DELIVERABLE

D3.2 – Specification of Integrated and Modular Architecture for Drones

Project Title	COMP4DRONES
Grant Agreement number	826610
Call and topic identifier	H2020-ECSEL-2018
Funding Scheme	Research & Innovation Action (RIA)
Project duration	36 Months [1 October 2019 – 30 September 2022]
Coordinator	Mr. Mauro Gil (INDRA)
Website	www.comp4drones.eu

ECSEL Joint Undertaking
Electronic Components and Systems for European Leadership

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826610

Document fiche	
Authors:	See Page 4
Internal reviewers:	Muhammad Tufail – SHERPA Jean-Patrick Mascomere – TOTAL
Work Package:	WP3
Task:	T3.1
Nature:	R
Dissemination:	PU

Document History			
Version	**Date**	**Contributor(s)**	**Description**
V0.6	08/04/2021	Mahmoud Hussein, Reda Nouacer	Initial version of the reference architecture
V1.0	17/06/2021	See Page 4	First draft of the complete deliverable
V1.1	02/07/2021	Mahmoud Hussein, Reda Nouacer	Deliverable ready for internal review
V1.1	12/07/2021	Muhammad Tufail, Jean-Patrick Mascomere	Internal review process completed
V1.2	21/07/2021	Mahmoud Hussein, Reda Nouacer	Final version of the deliverable
V0.6	08/04/2021	Mahmoud Hussein, Reda Nouacer	Initial version of the reference architecture

Keywords:	Reference Architecture, Flight Management, Flight Control, Flight Navigation, Mission Management, Payloads, Tools, and Hardware Platforms
Abstract (few lines):	This deliverable provides the **specification** of the **COMP4DRONES reference architecture**. To provide this specification, first, a list of **requirements** that drive the architecture, and a number of underlying **concepts** are presented. Second, a set of **building blocks** of drone systems and their **interaction patterns** are identified and described. Third, the blocks and the patterns are used to define the reference architecture. The architecture includes four main clusters: **flight management, flight control, flight navigation,** and **mission management**. Finally, key **enabling technologies** that support the drone architecture are discussed.

DISCLAIMER

ACKNOWLEDGEMENT

This document is a deliverable of COMP4DRONES project. This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement Nº 826610

D3.2 Authors:

Partner Name	Contributors
INDRA	Adrian Irala
IMEC-BG	Hiep Luong, Michiel Vlaminck, Murali Jayapala
BUT	Svetozar Nosko
UWB	Miroslav Flídr, Ondřej Severa
ENAC	Fabien Bonneval, Gautier Hattenberger
SIEMENS	Federico Cappuzzo
SCALIAN	Raphaël Lallement
ENSMA	Syed Ali Ajwad, Yassine Ouhammou, Matheus Ladeira
UNIMORE	Andrea Marongiu, Alessandro Capotondi
UNISANNIO	Giuseppe Silano, Luigi Iannelli, Valerio Mariani
UNISS	Daniel Madronal Quintin, Francesca Palumbo, Tiziana Fanni
UNIVAQ	Mario Di Ferdinando, Stefano Digennaro
EDI	Rihards Novickis, Daniels Jānis Justs
ACORDE	Fernando Herrera, David Abia
HIB	Diego Fuentes
IKERLAN	Iñigo Muguruza, Leire Rubio
CEA	Mahmoud Hussein, Reda Nouacer
AI	Stefano Delucchi
SM	Jiri Bartak, Petr Bartak
UDANET	Domenico Bianchi, Maurizio Preziuso
MODIS	Daniela Parletta
ROT	Niccolò Cometto, Diego Grimani, Maria Luisa Scalise
IFAT	Dominic Pirker, Rainer Matischek

Table of Contents

Table of Figures

Table of Tables

Definitions, Acronyms and Abbreviations

Acronym	Title
ABB	Architecture Building Block
AP	Architecture Pattern
APF	Artificial Potential Function
BDS	BeiDou Navigation Satellite System
BVLOS	Beyond Visual Line of Sight
DAA	Detect and Avoid
FPGA	Field Programmable Gate Arrays
GCS	Ground Control Station
GMC	Generic Mission Controller
GNC	Guidance, Navigation and Control
GNSS	Global Navigation Satellite System
GLONASS	Global Navigation Satellite System
GPS	Global Positioning System
HSI	Hyperspectral Imaging
HSM	Hardware Security Module
IoT	Internet of Things
IPS	Indoor Positioning System
IIRA	Industrial Internet Reference Architecture
MSaaS	Modelling and Simulation as a Service
QoS	Quality of Service
RPA	Remotely Piloted Aircraft
SA	Situation Awareness
SLAM	Simultaneous Localization and Mapping
UAS	Unmanned Aerial System
UAV	Unmanned Aerial Vehicle
UTM	Unmanned Traffic Management

Executive Summary

In this deliverable, we provide the **COMP4DRONES reference architecture** that enables the **easy integration** and **customization** of systems that are embedded in drones. To do that, first, we start by presenting **requirements** underlying/deriving the reference architecture. These requirements are divided into two main groups: **architectural requirements**, and **architectural elements' requirements**. The architectural requirements are generic and driven from the objective of having modular and customizable architecture that eases the development and integration of drone systems. The elements' requirements are specific to features of the drone system. These features include flight planning, flight navigation, flight control, data management, mission management, etc.

Second, to define the reference architecture based on its requirements, a set of **underlying concepts** are presented. These concepts include a **strategy** to follow in defining the architecture, a possible **decomposition** of the drone system, and a **template** for specifying the **architecture blocks**. The strategy for defining the architecture starts with identifying the different blocks of the system and their interaction patterns, and then the blocks are linked together to form the reference architecture.

Third, the different building **blocks** of the drone systems and their interaction **patterns** are identified and presented. The blocks are classified into three domains: **control**, **operations (management)**, and **usage**. The control domain includes the blocks for flight planning, guidance, and control together with the actuation, perception, and communication as a support. The management domain includes blocks for managing the health and the payload of the drone. The usage domain includes blocks for data and mission management. In each of these domains, there are patterns that show/specify the interactions between the different blocks. These patterns include: flight planning, health management, data management, flight control, etc.

Fourth, based on the identified building blocks and their interaction patterns, we have clustered the different blocks into four main groups: **flight control**, **flight management**, **flight guidance**, and **mission management**. These groups are then put/linked together to form the **reference architecture** of the drone systems.

Finally, in addition to the identified building blocks, a number of key-enabling **technologies** are presented. These technologies support one or many building blocks of the system, and they include **hardware**, **basic software**, **sensing**, and **data processing** components.

1 Introduction

This deliverable provides the specification of the **COMP4DRONES** reference architecture for Unmanned Aerial Systems (UAS). The reference architecture is defined at different levels of abstraction. By staying at a higher level of abstraction, it enables the identification and comprehension of the most important issues and patterns across its applications in many different use-cases (i.e., broad applicability). By avoiding specifics, the reference architecture allows subsequent designs to follow the reference architecture without the encumbrance of unnecessary and arbitrary restrictions. The architecture aims to provide a template with common vocabulary to stress commonality for drone systems, and consists of a list of drone functions/features and some indication of their interfaces and interactions with each other and with external functionalities.

In the field of software architecture[1], many empirical studies have shown the following common benefits of adopting a software reference architecture within an organization: (a) improvement of the software systems interoperability by establishing a standard solution and common mechanisms for information exchange; (b) reduction of the development costs of the software projects through the reuse of common assets; (c) improvement of the communication inside the organization because stakeholders share the same architectural mindset. In the same manner, the adoption of the **COMP4DRONES** reference architecture will accelerate the unmanned aerial systems delivery through the re-use of an effective solution, and avoiding mistakes by learning from experience.

The **COMP4DRONES** architecture distils and abstracts common characteristics, features and patterns from use cases defined in the project as well as elsewhere. It will be refined and revised continually as feedback is gathered from its application to the use-cases developed in project as well as real-world deployment of drone systems. The **COMP4DRONES** reference architecture design is also intended to take into account today's available technologies for identifying technology gaps based on the architectural requirements. This will in turn drive new technology development efforts by the drone system community.

It is important to understand that unmanned aerial system (drone system) is not only formed by the drone (i.e., Unmanned Aerial Vehicle (UAV)) but also includes external infrastructure. Figure 1 shows common blocks of a standard UAS. The common blocks include:

- The *flight sensors* that are responsible for providing information to perform a flight. These sensors include Global Navigation Satellite System (GNSS) receiver, inertial sensors, altimeters, pressure sensor, etc.
- Flight *actuators* (e.g., engines, spoilers, rudder, stabilizer, etc.) which are responsible for acting the commands coming from the flight control to follow the trajectory demanded by the navigation system.
- The *UAV avionics* that are formed by all the electronic systems that allow the UAV's autonomous flight (e.g., planning, guidance, control, etc.).
- The *payload* which is made by one or several sensors/actuators needed to carry out the drone mission. The most common payloads are cameras, but there exist some different payloads such as radars, LiDARs, environmental sensors, etc.
- *Communication links* that enable the wireless communication between the UAV and the ground control station. It is common to have different communication links for command and control (commonly known as C2 – Command and Control connection) and for the payload. It is also

1 Martinez-Fernandez, Silverio; Medeiros Dos Santos, Paulo Sergio; Ayala, Claudia P.; Franch, Xavier; Travassos, Guilherme H. (2015). "Aggregating Empirical Evidence about the Benefits and Drawbacks of Software Reference Architectures". 2015 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM). pp. 1–10. doi:10.1109/ESEM.2015.7321184. hdl:2117/80457. ISBN 978-1-4673-7899-4.

important to highlight that those communications can be ground-to-air (direct) or through satellite (indirect).

- External infrastructures such as ground segment where the ground control station (GCS), Unmanned Traffic Management (UTM), etc. are located. Such external infrastructures are used for data and mission management.

Figure 1: General UAS composition

Another important aspect to consider is the flow of information between the UAV and its environment. The two most important operational connections from a safety point of view are: 1) the bidirectional connection between the UAV and the ground control station, and 2) the flow of information from the environment to the sensors.

This document structure is as follows. First, we start by presenting the requirements that drive the drone system architecture in Section 0. The strategy to define the reference architecture is described in Section 0. The architecture's building blocks and interaction patterns are described in Section 0. The reference architecture and its application to two drone systems (i.e., piloted and semi-autonomous drones) are discussed in Section 5. In Section 6, the different enabling technologies that support the architecture are described.

2 Requirements for Drone System Architecture

The primary stakeholders for the drone architecture are the engineers and developers that will implement the architecture in their applications. Thus, most of the effort in designing the architecture should be to meet the engineers'/developers' needs for the drone system. These needs can be divided into two groups: architecture requirements and requirements for architecture elements (described in the next sub-sections).

2.1 Overall Architecture Requirements

The following are requirements that have been collected from researchers and industrials to help defining exactly what a **modular/customizable** architecture for drone systems needs to specify[2]:

- The architecture shall support the specification of the system functions **independently from the implementation aspects** (i.e., independent from any specific modelling and programming language).
- The architecture shall **allow and facilitate the mapping to platform dependent models** (i.e., annotation, extension of SW languages, components, HW processors, and other resources the system functions will be mapped to).
- The architecture shall enable the **capture of the communication protocol** (communication behaviour) among modules/functions.
- The architecture shall enable **low coupling** between the system modules.
- The architecture shall provide mechanisms to explicitly describe overall drone system and its configuration, and moreover to **facilitate the instantiation of modules by indicating their potential auto-configuration** for the different target mappings (in case such a feature is supported).
- The architecture shall support and facilitate the use of libraries for enabling **reuse of already developed modules**, and modules of the same type shall be compatible.
- The architecture shall support homogeneous encapsulation (common I/O interfaces on modules) for the same type of algorithms for **seamlessly pick and replace**.

2.2 Specific Requirements

In the previous sub-section, several requirements that are either common for all architecture elements or refer to aspects that are transversal (i.e., touch several elements or their interrelations) are presented. In the following, we present requirements more related to specific architecture's elements. These specific requirements are presented and associated to the general functionalities and sub-functionalities of UAS presented in Figure 1. Such a decomposition is directly related to and supports the different activities that need to be performed since a request is received to do a mission, until the completion of its successful execution. Those activities include perception, actuation, flight planning, flight guidance, flight control, health management, payload management, communication, data management, mission planning and supervision (i.e., mission management).

2.2.1 Perception

The **perception** shall:

- Provide geo-referenced position, attitude, and velocity upon a global time scale (e.g., GPS and UTC).
- Provide weather information.

[2] https://commons.erau.edu/edt/264/

- Provide the status of the UAV environment.

2.2.2 Actuation

The **actuation** shall:

- Compute the command for each actuator from the command vector (output of attitude and position controllers).
- Prevent, if applicable, saturation of actuators that would reduce the stability of the flight (priorities between attitude, heading, and position can be defined).

2.2.3 Flight Planning

To plan a UAV flight, two features are needed: trajectory planning and trajectory validation.

The **trajectory planning** shall:

- Be able to find a sequence of valid waypoints that moves the UAV from the source to destination based on the route planned by the mission planner.
- Be capable to perform:
  - Fuel calculations.
  - Weight and balance take-off and landing data calculations.
  - Terrain avoidance warning and minimum altitude calculations.
- Be able to re-plan the trajectory when needed.

The **trajectory validation** shall:

- Evaluate the validity of the flight plans requested. The possible outputs of the validation process shall be “planned”, “manual” or “denied” status of the trajectory.
- Provide alternative flight plans when possible if the original flight plan requested was denied.
- Be able to receive the authorization request for an alternative flight plan proposed.
- Perform the complete authorization process.
- Provide the authority with each manual flight plan for its manual authorization.
- Modify airspace allocation when a flight manager or operation manager becomes unavailable.
- Detect trajectory conflicts between different agents.
- Send to the drone operator a notification message with the complete output of the authorization process.

2.2.4 Flight Guidance

The flight guidance is responsible for trajectory execution, geo-fence preservation, and obstacle detection and avoidance.

The **trajectory execution** shall:

- Coordinate the execution of the drone trajectory (i.e., performing waypoint navigation).
- Calculate the set-points for position, velocity, and attitude controllers while considering the physical constraints and properties of the drone.

The **obstacle detection and avoidance** shall:

- Obtain data about the UAV surroundings.
- Process the UAV surrounding data to create a map of the UAV environment.
- Detect an obstacle from the UAV environment’s map.
- Know the drone type and its manoeuvring capabilities.

- Calculate a valid new trajectory to avoid the obstacle.
- Be configurable, to select the sensor or algorithm used in the sensor data processing and map generation.

The **geo-fence preservation** shall:

- Monitor the drone position and the distance between the drone and the geo-fence, and generate a warning if the distance becomes less than a certain threshold.
- Generate a set of commands and execute them to fly away the drone from the fence.
- Alter the drone trajectory if required when actions taken to preserve the geo-fence is executed.

The **commander executor** shall:

- Receive the commands from mission supervision or trajectory execution.
- Execute the commands in a safe manner.
- Allow the pilot to safely take-off, land, and navigate the drone manually.

2.2.5 Flight Control

Different controllers are necessary to fly the drone, which are position, velocity, attitude and rate controllers.

The **position and velocity** control shall:

- Calculate the desired attitude and thrust, while considering constraints on speed and supporting a speed control input when used.
- Support the control and constrain of the required speed metrics, i.e., airspeed (fixed-wing) or ground-speed (rotorcraft).

The **attitude and rate control** shall:

- Calculate the command vector to stabilize the orientation of the UAV, while taking into account constraints on rotation speeds.
- Calculate the command vector to stabilize the body rates of the UAV when used with rates inputs.

2.2.6 Health Management

The health management include power management, fault management, and diagnostics.

The **power management** shall:

- Be capable of restoring power in sufficient time to avoid loss of critical mission data or loss of UAV control during power outages.
- Enable an uninterrupted power supply for critical phases of mission execution.
- Estimate the state of the available power.
- Provide the actual power status and remaining operation time.

The **fault management** shall:

- Detect faults during UAV boot-up, and its mission execution.
- Provide management for faults and create triggers for failsafe actions.
- Switch to emergency mode if an error condition (fault) is detected.

The **diagnostic** shall:

- Generate diagnostics data (out of monitored status) for delivery to mission supervision, and to fault and power management.
- Store the diagnostics data.

