestation detection 51
5.3.3 Weed detection 52
6 Mission Control Centre 54
6.1 Implemented functionalities 54
6.2 Requirements 56
6.2.1 Technical requirements 56

6.2.2 Functional requirements 57
6.3 Data models 57
6.4 Application Programming Interfaces (APIs) 57
6.5 Graphical User Interfaces (GUIs) 57
6.6 Installation 58
6.7 Prototype availability within FlexiGroBots 58
6.8 Release planning 58
7 Conclusions 60
References 61

List of Figures

Figure 1 Deliverables linked to D3.1 15
Figure 2 Differentiation and synergies between T3.1, T3.4 and WP3-4-5 18
Figure 3 Kubeflow GUI. Source: https://www.kubeflow.org/docs/components/central-dash/overview/ 20
Figure 4: Current version of the IDS Testbed 24
Figure 5 IDSA Dataspace connector REST API endpoints. Source: https://international-data-spaces-association.github.io/DataspaceConnector/Documentation/v6/RestApi 26
Figure 6 Extract of IDSA Metadata broker REST API endpoints. Source: https://app.swaggerhub.com/apis/idsa/IDS-Broker/1.3.1 26
Figure 7 Open Data Cube (ODC) components. Source of original image: [47] 30
Figure 8 Data Cube Explorer 33
Figure 9 Jupyter Notebook with some sample code and results for the processing of Sentinel 2 images in the Spanish pilot area 34
Figure 10 Example of FlexiGroBots GIS plugin-in DSS Web App 48
Figure 11 Overview of MCC functionalities 54
Figure 12 MCC detailed architecture 55
Figure 13 Application settings in QGC GUI 57
Figure 14 Plan view of QGC GUI 58

List of Tables

Table 1 Comparison of data management tools 19
Table 2 Comparison of MLOps tools 19
Table 3 User stories for the AI platform 22
Table 4 User stories for the ADS 28
Table 5 User stories for the geospatial processing and services 37
Table 6 User stories for the SLAM 40
Table 7 User stories for people detection, location and tracking 42
Table 8 User stories for people behaviour estimation 44
Table 9 User stories for moving objects detection, location and tracking 46
Table 10 User stories for GIS plug-in 49
Table 11 User stories for disease detection 51
Table 12 User stories for disease detection 52
Table 13 User stories for weed detection 53
Table 14 Comparison between RHEA Mission Control Centre and QGC 56
Table 15 User stories for MCC 59

List of Abbreviations

Abbreviation / acronym	Description
ADS	Agriculture Data Space
AI	Artificial Intelligence
API	Application Programming Interface
AWS	Amazon Web Services
CA	Certification Authority
CLI	Command Line Interface
COG	Cloud-Optimized GeoTiffs
CRUD	Create, Read, Update, Delete
CUDA	Compute Unified Device Architecture
CV	Computer Vision
DAPS	Dynamic Attribute Provisioning Service
DBS	Data Base System
DF	Dronecode Foundation
DSS	Decision Support System
DVC	Data Version Control
Dx.y	Deliverable number y belonging to WP x
EC	European Commission
ELSEC	Ethical, Legal, Societal, Economic, Cultural
EO	Earth Observation
EU	European Union
FDM	FOODIE Data Model
GCP	Google Cloud Platform
GCS	Google Cloud Storage
GDAL	Geospatial Data Abstraction Library
GIS	Geospatial Information System
GPL	General Public License
GPS	Global Positioning System
GPU	Graphics Processing Unit
GUI	Graphical User Interface

Abbreviation / acronym	Description
HDFS	Hadoop Distributed File System
ICT	Information and Communication Technologies
IDS	International Data Spaces
IDSA	International Data Spaces Association
JPEG	Joint Photographic Experts Group
JRE	Java Runtime Environment
LFS	Large File Storage
LPWAN	Low Powe Wide Area Network
MCC	Mission Control Centre
ML	Machine Learning
MVDS	Minimum Viable Data Space
NDVI	Normalised Difference Vegetation Index
ODC	Open Data Cube
OGC	Open Geospatial Consortium
OS	Operating System
QGC	QGroundControl
QGIS	Quantum Geographic Information System
REST	Representational State Transfer
RGB	Red, Green, Blue
RHEA	Robot Fleets For Highly Effective Agriculture And Forestry Management
ROI	Return of Investment
SDK	Software Development Kit
SLAM	Simultaneous Location And Mapping
SQL	Structured Query Language
SSH	Secure Shell
STAC	Spatio-Temporal Asset Catalogue
TRL	Technology Readiness Level
TSC	Technical Steering Committee
UAV	Unmanned Aerial Vehicle
UC	Use-case
UGV	Unmanned Ground Vehicle

Abbreviation / acronym	Description
UI	User Interface
US	United States
VM	Virtual Machine
WCS	OGC Web Coverage Service
WMS	OGC Web Map Service
WP	Work Package

Executive Summary

The main mission of deliverable D3.1 is to present the initial version of the components that compose the FlexiGroBots platform, being the first outcome of WP3 -*Platform development*. Two more versions of the deliverable will be published during the execution of the project: D3.2 in December 2022 (M24) and D3.3 in December 2023 (M36). The three deliverables correspond to the main releases of the FlexiGroBots platform's components software prototypes, but descriptive reports have been also prepared to explain the work done during the development process and the achieved results.

FlexiGroBots platform is devoted to enabling the usage of flexible and heterogeneous multi-robot systems for intelligent automation of precision agriculture operations. The platform considers by design the creation of an embryonic Agriculture Data Space (ADS) to support robotics missions leveraging the vision of the International Data Spaces Association (IDSA) and the existing building blocks already implemented by its community. The ADS will support the three pilots to realize a data value chain that maximizes synergies, collaboration, and trading around data, while ensuring data sovereignty, data governance and security in data sharing / exchange across companies, domains and international borders. The key Data Space building blocks will be extended and particularised for the needs and requirements of the stakeholders and systems participating normally in the usage of robotics systems for precision agriculture.

Within the ADS, the FlexiGroBots platform includes data analytics to create AI-driven multi-robot systems and advanced services. On one hand, it supports Machine learning operations (MLOps) addressing the complete lifecycle of the models and introducing powerful functionalities like Automated Machine Learning (AutoML). On the other hand, additional enablers are being developed to deploy geospatial data management, access and processing capabilities following OGC standards.

To maximise the replicability and impact of the FlexiGroBots platform in a wide range of use-cases beyond the crops targeted by the project's pilots, a set of common and general applications and services are being implemented with multiple purposes, e.g., robots' navigation, detection of diseases or pests. Their goal is to be general enough to be used off the shelf by more new farmers.

Finally, FlexiGroBots Mission Control Centre (MCC) offers a complete and vendor-independent solution to plan, execute and monitor the operation of fleets of flexible and heterogeneous robots, providing the maximum standards in terms of safety. It is seamlessly integrated with the rest of the components of the platform and with third-party systems through the ADS.

For each one of the components of the platform, the deliverable describes in detail the status at M12 of the project regarding implemented functionalities, technical requirements, data models, APIs, GUIs, installation procedure, prototype availability and planning for the next steps.

All these aspects will evolve during the project lifetime, benefiting especially from the interactions and feedback received from the three project’s pilots of WP4, WP5 and WP6. They will also receive the influence from the different activities executed in the scope of WP2 (i.e., standardisation, ethical and legal assessment) and WP7 (i.e., dissemination, communication, business development). The updates will be reflected in D3.2 and D3.3 although the development of the prototypes will be done in an agile and continuous manner, being available in the project GitHub repository: https://github.com/FlexiGroBots-H2020

1 Introduction

1.1 Purpose of the document

FlexiGroBots project is an Innovation Action aiming to build a platform for flexible heterogeneous multi-robot systems for intelligent automation of precision agriculture operations, providing multiple benefits to farmers around the world. In this vision, fleets of heterogeneous robots will be able to execute complex missions in an orchestrated and coordinated way overcoming some of the main barriers that currently limit the adoption of unmanned vehicles and robotics technologies in the agriculture domain.

In reference to the addressed FlexiGroBots solution, WP3 is focused on the development of an innovative ICT platform prototype which is composed of the following elements:

- Artificial Intelligence (AI) platform.
- Common data enablers and services.
- Geospatial enablers and services.
- Common application services.
- Mission Control Centre.

The development of the software prototypes of the different components follows an agile methodology that starts from the formalised requirements and the platform technical architecture proposed by D2.2 [1]. From there, an agile methodology is being followed to deliver increasingly mature software prototypes until reaching the final version with a TRL 6-7.

The goal of this document is to describe the status of the software prototypes of FlexiGroBots platform components at the end of the first year of the project, covering the following aspects:

- Implemented functionalities.
- Requirements: technical and functional.
- Data models.
- Application Programming Interfaces (APIs).
- Graphical User Interfaces (GUIs).
- Installation procedure.
- Prototype availability within FlexiGroBots.
- Release planning.

In the last point, the consortium has decided to adopt a modern software development based on the Agile philosophy. Following this approach, for each one of the components to be implemented within the scope of the project, a product backlog will be created, defining the list of tasks or *user stories* that must be completed. The product backlog will be visible and shared with all the partners since it will be published in the present document but handled also through the project GitHub repository. Due to the collaborative nature of the project and the need to synchronize the activities of the different involved entities, development sprints of four weeks of duration will be applied to produce incremental versions of the software prototypes in an iterative way. At the beginning of each sprint, involved partners will select the tasks of the backlog with a higher priority, taking into account also the needed effort that will be estimated through *story points*. The prediction of the time needed for the team to complete a certain user story or task by means of story points is a widely extended practice in agile software development that aims to increase the accuracy in the estimations with respect to using hours and to provide a standard way to compare the complexity to address several tasks [2]. In the FlexiGroBots project, story points are calculated using the Fibonacci sequence. At the end of each sprint, all WP3 partners will have a retrospective meeting to review the results and to plan the next one. The same methodology is applied in the implementation of the pilots as explained in deliverable D2.7 [3].

In some cases, the level of maturity in the development does not allow providing some certain details at the moment of writing D3.1. This is mainly the case for the specification of data models in the common application services. It is important to emphasize that D3.1 contains a fixed view or screenshot of the status of the prototypes' development at M12 of the project. The implementation will continue in the next months and therefore all these details will be progressively incorporated into the project GitHub repository [4]. In the next checkpoint in M24, the updated version of this deliverable (D3.2) will further elaborate on all the aspects presented in this document.

It must be noted that in order to maximise the project outreach and impact, the partners have made the decision to follow an open-source philosophy in most of the components, using also as a baseline for the platform implementation already existing technologies that will be improved, extended and tailored for the objectives of the project. Thus, the project will publish software results in a GitHub repository. The same tool has been also selected to manage the corresponding product backlogs.

1.2 Relation to other project activities

Deliverable D3.1 is the first outcome of WP3 - *Platform development*, wrapping the results of several tasks: *T3.1 - AI platform*, *T3.2 - Common data enablers and services*, *T3.3 - Geospatial enablers and services*, *T3.4 - Common application services* and *T3.5 - Mission Control Centre*. The content of D3.1 summarises the implementation status at the end of the first year of the project for the prototypes built in these five tasks. The corresponding functionalities have been implemented taking into account the requirements (functional and non-functional) and the technical architecture proposed by D2.2 for the platform but also after an exhaustive assessment of the pilot's specifications released by D4.1, D5.1 and D6.1, and the joint analysis of D2.7. The work reflected in D3.1 will be improved and completed following a continuous, iterative and agile development methodology. New versions of the platform will be released with more advanced functionalities so that they can be integrated, demonstrated and assessed by the pilots. The results will be reported on D4.2, D5.2 and D6.2 at the end of the second year of the project (M24, December 2022). Again, a holistic vision will be offered through D2.8. A new version of the platform prototype will be also officially published at the same moment with D3.2. The collaboration and synergies between *WP2 - Requirements, architecture and standardization* and WP3 will be collected also in D2.4 and D2.6. In the former, relevant standards to be taken into account in the development of the FlexiGroBots platform will be evaluated. In the latter, the focus will be on the ELSEC assessment and the extraction of guidelines and recommendations.

A second and updated version of this deliverable, D3.2, will be released in M24 of the project, which corresponds to December 2022. The final one, D3.3, will be published in M36 of the project, December 2023.

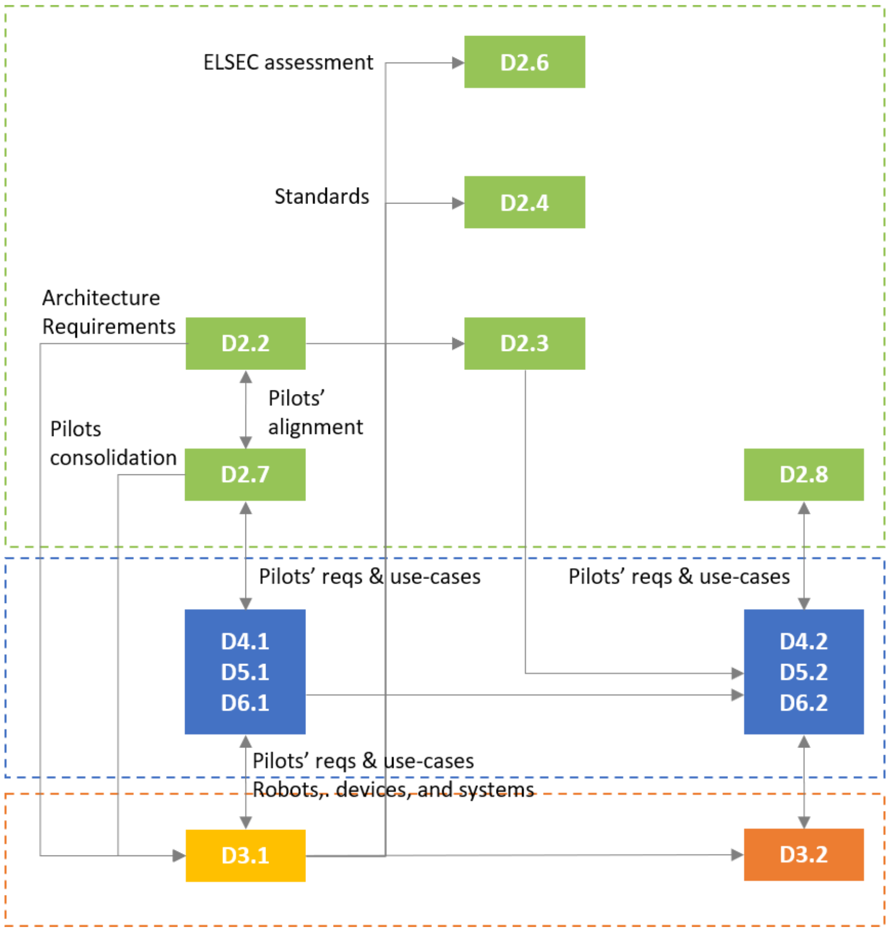


Figure 1 Deliverables linked to D3.1

1.3 Structure of the document

This document is structured in seven major sections:

- **Section 1** introduces the motivation for the document, its link to other tasks and deliverables and describes the structure.
- **Section 2** presents the status of the prototype for the Artificial Intelligence platform.
- The Common Data Services that are being used for the development of the embryonic Agriculture Data Space (ADS) are discussed in **Section 3.**
- **Section 4** is focused on the geospatial enablers and services.
- **Section 5** includes the information for the common application services. This chapter is divided into one subsection for each specific service.
- The Mission Control Centre is introduced in **Section 6.**
- **Section 7** explains the conclusions of the deliverable, providing a vision for the related tasks and documents, especially for the next iteration of this document, D3.2.