2.2.7 Payload Management

The **payload management** shall:

- Allow the setting of the payload parameters (e.g., exposure time, when to start capture and when to stop capture, and orientation of the payload).
- Coordinate the execution of the payload work plan.
- Capture mission-specific data.
- Execute mission-specific action (e.g., drop a package).
- Be able to provide processed and unprocessed data to the user/ground controller.
- Be able to tag/stamp the captured data with geo-referenced location and time.

2.2.8 Communication

The communications shall support interactions between system parts through:

- Data links for command and control, and the UAV payload data.
- A simultaneous uplink and downlink capability.
- The capability to connect to a local area network.
- The ability to use radio links for digital message transmission.
- Securing the data transmission.

2.2.9 Data Management

Two source of data exists in the drone system: data about the UAV and data concerning its environments. These data are collected/managed through flight logger, prognostics, data analytics and storage:

The **flight logger** shall:

- Keep track of
  - All flight-related state variables such as position, orientation, and speed.
  - The status of individual platform components such as motors, battery, and processor operation.
  - The status of the sensors and their measurements, as well as the actuators and their implemented values.
  - Faults that have occurred during systems execution.

The **prognostics** shall:

- Foresee problems related to the state of charge of the battery as being insufficient or putting at risk the execution of the mission with a certain safety.
- Foresee situations of possible collision danger based on the planned trajectory, the objects detected, the UAV current location and expected trajectory as can be derived from the UAV sensors.
- Identify possible hardware failures based on anomalous signals.

The **payload data analytics** shall:

- Extract high level information from raw data.
- Perform predictive analysis to forecast the status of the drone environment.

The **data storage** shall:

- Provide a means of inputting/storing data.
- Be able to communicate with a server to receive, extract, and push data.

- Be capable of adding additional storage without a major hardware reconfiguration to meet the growth of storage requirements.
- Enable access of data to only authorized parties.

2.2.10 Mission Management

The mission management includes two main features/functions: mission planning and mission supervision.

The **mission planning** shall:

- Be capable of importing map information.
- Be capable of providing point-and-click route and payload planning.
- Provide override of payload and UAV automated/preprogrammed inputs.
- Be capable of storing mission plans and exporting them.
- Be capable of changing the mission plan during mission execution.

The **mission supervision** shall:

- Display the location and status of the UAV.
- Provide dynamic mission and sensor re-planning during the mission execution.
- Receive, process, format, store and retrieve flight and payload data and perform exploitation of payload data.
- Have the capability to receive and control payloads.
- Provide the operator a caution/warning when the UAV has a malfunction.

3 Underlying Concepts

The architecture of a system or a federation of systems describes "the fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution". Thus, the system architecture provides a plan or a blueprint for the system. Architectures can be designed at various levels of abstraction. Thus, in this section, we describe a strategy for defining the different levels of abstraction of the drone system. This strategy is adopted from the Modelling and Simulation as a Service (MSaaS[3]) architecture. We also describe the basic decomposition of the drone system following the Industrial Internet Reference Architecture (IIRA[4]) standard, where the drone system is one of the fastest growing and innovative sectors of Internet of Things (IoT)[5]. Furthermore, we present a template for specifying the different blocks of drone systems.

3.1 Strategy for Defining the Reference Architecture

There is little consensus in general on the various levels of abstraction of a reference architecture or on how to name them. Thus, in MSaaS reference architecture, two notions are central: Architecture Building Block (ABB), and Architecture Pattern (AP).

ABBs are the elements that constitute an architecture, and each ABB should have attributes that specify its functions. APs are high-level suggestions for ways of combining ABBs. There are several levels of abstraction for architectures. At the highest level, an architecture ontology might declare types of ABBs and APs. For example, ABB types such as "(business) process", "service", "repository", "service container"; and AP types such as "consumer pattern", "service invocation pattern", that are pertinent for any Service Oriented Architecture (SOA). Next, the actual ABBs and APs of the various ABB types and AP types can be used for declaring a domain-specific overarching architecture.

The manner in which ABBs and APs are specified might be standardized. For example, an ABB representing a service would be of type "service" and its specification may follow some standard for service specification. Then, a reference architecture is designed by composing the ABBs guided by the APs from the overarching architecture. In addition, an architecture topology (or several) should be designed at the reference architecture level to delineate intended systems boundaries and the boundaries in which interoperability standards are enforced. From a reference architecture, individual solution architectures (also called target architectures) that specify solution implementations may be derived. There should be methods for refining architectures at one abstraction level to the next. The spectrum of architecture abstraction levels and such methods are what is referred to as an architecture framework.

3 https://nmsg.sto.nato.int/themes/msaas
4 https://www.iiconsortium.org/IIRA.htm
5 https://www.gsma.com/iot/aviation/

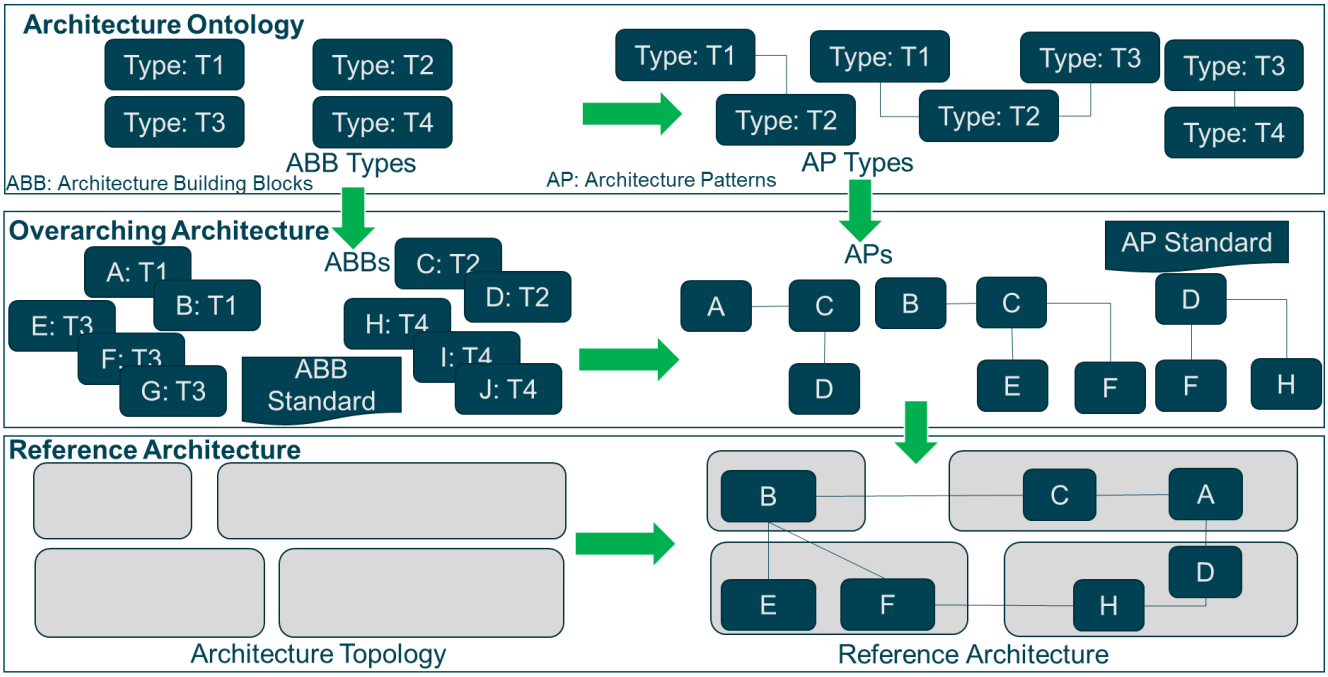

Figure 2: Strategy for specifying the reference architecture

The notions of "overarching architecture" and "reference architecture" are different. In Figure 2, an architecture ontology provides types of Architecture Building Block (ABB) and Architecture Pattern (AP). An overarching architecture consists of specific ABBs and APs of various types, with standards for specifying ABBs and APs. Various architecture topologies specifying system and interoperability boundaries help in designing reference architectures using ABBs and APs. From this, solution architectures with implementation-specific systems or solution implementations can be designed/derived.

To specify the **COMP4DRONES** reference architecture, we follow the same strategy used for specifying the MSaaS reference Architecture. Following this strategy, several levels of abstraction are used for the architecture specification.

At the highest level, an architecture ontology declares the types of Architecture Building Blocks (ABBs) and Architecture Patterns (APs) as shown in Figure 2. For example, in the drone domain, ABB types such as "guidance", "navigation", "control"; and AP types such as 'detect and avoid pattern', 'geofencing pattern' can be declared. Next, the actual ABBs and APs of the various ABBs types and APs types can be used for declaring a domain specific overarching architecture. For example, an ABB representing a trajectory planning would be of type "guidance" and its specification may follow some standards for planning algorithms specification. Then, the reference architecture is designed by composing ABBs guided by APs from the overarching architecture. Finally, from the reference architecture, individual solution architectures for a specific drone system can be derived.

3.2 Drone System Decomposition

To manage the drone system complexity, there is a need for an architecture framework to decompose the system into appropriate categories/sub-systems. Such a framework allows a systematic evaluation of such systems, as well as the resolution necessary to architect and build such systems. The drone system is an example IoT system, and then we are going to adopt a reference architecture from the IoT domain to decompose the drone system.

In the IoT domain, the Industrial Internet of Things Reference Architecture (IIRA) has been introduced. The IIRA is the outcome of applying the "ISO/IEC/IEEE 42010:2011"[6] Systems and Software Engineering–Architecture Description' standard to Industrial Internet of Things systems (IIoT). It first identifies and highlights the most important architectural concerns commonly found in IIoT systems across industrial sectors, and classifies them into four viewpoints along with their respective stakeholders. These four viewpoints are: business viewpoint, usage viewpoint, functional viewpoint, and implementation viewpoint.

The most relevant viewpoint for this deliverable is the functional viewpoint. This viewpoint is an architecture viewpoint that frames the concerns related to the functional capabilities and structure of an industrial internet of things (IIoT) system and its blocks. The functional viewpoint decomposes a typical IIoT system into functional domains highlights the important building blocks that have wide applicability in many industrial verticals. The functional domains are: control domain, operations domain, information domain, application domain, and business domain.

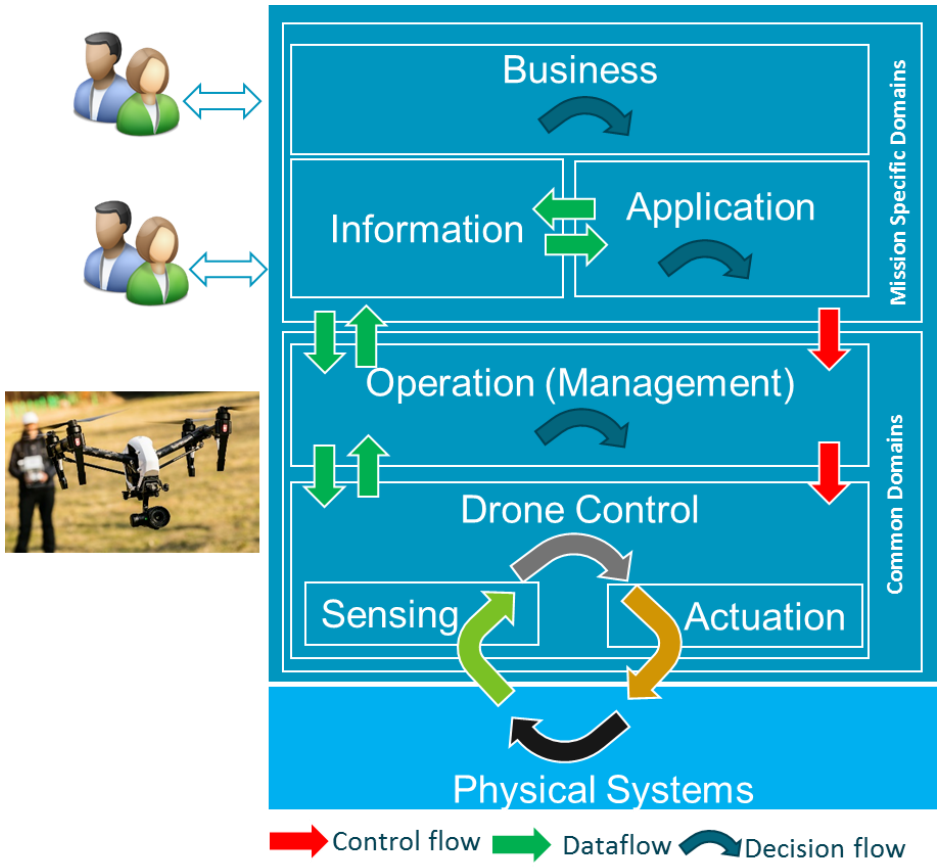

Figure 3: Structure for drone system

Inspired from the IIRA, a structure for a drone system architecture is shown in Figure 3. For the drone domain, the functional structure can be divided into two main parts: common functions to perform the flight, and functions for mission-specific operations. Relative to the IIRA functional architecture, the control and operation domains correspond to common flight functions, while the business, information and application domains correspond to the mission-specific functions. In the following section, we describe these domains.

Data and control flows take place in and between these functional domains. Figure 3 illustrates how the functional domains relate to each other with regards to data and control flows. Green arrows show how

6 ISO/IEC/IEEE: "ISO/IEC/IEEE 42010 Systems and software engineering -- Architecture description", 2011

data flows circulate across domains. Red arrows show how control flows circulate across domains. Other horizontal arrows illustrate some processing taking place within each domain, to process input flows and generate new forms of data or control flows.

Controls, coordination and orchestration exercised from each of the functional domains have different granularities and run-on different temporal cycles. As it moves up in the functional domains, the coarseness of the interactions increases, their cycle becomes longer and the scope of impact likely becomes larger. Correspondingly, as the information moves up in the functional domains, the scope of the information becomes broader and richer, new information can be derived, and new intelligence may emerge in the larger contexts.

3.3 Template for the Architecture Blocks

In the robotics domain, there are two modelling approaches: SmartSoft[7] and openRTM[8]. SmartSoft looks at each system block model in terms of communication patterns, which include clearly-defined semantics for interfacing these blocks, while openRTM hides the implementation of the common interface from the block developer and the integrator who make a system by combining the blocks. As a result, the block developer can focus on the implementation of the main logic and the integrator can focus on the whole system design without worrying about the implementation details of the block.

Inspired from openRTM and SmartSoft block architectures, we have proposed a template for the generic architecture blocks as shown in Figure 4.

The block template contains the following elements:

- **Introspection and Configuration**: Introspection is the mechanism to acquire meta-information of the generic block, while the configuration function is used for changing the user-defined parameters externally at deployment time or runtime.
- **Service Ports**: Service ports are used for providing functions and using external functions in a command-base. There are two types of interfaces - Provider (Provided interface) and Consumer (Required interface). The Provided interface provides functions to external units. Required interface is for requesting/using the function on external blocks.
- **Data Ports**: A data port is used for transmitting or receiving data. There are two types: input ports (for reading data through readers), and output ports (for writing data through writers).
- **Block Behaviour**: The behaviour is corresponding to the implementation of functions into a block. It includes common block states such as inactive, active, error, etc.

7 http://smart-robotics.sourceforge.net/
8 https://openrtm.org/openrtm/en/doc/aboutopenrtm/rtc_architecture

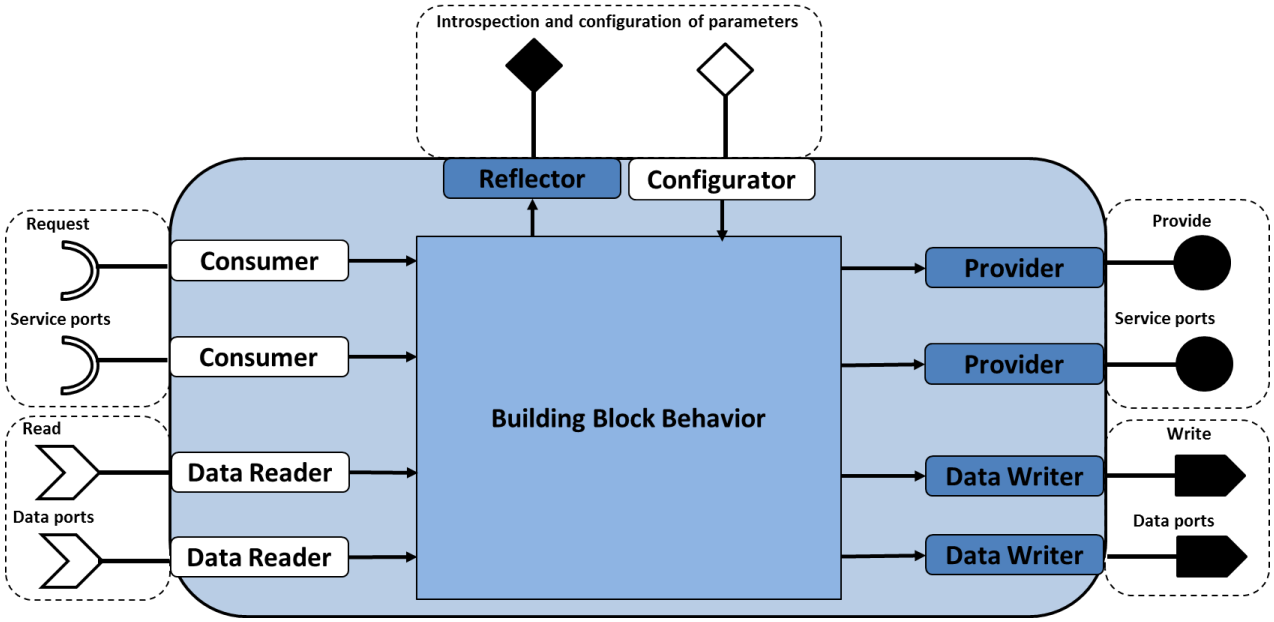

Figure 4: A template for the architecture generic blocks

4 Reference Architecture Building Blocks and Patterns

In this section, we describe the different architecture building blocks of drone systems and their patterns of interactions. First, an architecture building block (ABB) represents a block of the drone reference architecture. It describes a capability and serves as reference for the creation of a solution architecture and solution implementation. A capability is an ability that an ABB possesses and is expressed in terms of requirements. An ABB is technology-aware and provides references to enabling technology. Enabling technology provides the means for the technical realization of an ABB in a so-called "solution".

Second, an architecture pattern (AP) expresses a fundamental structural organization schema for software systems. It provides a set of predefined subsystems, specifies their responsibilities, and includes rules and guidelines for organizing the relationships between them. Architecture patterns are best practices for creating architectures from ABBs[9]. Patterns help the architect in the construction of architecture models and composition of architecture views for a solution architecture. Patterns can be generic or specific to a certain domain.

4.1 General Structure of UAS

The building blocks of unmanned aerial system can be divided into two main groups: the drone/UAV, and the ground control (GC). These two groups also interact with the external services as shown in Figure 5.

First, the different blocks of the UAV include guidance, navigation, control, storage, perception, actuation, security, payload management, and health management. These blocks are responsible for executing a mission defined/guided by the ground control. They also interact with each other internally and communicate externally through radio, Wi-Fi, and satellite communications. More details about these blocks and their interactions are described in Section 4.2 and Section 4.3.

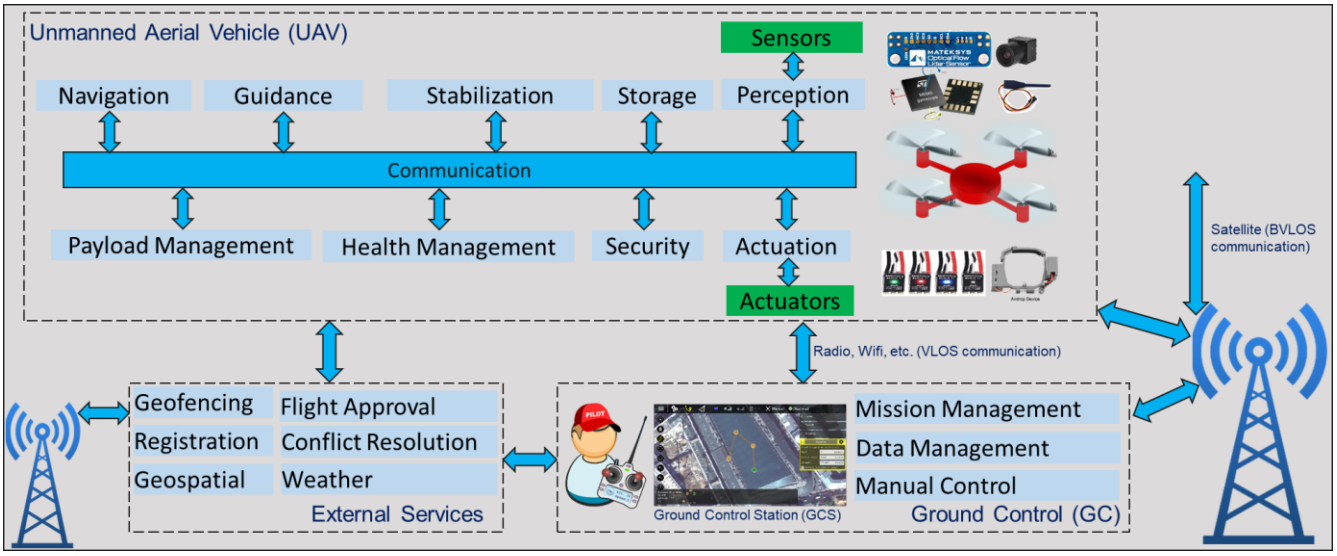

Figure 5: The building blocks of UAS and their interactions with external services

Second, the ground control (described in Section 4.4) includes a ground control station (GCS) to perform mission and data management and the manual control of the UAV either by the pilot through a remote control or through the ground control station. Finally, the external services support the operation of the

9 Frank Buschmann et al, "Pattern-Oriented Software Architecture, Volume 1: A System of Patterns", 1996

drone by providing maps, weather information, drone, registration, validating planned trajectory. It also keeps tracking the drone and warn the pilot when the geo-fence is to be violated.

The blocks shown in Figure 5 can be mapped/distributed to the different domains identified by the industrial internet reference architecture (see Section 3.2). In the following, we describe the different domains and their building blocks and interaction patterns.

4.2 Control Domain

The control domain is a functional domain for specifying/implementing the UAV system control. It represents the collection of functions that are supported/provided by control and automation systems as shown in Figure 6.

4.2.1 Overview

The control domain comprises a set of common functions as depicted in Figure 6. Their implementation may be at various levels of complexity and sophistication depending on the systems, and, in a given system, some blocks may not exist at all. These functions are provided by many control and automation units in various forms and in different architectures. The core functions of the functional domain comprise fine-grained closed-loops, reading data from sensors and analysing these data (i.e., *perception and storage*), applying rules and logic (i.e., flight *planning* and *guidance*) based on acquired data and stored information (e.g., geo-fence), and exercising control over the physical system (i.e., flight control) through actuators (i.e., "*actuation*"). Such closed-loop systems usually require high timing accuracy and resolution.

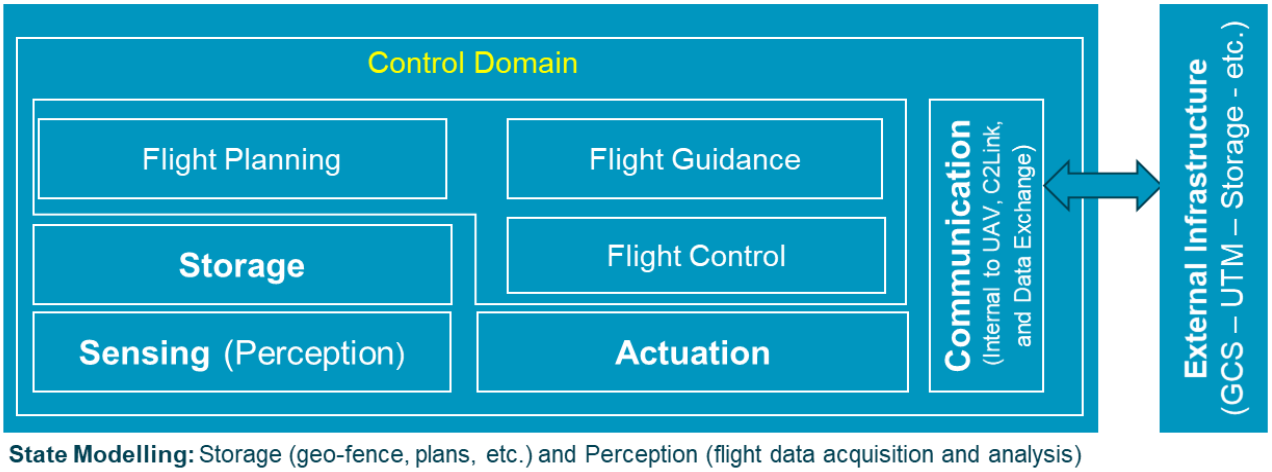

Figure 6: Control domain

The executor (i.e., planning, guidance, and control) executes control logic related to the understanding of the states, conditions, and behaviour of the system under control and its environment in accordance with control objectives. The control objectives may be programmed or otherwise set by static configuration, be dynamic under the authority of local autonomy, or be advised dynamically by systems at higher tiers. The outcome of the control logic is a sequence of actions to be applied to the system under control through actuation. It may also lead to interactions with external systems through *communication* mechanism.

The executor can be designed following the GNC[10] (Guidance, Navigation and Control) framework. The GNC framework is a branch of engineering dealing with the design of systems to control the movement of vehicles, especially, automobiles, ships, aircraft, and spacecraft. In many cases, these functions can be performed by trained humans. However, because of the speed of, for example, a rocket's dynamics,

10 https://www.nasa.gov/sites/default/files/atoms/files/gnc.pdf

human reaction time is too slow to control this movement. Therefore, systems are used for such control. Even in cases where humans can perform these functions, it is often the case that GNC systems provide benefits such as alleviating operator work load, smoothing turbulence, fuel savings, etc. In addition, sophisticated applications of GNC enable automatic or remote control.

In the following sub-sections, we describe the different building blocks of the control domain and their interaction patterns.

4.2.2 Control Domain Building Blocks

The control domain contains the following blocks: planning, guidance, control, actuation, perception, storage, and communication.

4.2.2.1 Flight Planning

The flight planning refers to the determination of the desired and valid path of travel (the "trajectory") from the vehicle's current location to a designated target, as well as desired changes in velocity, rotation, and acceleration for following that path[11]. It consists of two sub-blocks: trajectory planning and trajectory validation (see Figure 7).

The *trajectory planner* provides a sequence of way-points based on a mission's objectives (e.g., mapping and surveillance), and constraints such as terrain, weather, air space and fuel. This is often done manually by the pilot that directly specifies the way point sequence, although there exist numerous algorithms than can make the plan automatically. Plans can change during flight as re-planning can be necessary due to new information becomes available.

The *trajectory validation* is the process of communicating with the external services to validate the flight trajectory. The external services can process drone plans and checks for compliance with the applicable national regulations. Depending on the customized workflow, the systems can authorize the plans automatically to reduce response time and automating most of the workload. If the drone operation cannot be authorized automatically, the drone operator can request authorization from a supervisor such as Air Traffic Control (ATC), local government, or law enforcement.

4.2.2.2 Flight guidance

The flight guidance refers to the process of monitoring and controlling the movement of a vehicle from one place to another[12]. It consists of four main sub-blocks: trajectory execution, detect and avoid, geofencing, and command executor (see Figure 7).

The *trajectory execution* specifies how the UAV should travel between the way-points. This is an automatic function that considers the specifications of the UAV in order to generate a path consisting of segments of lines and circles that are compatible with the UAV's turn and climb capabilities.

Detect and Avoid (DAA) block allows drones to integrate safely into civilian airspace, avoiding collisions with other aircraft, buildings, power lines, birds and other obstacles. This block observes the environment surrounding the drone, decides whether a collision is imminent, and generates a new flight path in order to avoid collision. It contains two main blocks: obstacle detection and obstacle avoidance. First, the *obstacle detection* consists of the sensing and the situation awareness (SA). The sensing is the reading, conditioning and filtering of the sensors to receive distances, position changes, orientations, and pitch of obstacle. These data are then fed to the situation awareness, whose task is to process the multiple information from the sensing to determine the minimum distance available to the front, right, rear and left of an obstacle as well as its longitudinal and velocity. Such information is used to determine whether a collision may arise or not. Second, the *collision avoidance* is used when an obstacle is detected on the way to the target position in order to carry out the mission, this block finds a trajectory

11 Draper, C. S.; Wrigley, W.; Hoag, G.; Battin, R. H.; Miller, E.; Koso, A.; Hopkins, A. L.; Vander Velde, W. E. (June 1965). Apollo Guidance and Navigation (Report). Massachusetts: Massachusetts Institute of Technology, Instrumentation Laboratory
12 Rell Pros-Wellenhof, Bernhard (2007). Navigation: Principles of Positioning and Guidance. Springer. pp. 5–6

that avoids the detected obstacle. The block should also consider far obstacles by flowing/finding a trajectory where the least number of obstacles exist.

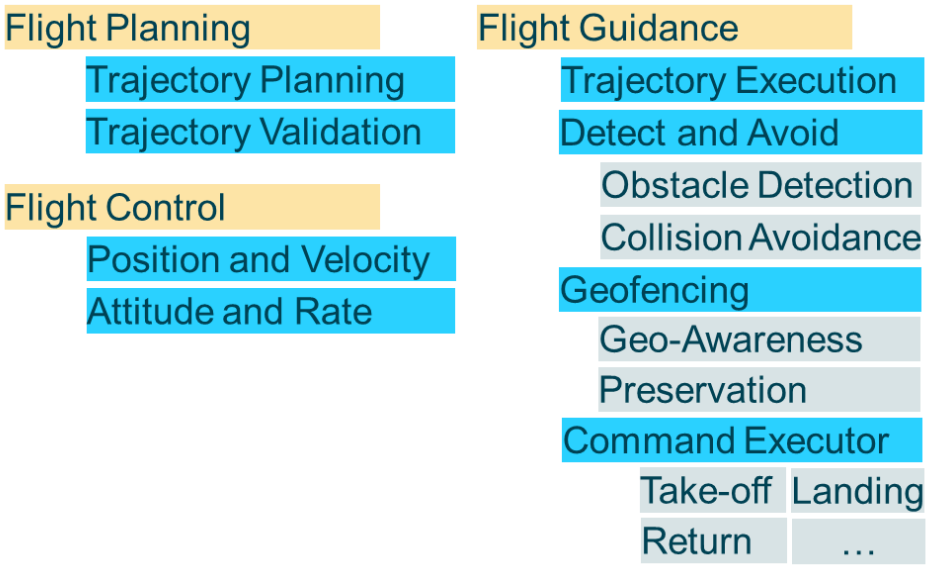

Figure 7: Flight planning, guidance, and control

The *geofencing* block is responsible for keeping the drone flying without violating the geo-fence area defined by the regulation authorities. It contains two functions: geo-awareness and geo-fence preservation. First, the *geo-awareness* function is to have a safety distance to the fence threshold. If the distance to a given fence gets closer than this threshold, a warning is sent to the operator/pilot. Second, the *geo-fence preservation* is responsible for flying the UAV away when the fence is (will be) violated while considering the running flight plan.

The *command executor* is responsible for mode switching and executing the different mission commands. The commands can be: (a) navigation commands that are used to control the movement of the vehicle, including take-off, moving to and around waypoints, changing altitude, and landing, (b) do commands that are for auxiliary functions and do not affect the vehicle's position (for example, setting the camera trigger distance, or setting a servo value), and (c) condition commands that are used to delay do commands until some condition is met, for example the UAV reaches a certain altitude or distance from the waypoint.

4.2.2.3 Flight Control

Control refers to the manipulation of the forces, by way of steering controls, thrusters, etc., needed to execute commands while maintaining vehicle stability[13].

The *flight control* specifies changes in the UAV's thrust, yaw, pitch, and roll to follow the specified path (typically represented as a sequence of line and circle segments) through the position and altitude controls. It also tracks errors if the UAV is deviating from the desired path due to path updates, winds and other disturbances using different controllers (see Figure 7).

The *position and velocity* controller takes the inputs from the trajectory generation and compute the desired attitude and desired thrust.

The *attitude and rate* controller takes the desired attitude (or body rates), and compute the desired command to keep the drone stable, most of time around body axis (roll, pitch, yaw).