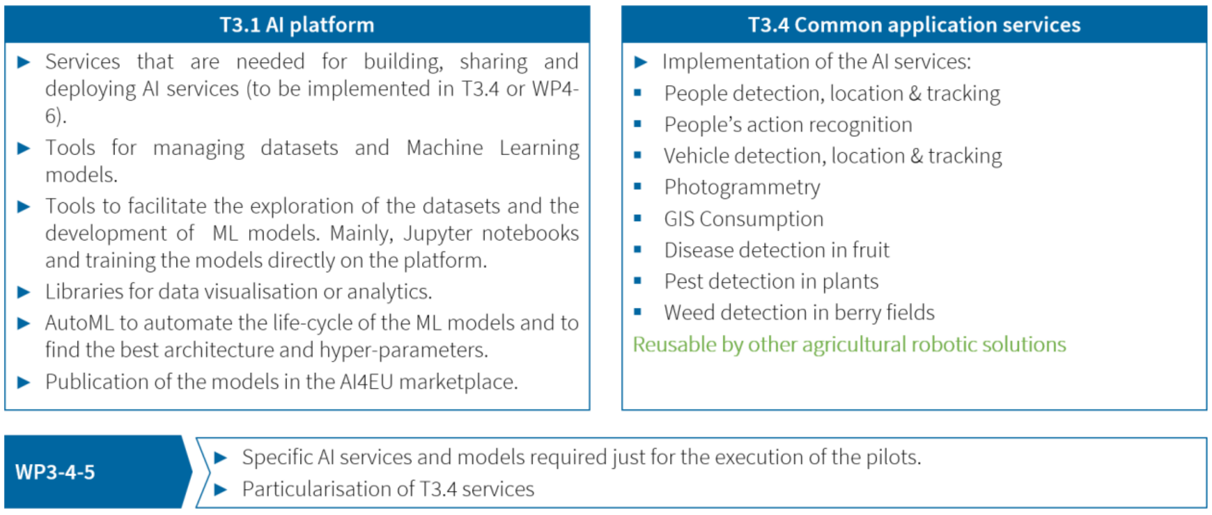
2 Artificial Intelligence platform

2.1 Implemented functionalities

FlexiGroBots has dedicated components to support building, sharing and deploying AI services in the context of the Agriculture Data Space (ADS) and having as an additional objective to be interoperable with the results of the Artificial Intelligence on-demand platform produced by the H2020 AI4EU project. The main services implemented by FlexiGroBots AI services are the following:

- Abstraction of computing resources and software toolchains so that data scientists can get access to an off-the-shelf power Machine Learning (ML) environment.
    - Management of computing resources (cloud and/or high-performance computing) to run any ML process according to your processing needs.
    - Kubernetes [5] infrastructure support to have interoperability with the main Infrastructure as a Service (IaaS) or Platform as a Service (PaaS) solutions
    - Quotas for users about how to exploit the infrastructure. Multi-tenant support.
- Toolbox of well-known ML frameworks, libraries or SDKs to allow developing models for precision agriculture applications and heterogeneous robots.
- Workflow engine for the automation of ML processes following MLOps and AutoML paradigms. It should cover the complete lifecycle of ML models, i.e., data ingestion, data exploration, data pre-processing, feature engineering, models training, architecture exploration, hyperparameters optimization, packaging, deployment, monitoring, and optimization.
- A repository of datasets shared as part of the ADS that can be used to train ML models.
- A repository of ML and AI-driven robotics services built using the platform functionalities for the purpose of the project's pilot and common application services.
    - User's own storage for their own experiments and developments.
    - Access control functionalities.
    - Concept of experiments and training to drive Transfer Learning.
- Multi-platform support (x86, ARM, GPU, TPU) based on Docker containers.
    - Training options to use specific hardware.
    - Generation and optimization of models for serving considering different architectures and frameworks.
- Integration of development tools: Jupyter Notebooks, Software Development Kits (SDKs) and/or Command Line Interfaces (CLI).

Therefore, the usage of the FlexiGroBots AI platform will be of paramount importance in several aspects of the project since it will be supportive technology required by data scientists and Machine Learning engineers to produce innovative ML models and AI-powered applications. Thus, a close interaction will take place between the results of task *T3.1 – AI platform*, *T3.4 – Common Application Services* and the three work packages focused on the pilots WP4, WP5 and WP6. Figure 2 shows the split between the scope of these three realms.


T3.1 AI platform

- Services that are needed for building, sharing and deploying AI services (to be implemented in T3.4 or WP4-6).
- Tools for managing datasets and Machine Learning models.
- Tools to facilitate the exploration of the datasets and the development of ML models. Mainly, Jupyter notebooks and training the models directly on the platform.
- Libraries for data visualisation or analytics.
- AutoML to automate the life-cycle of the ML models and to find the best architecture and hyper-parameters.
- Publication of the models in the AI4EU marketplace.

T3.4 Common application services

- Implementation of the AI services:
  - People detection, location & tracking
  - People's action recognition
  - Vehicle detection, location & tracking
  - Photogrammetry
  - GIS Consumption
  - Disease detection in fruit
  - Pest detection in plants
  - Weed detection in berry fields

Reusable by other agricultural robotic solutions

WP3-4-5

- Specific AI services and models required just for the execution of the pilots.
- Particularisation of T3.4 services

Figure 2 Differentiation and synergies between T3.1, T3.4 and WP3-4-5

Although initially, the usage of Acumos AI [6] was proposed, the partners performed at the beginning of the project an exhaustive analysis of the state-of-the-art of existing technology solutions to ensure that the FlexiGroBots project leverages the most recent advanced in the field of MLOps. The study was done considering the requirements and use-cases specified during the first months of the project and which are described in detail in the deliverable D2.2 [1]:

- AI_UC1 – Data management.
- AI_UC2 – Experiments management.
- AI_UC3 – Trainings management.
- AI_UC4 – Models management.

Since the FlexiGroBots project is an innovation action and it must produce results with a high TRL (6-7), it must re-use and integrate as much as possible available but innovative components (especially open source), particularizing them to reach the project's goals. Following this idea, for each one of the use-cases, some candidate technologies have been studied.

Tool	License	Data forms	Storage technology	Multi-user	Kubernetes support	ML frameworks
DVC [7]	Apache 2.0	ALL	Amazon S3, MinIO, Microsoft Azure Blob Storage, Google Drive, Google Cloud	Yes	N/A	MLFlow

Tool	License	Data forms	Storage technology	Multi-user	Kubernetes support	ML frameworks
			Storage, SSH, HDFS			
Git LFS [8]	MIT	ALL	N/A	Yes	N/A	ALL
lakeFS [9]	Apache 2.0		S3, GCS, or Azure Blob	Yes	Yes	MLFlow, Metaflow
Pachyderm [10]	N/A	ALL	S3	Yes	Yes	Label Studio, Seldon, ClearML
Delta Lake [11]	Apache 2.0	ALL	S3, GCS, and HDFS	Yes	No	MLFlow

Table 1 Comparison of data management tools

Tool	License	Data management	Experiments management	Pipelines	Serving	Multi-user	Kubernetes support
Kubeflow [12]	Apache 2.0	Yes	Yes	Yes	Yes	Yes	Yes
MLFlow [13]	Apache 2.0	Yes	Yes	No	Yes	No	Yes

Table 2 Comparison of MLOps tools

As a result of this initial analysis, the FlexiGroBots AI platform is initially based on Kubeflow. Data persistence will rely on MinIO [14], which can be easily integrated with Kubeflow.

2.2 Requirements

2.2.1 Technical requirements

2.2.1.1 MinIO

Minimum hardware requirements are:

- Dual Intel® Xeon® processor.
- 128GB RAM memory.

2.2.1.2 Kubeflow

The installation of Kubeflow can be done on multiple platforms [15], e.g., AWS, Azure, GCP, OpenShift or even MicroK8s.

The deployment can be done in an existing Kubernetes cluster with the following characteristics:

- Kubernetes version is 1.14.
- At least a worker node with:
  - 4 CPUs
  - 50 GB storage.
  - 12 GB memory.

2.2.2 Functional requirements

Not relevant.

2.3 Data models

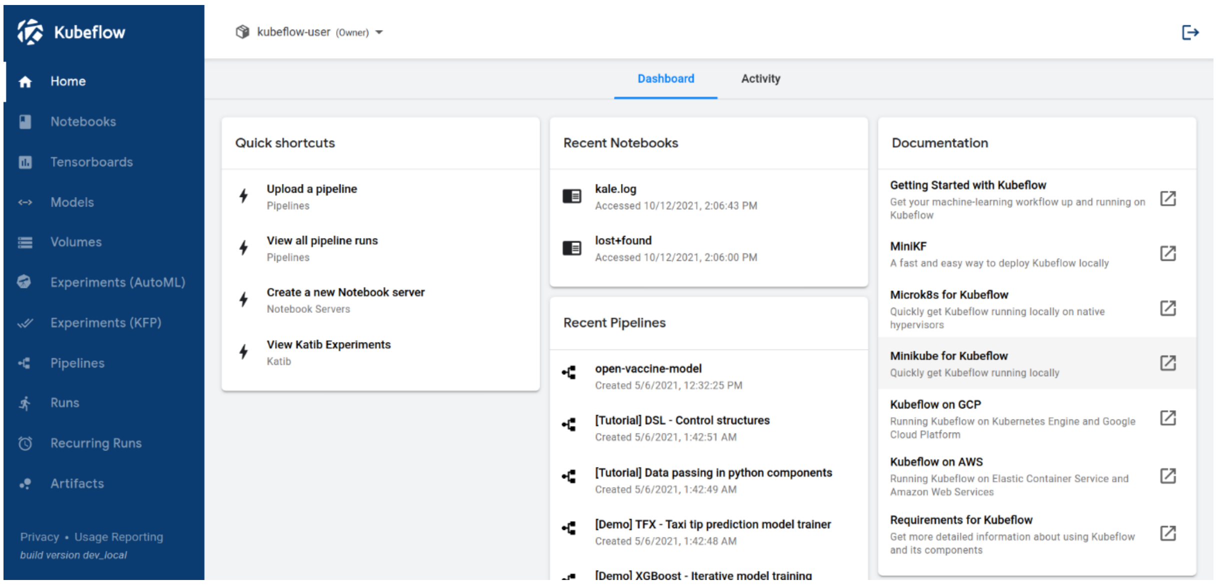
The AI platform will be agnostic with respect to any data format or model. Ingestion will be achieved through a specific connector following the principles of the ADS.

2.4 Application Programming Interfaces (APIs)

Kubeflow provides a Python SDK to manage and execute ML pipelines [16]. A RESTful API is also available [17]. It supports also main ML libraries like TensorFlow, PyTorch or scikit-learn.

2.5 Graphical User Interfaces (GUIs)

Kubeflow has a central GUI to get access to all available functionalities. A screenshot is included in Figure 3 below.



Quick shortcuts

Upload a pipeline
Pipelines

View all pipeline runs
Pipelines

Create a new Notebook server
Notebook Servers

View Katib Experiments
Katib

Recent Notebooks

kale.log
Accessed 10/12/2021, 2:06:43 PM

lost+found
Accessed 10/12/2021, 2:06:00 PM

Recent Pipelines

open-vaccine-model
Created 5/6/2021, 12:32:25 PM

[Tutorial] DSL - Control structures
Created 5/6/2021, 1:42:51 AM

[Tutorial] Data passing in python components
Created 5/6/2021, 1:42:49 AM

[Demo] TFX - Taxi tip prediction model trainer
Created 5/6/2021, 1:42:48 AM

[Demo] XGBoost - Iterative model training

Documentation

Getting Started with Kubeflow
Get your machine-learning workflow up and running on Kubeflow

MiniKF
A fast and easy way to deploy Kubeflow locally

Microk8s for Kubeflow
Quickly get Kubeflow running locally on native hypervisors

Minikube for Kubeflow
Quickly get Kubeflow running locally

Kubeflow on GCP
Running Kubeflow on Kubernetes Engine and Google Cloud Platform

Kubeflow on AWS
Running Kubeflow on Elastic Container Service and Amazon Web Services

Requirements for Kubeflow
Get more detailed information about using Kubeflow and its components

Figure 3 Kubeflow GUI. Source: https://www.kubeflow.org/docs/components/central-dash/overview/

Document name:	D3.1 FlexiGroBots Platform v1					**Page:**	20 of 63
Reference:	D3.1	**Dissemination:**	PU	**Version:**	1.0	**Status:**	Final

2.6 Installation

The installation procedure for Kubeflow depends on the underlying platform. In the case of FlexiGroBots, the installation will be done in a bare-metal server and therefore manifests are used directly [18].