13 Stengel, R. F. Toward Intelligent Flight Control, IEEE Trans. Systems, Man, and Cybernetics, Vol. 23, No. 6, November–December 1993, pp. 1699–1717.

4.2.2.4 Actuation

The actuation is the function that writes data and control signals to an actuator to execute the flight commands. It contains two elements: mixer and actuators drivers (see Figure 8).

The **mixer** takes commands (e.g., turn right) and translates them into individual motor commands, while ensuring that some limits are not exceeded. This translation is specific for a vehicle type and depends on various factors, such as the motor arrangements with respect to the center of gravity, or the vehicle's rotational inertia.

The *drivers* are a set of software modules that have the actual interface with the physical actuators to perform the specific actions.

4.2.2.5 Perception

A data abstraction function may be needed for cleaning, filtering, de-duplicating, transforming, normalizing, ignoring, augmenting, mapping and possibly persisting data before the data are ready for analysis or destroyed. Perception deals with understanding the states, conditions and behaviours of the systems under control, and those of peer systems by interpreting and correlating data gathered from sensors and peer systems (i.e., creating a system model). The complexity and sophistication of modelling of the system under control varies greatly. It may range from straightforward models (such as a simple interpretation of a time series of the wind measurements), to moderately complex (a prebuilt physical model of a drone), to very complex and elastic (models built with artificial intelligence possessing learning and cognitive capabilities). These modelling capabilities (sometime referred to as edge analytics) are generally required to be evaluated locally in control systems for real-time applications. Edge analytics are also needed in cases where it is not economical or practical to send a large amount of raw sensor data to remote systems to be analysed even without a real- time requirement.

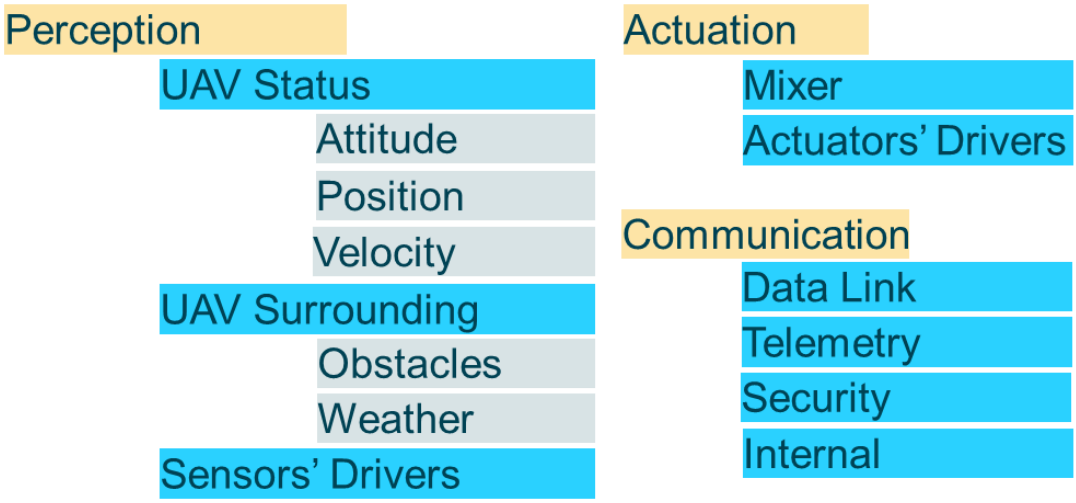

Figure 8: Flight perception, actuation, and communication

The perception block includes the necessary functions for providing the information required by the other architecture blocks. It includes: UAV status (i.e., attitude, heading, positioning, and velocity), UAV surrounding, weather, and sensor's drivers (see Figure 8).

An *attitude* block provides attitude and heading information for the drone including roll, pitch and yaw. These measurements can then be used to derive an estimate of the drone attitude.

The *drone position* can be estimated using Global Navigation Satellite System (GNSS-based) or, when GNSS are denied using methods such as Simultaneous Localization and Mapping (SLAM). First, the *GNSS-based* block uses one of the existing positioning systems such as the Global Positioning System

(GPS), Global Navigation Satellite System (GLONASS), BeiDou Navigation Satellite System (BDS) and Galileo. Second, the *SLAM* block is the computational process of constructing or updating a map of an unknown environment, while simultaneously keeping track of a drone location within it when GNSS-based devices are not available (e.g., indoor situations).

The *velocity estimator* provides the drone speed. The speed can be directly measured using a speed sensor, or estimated using other sensors such as the drone's camera.

The *UAV surrounding* provides information about the drone environment including static and moving obstacles, and weather information such as wind speed.

The *drivers* are software modules that have a direct interface with the physical sensors to acquire the needed data.

4.2.2.6 Communication

Communication connects sensors, actuators, controllers, gateways and other edge systems. The communication mechanisms take different forms, such as a bus (local to an underlying system platform or remote), or networked architecture for command and control and service data exchange with external systems. Also, Quality of Service (QoS) characteristics such as latency, bandwidth, jitter, reliability, redundancy and resilience should be taken into account.

Within the communication function, a connectivity abstraction function may be used to encapsulate the specifics of the underlying communication technologies, using one or more common APIs to expose a set of connectivity services. These services may offer additional connectivity features that are not otherwise available directly from the underlying communication technologies like reliable delivery, auto-discovery and auto-reconfiguration of network topologies upon failures. The communication can have four forms (see Figure 8):

A *data link* is the means of connecting the UAV with the external infrastructure for the purpose of transmitting and receiving digital information (i.e., service data communication).

Telemetry is a digital two-way data stream dedicated to ground control, which can both send data about the flight down to a ground control station, and send command up to the autopilot.

Internal communication is used for exchanging data between the different blocks of the UAV or the ground control station. Such communication can be performed using publish-subscribe model or through directed links between the blocks.

Communication security is the prevention of unauthorized interceptors from accessing communications medium, while still exchanging data between the UAV system blocks.

4.2.3 Control Domain Patterns

Within the control domain, there are a number of patterns of interaction between its blocks. In the following, we describe these patterns.

4.2.3.1 Flight Planning Pattern

The flight planning pattern is shown in Figure 9. In this pattern, the planning block uses the maps, geo-fence limitation, UAV power and the destination to find a trajectory. This trajectory is then validated by the validation block and sent for execution by the flight guidance.

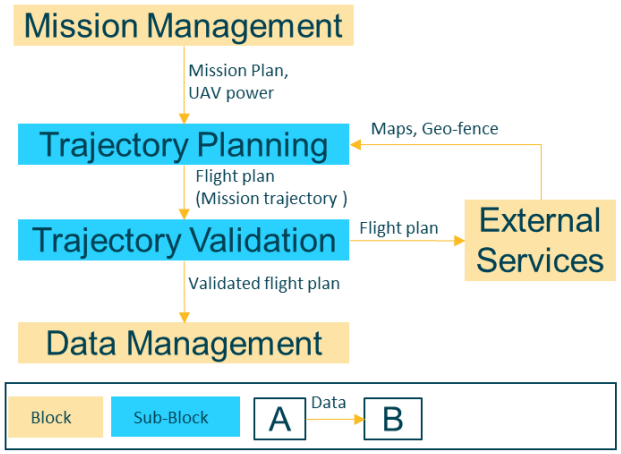

Figure 9: Flight planning pattern

4.2.3.2 *Flight Guidance Pattern*

The flight guidance consists of four main sub-blocks: trajectory execution, detect and avoid, geofencing, and commander (see Figure 10). The commander is used for executing the commands coming from the mission management and trajectory execution. The trajectory execution interacts with the flight guidance to get the trajectory details and then starts the waypoint navigation through the flight control. During the flight, detect and avoid block detects obstacles in the surroundings and executes an algorithm to find a strategy to avoid such obstacles through trajectory update or through direct manipulation of the drone using the actuation module in emergency situations (see Figure 11). In the same manner, the geo-fencing block uses its geo-awareness module to keep track of the UAV position and detects any geo-fence violation. In case of violation detected, the geo-fence preservation either updates the trajectory or take a direct control of the drone.

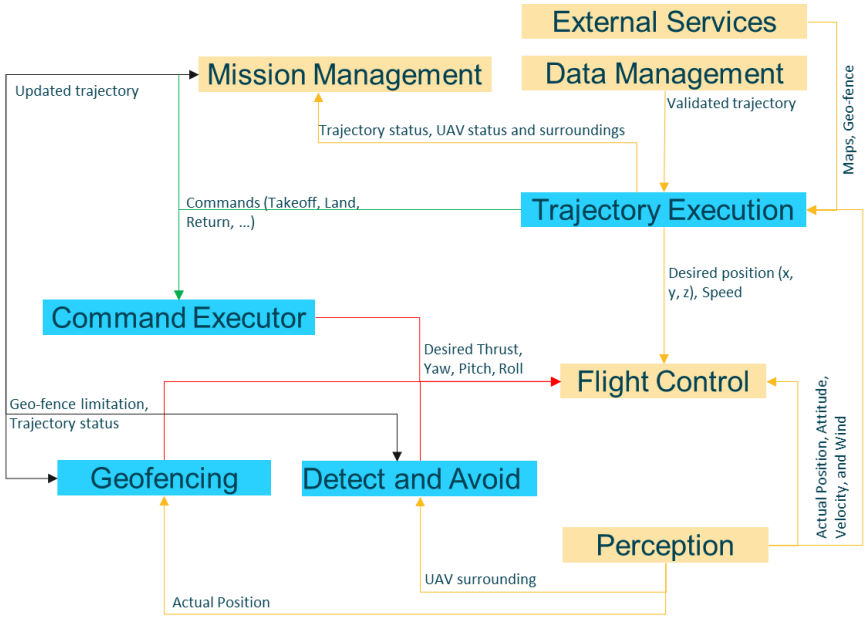

Figure 10: Flight guidance pattern

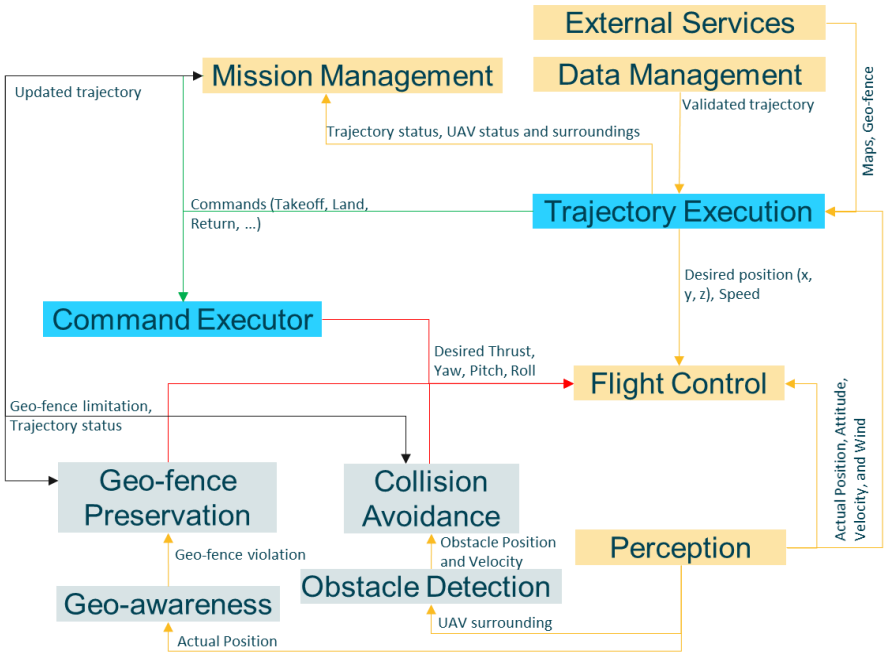

Figure 11: Detect and avoid and geofencing patterns

4.2.3.3 *Flight Control Pattern*

The flight control specifies changes in the UAV thrust, yaw, pitch, and roll to follow the specified path (typically represented as a sequence of line and circle segments) through the position and altitude controls. It also tracks and correct errors if the UAV is deviating from the desired path due to path updates, wind, and other disturbances (see Figure 12).

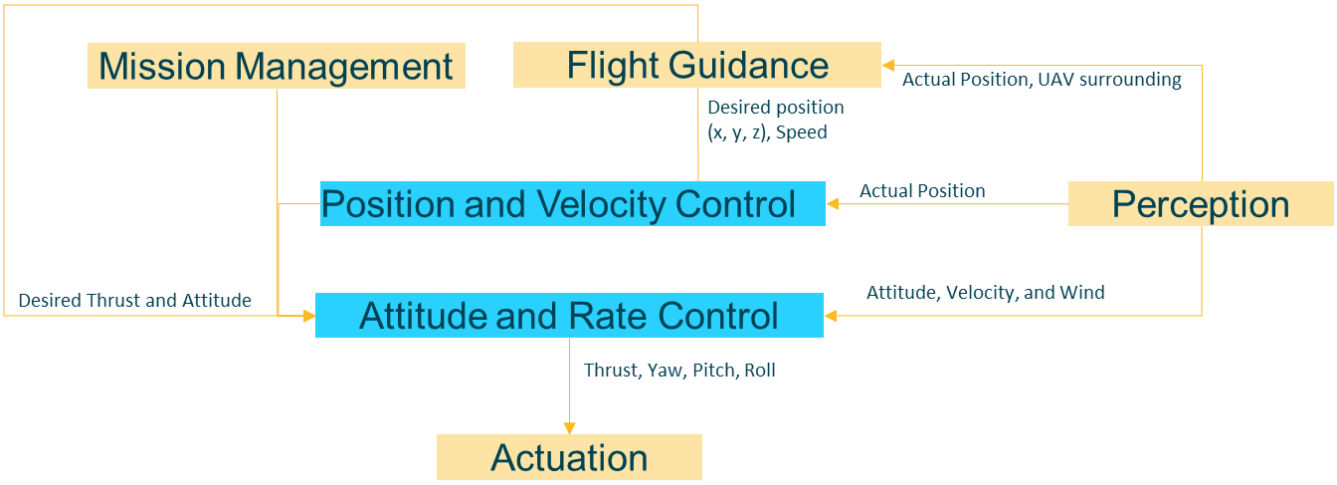

Figure 12: Flight control pattern

4.2.3.4 *Guidance, Navigation, and Control pattern*

The different blocks that enable a drone system to fly from a source to a destination and their interactions are shown in Figure 13. The blocks include: flight guidance, flight control, perception (including the UAV status), and actuation.

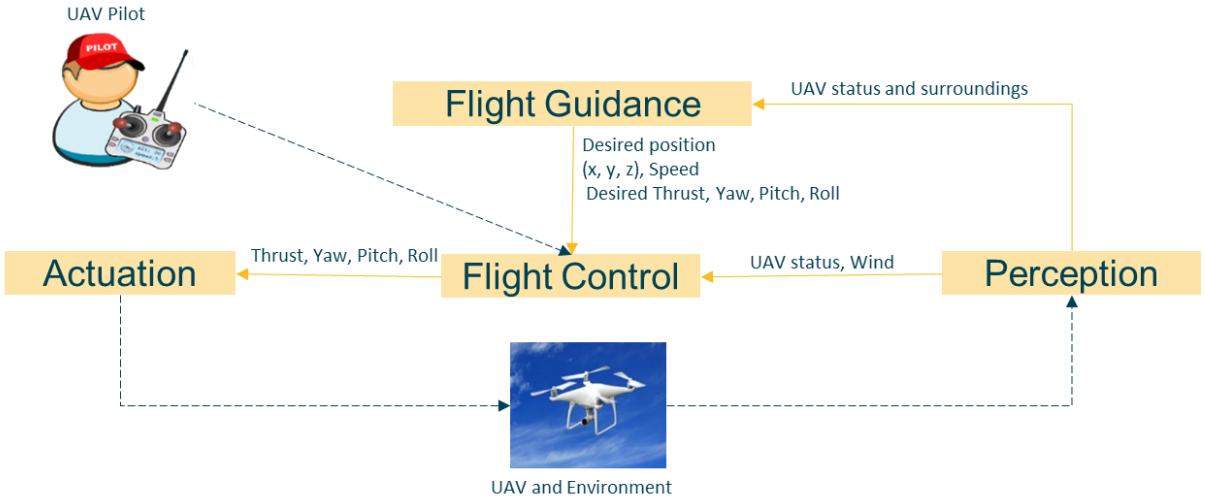

Figure 13: Guidance, navigation, and control pattern

The flight planning is responsible for generating a valid trajectory (i.e., a flight plan) based on an intended mission, maps, geo-fence limitation and UAV power. This valid plan is then fed to the flight guidance to be executed while taking into account the geo-fence limitation and detecting and avoiding obstacles. The desired waypoints generated by the guidance are translated to commands through the flight control and executed with the help of the actuation block. During the flight, the UAV status block inside the perception gives the flight guidance and control the necessary information to execute their intended functions correctly.