2.7 Prototype availability within FlexiGroBots

- MinIO is available at https://minio.flexigrobots-h2020.eu
- Kubeflow is available in https://kubeflow.flexigrobots-h2020.eu/

2.8 Release planning

- 12/2021:
    - Complete deployment of MinIO and Kubeflow.
- 01/2022:
    - Interoperability with AI4EU.
    - Examples of ML pipelines running in the AI platform.
- 02/2022 – 04/2022:
    - Integration of AI platform into the ADS.
    - Integration of AI platform with pilots' systems.
- 05/2022 (onwards):
    - Additional features, feedback from pilots.
- 12/2022:
    - Final prototype.

User story ID	User story	Priority	Story points
AI_US_01	Development of an IDSA connector for MinIO.	High	3
AI_US_02	Integration of Kubeflow with AI4EU for re-using and publication of artefacts.	High	5
AI_US_03	Testing of AutoML operators	High	1
AI_US_04	Integration of models for common application services	Medium	7
AI_US_05	Integration of models for pilot 1	High	8
AI_US_06	Integration of models for pilot 2	High	8
AI_US_07	Integration of models for pilot 3	High	8

User story ID	User story	Priority	Story points
AI_US_08	Optimisation for multiple architectures including ARM processors.	Medium	13
AI_US_09	Integration of *codecarbon* library	Low	2
AI_US_10	Performance monitoring and retraining functionality	Medium	21

Table 3 User stories for the AI platform

3 Common data services

The specifications of IDS described in the Reference Architecture Model form the basis for creating standard, secure and sovereign data sharing capabilities for the FlexiGroBots platform based on European values, which ensure equal opportunities and trust among data sharing entities through a federated design. The IDS roles and essential components have been described in detail in D2.2 “Requirements and platform architecture specifications” [1].

IDSA has created a GitHub [19] page where all links to important technical documents along with available Open-Source (OS) projects built around IDS concepts are available. This main repository of IDSA on GitHub provides an overview and required information on IDS Open-Source Landscape that the FlexiGroBots platform will be based on to create the Minimum Viable Data Spaces (MVDS) that will offer the common data services. The OS project has been initiated so that the IDS standard is designed in a transparent and reusable way among various developers of IDS components. It is open for external contributors that might want to contribute to the implementations either by joining technical development groups or by sharing comments and inputs.

As it is stated in [20], “*IDS-G is the point of truth for specifications of the IDS and its components in the IDS GitHub page. It is also public for everyone and contains the approved specifications that were confirmed by the IDS Technical Steering Committee (IDS-TSC) and the IDSA Working Groups. IDS-G publishes quarterly releases with new approvals by the Working Groups and the TSC*”.

A very important part of the available material that IDSA provides is the GitHub documentation page [21], which is meant to support everyone who wants to either build IDS components or implement and/or contribute to the existing open-source components. It will link you to relevant sources and will provide guidance on your way. The majority of the projects on IDSA OS Landscape are openly available under Apache 2.0 license, allowing them to be reused.

3.1 Implemented functionalities

For the purposes of the project, the Dataspace Connector will be used from the different pilots, which is an implementation of an IDS connector component following the IDS Reference Architecture Model. The connector has been an ongoing project of the Data Business department of the Fraunhofer ISST [22]. In the first year of the project, VTT and Atos successfully demonstrated the exchange of data between two instances of the Dataspace Connector. In the second year of the project instances of the connector will be created from all the entities that need to exchange data in a sovereign way and the testing phase will start.

Atos and VTT will develop the rest of the necessary components based on the open-source components of IDSA.

The implementations of the connectors for the projects will be tested using the IDS Testbed [23], which is a setup of Open-Source IDS components that can verify that a component:

- Implements the IDS specifications for establishing connections and communication.
- And, thus, can work in an interoperable way with all IDS components in the testbed setup.

The current version of the testbed that is characterized as a minimal setup with essential and already available components is depicted below.

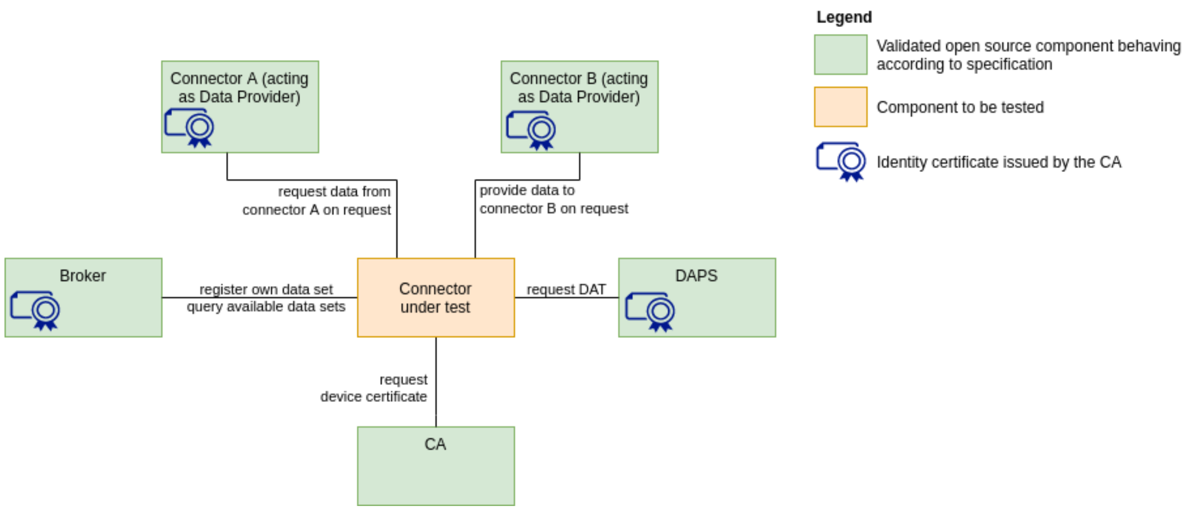

Legend
Validated open source component behaving according to specification
Component to be tested
Identity certificate issued by the CA
Connector A (acting as Data Provider)
Connector B (acting as Data Provider)
request data from connector A on request
provide data to connector B on request
Broker
register own data set
query available data sets
Connector under test
request DAT
DAPS
request device certificate
CA

Figure 4: Current version of the IDS Testbed

IDS Testbed will be used to test the behaviour of the connector in terms of interoperability against IDS components (Connector, DAPS, CA, Metadata Broker) and will be the final step for the component to be characterized certification-ready.

3.2 Requirements

3.2.1 Technical requirements

The deployment of ADS Common Data Services is based on the usage of Docker containers. In principle, this is the only technical requirement foreseen so far.

In the specific case of the metadata broker, the minimum recommended configuration is:

- 20GB of free disk space.
- 64bit quad-core processor.
- Ubuntu 20.04 is higher.
- Docker version 20.10.7 or higher.

3.2.2 Functional requirements

All systems that will become part of the FlexiGroBots ADS should be compliant with IDS-RAM principles. Specific connectors should be implemented to allow exchanging data according to this vision.

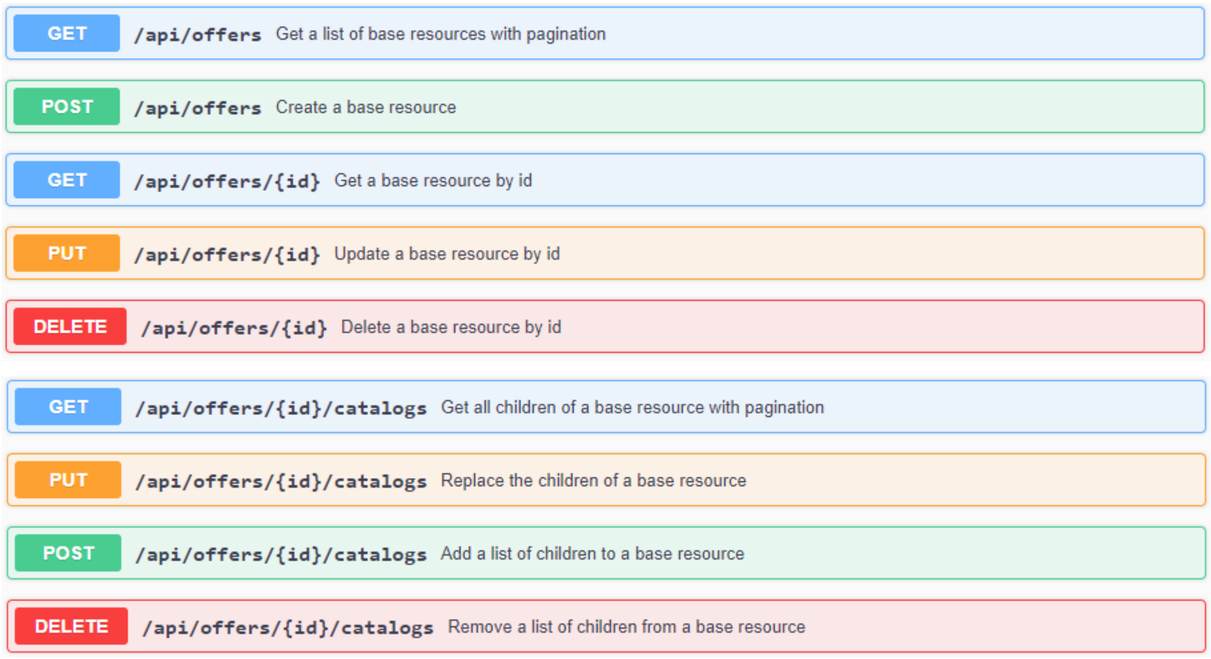
3.3 Data models

Common Data Services of the FlexiGroBots platform will use the Information Model of IDS [24] as the domain-agnostic, common language that represents an essential agreement shared by the participants and components of the IDS, facilitating compatibility and interoperability. The information model will only support the description, publication and identification of data products and reusable data processing software (both referred to the IDS-RAM as "Digital Resources", or simply "Resources"). Once the relevant Resources are identified, they can be exchanged and consumed via semantically annotated, easily discoverable services.

The FlexiGroBots Consortium will define a domain-specific vocabulary that will be provided to the Agriculture MVDS participants on top of the information model allowing them to understand each other in terms of the data they exchange.

3.4 Application Programming Interfaces (APIs)

A description of the REST API of the IDSA connector is available at [25]. It contains endpoints to execute CRUD operations to manage resources, catalogues, representations, artefacts, agreements and contracts.


GET /api/offers Get a list of base resources with pagination
POST /api/offers Create a base resource
GET /api/offers/{id} Get a base resource by id
PUT /api/offers/{id} Update a base resource by id
DELETE /api/offers/{id} Delete a base resource by id
GET /api/offers/{id}/catalogs Get all children of a base resource with pagination
PUT /api/offers/{id}/catalogs Replace the children of a base resource
POST /api/offers/{id}/catalogs Add a list of children to a base resource
DELETE /api/offers/{id}/catalogs Remove a list of children from a base resource

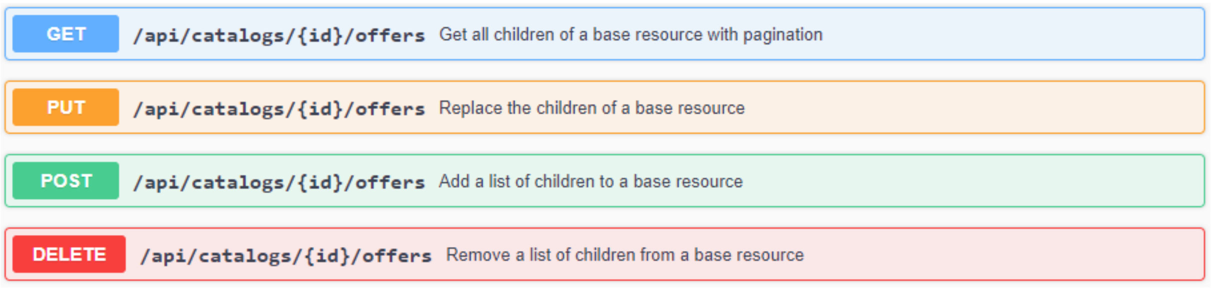

GET /api/catalogs/{id}/offers Get all children of a base resource with pagination
PUT /api/catalogs/{id}/offers Replace the children of a base resource
POST /api/catalogs/{id}/offers Add a list of children to a base resource
DELETE /api/catalogs/{id}/offers Remove a list of children from a base resource

Figure 5 IDSA Dataspace connector REST API endpoints. Source: https://international-data-spaces-association.github.io/DataspaceConnector/Documentation/v6/RestApi

In the case of the metadata broker, there is also a REST API available in SwaggerHub [26].