4.3 Operations/Management Domain

The operations domain is a functional domain for managing the control domain. It includes a collection of functions responsible for managing, monitoring and optimizing the systems in the control domain. This opens opportunities for added business and customer value as set out by higher-level business-oriented domains.

4.3.1 Overview

Figure 14 shows how operations in a drone system can be supported through a suite of interdependent operations support functions. These operations include health and payload management.

Health management is the capability to preserve the UAV system's ability to function as intended. This feature makes the system able to contain, prevent, detect, diagnose, respond to, and recover from conditions that may interfere with nominal system operations to ensure the UAV flight safety.

Payload management is a capability for managing the drone payload. It is responsible for sending commands to the payload either for data collection or to perform an action.

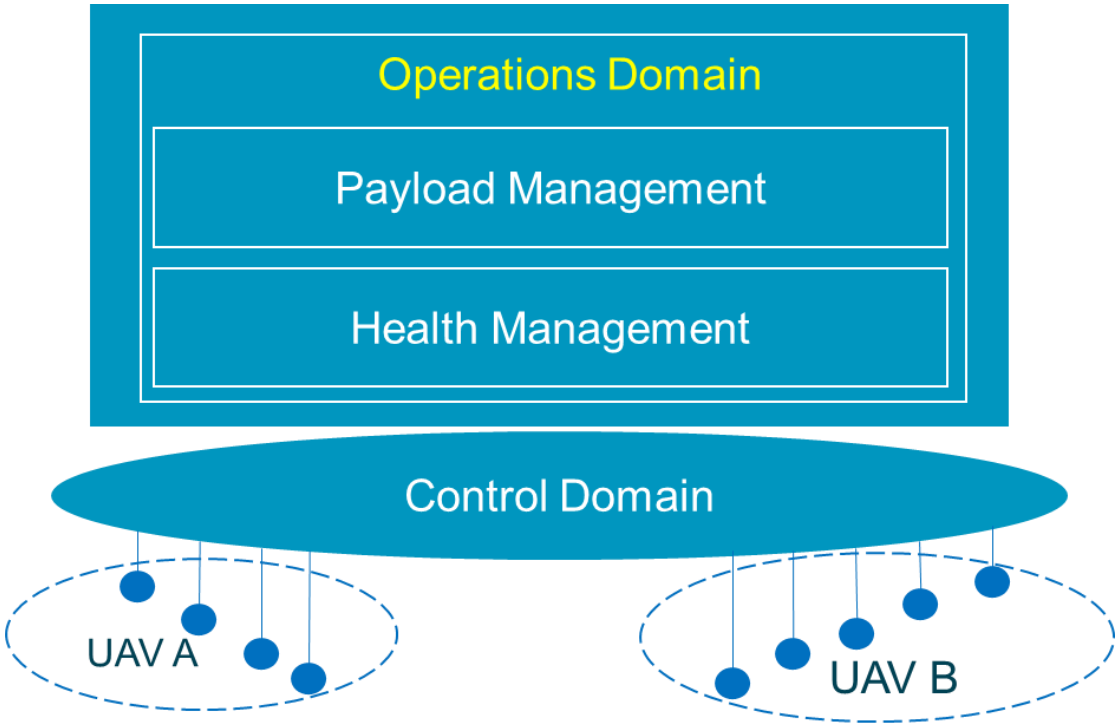

Figure 14: Operations domain

4.3.2 Operation Domain Blocks

The operation domain contains two main blocks: health and payload management. We describe them in this section.

4.3.2.1 Health Management

The health management block is responsible for analysing flight data and for responding to problems in the power supply or faults that can occur in the drone engine, sensors, actuators, etc. It consists of three main blocks: power management, fault management, and diagnostics.

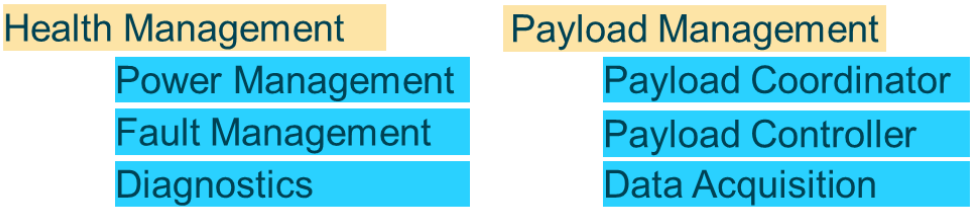

Figure 15: Health and payload management

Power management continuously monitors important power supply parameters, while dealing with the varying power demands of the many aspects of the UAV's operation. It also optimizes the usage of the power source. For example, when a battery is the power source, the power management monitors battery voltage, current, temperature, state of charge, state of health and other parameters, and may calculate additional information based on these and takes actions to overcome battery problems and ensure safe mission execution. In addition to managing the battery usage, the battery management system can also protect the battery during charging, safeguarding against conditions such as over-current or over-voltage.

Fault management is in charge of detecting and accommodating failures of the UAV. Faults in UAV are commonly represented as either actuator, plant, or sensor failures. The fault management technique

depends on the type of the failure experienced. Plant and actuator failures in the UAVs generally result from mechanical or structural failures, and often require adaptation of the control system. Sensor failures can be a major source of error in the UAVs, particularly when a UAV uses less reliable sensors due to design restrictions. To detect and manage these three types of failures, a combination of techniques would be required.

Diagnostics enables the detection of problems' occurrences. It is responsible for real-time monitoring of key performance indicators of a drone, collecting and processing health data with intelligence, so that it can diagnose the real cause of a problem, and then alerting on abnormal conditions and deviations. It assists operation and maintenance personnel to reduce the response time between detecting and addressing a problem.

4.3.2.2 Payload Management

The payload management block is responsible for managing the payload added to the UAV to execute certain mission. It consists of three main blocks: payload coordinator, payload controller, and data acquisition.

Payload coordinator takes the mission plan as an input and keeps track of the UAV position. When a location is reached where data acquisition (e.g., capture an image) or an action needs to be taken by the drone (e.g., drop a package), specific commands are issued to the payload controller or the data acquisition block to perform the drone mission.

Payload controller executes the commands coming from the payload coordinator. Such actions are either performed by sensors (e.g., change a camera angle), or through actuators (e.g., pick up a package).

Data acquisition is the process of sampling signals that measure real world physical conditions relevant to the drone mission (e.g., status of a collapsed building), and converting the resulting samples into digital values (e.g., constructing 3D image of the building) that can be later on manipulated by a computer.

4.3.3 Operation Domain Patterns

Two patterns exist in the operation domain. These patterns are for managing the health and the power of the UAV.

4.3.3.1 Health Management Pattern

The health management block receives the UAV status from the UAV status block. The status is then analysed to detect low power supply or failures within the system as shown in Figure 16. Such indications are then taken by the flight control to switch from normal to emergency mode. In this mode, the power and fault management blocks issue some commands to the UAV through the flight control to keep the UAV mission safe. Such commands include safe landing in case of failure, return home when power level goes under certain percentage, etc.

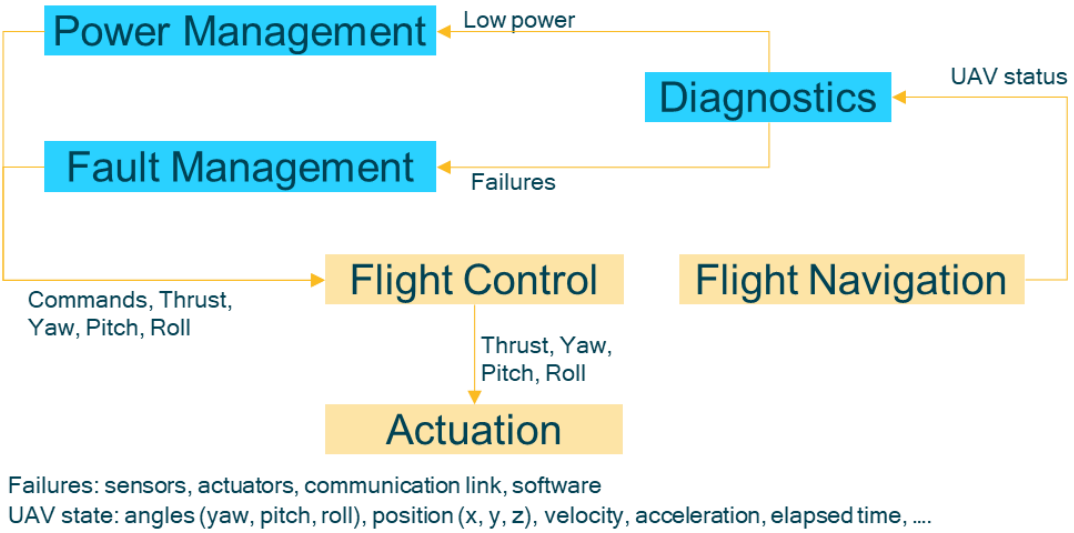

Figure 16: Health management pattern

4.3.3.2 Payload Management Pattern

To perform a drone mission, a number of payloads are added to the drone. Then, during the mission, the payload coordinator communicate with both the data acquisition and the payload controller through commands either to request data or control the payload as shown in Figure 17.

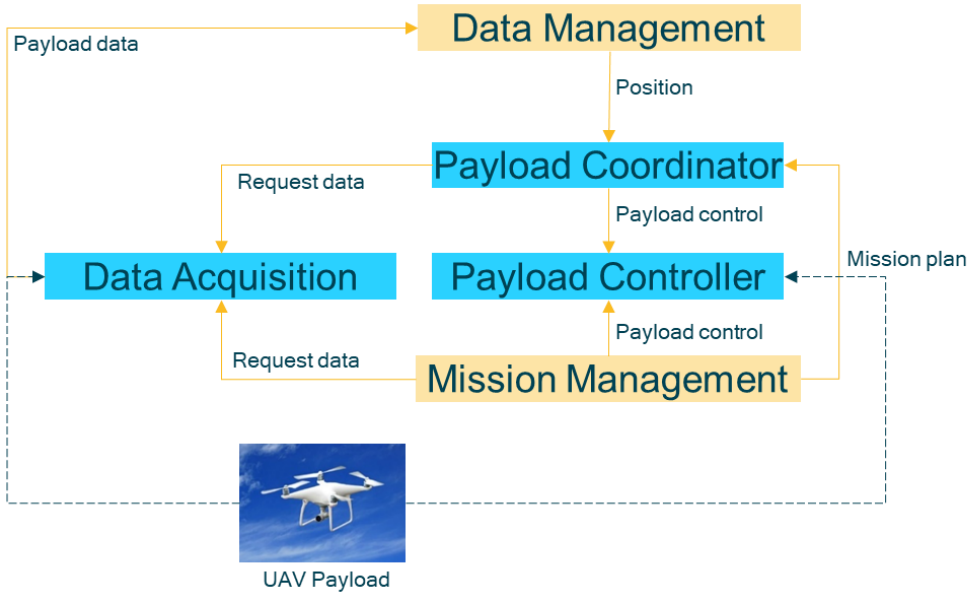

Figure 17: Payload management pattern

4.4 Usage and Business Domains

The usage and business domains are the mission-specific functions in the drone system. These functions are specified based on the intended drone mission. Figure 18 illustrates the functional decomposition of the usages (i.e., information and application), and the business domains.

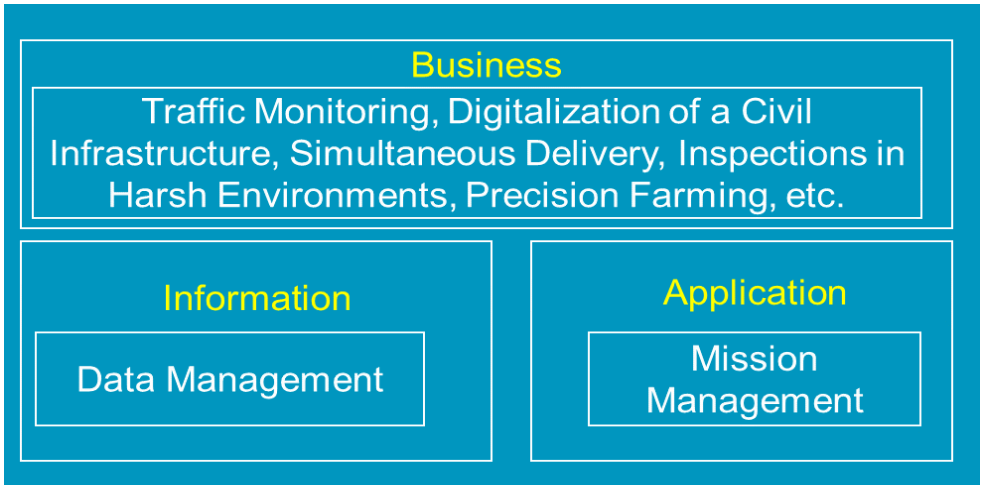

Figure 18: Business and usage domains

4.4.1 The information domain

The *information domain* is a functional domain for managing and processing data. The *data collection and analysis* functions in this domain are complementary to those implemented in the control domain. In the control domain, the functions participate directly in the immediate control of the physical systems, while in the information domain they are for aiding decision-making, optimization of system-wide operations, and improving the system models over the long term.

4.4.1.1 Data Management Block

The data management block consists of capabilities for data persistence and storage, data distribution, semantic transformation, etc. as shown in Figure 19. These functions can be used in an online streaming mode in which the data are processed as they are received to enable real-time analytics. They may be used in an offline batch mode (e.g., seismic sensor data collected and accumulated in an offshore oil platform that does not have high-bandwidth connectivity to the onshore datacenter). Data governance functions may be also included for data security, data access control and data rights management, as well as conventional data management functions related to data resilience (replication, snapshotting, restore, backup & recovery, and so on).

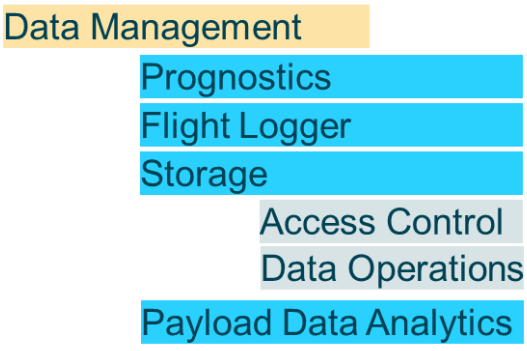

Figure 19: Flight and payload data management

The *Prognostics* block serves as a predictive analytics engine of the drone system. It relies on historical data of drone operation and performance, engineering and physics properties of the drone, and modelling information. The main goal is to identify potential issues before they occur and provide recommendations on their mitigation.

The *flight logger* keeps track of the UAV status, and records it for latter analysis by the prognostics block. By default, logging is automatically started when UAV is arming, and stopped when disarming. A new log file is created for each drone mission.

The *storage* block is responsible for storing all the needed information to execute a mission. Example information needed for executing a flight includes flight plans, geo-fence limitation, UAV model (e.g., UAV power supply). During mission execution, it also stores the generated information about the vehicle health during the flight (e.g., power level, engine temperature, etc.), and information about the drone environment. The storage block also has access control, which is a method of allowing access to drone's sensitive data only to those people who are allowed to access such data and to restrict access to unauthorized persons.

The *payload data analytics* encapsulates a set of functions for data modelling, analytics and other advanced data processing, such as rule engines. The analytic functions may be done in online/streaming or offline/batch modes. In the streaming mode, events and alerts may be generated and fed into functions in the application domains. In the batch mode, the outcome of analysis may be provided to the business domain for planning or saved as information for other applications.

4.4.1.2 Data Management Pattern

The data management is centered on the storage block, where the flight information coming from flight logger and the data coming from payload management are stored (see Figure 20). The flight data are analysed using the prognostic block to anticipate expected problems in the drone power supply, engines, sensors, actuators, etc. The payload data are analysed based on the mission objective to provide useful information to the business domain.

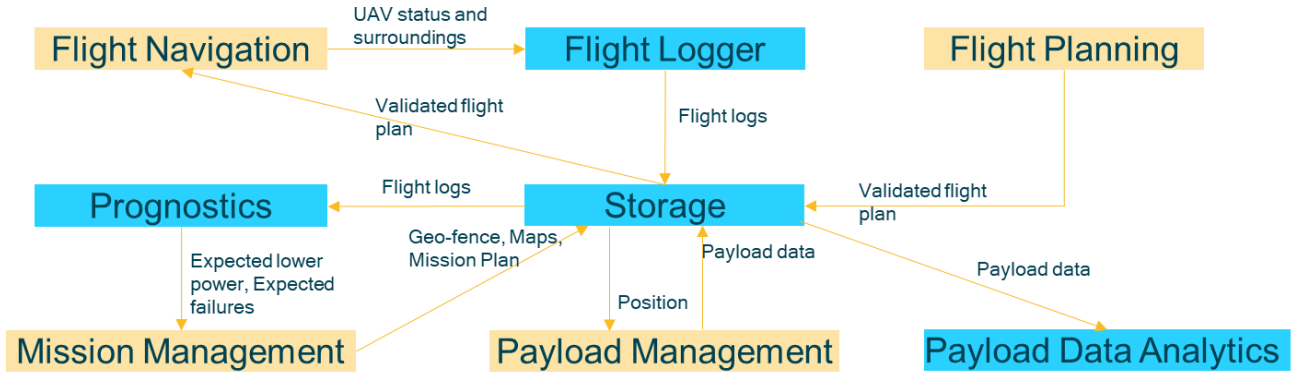

Figure 20: Data management pattern

4.4.2 The application domain

The application domain is a functional domain for implementing application logic (i.e., mission management). It represents a collection of functions implementing application logic that realizes specific business functionalities. Functions in this domain apply application logic, rules and models at a coarse-grained (high level) for optimization in a global scope. They do not maintain low-level continuing operations, as these are delegated to functions in the control domain that must maintain local rules and models in the event of connectivity loss. Requests to the control domain from the application domain are advisory, so as not to violate safety, security, or other operational constraints.

4.4.2.1 Mission Management Block

The mission management block comprises logics (rules, models, engines, activity flows, etc.) implementing specific functionality that is required for the use case under consideration. It is expected that there are great variations in these functions in both its contents and its constructs among the use cases. However, there are common functions between the different missions/use cases. These

functions are used for planning and monitoring the execution of a drone mission (i.e., mission planning and mission supervision).

Mission planning is the process of producing a detailed flight schedule which concerns the calculation of the route for UAV(s) and the work plan of the payloads (sensors and actuators). A correct mission plan should meet the mission constraints set for the mission objective. The UAV should complete the mission by traveling from where it is initially located and should follow the plan until it reaches the desired destination. The objective function set for the mission may include, for example, planning the mission along the shortest routes, which leads to reduced fuel consumption. This objective function gives the greatest guarantee that the mission can be completed when the UAVs are not threatened by other danger.

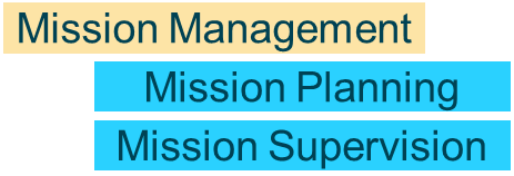

Figure 21: Mission management block

Mission supervision represents a set of functions that the UAS exposes through user interface to enable human interactions with the UAS. It allows UAV operators to communicate with and control a drone and its payloads, either by setting parameters for autonomous operation or by allowing direct control of the UAV, usually from the point of launch (mission start) until landing or the end of the mission.

4.4.2.2 Mission Management Pattern

The mission management is used for planning and executing a specific drone mission. To plan a mission, it receives maps and a geo-fence area from externals services and based on the mission objectives, a mission plan (route and payload work plan) is created. This plan is used by the flight plan to generate the flight trajectory and by the payload management to control the payload for achieving the mission objectives. The mission supervision (a) monitors the mission through the information coming from flight guidance, (b) controls the flight and payload manually, and (c) is notified by expected failures and power problems in advance using the prognostic block (see Figure 22).

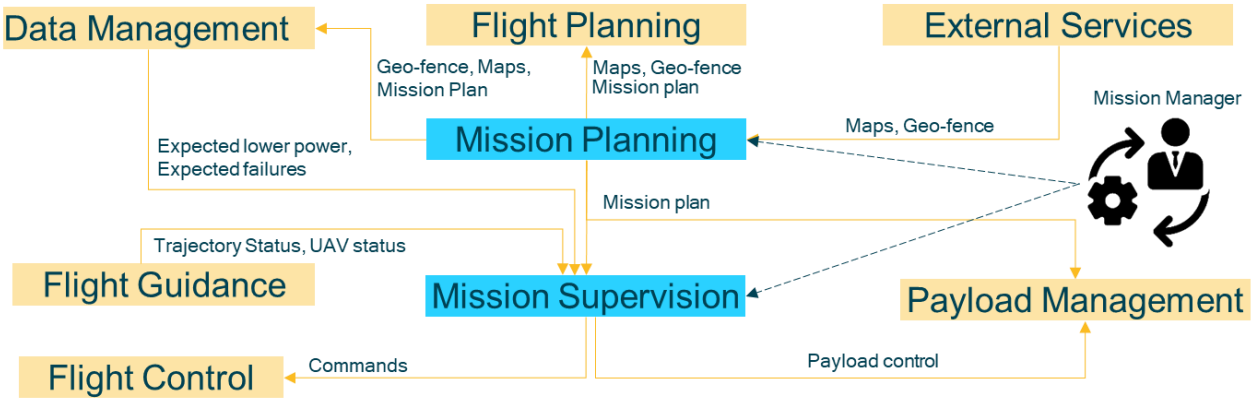

Figure 22: Mission management pattern

4.4.3 The business domain

The business domain implements business functional logic. It represents business functions supporting business processes and procedural activities. Examples of these business functions include traffic monitoring, digitalization of a civil infrastructure, simultaneous delivery, precision farming, inspections in harsh environments, etc.

5 The Reference Architecture

In this section, we present the overall reference architecture, and how its blocks and patterns can be clustered and instantiated based on the drone autonomy level.

5.1 Overall UAS Blocks

A summary of all the blocks that have been described in Section 0 is shown in Figure 23. In the following sub-section, we describe how these blocks are distributed in Remotely Piloted Aircraft (RPA) and semi-autonomous drone.

Figure 23: Overall blocks of a drone system

5.2 Remotely Piloted Aircraft System (RPAS)

ICAO explains the term Remotely Piloted Aircraft (RPA)[14] as "An unmanned aircraft which is piloted from a remote pilot station expected to be integrated into the air traffic management system equally as manned aircraft and, real-time piloting control is provided by a licensed remote pilot." ICAO also defined the term RPA system as "A set of configurable elements consisting of a remotely-piloted aircraft, its associated remote pilot station(s), the required command and control links and any other system elements as may be required, at any point during flight operation."

Following the definition of RPAS and the building blocks identified in Section 0, the distribution of the blocks to the drone platform and the ground station with a (licensed) pilot are shown in Figure 24. Most of the building blocks/functions are performed from the ground, while only the automatic control is done through the drone platform following the commands received from the ground.

14 https://www.icao.int/safety/UA/Documents/ICAO%20RPAS%20CONOPS.pdf

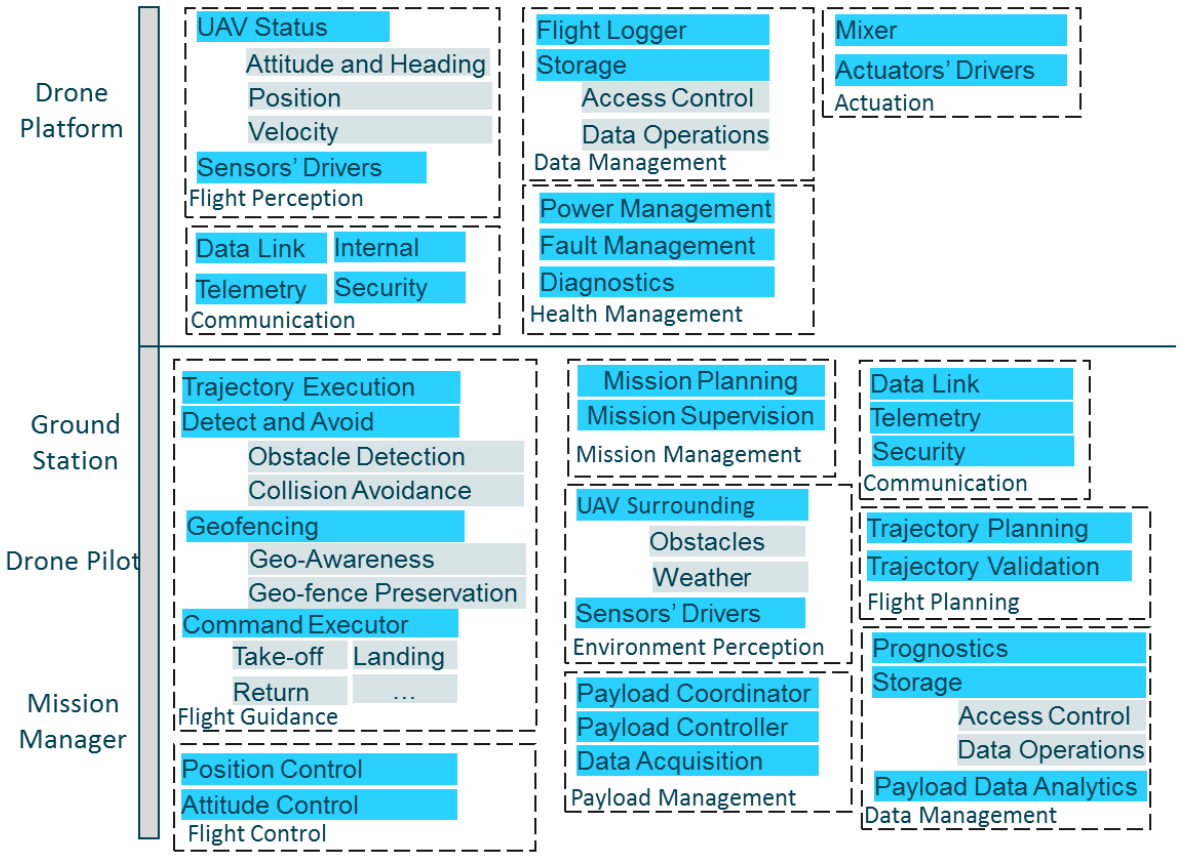

Figure 24: Distribution of the building blocks for RPAS drone

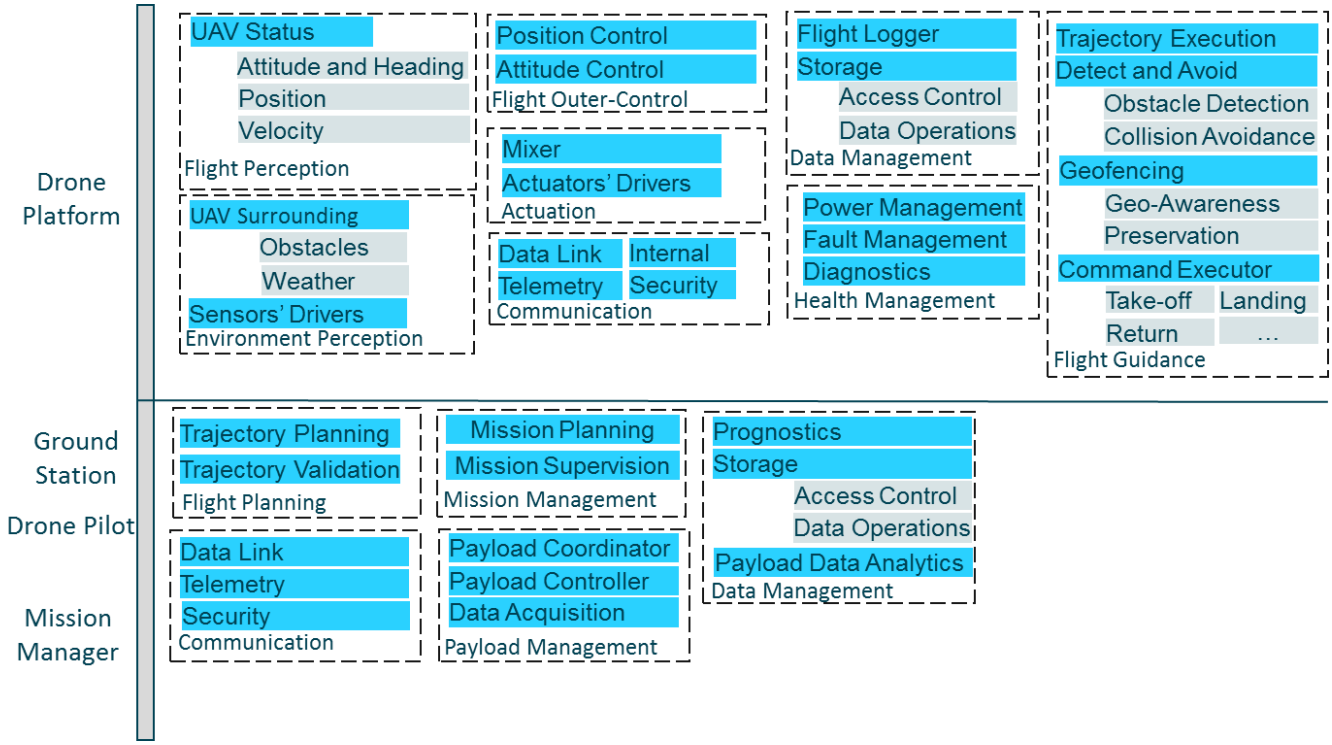

Figure 25: Distribution of the building blocks for semi-autonomous drone

5.3 Semi-Autonomous Drone (BVLOS)

A (semi-)autonomous drone is a drone that can operate with little human intervention[15]. In other words, the drone can take off, carry out missions, and land in an autonomous manner. In the case of such type of drones, a management software coordinates missions and pilots the aircraft instead of a human (i.e., the drone is piloted by software instead of a human). To achieve such drone autonomy, most of the UAS functions need to be automated and put on the drone platform to reduce the human interactions as shown in Figure 25.

5.4 Architecture Topology

The building blocks can exist either in the drone platform or in the ground control based on the drone autonomy level as described in the previous sections. Therefore, the architecture topology can be defined through clustering the building blocks based on their dependences (coupling). The main clusters identified are: flight control, flight management, flight navigation, mission management, and communication as shown in Figure 26.