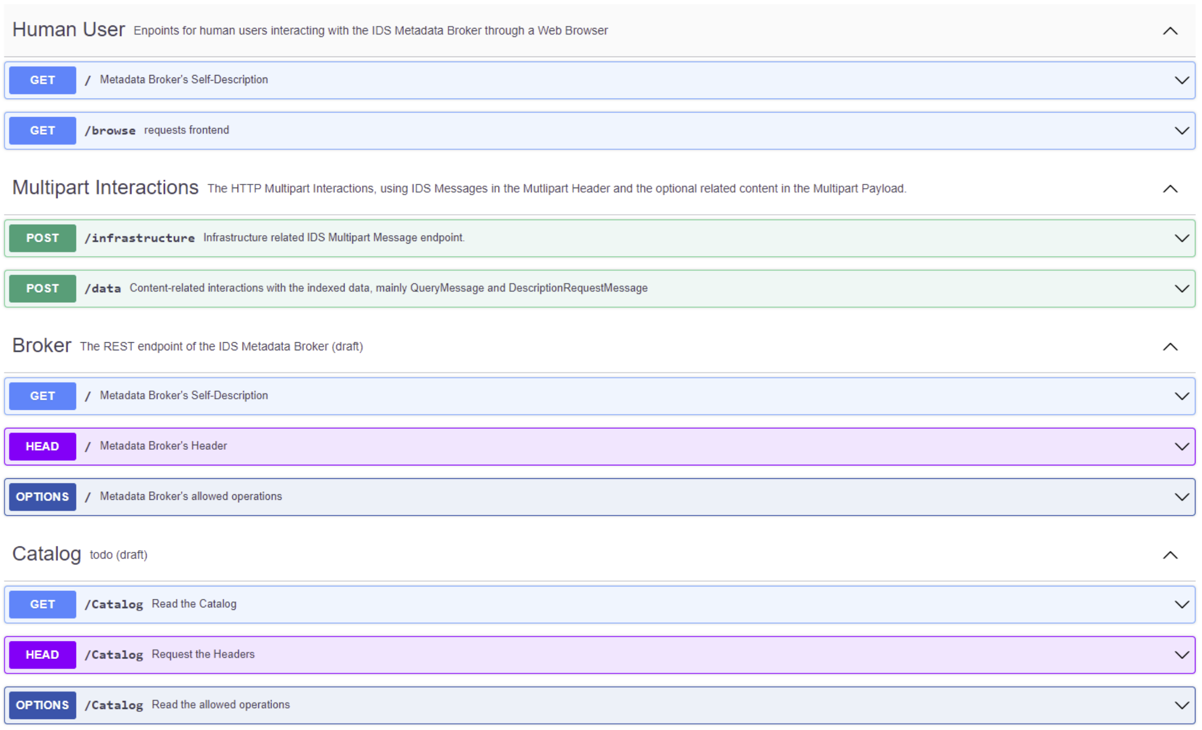

Human User Enpoints for human users interacting with the IDS Metadata Broker through a Web Browser
GET / Metadata Broker's Self-Description
GET /browse requests frontend
Multipart Interactions The HTTP Multipart Interactions, using IDS Messages in the Mutlipart Header and the optional related content in the Multipart Payload.
POST /infrastructure Infrastructure related IDS Multipart Message endpoint.
POST /data Content-related interactions with the indexed data, mainly QueryMessage and DescriptionRequestMessage
Broker The REST endpoint of the IDS Metadata Broker (draft)
GET / Metadata Broker's Self-Description
HEAD / Metadata Broker's Header
OPTIONS / Metadata Broker's allowed operations
Catalog todo (draft)
GET /Catalog Read the Catalog
HEAD /Catalog Request the Headers
OPTIONS /Catalog Read the allowed operations

Figure 6 Extract of IDSA Metadata broker REST API endpoints. Source: https://app.swaggerhub.com/apis/idsa/IDS-Broker/1.3.1

The implementation of the DAPS component will be done following the specification available at [27].

3.5 Graphical User Interfaces (GUIs)

Not relevant for common data services.

3.6 Installation

The installation of the connector for each system participating in the ADS will depend on the specific implementation that is performed. Nevertheless, Docker containers will be the standard mechanism followed by the project.

For the Metadata broker, the installation procedure is documented in detail in [28].

3.7 Prototype availability within FlexiGroBots

At M12 of the project, VTT and Atos achieved to demonstrate the exchange of data between the systems managed by these two companies, i.e., VTT data lake and Atos AI platform. The source code used for this development can be found at [4].

3.8 Release planning

- 01/2022:
    - Data exchanges between FlexiGroBots entities through custom IDSA-compliant connectors.
    - Usage and deployment of IDS Metadata Broker.
- 02/2022 – 05/2022:
    - Implementation of prototypes for DAPS and CA.
- 06/2022 (onwards):
    - Pilots' integration.
- 12/2022:
    - Final prototypes.

User story ID	User story	Priority	Story points
ADS_US_01	Deployment and usage of metadata broker	High	3
ADS_US_02	Implementation of connector for AI platform	High	2
ADS_US_03	Implementation of connector for geospatial services.	High	2
ADS_US_04	Implementation of connector for MCC.	Medium	2
ADS_US_05	Implementation of connectors for Pilot 1	High	5
ADS_US_06	Implementation of connectors for Pilot 2	High	5
ADS_US_07	Implementation of connectors for Pilot 3	High	5
ADS_US_08	Development of DAPS component	Medium	8

User story ID	User story	Priority	Story points
ADS_US_09	Development of CA component	Low	13

Table 4 User stories for the ADS

4 Geospatial enablers and services

The aim of the FlexiGroBots geospatial enablers and services is to provide a set of general-purpose services and features that facilitate the access, visualization, and processing of geospatial datasets necessary for performing the daily activities on the farm. Therefore, and by building on top of (and complementing) the common enablers provided by tasks T3.4 and T3.5, the focus of this task (T3.3 "Geospatial enablers and services") is threefold:

- Deployment of geospatial data management and access and processing services.
- Implementation of additional data exchange connectors (not offered by T3.4) specialized for geospatial data formats; and
- Implementation of supporting GIS processing services (invoked via OGC WPS API) required by the data workflows in the pilots. In some cases, it will lead to the implementation of advanced services combining Big Data Copernicus and AI.

Furthermore, FlexiGroBots, the precision agriculture domain in general, heavily depends on a variety of heterogeneous spatially enabled information, being Earth Observations (EO) - either collected from satellite imagery and/or UAVs - one of the main inputs for several other agricultural ICT services (e.g., vegetation indices calculation, pest detection, robots mission planning, etc.). Thus, one of the key geospatial components planned for deployment within FlexiGroBots agricultural data space is a data cube (in this case, leveraging on the open-source solution Open Data Cube (ODC)) for facilitating the management and access to the Copernicus satellite imagery (e.g., Sentinel 2) required by the pilots as well as other Earth Observation datasets produced by them (e.g., raster images from the UAVs). This component, together with others, will be the base for implementing other geospatial processing services (some of which will be powered by AI) in the next phases of the project.

4.1 Implemented functionalities

As mentioned before, the provision and deployment of the data cube service have been the priority for the first release of the FlexiGroBots platform as it will be the base for developing other (geospatial and/or AI-based) services for the pilots.

FlexiGroBots data cube leverages the Open Data Cube (ODC) [29], which is an open-source solution for the management and analysis of Earth Observation data.

accessing, managing, and analysing large quantities of Geographic Information System (GIS) data - namely Earth observation (EO) data. It presents a common analytical framework composed of a series of data structures and tools which facilitate the organization and analysis of large, gridded data collections. As it is explained in [30], "*The Open Data Cube system is designed to (1) Catalogue large amounts of Earth Observation data; (2) Provide a Python-based API for high-performance querying and data access; (3) Give researchers and other users* *easy ability to perform Exploratory Data Analysis (by means of Jupyter notebooks); (4) Allow scalable continent-scale processing of the stored data, and (5) Track the provenance of all the contained data to allow for quality control and updates*".

The following image (based on the work done in [30]) depicts the current deployment and availability of tools provided by the ODC which are currently available in the FlexiGroBots platform (marked in green colour those already available and in yellow the ones that are being integrated at the time of writing this document).

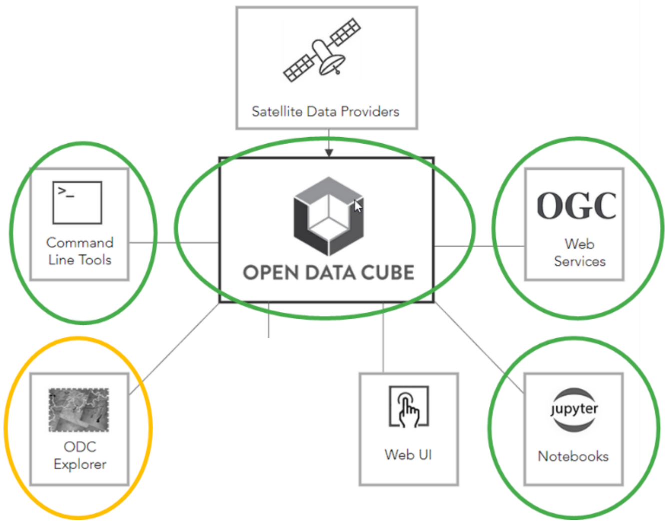


Figure 7 Open Data Cube (ODC) components. Source of original image: [30]

- **Command Line Tools** so that it is possible for developers to interact with the ODC. These include scripts for registering EO data products definitions in the data cube database and indexing the actual EO products (e.g., Sentinel 2 images) by using their accompanying metadata records. Thus, the data cube is aware of their existence and can offer them via the corresponding OGC WMS and WCS APIs.
- **Open Data Cube Explorer** is being currently integrated to provide a visual interface for the exploration of the inventory.
- **Web User Interface (UI)** to obtain visually the outputs of the algorithms.
- **Jupyter Notebooks** contain detailed examples and allow interactive development.
- **Open Geospatial Consortium (OGC) Web Services**: Standard and interoperable service interfaces provided by the ODC in order to facilitate the access to the gridded data to other non-ODC applications (e.g., map visualizers, AI/robotic services using raster data as input). FlexiGroBots ODC currently supports OGC Web Map Service (WMS) and OGC Web Coverage Service (WCS) interfaces, which allows visualizing as images the gridded data and access to it in its raw format respectively. A third OGC interface will be made available in the coming months, that is, OGC Web Processing Service (WPS), allowing to directly

implement within the data cube itself various processes using the satellite/UAVs datasets (e.g., NDVI calculation) and offer them via a common API interface for being used by the pilots’ services/systems/tools within FlexiGroBots.

4.2 Requirements

4.2.1 Technical requirements

The deployment and installation of the different data cube components (see section 4.7) require a Linux-based server (currently tested in an Ubuntu 20.04 virtual machine), as well as Docker and Docker Compose software as each data cube component is provided and run in a separate container.

Storage requirements are low given that the data cube doesn’t store itself the gridded data (either from the satellite or UAVs), but it only stores within its internal PostgreSQL + PostGIS database the metadata about them (the datasets themselves are located somewhere else, e.g., AWS, Azure or a local pilot server with the UAVs captured imagery).

Concerning memory and CPU requirements, according to their technical guidelines [31]:

“The ability to run concurrent processes is based on the number of cores, and the size of each concurrent process depends on the amount of available memory. Analyses on large datasets often require splitting a large geospatial and temporal region into smaller ‘chunks’ for more efficient processing. A ‘chunk’ in this context refers to a smaller portion of a larger dataset; large analysis regions are broken into many smaller subsets and processed independently. As a general rule of thumb,

- *At least one core per desired concurrent user: More per user is desired if execution speed is a concern, for example, each analysis run using our UI is split into 5+ parts that are then computed concurrently*
- *At least one gigabyte of memory per core: Depending on your analysis case, this may need to be higher. Most of our algorithms allow for temporal as well as geospatial chunking, so each process’ memory usage is fairly low”*.

Besides, from our current testing, it has been noticed that the performance of ODC decreases due to non-proximity with the Sentinel-2 datasets (currently using the free copy provided by AWS). Therefore, in order to improve it, one possibility (instead of having to store a copy of all the necessary Sentinel 2 or Landsat imagery locally) is to deploy an instance of the ODC in AWS, so it can benefit from directly accessing those datasets and improve the data access and processing times.

4.2.2 Functional requirements

Before the indexing of the satellite and UAVs imagery can take place, it is necessary to register in the data cube the EO products definitions for the types of images to be ingested. Fortunately, the ODC already provides several of these definitions for many of the most common EO data products, as is the case for Sentinel 2 level 2A product or the Landsat 7 and 8. A comprehensive list of products definitions already prepared can be found here: https://github.com/digitalearthafrica/config/tree/master/products.

In a similar manner, for each type of image product produced by the UAVs, it is necessary to define and register the corresponding EO. In this case, this can be done by taking one of the above-mentioned products definitions as a template and modifying it accordingly to the concrete details of the image produced by the UAV.

Concerning the imagery storage, as mentioned before, ODC doesn't store itself the gridded data (as the recommended approach [32] is just to index it by using the metadata that accompanies it) and therefore, the imagery must be in other (local or cloud) repository.

Nevertheless, the existence of metadata records for each image scene is mandatory (or it must be (semi-)automatically crafted upfront) for the data cube to be able to index them. The ODC can directly process STAC and EO3 metadata files [1]. In the case of the UAVs EO datasets, it is necessary to automatically generate these STAC metadata files from the data itself since these metadata records don't exist.

4.3 Data models

In principle, the ODC can use as input (or provide as output) any grid specification supported by GDAL. Nevertheless, the most common ones will be used, i.e., GeoTiff, JPEG2000 and COG (Cloud-Optimized GeoTiffs).

Concerning the metadata necessary for indexing the EO data products, ODC is able to process metadata records based on STAC [33] and EO3 formats.

4.4 Application Programming Interfaces (APIs)

The ODC provides two different types of APIs for accessing and processing the gridded imagery:

- A Python SDK library, which allows to directly manage the data cube as well as to access and process in a more specific manner the resources. A full specification of the API can be found here: https://opendatacube.readthedocs.io/en/latest/api/index.html

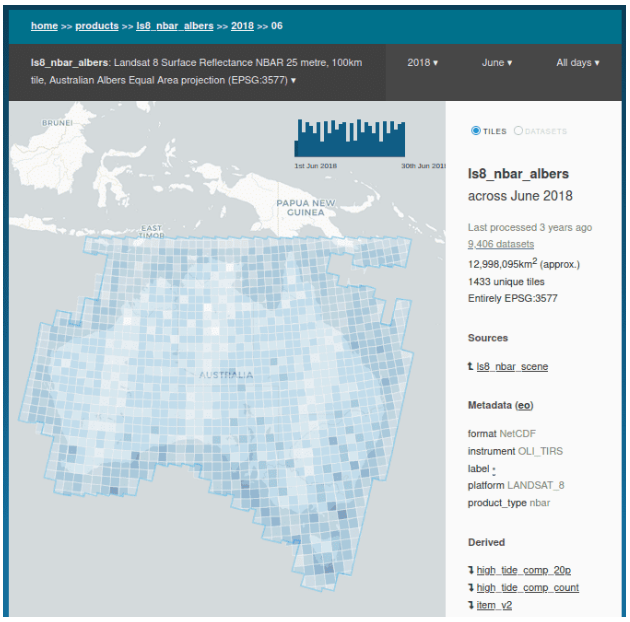
[1] AWS already provides the STAC metadata together with the Sentinel 2 imagery, which is very convenient for indexing it with the ODC.

- A set of OGC service interfaces, which allow to interact with an Open Data Cube instance using client software (for example, Desktop GIS like QGIS, or a web mapping library like Leaflet), through the use of Web Map Service (WMS) [34] and Web Coverage Service (WCS) [35] standards. A third OGC interface, Web Processing Service (WPS) [36], will be made available in the next months, allowing to implement and execute directly over the data cube some processing algorithms.