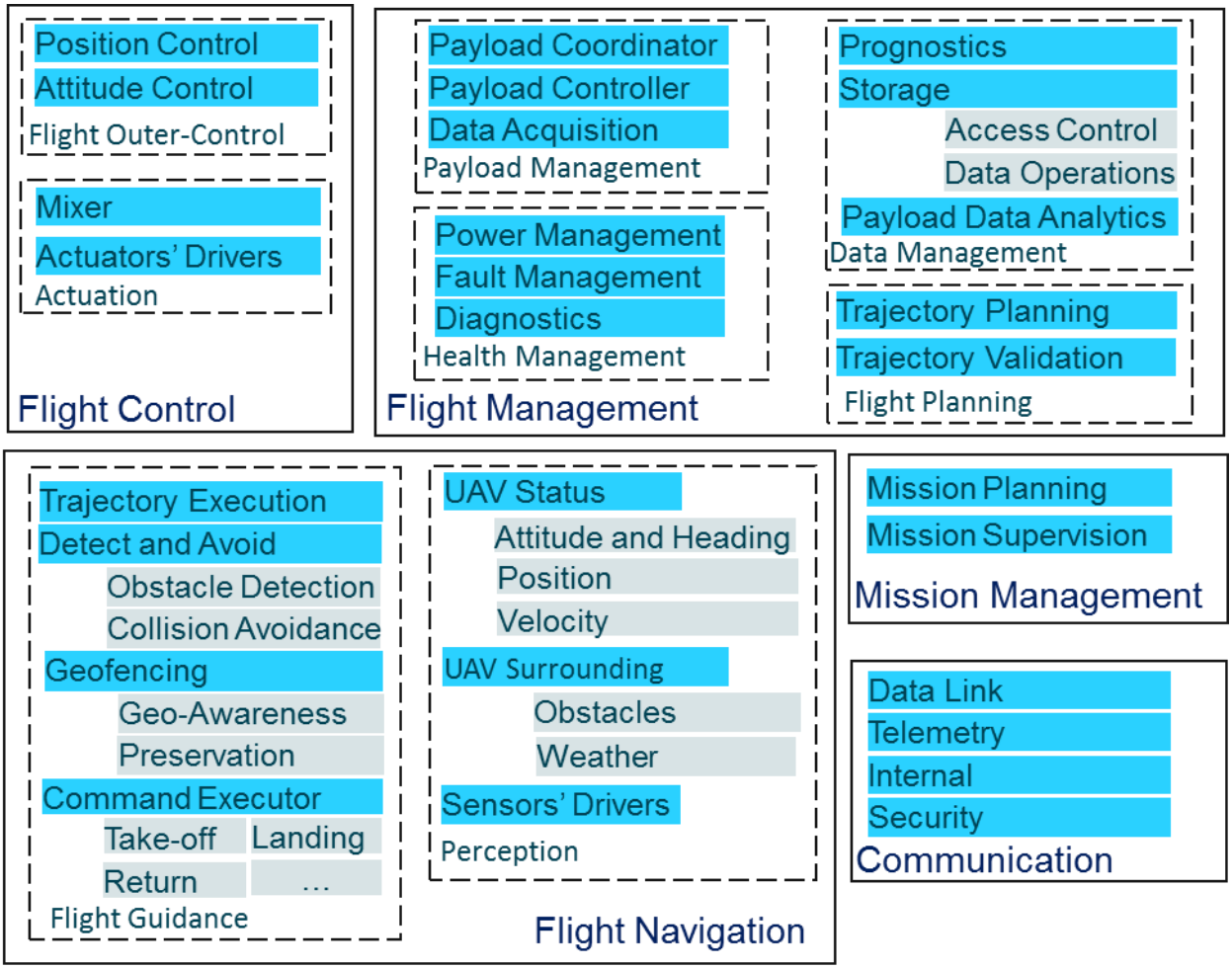

Figure 26: Clusters of the reference architecture building blocks

15 https://percepto.co/what-are-the-differences-between-uav-uas-and-autonomous-drones/

5.5 Overall Architecture

The overall reference architecture of a drone system is shown in Figure 27. This architecture shows the interactions between the different blocks of the UAS. It is divided into four main groups: flight navigation, flight control, flight management, and mission management. First, the flight navigation includes the drone perception to gather information needed to navigate the drone from one location to another, while avoiding obstacles and preserving the geo-fence using the flight guidance. Second, the flight control executes the guidance commands to fly the drone from one place to another through the drone actuation. It also executes commands coming from the pilot directly.

Third, the flight management contains functions for planning the flight trajectory and managing the UAV payload, data, and health. Fourth, the mission management has the mission planning and supervision functionality that are managed by the mission manager. Finally, there are external services that provide information for mission and flight planners to plan a valid mission and trajectory.

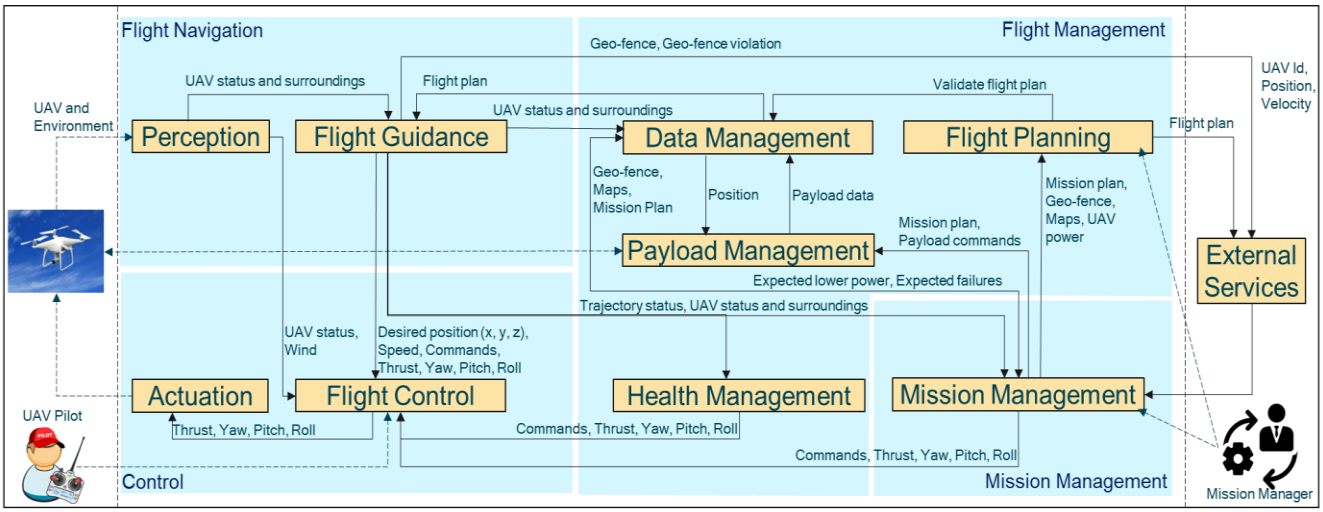

Figure 27: Flat view of the overall drone system architecture

6 Architecture Enabling Technologies

The reference architecture described in Section 5 includes the different building blocks of the UAS and their interactions. In addition to the building blocks, there are some cross-cutting concerns (technologies) that are needed to support the architecture. In the following, we give the specification of such technologies.

6.1 Hardware

To support the reference architecture a number of hardware components are being developed in **COMP4DRONES** project. In the following, we describe some of them.

6.1.1 Onboard Programmable and Reconfigurable Compute Platform Design Methodology - UNIMORE and UNISS

Some of the components described in the Reference Architecture (Figure 27) could be very demanding in terms of computational capability and memory footprint (e.g., Perception). Also, some other parts could require timing predictability (e.g., Actuation, and Flight Control) or flexible interoperability and portability (e.g., Payload Management, and Data Management).

Modern heterogeneous FPGA-based System-on-Chips (SoC) combine the benefits of both hardware and software computing infrastructures to tackle all these objectives/needs. They are suitable to serve on-board computationally intensive workloads while maintaining operational constraints related to the power envelope, the form-factor, and the interoperability and connectivity within standard drone software stacks. However, the more heterogeneous components are integrated into any computing platform, the more complex the integration process becomes (see D3.1 - Section 4.3.1).

The Onboard Programmable and Reconfigurable Compute Platform Design Methodology provided by UNISS and UNIMORE is intended to solve the integration with the legacy software components, the programming interfaces and the management of many heterogeneous components. This methodology combines 1) an Open-Source FPGA overlay that enables plug-and-play deployment of Hardware Processing Elements (HWPE) in typical drone workloads [UNIMORE], and 2) a methodology (Multi-Dataflow Composer - MDC) to design Coarse-Grained Reconfigurable Co-processing Units to be used when defining and integrating those HWPEs into the overlay compute clusters [UNISS].

The proposed technology and related design methodology can be seen as an enabling technology for the fast and effective deployment of mission-specific companion computers, targeting ease of deployment of FPGA accelerators and programmability of the whole platform. This methodology is orthogonal concerning the Drone Reference Architecture, and it can potentially be used to target a large set of components such as Perception, Flight Guidance, and Data Management. Specifically, WP4 includes some examples of Application-Specific Accelerators, controllable via OpenMP API, and can perform pre-defined computing tasks.

6.1.2 Efficient Digital Implementation of Controller on FPGAs – UNIVAQ

In the context of **C4D** project, a component provided by UNIVAQ aims to provide an efficient methodology for the digital implementation of controller on FPGAs. The proposed methodology for the efficient implementation of controllers on FPGA is focused on retiming and pipelining. The former is a transformation technique used to change the locations of the delay elements in a circuit without affecting the input/output characteristics of the circuit. Pipelining is a special case of retiming used to reduce the critical path, introducing pipelining latches along the data path. Shortening the critical paths, one can increase the clock speed or the sample speed, or one can reduce the power consumption at the same speed.

The pipelined implementation technique, here proposed in the context of UAVs control, allows lower execution times, and hence smaller sampling times, than its naive implementation. Moreover, the power

consumption of the pipelined control algorithm is lower. These two aspects constitute the main benefits of using a pipelining technique for the implementation of UAV control algorithm on an FPGA. In other words, the main aim behind this component is to provide an implementation guideline for UAVs control algorithms showing that, when technological solutions such as FPGAs are used, the pipelining methodology can be successfully applied to obtain lower sampling periods, thereby allowing the implementation of more sophisticated controllers for UAVs. The efficiency of the proposed implementation methodology is shown by developing on FPGA a robust sampled—data controller for the autonomous navigation of drone which has been designed in the context of WP4 for **C4D** project.

Finally, through experimental results, it can be shown that the pipelining methodology also allows taking into account the energetic aspects of the controller implementations, and not only the controller performance. This is another very important aspect for onboard systems as in UAVs. Thus, considering the reference architecture defined in **C4D** project, the provided technique aims to support the following building blocks: Trajectory Execution; Detect and Avoid; Command Executor; Trajectory Planning; Trajectory Validation; Position Control; Attitude Control; Stabilization; Power Management; Fault Management; Payload Controller; Data Acquisition; Payload Data Analytics.

6.1.3 Modular SoC-based Embedded Reference Architecture – EDI

The currently utilized embedded flight controller platforms are based on the sequential processing paradigm, which is a limiting factor for the drone's onboard computational capacity. A potential solution is to utilize the modern heterogeneous systems that incorporate different computational paradigms (Microprocessor Unit (MPU), FPGA, and Graphics Processing Unit (GPU)) as they promise the advantages of better computational capabilities and power efficiency while not introducing additional system complexity. We utilize our experience in heterogeneous SoC systems to develop a SoC-based embedded reference architecture, which we anticipate to be the core of future autonomous flight controllers.

To manage the additional complexity of the heterogeneous SoCs, we adopt a blackboard pattern from software development in combination with shared memory between the software and hardware parts of the system (see Figure 28). The reference platform is running Linux to enable the reuse of the available software stack. The idea is to abstract the FPGA accelerators efficiently to manage the complexity of interfacing hardware using shared libraries in cooperation with Linux modules. The software components, which may or may not utilize the hardware accelerators, are managed using the system architecture frameworks: COMPAGE: https://gitlab.com/rihards.novickis/compage (for component management), and ICOM: https://gitlab.com/rihards.novickis/icom (for component inter-communication).

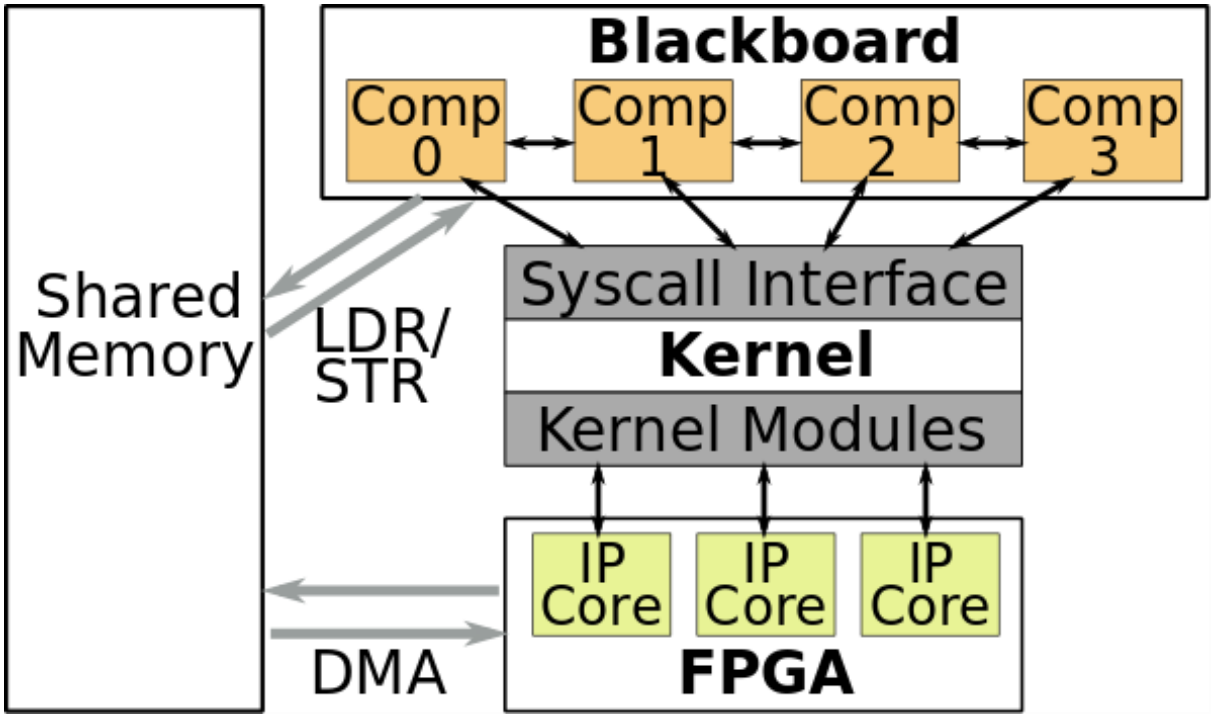

Figure 28: Blackboard pattern for managing the complexity of SoCs

This reference architecture is based on the Xilinx Ultrascale+ MPSoC System on Module (SoM), which is deployed on the drone using a tailored carrier board (see Figure 29). The reference architecture will include a methodology on the algorithm separation between concurrent (digital logic) and sequential processing paradigms. The approach is envisioned to enable a new class of algorithms for the onboard execution. For the project duration, we hope to validate the possibility of running SLAM based algorithms on the drone itself.

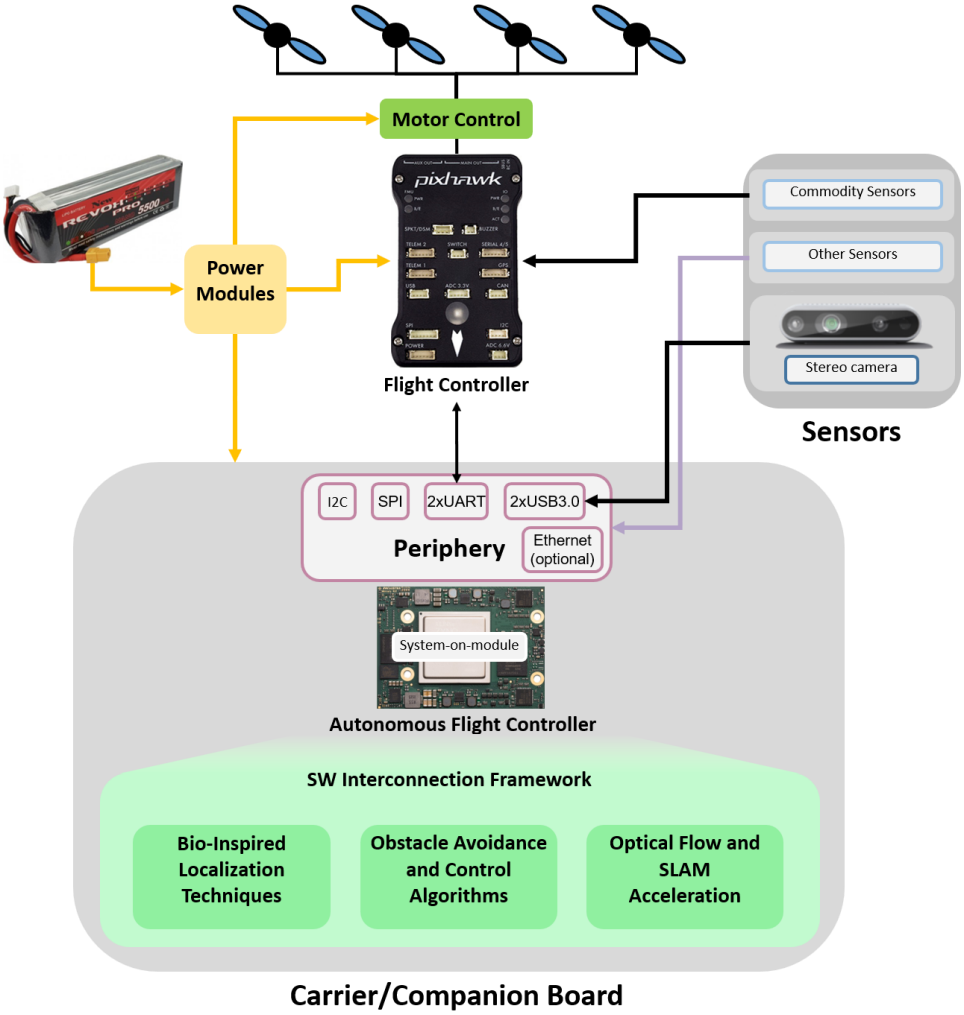

Figure 29 : Reference architecture is based on the Xilinx Ultrascale+ MPSoC System on Module (SoM)

6.1.4 Highly Embedded Customizable Platform for SLAM technique – MODIS

The Highly Embedded Customizable Platform aims to support MODIS's SLAM component (see Section 6.3.3) through the deployment of a range of sensors collecting odometry, geo-magnetic field measurements and assessing the distance from fixed points within the map.