4.5 Graphical User Interfaces (GUIs)

The ODC provides two means of visually interacting with it:

- Data Cube Explorer (planned for deployment in FlexiGroBots in the next months): it is a web application for searching and browsing the metadata available from an Open Data Cube. It has rich visualisation abilities to show the available data extents and can be used to browse the provenance of indexed data.



ls8_nbar_albers: Landsat 8 Surface Reflectance NBAR 25 metre, 100km tile, Australian Albers Equal Area projection (EPSG:3577) ▾

ls8_nbar_albers
across June 2018

Last processed 3 years ago
9,406 datasets
12,998,095km^2 (approx.)
1433 unique tiles
Entirely EPSG:3577

Sources

ls8_nbar_scene

Metadata (eo)

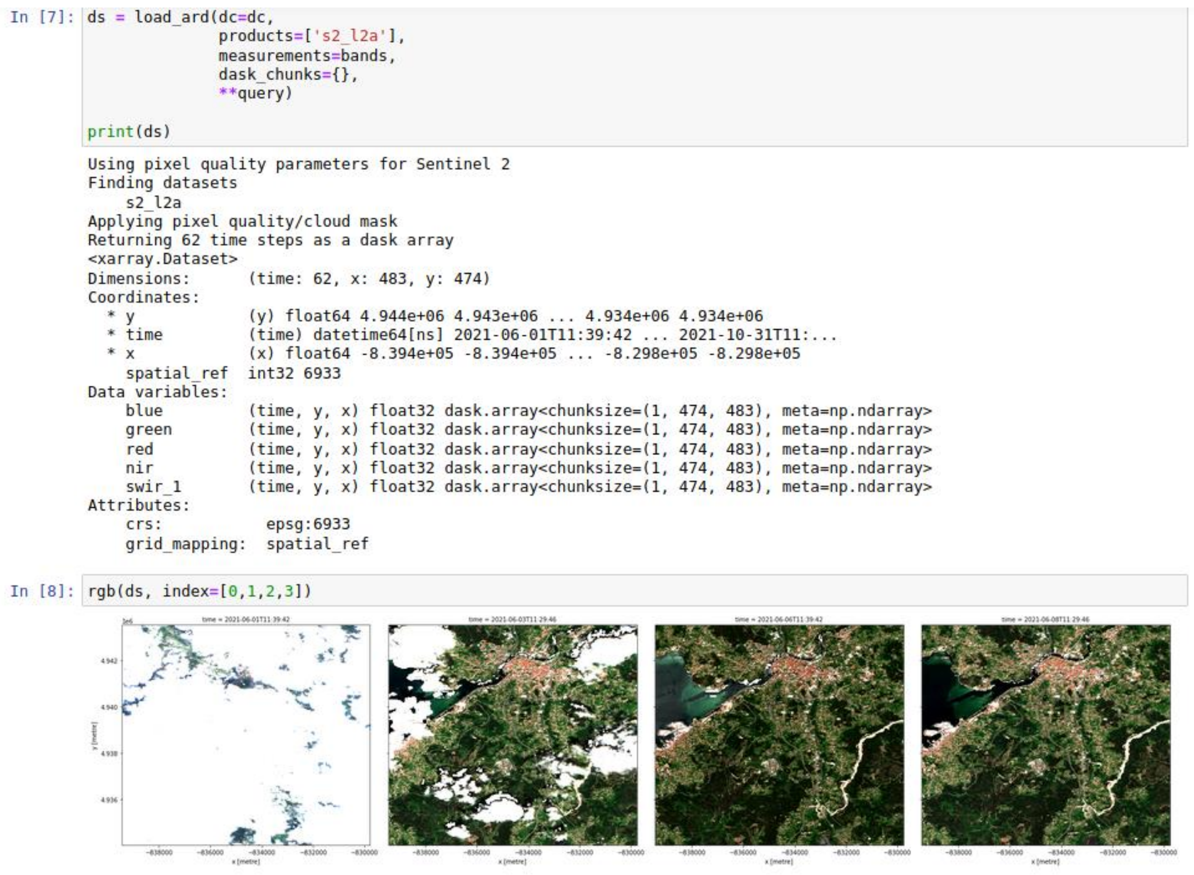
format NetCDF
instrument OLI_TIRS
label :
platform LANDSAT_8
product_type nbar

Derived

high_tide_comp_20p
high_tide_comp_count
item_v2

Figure 8 Data Cube Explorer

- Jupyter notebooks: which give consultancy services and/or expert farmers easy ability to perform Exploratory Data Analysis and testing of algorithms over the gridded data previous to putting them into production.



```
In [7]: ds = load_ard(dc=dc,
                      products=['s2_l2a'],
                      measurements=bands,
                      dask_chunks={},
                      **query)

        print(ds)

        Using pixel quality parameters for Sentinel 2
        Finding datasets
            s2_l2a
        Applying pixel quality/cloud mask
        Returning 62 time steps as a dask array
        <xarray.Dataset>
        Dimensions:      (time: 62, x: 483, y: 474)
        Coordinates:
          * y            (y) float64 4.944e+06 4.943e+06 ... 4.934e+06 4.934e+06
          * time         (time) datetime64[ns] 2021-06-01T11:39:42 ... 2021-10-31T11:...
          * x            (x) float64 -8.394e+05 -8.394e+05 ... -8.298e+05 -8.298e+05
            spatial_ref  int32 6933
        Data variables:
            blue         (time, y, x) float32 dask.array<chunksize=(1, 474, 483), meta=np.ndarray>
            green        (time, y, x) float32 dask.array<chunksize=(1, 474, 483), meta=np.ndarray>
            red          (time, y, x) float32 dask.array<chunksize=(1, 474, 483), meta=np.ndarray>
            nir          (time, y, x) float32 dask.array<chunksize=(1, 474, 483), meta=np.ndarray>
            swir_1       (time, y, x) float32 dask.array<chunksize=(1, 474, 483), meta=np.ndarray>
        Attributes:
            crs:           epsg:6933
            grid_mapping:  spatial_ref

In [8]: rgb(ds, index=[0,1,2,3])
```

time = 2021-06-01T11:39:42
time = 2021-06-03T11:29:46
time = 2021-06-06T11:39:42
time = 2021-06-08T11:29:46
1e6
4.942
4.940
4.938
4.936
y [metre]
-838000
-836000
-834000
-832000
-830000
x [metre]

Figure 9 Jupyter Notebook with some sample code and results for the processing of Sentinel 2 images in the Spanish pilot area

4.6 Installation

The current instance of the ODC deployed in FlexiGroBots platform comprehends running (in an orchestrated manner by means of Docker Compose) of the following separate docker images:

- postgis/postgis:14-3.1-alpine: Postgresql + PostGIS database used for storing all information necessary for correctly managing the ODC metadata, the EO datasets products definitions, the EO products indexed as well as for exposing them through the OGC WMS and WCS service interfaces.
- opendatacube/datacube-index [37]: provides a series of Python scripts for i) registering the EO data products (defined in separate yaml files) into the ODC database, and ii) indexing the EO data products in the ODC.
- opendatacube/ows [38] [39]: provides the OGC WMS, WMST and WCS services on top of the data cube
- opendatacube/cube-in-a-box [40]: provides a complete Jupyter Hub environment (together with all ODC Python SDK libraries, Numpy, Xarray, Pandas and other commonly

used Data Science Python libraries) for working with and executing Jupiter Notebooks that make use of the underlying ODC features and offered EO datasets.

Due to the relative complexity for setting up, deploying and having run all ODC components, three docker-compose files have been prepared with the necessary configurations (e.g., environment variables, a common shared network for permitting that all ODC components are "visible" and accessible to each other, etc.).

The following showcases the typical steps necessary for having a running ODC instance for the Spanish pilot:

1) Registration of EO data products definitions and indexation of Sentinel 2 data products

   1.1) Get the ODC database and the indexer containers up and running

```
$ docker-compose -f docker-compose.products.yaml up
```

   1.2) Attach to the index container and register the Sentinel 2 data product definition

```
$ docker exec -it odc-indexer bash
$ datacube -v system init (this prepares the PostgreSQL database tables)
$ dc-sync-products /conf/products.csv (Add a product definition for Sentinel-2)
```

   1.3) Indexing available Sentinel 2 products [2] for the Spanish pilot (by providing a spatial bounding box and temporal range)

```
$ stac-to-dc --bbox='-10,41,-9,42' --catalog-href='https://earth-
search.aws.element84.com/v0/' --collections='sentinel-s2-l2a-cogs' --datetime='2021-01-
01/2021-12-20'"
```

2) Expose ODC EO data products via OGC Services

   2.1) Get the OGC services up and running

```
$ docker-compose -f docker-compose.ows.yaml up
```

   2.2) Attach to the odc-ows container and update the database tables related to the ows services since there are new EO datasets available [41].

```
$ docker exec -it odc-ows bash
$ datacube-ows-update –views
$ datacube-ows-update
```

3) Deploy Jupyter Hub and sample notebooks

```
$ docker-compose -f docker-compose.jupyter.yaml up
```

[2] Sentinel 2 products are retrieved from Amazon Web Services (AWS), given the convenience of using COG format and the already available STAC metadata for each Sentinel 2 image.

4.7 Prototype availability within FlexiGroBots

Original ODC components code is hosted in the official Open Data Cube GitHub account (https://github.com/opendatacube). However, these have been forked into FlexiGroBots dedicated GitHub repository (https://github.com/FlexiGroBots-H2020) in order to add some specific changes and configuration files required by the project.

In the next weeks, it is planned to deploy a more stable version of the FlexiGroBots ODC instance either in one of the project servers and/or in AWS in order to take advantage of the proximity to the EO data imagery (yet to be decided).

4.8 Release planning

User story ID	User story	Priority	Story points
GEO_US_01	Deployment of basic ODC instance (comprehending PosGIS database, odc-index, odc-ows and odc-jupyter hub) in Atos/AWS.	High	3
GEO_US_02	Registration of Sentinel 2 EO product definition and indexing data for the 3 FlexiGroBots pilots' areas	High	5
GEO_US_03	Description and registration of EO product definition for the UAVs imagery produced by the 3 FlexiGroBots pilots	High	3
GEO_US_04	Processing of FlexiGroBots pilots UAVs produced imagery into COG format in order to improve performance when used by ODC	Medium	2
GEO_US_05	Implementation of Python script for automatically producing STAC metadata out of FlexiGroBots UAVs imagery (preferably from the COGs) and indexing it into the ODC	High	5
GEO_US_06	Deployment of ODC Explorer to facilitate the manual visualization and exploration of available datasets offered by ODC	Low	3
GEO_US_07	Deployment and configuration of odc-wps on top of the ODC in order to enable the possibility to directly execute specific algorithms using the EO data offered by the ODC using the OGC WPS API	Medium	6
GEO_US_08	Implement algorithms (to be discussed and prioritized with pilots, e.g., NDVI calculation) using ODC EO data and expose them via the OGC WPS API	Medium	8

User story ID	User story	Priority	Story points
GEO_US_09	Index other additional EO/raster-based datasets potentially useful/necessary for the FlexiGroBots pilots (e.g., Landsat 8 imagery, DEM, etc.)	Low	4

Table 5 User stories for the geospatial processing and services

5 Common application services

FlexiGroBots will count with several software modules that are designed to be shared and reused between the project's Pilots as well as with other robotic agricultural solutions. These common applications and services will be developed within Task 3.4 of the project. Thus, the partners of the consortium are collaborating to find common services that could benefit all Pilots and enhance the overall capabilities of the FlexiGroBots tool. At this stage of the Task's development, some of the modules are still being defined. Also, the final number of services could increase or decrease depending on the project's requirements. The following subsections present the status of the selected common services, along with their description. The different services have been classified into three categories: (i) situation awareness, (ii) utility and (iii) generalization. More information on each category is provided in the correspondent category.

5.1 Situation Awareness

The services under the category "situation awareness" are designed to increase the robot's capacity to detect and understand its surroundings. All four proposed services are computer vision models, all of them featuring deep learning technology.

5.1.1 SLAM

The partners are developing a Simultaneous Location And Mapping (SLAM) software module. This service will allow robots to create 3D reconstructions of their surroundings, processing clouds of points into maps. Also, they will be able to locate themselves within the created maps. This will be done by processing a sequence of images (video) from a monocular camera located in the UGV itself. This service could be implemented in any UGV equipped with a camera. Since all three pilots are using vehicles with cameras, they all will benefit from this module, creating maps of the different fields and enabling each UGV to locate themselves within those maps.

The software will use classical computer vision techniques for analysing each image in search of landmarks. Each frame of the video will have its own landmarks, a combination of reference points that are intrinsic to the image and allow the camera to position itself in the image. The recognition, movement and variation of those landmarks in successive images allows the module to recreate the camera's movement and create a map (visual odometry).

Also, this module will use the translation matrices (that combine intrinsic properties of the camera with the quaternion description of the movement) and the landmarks to feed a trained deep neural network that will output a 3D reconstruction of the scene.

5.1.1.1 Implemented functionalities

As of M12, we have a SLAM module running in a monocular camera. It is capable of creating maps and locating the camera within them. It still has room for improvement and has only been tested on recordings of Pilot 1 grape farm.

5.1.1.2 Technical requirements

This software module needs an Ubuntu OS and some open-source libraries installed. It requires C++ compilers and Python. We can also run this solution in a containerized manner (Docker). Although it can run in CPU, a machine with access to CUDA libraries for GPU usage is recommended. This SLAM module can run with recorded video or a live camera.

5.1.1.3 Functional requirements

For actual usage of the SLAM common service, the vehicle/UGV should count with a calibrated monocular camera, power supply and a device for computing the module. That last device can be a laptop if the UGV has an internet connection, or an edge device if no connection is provided (in that case the computation must be done in the field).

5.1.1.4 Data models

There are no data models specified in this module so far. It will be done during the second year of the project.

5.1.1.5 Application Programming Interfaces (APIs)

SLAM service uses several open-source libraries, mainly for Python and also for C++. OpenCV, PyTorch, NumPy and Pangolin are some examples.

If the deployment is to be containerized, also Docker will be used.

5.1.1.6 Graphical User Interfaces (GUIs)

This module counts with a custom GUI for showing the generated maps, landmarks, keyframes, camera pose and trajectory. It uses Pangolin for visualization.

5.1.1.7 Installation

The module can be installed in an Ubuntu 18.XX or Ubuntu 20.XX machine that has C++ and Python installed. A document with the needed requirements will be released so any common package manager can create an environment and automatically install the necessary packages

5.1.1.8 Prototype availability within FlexiGroBots

This module has not yet reached the stage of a functional prototype.

5.1.1.9 Release planning

User story ID	User story	Priority	Story points
CAS_SLAM_US_01	Develop software module for camera pose, image keypoints and visual odometry.	High	5
CAS_SLAM_US_02	Develop 3D reconstruction software for generating realistic mesh.	High	6
CAS_SLAM_US_03	Link the SLAM and the 3D Reconstruction to create an integrated module.	High	4
CAS_SLAM_US_04	Develop software for fine-grain interactive camera calibration.	Medium	2
CAS_SLAM_US_05	Create virtual maps and 3D meshes from Pilot 1 and Pilot 3 scenes.	Low	4
CAS_SLAM_US_06	Upload the application to the AI platform and prepare deployment for required devices.	Low	4

Table 6 User stories for the SLAM

5.1.2 People detection, location and tracking

This computer vision module will allow any UGV equipped with a monocular camera to detect people in the images. The detection has three features:

i. Location within the image: a bounding box will be drawn around the person, also estimating the centroid of the box where the person is exactly located within the image.
ii. Estimated distance to the camera (in meters): the software would also output the predicted distance between the centroid of each detected person and the camera itself. This will need a calibration of the camera to work properly.
iii. Tracking: the software will also recognize the intrinsic characteristics of each detected person and assign an identification code. This will allow tracking each individual in a video sequence, enabling posterior processing like trajectory estimation (which is not part of this particular service).

This application is designed to enhance the safety of robotic operations in environments where they interact with humans. This can be implemented in every UGV equipped with monocular cameras, and thus is a common service that could benefit all three pilots.

This service is powered by deep neural networks that extract the features of the image and process them to form its predictions. Also, statistical algorithms like Kalman filters are used.

5.1.2.1 Implemented functionalities

We have successfully implemented people detection and location. The pending work is to calibrate the camera to get a more precise distance estimation and to implement the tracking to the model. Also, Kalman filters need to be implemented.

5.1.2.2 Technical requirements

This module needs an Ubuntu 18.XX or Ubuntu 20.XX with Python. For optimum inference speed GPU compatible with CUDA drivers will be needed as well. This model can run on video or live cameras.

5.1.2.3 Functional requirements

This module needs a monocular camera and a computing unit, whether a PC or an edge device, depending on the connection availability.

5.1.2.4 Data models

There are no data models specified in this module so far. It will be done during the second year of the project.

5.1.2.5 Application Programming Interfaces (APIs)

It needs several Python open-source APIs. PyTorch, NumPy and OpenCV are the most important ones.

5.1.2.6 Graphical User Interfaces (GUIs)

The software outputs the processed video with some markings and drawings showing the outputs of the model.

5.1.2.7 Installation

We will provide a “requirements” document containing the dependencies so it can be automatically installed by common python package managers.

5.1.2.8 Prototype availability within FlexiGroBots

The module is not yet ready for the prototype.

5.1.2.9 Release planning

User story ID	User story	Priority	Story points
CAS_PLDT_US_01	Develop a software module for detecting people in images (bounding box).	High	3
CAS_PLDT_US_02	Develop a module for estimating distances of the detected people towards the camera.	High	6
CAS_PLDT_US_03	Develop camera calibration software and make camera calibration and testing.	High	3
CAS_PLDT_US_04	Develop software for assigning a unique ID to each detected person (tracking)	Medium	4
CAS_PLDT_US_05	Evaluate and test in field conditions (Pilots 1 and 3)	Low	4
CAS_PLDT_US_06	Upload the application to the AI platform and prepare deployment for required devices.	Low	4

Table 7 User stories for people detection, location and tracking

5.1.3 People behaviour estimation

Besides the people detection, this other module also performs "pose estimation", which is the detection of body keypoints or landmarks that describe the posture of a person. With those estimations this module will analyse the change in the pose from frame to frame to classify the person's actions: sitting, running, squatting, etc. This will give the UGVs an estimation of intent and could be used by an additional model to help the robot's decision-making.

As it happens with the previous application explained in Section 5.1.2, this module is designed to enhance the safety of robotic operations. Also, it is a common module that could be used in any of the pilots, as long as they have cameras at ground level.

This computer vision model uses deep learning technology for detecting the keypoints and analysing them for recognizing actions.

5.1.3.1 Implemented functionalities

At the moment, the module makes people detection (bounding boxes) and poses estimation (keypoint detection). The next step would be to train a classifier to recognize actions by analysing several frames.

5.1.3.2 Technical requirements

An Ubuntu OS with Python installed is the minimum technical requirement. It needs a monocular camera for live inference. GPU with access to CUDA drivers is highly recommended to reach optimum processing speed.

5.1.3.3 Functional requirements

For this module to run in the Pilots, it needs monocular cameras installed in the UGVs along with a computer, whether it is a PC, or an edge computing device (like NVIDIA Jetson Nano).

5.1.3.4 Data models

There are no data models specified in this module so far. It will be done during the second year of the project.

5.1.3.5 Application Programming Interfaces (APIs)

The people behaviour estimation module needs several Python libraries like OpenCV, PyTorch and NumPy.

5.1.3.6 Graphical User Interfaces (GUIs)

The software outputs the processed video, drawing the predictions over the original frames.

5.1.3.7 Installation

We will provide a document with the required Python dependencies in a format that can be interpreted by common Python package managers to install them.

5.1.3.8 Prototype availability within FlexiGroBots

There is no functional prototype yet.

5.1.3.9 Release planning

User story ID	User story	Priority	Story points
CAS_PBE_US_01	Develop a software module for detecting people in images (bounding box).	High	3
CAS_PBE_US_02	Develop a module for estimating pose and body keypoints.	High	4
CAS_PBE_US_03	Train model for classifying poses into actions to estimate the behaviour	High	6
CAS_PBE_US_04	Evaluate and test in field conditions (Pilots 1 and 3)	Low	4
CAS_PBE_US_05	Upload the application to the AI platform and prepare deployment for required devices.	Low	4

Table 8 User stories for people behaviour estimation

5.1.4 Moving objects detection, location and tracking

This application is designed to be used from UAV instead of UGV. The images to be analysed would come from drones flying at around 100 hundred meters high, giving a bird's eye view of the farming field. As different human workers, tractors, robots and machines are expected to be working simultaneously in the field, this model is designed to increase the overall safety of the farm's operations.

The detection algorithm of this application will be able to:

i. Detect the different moving objects (e.g., agricultural vehicles, people and UGVs) from a drone's monocular camera. Each individual object will be marked by drawing a bounding box around it.

ii. Location within the image. Once the camera and the model are calibrated, the module can use the pixel coordinates of the bounding boxes to estimate distances between the different objects and locate them within the image.

iii. Tracking. Each object will have a unique identification code that will allow the software to track it from frame to frame. Analysing the relative position of each object in a sequence of images will allow additional models to estimate trajectories.

At least two of the pilots are using drones to make status maps of the fields (Pilot 1 and Pilot 2). Having this common module in their drone's cameras will enhance the overall safety of the operations.

This computer vision module will use deep neural networks to analyse the images and predict the bounding boxes of each vehicle. These machine learning models will be trained with datasets acquired and annotated by partners of the FlexiGroBots project.

5.1.4.1 Implemented functionalities

No functionalities have been yet implemented.

5.1.4.2 Technical requirements

This module will use Python libraries installed on Ubuntu OS (18.XX or 20.XX).

All trainings and inference will be executed in a GPU-accelerated manner, needing GPU with access to CUDA drivers.

Annotated datasets are also needed for the training rounds.

5.1.4.3 Functional requirements

The software can run directly in UAVs if they have monocular RGB cameras and a computer with GPU acceleration onboard (edge device). If the drone does not have such devices, the processing can be served via the internet. In that case, the images will be processed on a PC and the communications should be done via the internet.

5.1.4.4 Data models

There are no data models specified in this module so far. It will be done during the second year of the project.

5.1.4.5 Application Programming Interfaces (APIs)

This is still in the definition phase. It is planned to use common Python libraries like OpenCV, NumPy, TensorFlow and PyTorch, among others.

5.1.4.6 Graphical User Interfaces (GUIs)

To be defined.

5.1.4.7 Installation

A document containing the needed dependencies will be released. It will follow a format compatible with common Python package managers to facilitate a semi-automated installation.

5.1.4.8 Prototype availability within FlexiGroBots

No prototype is available yet.

5.1.4.9 Release planning

User story ID	User story	Priority	Story points
CAS_MODLT_US_01	Annotate video datasets featuring farming objects from atop.	High	6
CAS_MODLT_US_02	Train model for detecting farming objects from atop and build corresponding software module	High	6
CAS_MODLT_US_03	Evaluate and test in field conditions (Pilots 1, 2 and 3)	Medium	4
CAS_MODLT_US_04	Upload the application to the AI platform and prepare deployment for required devices.	Low	4

Table 9 User stories for moving objects detection, location and tracking

5.2 Utility

The “utility” category encompasses models that are used as intermediate steps in the different data processing pipelines. Although each Pilot has its own specific modules, many share common data pre-processing requirements.

5.2.1 GIS plug-in

The main goal of this common application service is to facilitate the consumption of geospatial information from the Open Data Cube tailored and deployed as part of T3.3 and described in detail in chapter 4 of this document. The GIS (Geospatial Information System) plug-in will extend the GeoServer service, also creating a web application to show consumed information like NVDI maps or pilots’ plots.

5.2.1.1 Implemented functionalities

DSS Web App is ready to consume OGC services and show layers published in FlexiGroBots GIS servers. There is a JavaScript SPA consuming REST services and showing geospatial information. It’s necessary to implement a REST client to obtain data from FlexiGroBots.

5.2.1.2 Technical requirements

The main goals in the topic utility “GIS plug-in” are:

1. Use GeoServer extension, Smart Data Loader. The requirements for these specifications are:
    - Virtual machine.

    - Azure VM: Standard D2 v2
    - Azure Database for PostgreSQL server
        - Postgres Database + PostGIS extension.
- GeoServer installation.
- Smart Data Loader installation.

2. For the DSS web platform the requirements are:
    - JavaScript web page.
        - Azure VM: B1s standard.
    - GIS front-end framework: OpenLayers.
    - Back-end service to support web page.
        - Azure VM: B2s standard. (APIs)
        - Azure Database for PostgreSQL server
            - Postgres Database + PostGIS extension.

5.2.1.3 Functional requirements

Share the information stored in the Pilot's different DBS, using different data models.

Consume the OGC services offered by the FlexiGroBots platform.

5.2.1.4 Data models

An extension of the Foodie Data Model will be used as a FlexiGroBots standard.

5.2.1.5 Application Programming Interfaces (APIs)

Pilot 1 DSS Web Platform will offer an API through the GeoServer plugin to share data with other partners/stakeholders.

5.2.1.6 Graphical User Interfaces (GUIs)

The DSS Web App will look like the following image and will show layers offered by the FlexiGroBots OGC services.

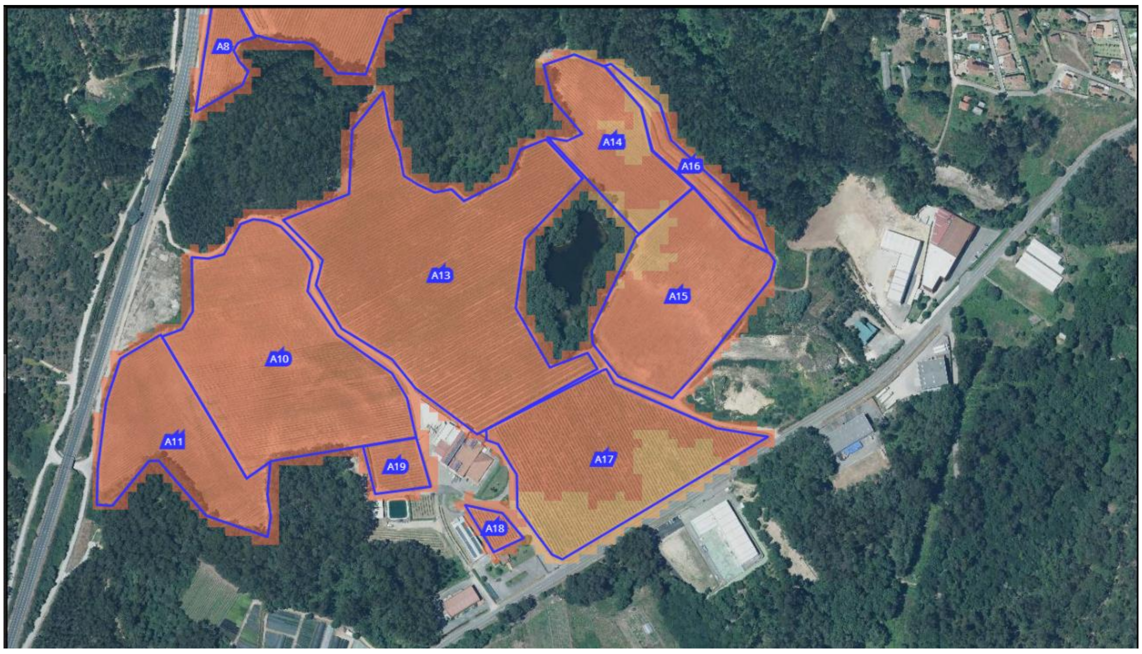


Figure 10 Example of FlexiGroBots GIS plugin-in DSS Web App

5.2.1.7 Installation

To install GeoServer, follow the instructions in the official website guide [42].

The process to install Smart Data Loader extension is the same as other GeoServer extensions. Follow the instructions on the official community modules website [43].

5.2.1.8 Prototype availability within FlexiGroBots

It will have a web app capable to consume OGC FlexiGroBots endpoints and will share its own data in a standard data model (FDM extension).

5.2.1.9 Release planning

User story ID	User story	Priority	Story points
CAS_GIS_US_01	Define Pilot 1 data model	High	5
CAS_GIS_US_02	Discuss Foodie Data Model shared version	High	3
CAS_GIS_US_03	Provide Azure resources	High	4
CAS_GIS_US_04	Install GeoServer and Smart Data Loader extension in Pilot 1.	High	4
CAS_GIS_US_05	Define Smart Data Loader conversion schema.	Medium	3
CAS_GIS_US_06	Develop Pilot 1 DSS Web App user interfaces.	Medium	5

User story ID	User story	Priority	Story points
CAS_GIS_US_07	Develop Pilot 1 DSS back-end services.	Medium	5
CAS_GIS_US_08	Testing and validation.	Low	3

Table 10 User stories for GIS plug-in

5.3 Generalization

Each of the pilots has specific computer vision modules to detect different types of pests (diseases, insects or weeds). The modules under the category "generalization" will be abstractions of the Pilot's specific models so they can be reused in different pilots or agricultural solutions. Please read the following subsections to find the descriptions and examples.

5.3.1 Disease detection

Partners working in Pilot 1 are developing, within Task 4.2, a computer vision model to detect the *Botrytis* disease in grapes. A common service for detecting disease in fruit can be obtained from that model in several ways. This module will have to stall its development until the *Botrytis* detection for Pilot 1 is ready. Then we can work on abstracting that model and make it generic. This can be done mainly with two approaches:

i. Changing the *target*. In this case, the target for the classification and detection task is the disease "*botrytis*". One could apply transfer learning techniques to reuse that algorithm and transform it to detect other diseases, augment the number of detected diseases or, in the most generic case, make an anomaly detector that indicates whether a grapevine looks healthy or not.
ii. Changing the *domain*. In Pilot 1 they focus on a disease that affects grapes. Using domain adaptation techniques, the model can be reused to change the grape category or even the fruit itself, depending on the intrinsic characteristics of the disease.

Both approaches would require small datasets to make retraining of neural network models.

5.3.1.1 Implemented functionalities

No functionalities have yet been implemented.

5.3.1.2 Technical requirements

The final requirements depend on the specific module of Pilot 1, still in development.

5.3.1.3 Functional requirements

As this module needs the one from Pilot 1 to be completed, there are no defined requirements yet.

5.3.1.4 Data models

There are no data models specified in this module so far. It will be done during the second year of the project.

5.3.1.5 Application Programming Interfaces (APIs)

To de defined.

5.3.1.6 Graphical User Interfaces (GUIs)

Still an ongoing discussion.

5.3.1.7 Installation

Not yet defined.

5.3.1.8 Prototype availability within FlexiGroBots

No prototype is available yet.

5.3.1.9 Release planning

User story ID	User story	Priority	Story points
CAS_GDIS_US_01	Check the software module and requirements for Pilot 1's *Botrytis* detection.	High	2
CAS_GDIS_US_02	Build domain adaptation and/or transfer learning module for allowing re-training of the *Botrytis* model.	High	6
CAS_GDIS_US_03	Evaluate and test in field conditions with data from Pilot 3.	Medium	4
CAS_GDIS_US_04	Upload the application to the AI platform and prepare deployment for required devices.	Low	4

Table 11 User stories for disease detection

5.3.2 Insect infestation detection

Following the same approach as done in the section before (5.3.1), this module will be a generalization of the specific computer vision model that the partners are developing for Pilot 2 within Task 5.2. In this case, the same techniques can be applied to change the model from detecting "*meligethes aeneuss*" in rapeseed plants to "insects" in crops. This generalization would become a common service that can be useful to all pilots and for other solutions beyond this project. Like in the previous section, there are two main approaches: changing the target (transfer learning) or changing the domain (domain adaptation). In both cases, the most generic outcome would be to get an anomaly detection module for insects in plants. Like before, the detailed definition of this module depends on the final implementation of the specific model in T5.2.

5.3.2.1 Implemented functionalities

No functionalities have been implemented yet.

5.3.2.2 Technical requirements

The specific module of Pilot 2 is still in development.

5.3.2.3 Functional requirements

We need the finished version of the Pilot 2 insect detector, so there are no defined requirements yet.

5.3.2.4 Data models

There are no data models specified in this module so far. It will be done during the second year of the project.

5.3.2.5 Application Programming Interfaces (APIs)

To de defined.

5.3.2.6 Graphical User Interfaces (GUIs)

Not yet defined.

5.3.2.7 Installation

Still an ongoing discussion.

5.3.2.8 Prototype availability within FlexiGroBots

No prototype is available yet.

5.3.2.9 Release planning

User story ID	User story	Priority	Story points
CAS_GINS_US_01	Check the software module and requirements for Pilot 2's *Melighethes Aeneuss* insect detection.	High	2
CAS_GINS_US_02	Build domain adaptation and/or transfer learning module for allowing re-training of the insect detection model.	High	6
CAS_GINS_US_03	Evaluate and test in field conditions with data from Pilot 3.	Medium	4
CAS_GINS_US_04	Upload the application to the AI platform and prepare deployment for required devices.	Low	4

Table 12 User stories for disease detection

5.3.3 Weed detection

From specific weed species in blueberry fields.

The third "generalization" module has the same structure as the previous two (sections 5.3.1 and 5.3.2). In this case, partners of Pilot 3 will develop a module for detecting weeds in blueberry farms. The final list of weed species to be detected is not yet defined. Again, a common module will be abstracted from the specific weed detector by transfer learning and domain adaptation techniques, both of which require small datasets for retraining rounds. In this case, the most generic outcome would be a model that can discern the crops from the rest of the plants in the image, i.e., weeds. As it happens in the other two generalization modules, the definition and requirements of this common application will have to wait until the completion of the specific weed detection module for Pilot 3 (Task 6.2).

5.3.3.1 Implemented functionalities

No functionalities have yet been implemented.

5.3.3.2 Technical requirements

The definition of the final requirements needs to wait until the specific module of Pilot 3 is completed.

5.3.3.3 Functional requirements

There are no defined requirements yet.

5.3.3.4 Data models

There are no data models specified in this module so far. It will be done during the second year of the project.

5.3.3.5 Application Programming Interfaces (APIs)

To de defined.

5.3.3.6 Graphical User Interfaces (GUIs)

Still an ongoing discussion.

5.3.3.7 Installation

Not yet defined.

5.3.3.8 Prototype availability within FlexiGroBots

No prototype is available yet.

5.3.3.9 Release planning

User story ID	User story	Priority	Story points
CAS_GWEED_US_01	Check the software module and requirements for Pilot 3's weed detection model.	High	2
CAS_GWEED_US_02	Extend the weed detection module with customization features so it can be reused in other domains.	High	6
CAS_GWEED_US_03	Evaluate and test in field conditions with data from Pilot 1.	Medium	4
CAS_GWEED_US_04	Upload the application to the AI platform and prepare deployment for required devices.	Low	4

Table 13 User stories for weed detection

6 Mission Control Centre

6.1 Implemented functionalities

The FlexiGroBots Mission Control Centre (MCC) has the objective to design, plan and supervise heterogeneous multi-robot operations for precision agriculture tasks.

It will consist of the following components:

- A planner with functionalities to design and plan missions, leveraging reusable plans and tasks.
- A Graphical User Interface (GUI) so that the human operator can monitor and intervene in the missions at operating time, avoiding information overload and potentiating effective participation.
- A supervisor with functionalities for semi-automated monitor and adaption of the mission to ensure its success and execution according to the original plan. It will include functionalities for intelligent processing and notification of events and failures and to avoid overloading the operator with excessive information.

As it is described as part of the specification of the use-cases for the MCC included in deliverable D2.2, the FlexiGroBots MCC will cover the following steps during the planning of execution of the missions:

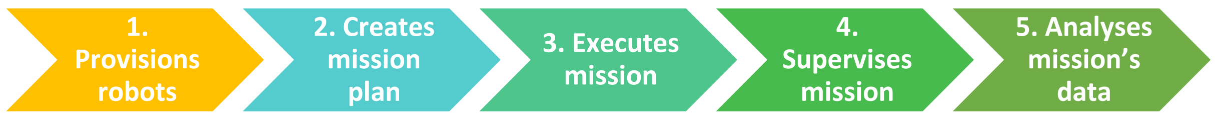


Figure 11 Overview of MCC functionalities

The design of MCC internal architecture is based on the lessons learnt as a result of the FP7 RHEA (Robot Fleets for Highly Effective Agriculture and Forestry Management) project, which was coordinated by CSIC, the technical coordinator of the FlexiGroBots project. RHEA architecture was based on the following components:

- A *Mission Manager* to automate the sequence of the work to be executed by the robotics fleets.
- A *Mission Planner* to decide the number and type of UGVs and UAVs to accomplish the mission and to generate action plans.
- A *Mission Controller* that decomposes the action plans into a sequence of commands understandable for each robot. This will not imply actually controlling the robot itself since most of the vehicles will be autonomous, especially for UGVs. Rather, the role of this component will be to extract from the overall mission plan, the subset of specific actions that must be executed by each individual robot (or fleet of robots controlled

by a Robot Fleet Management System) and to send them so that they can be executed. It will be in charge also of the dynamic synchronisation of actions or the update of the missions.

- A *Mission Supervisor* receives real-time low-level information from each robot (potentially from the Robot Fleet Management System) and reports alerts to the operator when unexpected events or failures are detected.
- A *Dispatcher* acts as a broker to distribute the information and messages between the different components of the MCC.

Figure 12 shows the architecture of FlexiGroBots MCC that is strongly inspired by the structure of the RHEA project.

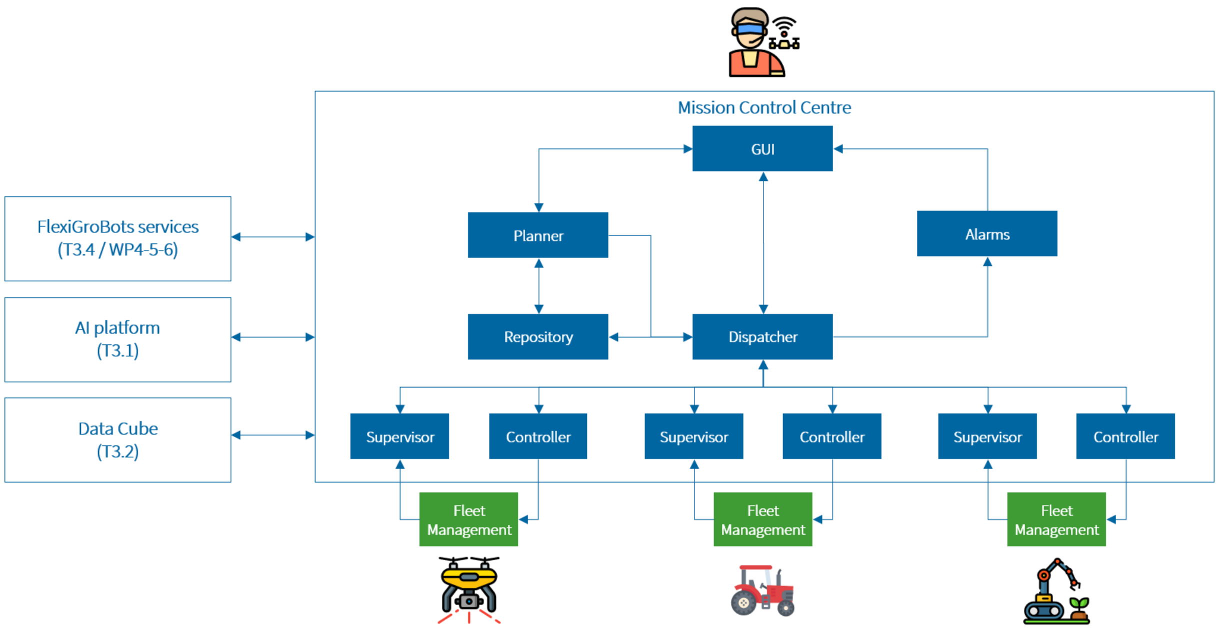


Figure 12 MCC detailed architecture

As the first step within task T3.5, the partners decided to analyse candidate technologies for the implementation of the MCC considering the requirements raised by the project's pilots and the overall objectives of the project. The comparison was done between RHEA and **QGroundControl (QGC)** [44], a promising open-source framework sponsored by the Dronecode Foundation (DF) [45] of the Linux Foundation [46] that allows controlling several vehicles' types that support PX4 Pro, Ardupilot or MAVLink protocol. The results of the analysis are included in Table 14.

	RHEA Mission Control Centre	**QGroundControl**
Technology	C++ (MMD) + MATLAB (Supervisor)	C++
Robots' provisioning	No	Yes. Extensions may be needed

	RHEA Mission Control Centre	QGroundControl
Mission planning	Yes	Yes. Focused on drones and single robots. Extensions needed.
Mission execution	Yes	Yes. Extensions are needed for ground robots and agriculture tasks.
Mission supervision	Yes	Yes. Extensions needed for ground robots and agriculture tasks
Analyses mission's data	No	No
Multiplatform support	No	Yes
Docker image	No	Yes
Commits	-	18,025 commits
Community	-	204 watchers, 2.6K forks
License	Open source	Dual license: Apache2.0 + GPLv3
Standards	-	MAVLink, PX4

Table 14 Comparison between RHEA Mission Control Centre and QGC

In the light of the results of the analysis and in order to have the maximum possible impact in the European robotics community, **FlexiGroBots MCC will be based on QGC as baseline technology**. Therefore, the implemented functionalities of the first prototype of the MCC rely completely upon QGC and they will be extended in the next iterations.

6.2 Requirements

6.2.1 Technical requirements

According to QGC [47] documentation, the minimum technical requirements are:

- Intel Core i5 processor.
- 8GB of RAM memory.
- Video graphics card Nvidia or AMD.

6.2.2 Functional requirements

QGC currently supports MAVLink enabled vehicles for both PX4 and ArduPilot stacks. Integration with DJI drones will be explored in the projects. Support for ground robots will be also implemented from scratch.

6.3 Data models

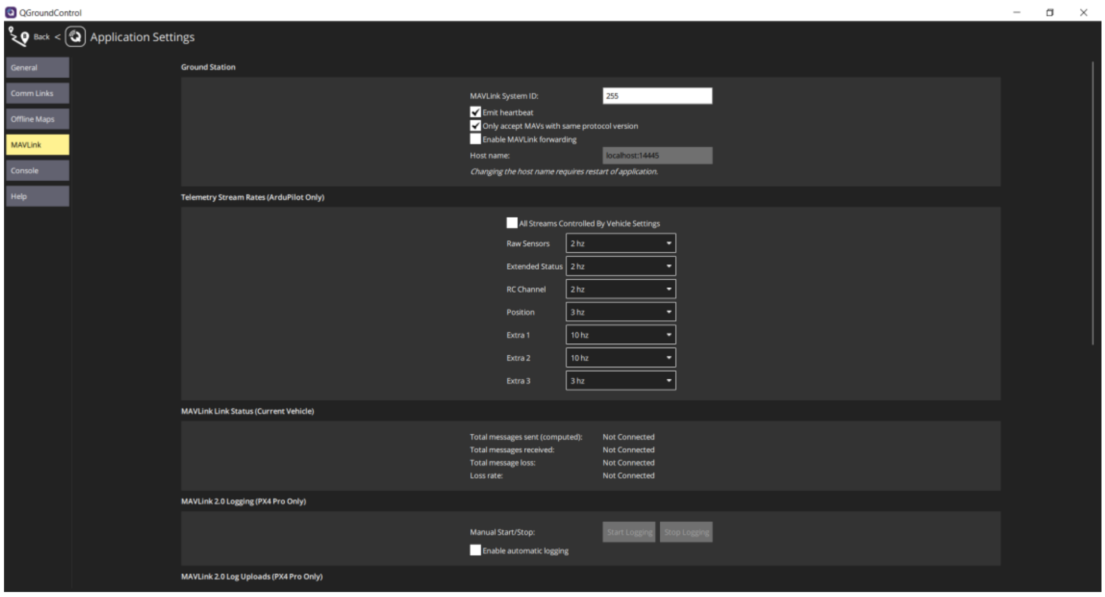
There are no data models specified in this module so far. It will be done during the second year of the project.

6.4 Application Programming Interfaces (APIs)

To de defined.

6.5 Graphical User Interfaces (GUIs)

QGC include a powerful GUI. Figure 13 includes a screenshot of the configuration of the application settings. It has options to manage general parameters, communication links, MAVLink settings etc.



Ground Station

MAVLink System ID: 255
✓ Emit heartbeat
✓ Only accept MAVs with same protocol version
☐ Enable MAVLink forwarding
Host name: localhost:14445
Changing the host name requires restart of application.

Telemetry Stream Rates (ArduPilot Only)

☐ All Streams Controlled By Vehicle Settings

Raw Sensors	2 hz
Extended Status	2 hz
RC Channel	2 hz
Position	3 hz
Extra 1	10 hz
Extra 2	10 hz
Extra 3	3 hz

MAVLink Link Status (Current Vehicle)

Total messages sent (computed):	Not Connected
Total messages received:	Not Connected
Total message loss:	Not Connected
Loss rate:	Not Connected

MAVLink 2.0 Logging (PX4 Pro Only)

Manual Start/Stop:
☐ Enable automatic logging

MAVLink 2.0 Log Uploads (PX4 Pro Only)

Figure 13 Application settings in QGC GUI

The options to plan robotics missions with QGC are shown in Figure 14.

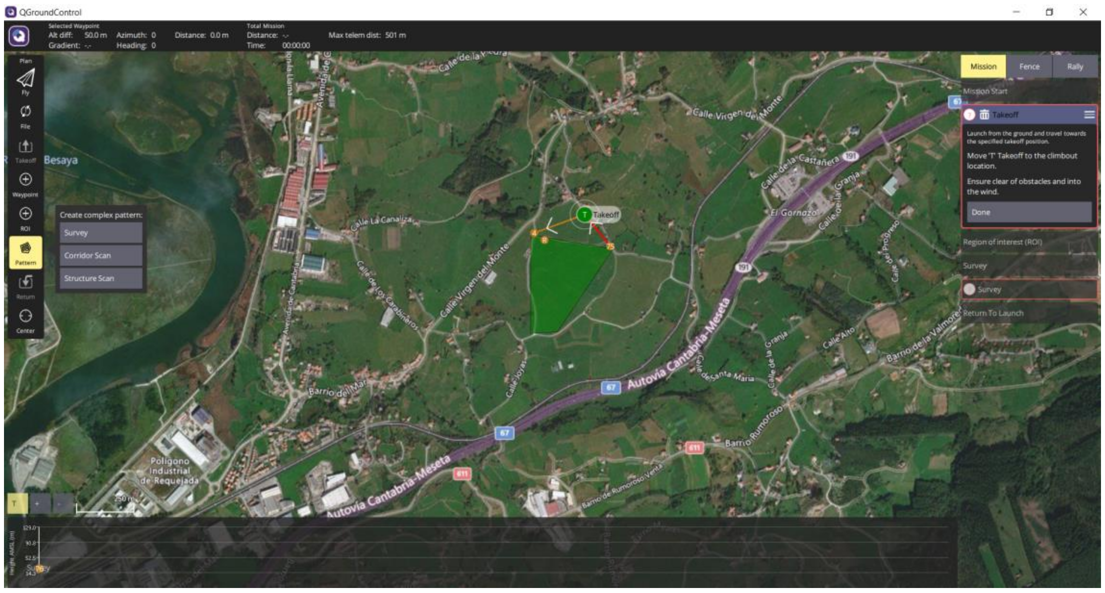


Figure 14 Plan view of QGC GUI

The current functionality is limited to planning missions of just one vehicle and completely focused on UAVs and without offering specialised functionalities for precision agriculture tasks (e.g., application of treatments, weeds removal, selection of types of crops). All these features will be developed and added by the FlexiGroBots project and published with an open-source license.

6.6 Installation

QGroundControl can be installed on multiple platforms: Windows, Mac OS X, Ubuntu Linux, Android and iOS.

6.7 Prototype availability within FlexiGroBots

A fork from the original QGroundControl repository has been done in FlexiGroBots GitHub repository: https://github.com/FlexiGroBots-H2020/qgroundcontrol.

It will be used for the development of new functionalities within the scope of the project.

6.8 Release planning

User story ID	User story	Priority	Story points
MCC_US_01	Create a simulation environment for development purposes.	High	1
MCC_US_02	Create controllers with ground robots in pilot 1 to send missions.	High	5

User story ID	User story	Priority	Story points
MCC_US_03	Create controllers with ground robots in pilot 2 to send missions.	High	5
MCC_US_04	Create controllers with ground robots in pilot 3 to send missions.	High	5
MCC_US_05	Create supervisors with ground robots in pilot 1.	High	2
MCC_US_06	Create supervisors with ground robots in pilot 2.	High	2
MCC_US_07	Create supervisors with ground robots in pilot 3.	High	2
MCC_US_08	Implement planner to support missions with multiple robots.	Medium	13
MCC_US_09	Implement events processing and alarms notification.	Medium	8
MCC_US_10	Implement DataSpace connector.	Low	3
MCC_US_11	Implement MQTT communication extension for QGC.	High	2

Table 15 User stories for MCC

7 Conclusions

This document presents the summary of the functionalities of the first version of the software prototypes that have been implemented for the FlexiGroBots platform at the end of the first year of the project. For each one of the components that are part of the system, the document describes the corresponding features, technical and functional requirements, APIs, GUIs, installation procedures, availability within the project and release planning.

In the case of the Artificial Intelligence platform, the available prototype is based on one of the most prominent open-source frameworks available at this moment: Kubeflow, which has been installed in a dedicated infrastructure provided by Atos. For information persistence, the solution is based on MinIO. Both technologies are being tailored to satisfy the requirements of the data scientists working in the development of Machine Learning models as part of the pilots' use-cases. In the next step, this solution will be integrated with the AI4EU catalogue and with AutoML capabilities.

IDSA open-source building blocks are being used as the core for the implementation of the Agriculture Data Space. Several connectors are being developed to interconnect the multiple digital systems that compose the FlexiGroBots platform (i.e., AI platform, open data cube, MCC) and the three pilots.

With respect to the geospatial services, an instance of the Open Data Cube has been also deployed and it is available for use for the three pilots.

The project has successfully specified a group of common AI-powered application services that will be embedded in the pilots but that could be easily applicable in many agricultural use-cases independently of the type of crop or technologies used. A first prototype is ready for the SLAM service.

Finally, the Mission Control Centre will allow planning, executing and supervising operations of multi robots' fleets for precision agriculture tasks. Again, the solution is based on a widely used open-source component, QGroundControl. It will be extended to create a custom version for agricultural robots' fleets.

The source code of the FlexiGroBots platform components is mostly available in the project GitHub repository, which is accessible to the whole European robotics community. New updates for the platform will be continuously delivered in the next months so that a more mature version can be integrated and validated during the second round of tests in the three pilots during the spring and summer months.

An updated version (D3.2) of this document will be also available at the end of the second year of the project, in December 2022. The final one (D3.3) will be published in December 2023, at the end of the project.

References

[1] "FlexiGroBots D2.2 – Requirements and platform architecture specifications".

[2] "JIRA documentation: Estimate in story points.," [Online]. Available: https://confluence.atlassian.com/jirasoftwareserver/estimate-in-story-points-938845204.html.

[3] "FlexiGroBots D2.7 – Pilot alignment and joint assessment report".

[4] "FlexiGroBots GitHub," [Online]. Available: https://github.com/FlexiGroBots-H2020/.

[5] "Kubernetes webpage," [Online]. Available: https://kubernetes.io/.

[6] A. AI. [Online]. Available: https://www.acumos.org/.

[7] DVC. [Online]. Available: https://dvc.org/.

[8] "Git LFS," [Online]. Available: https://git-lfs.github.com/.

[9] "lakeFS," [Online]. Available: https://lakefs.io/.

[10] "Pachyderm," [Online]. Available: https://www.pachyderm.com/.

[11] "Delta Lake," [Online]. Available: https://delta.io/.

[12] "Kubeflow," [Online]. Available: https://www.kubeflow.org/.

[13] "MLflow," [Online]. Available: https://mlflow.org/.

[14] "MinIO," [Online]. Available: https://min.io/.

[15] "Kubeflow installation," [Online]. Available: https://www.kubeflow.org/docs/started/installing-kubeflow/.

[16] "Kubeflow Pipelines SDK API," [Online]. Available: https://www.kubeflow.org/docs/components/pipelines/sdk-v2/.

[17] "Kubeflow Pipelines API," [Online]. Available: https://www.kubeflow.org/docs/components/pipelines/tutorials/api-pipelines/.

[18] "Kubeflow manifests installation," [Online]. Available: https://github.com/kubeflow/manifests#installation.

[19] "International Data Spaces Association GitHub repository," [Online]. Available: https://github.com/International-Data-Spaces-Association.

[20] "The International Data Spaces Association on GitHub," [Online]. Available: https://github.com/International-Data-Spaces-Association/idsa.

[21] "How To Build Data Spaces?," [Online]. Available: https://github.com/International-Data-Spaces-Association/idsa/tree/main/how-to-build-data-spaces.

[22] D. Connector. [Online]. Available: https://www.isst.fraunhofer.de/en/business-units/data-business/technologies/Dataspace-Connector.html.

[23] "IDS-testbed," [Online]. Available: https://github.com/International-Data-Spaces-Association/IDS-testbed.

[24] "Dataspace Connector GitHub project," [Online]. Available: https://github.com/International-Data-Spaces-Association/DataspaceConnector.

[25] "DataSpace Connector REST API," [Online]. Available: https://international-data-spaces-association.github.io/DataspaceConnector/Documentation/v6/RestApi.

[26] "IDS Metadata Broker API," [Online]. Available: https://app.swaggerhub.com/apis/idsa/IDS-Broker/1.3.1.

[27] "Dynamic Attribute Provisioning Service (DAPS)," [Online]. Available: https://github.com/International-Data-Spaces-Association/IDS-G/blob/main/Components/IdentityProvider/DAPS/README.md.

[28] "IDS Metadata Broker GitHub repository," [Online]. Available: https://github.com/International-Data-Spaces-Association/metadata-broker-open-core.

[29] "Open Data Cube," [Online]. Available: https://www.opendatacube.org/overview.

[30] B. R. J. P. G. Y. P. C. &. G. G. Chatenoux, "Bringing Open Data Cube into Practice.," in *In Workshop Material. GRID-Geneva & University of Geneva (pp. 3-3).*, 2019.

[31] "ODC System Requirements," [Online]. Available: https://www.opendatacube.org/system-requirements.

[32] "Open Data Cube FAQ," [Online]. Available: https://www.opendatacube.org/faq.

[33] "SpatioTemporal Asset Catalogs," [Online]. Available: https://stacspec.org/.

[34] "OGC Web Map Service," [Online]. Available: https://www.ogc.org/standards/wms.

[35] "OGC Web Coverage Service," [Online]. Available: https://www.ogc.org/standards/wcs.

[36] "OGC Web Processing Service," [Online]. Available: https://www.ogc.org/standards/wps.

[37] [Online]. Available: https://hub.docker.com/r/opendatacube/datacube-index.

[38] [Online]. Available: https://github.com/opendatacube/datacube-ows.

[39] [Online]. Available: https://hub.docker.com/r/opendatacube/ows.

[40] [Online]. Available: https://github.com/opendatacube/cube-in-a-box.

[41] “Installation of datacube-ows,” [Online]. Available: https://datacube-ows.readthedocs.io/en/latest/readme.html?highlight=datacube-ows-update%20%E2%80%93views#installation.

[42] [Online]. Available: https://docs.geoserver.org/stable/en/user/installation/index.html.

[43] [Online]. Available: https://docs.geoserver.org/stable/en/user/community/smart-data-loader/install.html.

[44] “QGroundControl,” [Online]. Available: http://qgroundcontrol.com/.

[45] “Dronecode Foundation,” [Online]. Available: https://www.dronecode.org/.

[46] “Linux Foundation,” [Online]. Available: https://www.linuxfoundation.org/.

[47] [Online]. Available: https://docs.qgroundcontrol.com/master/en/getting_started/download_and_install.htm.