In the context of the reference architecture, this component is part of the perception block. It relies on sensors' blocks and inputs to the block of the UAV status.

6.1.5 HW/SW System on Module for Object Detection and Positioning – IKERLAN

The IKERLAN MAMMUT compute module provides a hardware blueprint for testing different reference architecture blocks into a single system. Thanks to its heterogeneous-computing architecture, that includes general-purpose CPUs, Data Processing System (DPS)s, real-time capable CPUs, digital programmable logic and specialized cores for video encoding/decoding and communication means it allows the implementation of complex algorithms and the integration of high data-rate sensors. These characteristics bring to hardware level the needs of modular architecture, as different building blocks can be integrated within the developed component.

Considering all the building blocks, the best candidates for having support are obstacle detection and obstacle avoidance, as they require heavy computational capabilities and sensor integration, such as LiDARs, RADARs or RGB/IR/stereo cameras.

6.1.6 Hyperspectral (HSI) Cameras – IMEC-BG

Essentially, the HIS payload captures hyperspectral images, tags the images together with GPS coordinates, stores the images locally, processes the data to provide certain analytics, and sends the raw and classified/processed images to ground controller.

In the context of the reference architecture, IMEC's HSI payload is primarily part of the Payload Management block, but with relevant links to other blocks. This payload is composed of payload coordinator, controller and data acquisition (see Figure 30). The HSI payload relies on the following blocks in the reference architecture: Power for payload (Flight Planning), GPS information (Flight Guidance), and payload commands (Mission Management). In turn the HSI payload provides inputs to the following blocks in the reference architecture: hyperspectral images of the scene (Mission management), classified images or data analytics based on hyperspectral images (linked to External Services), and also perhaps as input to the UAV pilot to adjust the flight path (Flight Guidance).

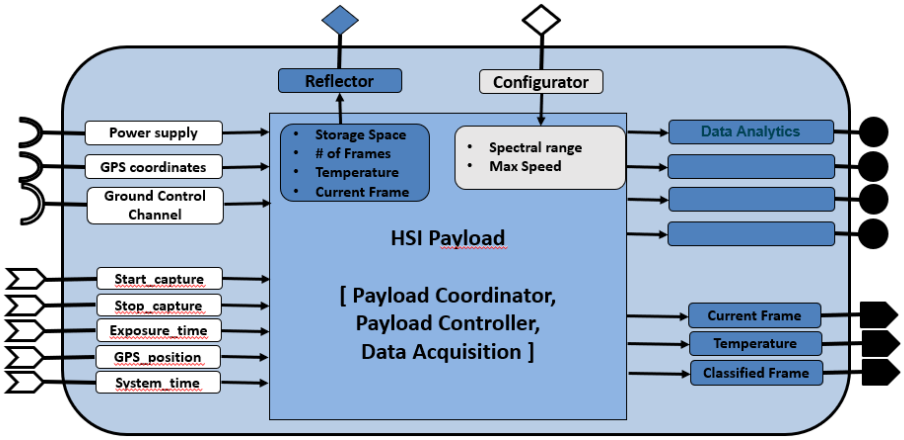

Figure 30: HSI payload management block

6.2 Basic Software

In this section, the different generic components that are part of the basic software of a drone are described.

6.2.1 Control Components that Implement Potential Barriers – ENSMA

The control component to implement potential barriers mainly serves three main objectives: a) geo-awareness, b) geo-fencing and c) obstacle avoidance. A drone must not enter in a restricted region (Geo-fence) while performing any task. The trajectory of the drone must be readjusted if it is going to violate the geo-fence as described by the authorities. Moreover, the potential barrier also helps the drone to avoid any collision with other drones or any objects in the path.

An Artificial Potential Function (APF) can be used to create the required potential barriers. APF produces a potential field in which the drone acts as a point. If the distance between the drone and the restricted region becomes less than a certain threshold, APF generates a repulsive force to fly away the drone and does not allow it to cross geo-fence boundary or collide with any other object.

In the context of reference architecture for **C4D**, control loop for potential barrier enhances the ability of the drone to perform safe autonomous operations in uncertain environment with unknown dynamic objects. The component supports navigation and mission planning building blocks by providing geo awareness, and avoiding any violation of the geo-fence.

6.2.2 Multi-agent Swarm Control- ENSMA

Many applications require multiple drones to work together in a cooperative manner in order to complete complex tasks quickly and efficiently. These applications sometimes require drones to move in a certain geometric shape and track a reference trajectory. This is known as formation tracking. Designing a control strategy to obtain the desired shape and tracking the reference trajectory while maintaining the shape is one of the fundamental problems in multi-agent systems. Cooperative behaviour of the drone can be controlled in a distributed manner where each drone shares information only with neighbours instead of all other drones in the network.

In **C4D** reference architecture, multi-agent swarm control component is aimed to provide the functionality of multi-drone consensus and formation tracking in distributed manner, while taking into account various constraints which are associated with the practical scenarios. The designed algorithm will only require position information of the neighbours and does not need their velocity or acceleration which means less communication resources (like bandwidth, etc.). The multi-agent swarm control component will also be able to deal with non-uniform and asynchronous sampling issue which is always present in the real application. The controller will also be able to avoid inter-agent collision while making the formation.

6.2.3 Complex System for Autonomous Drone Battery Management - UWB & SM

Droneport (DP) is a system for autonomous drone battery management. Droneport complies with reference architecture, dealing with drone power management, more precisely battery management. Regarding the reference architecture, the DP consists of a battery management unit for charging and storage, datalink to the drone, and telemetry data.

Power and storage management building blocks are implemented in Droneport, with datalink connection to the outside world. Users can use telemetry data via MAVLink protocol to monitor the state of the DP. The DP acts autonomously, it monitors the state of the batteries, controls charging and battery exchange process, provides necessary navigation information for drone landing and broadcast status of remaining batteries.

6.2.4 Smart and Predictive Energy Management System – UDANET

An energy management system is vital to optimize the energy life and the purpose of the system. It will continuously monitor important system parameters, while dealing with the varying power demands of the many aspects, the objectives of the mission and optimizing the usage of the energy.

In the context of the **C4D** project, this component aims to provide support for trajectory execution and power management building blocks. In fact, its main purpose is to identify excellent trajectories from an energetic point of view, knowing the initial and final point of the mission. The methodology has its roots in the resolution of optimal control problems with constraints, but it frees itself from the online resolution of them, trying to easily extract general rules from the study of the optimal choices that are made offline.

6.2.5 Generic Mission Controller – SCALIAN

SCALIAN has worked on developing a generic architecture, the EZ_Chains architecture, to allow fleets of UAVs to perform a variety of missions. The genericity allows the use of heterogeneous UAVs and to allocate them different missions. For instance, a fleet of 6 UAVs can perform logistic operations by dropping sensors over a huge area collaboratively, or a fleet of 4 UAVs can swipe scanned areas to build an orthophoto.

This generic architecture was dedicated to UAVs, and at its core the Aerial Mission Controller was designed to perform UAV missions. SCALIAN has then worked on making it suitable for other types of autonomous agents (i.e., Generic Mission Controller (GMC)). The EZ_Chains architecture shall be able to operate UAVs, rovers, and other types of autonomous system, for instance intelligent sensors and actuators (e.g. weather station, operation-area access control).

The Aerial Mission Controller has been demonstrated in simulations, software-in-the-loop tests, hardware-in-the-loop tests and real operations. The development of the GMC must also use this proof process to ensure a safe system. This constraint has been taken into account during the design and development phases.

The Generic Mission Controller has three main components: a task planner, a task monitor and a system watchdog. The task planner builds the sequences of tasks to carry out in order to complete the mission. Each agent has its instance and locally computes (distributed decision) its best actions based on the shared knowledge of the mission status (see Section 6.2.6 below). The task monitor takes this list of actions as input and controls that the agent executes it. Additionally, it ensures, through the whole mission, that all the agent systems are working as expected. It is connected to the system watchdog and triggers safety procedures when a system is detected as unhealthy. The watchdog is the component that measures the health of all the components. It relies on the components' reflector that has rules to know from this information if the component is heathy or not. When a component is detected to have issues, the watchdog has recipes on what to do, and depending on the component criticality it can try to solve the issue or it forwards the warning to the task monitor. In this last case, when a critical issue is detected, the task monitor will abort the mission and start a safety procedure depending on the failing component.

In summary, the generic mission controller provides the functionality of the mission planning and supervision blocks described in the reference architecture.

6.2.6 Fleet Knowledge Base – SCALIAN

SCALIAN has worked on developing a generic architecture, the EZ_Chains architecture, to allow fleets of agents to perform a variety of mission (see Section 6.2.5 above). The decision is distributed (i.e., each agent has to compute its list of actions to perform). However, the agents require an updated status of the mission. Thus, in EZ_Chains, a Knowledge Base (KB) serves the purpose of sharing the information through a reliable link inside the fleet.

As the mission progresses and the agents perform their actions, they report them to the KB allowing the other agents to plan accordingly. The KB also stores the operational constraints provided by the operators: flight area, geo-fencings, position and availability of emergency landing pads, and so on. Finally, the KB stores the declared flight path of each UAV. In a similar fashion to an ATM, the UAVs book air segments to prevent other UAVs to use them. It prevents collision since a given air segment is never used by two UAVs at the same time (i.e., tactical conflict resolution).

The Knowledge Base is composed of a relational database (similar to MySQL) and a communication system. The KB uses a centralized approach, where an instance is designated as the main, its role is to validate all the exchanges from the other agents to ensure reliability. The communication system offers four main features:

- A discovery process to identify and pair the agents with the main base.
- A transaction feature, that allows each agent to provide information that require a validation. For instance, booking an air segment requires that no other booking it since the agent started to compute its trajectory.
- A periodic system, that each agent uses to provide periodic information (heartbeat and position for instance, similar to telemetry).
- Finally, a file transfer module that allows to transfer large files. It is mainly used to send surveillance video feeds to the Ground Control Station.

In order to provide those features, a robust middleware has been used that provides most of the delivery and validity checks.

In the context of the **C4D** reference architecture, the knowledge base supports the mission supervision block through the frequent updates of the UAVs' fleet.

6.3 Sensing

To enable a mission execution, the drone should be equipped with a number of sensors to perceive the UAV status and its environment. Some of the sensing technologies support of **COMP4DRONES** project are presented in this section.

6.3.1 Ultra-Wideband-based Indoor Positioning - ACORDE

The Ultra-Wideband based Indoor positioning system/solution (IPS) of ACORDE enables the provision of real-time trustable 3D position and attitude to a drone in a long indoor infrastructure (e.g., tunnel) under construction. ACORDE IPS provides an optimized georeferenced positioning solution for the digitization task, for the posed constraints (3D positioning, challenging operational environment and geometry, cost constraints, RT positioning data at the drone), taking the most of the posed scenario (1 tag positioning) with innovative solutions at several aspects (a novel medium access, auto-positioning).

Figure 31 shows the building block of the ACORDE IPS tag device. This tag device is one element on the ACORDE IPS architecture (see section 6.6 of D3.1). Specifically, the tag is the element that will be embedded/integrated on-board the drone, and in charge to provide position, attitude, and velocity to the UAV within the indoor scenario. Notice that Figure 31 is an instantiation of the generic building block capturing a generic Geo-referenced positioning, and attitude estimation system (described in Annex A of this deliverable). Referring to that generic block, notice that it does not get GNSS observables at its input, but UWB-based ranges (reflecting the distance between the tag and anchors), and magnetic field data for position and attitude estimation.

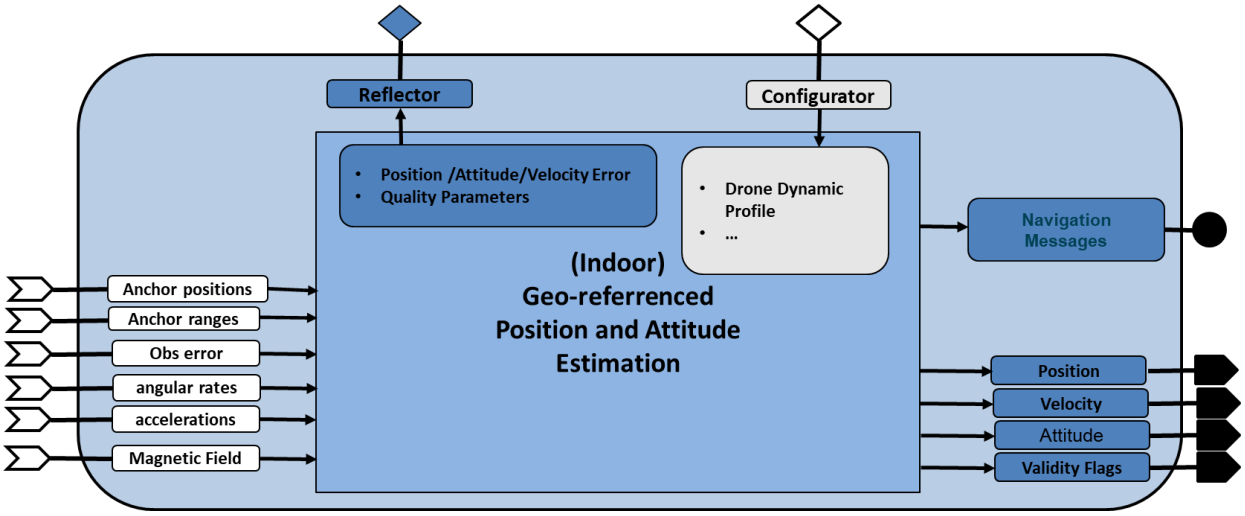

Figure 31: UWB-based indoor positioning system block

This component is an essential part of the UAV perception which enables the autonomous navigation of the drone in the indoor environment, including position, attitude stabilization and speed control. The navigation data provided by this component is an essential information for an autonomous trajectory track and eventually also indoor geo-fencing (e.g., to avoid collision with the tunnel bounds or obstacles provided that their geo-position is known in advance).

6.3.2 Outdoor Position and Attitude Estimation – ACORDE

The outdoor geo-referenced position and attitude estimation system of ACORDE (GLAD+) enables the provision of real-time trustable position and attitude to a drone in an outdoor scenario. ACORDE GLAD+ provides an optimized cost-performance (i.e., accuracy, precision, continuity, and integrity) solution by relying on state-of-the-art, low cost GNSS receivers and low-cost complementary sensing devices (i.e., IMU and barometer).

Figure 32 sketches the block diagram associated to the GLAD+ (see section 6.7 in D3.1). Similarly, as with the IPS tag shown in previous section, Figure 32 building block is a specific instance of the of the generic building block capturing a generic Geo-referenced positioning and attitude estimation system (see Annex A). Unlike the IPS tag of Figure 31, GLAD+ uses neither UWB ranges nor magnetic field information. Instead, GLAD+ realize/employs GNSS observables for an array of N antennas (shown as multiplicity N) for estimation of position, velocity and accurate attitude. Compared to its predecessor GLAD, GLAD+ also benefits from multi-constellation (shown as [M] multiplicity in Figure 32), integrating in **COMP4DRONES** Galileo observables for better accuracy and robustness on the estimation. Compared to GLAD, it also adds detection of jamming and spoofing for a higher integrity on the positioning solution.

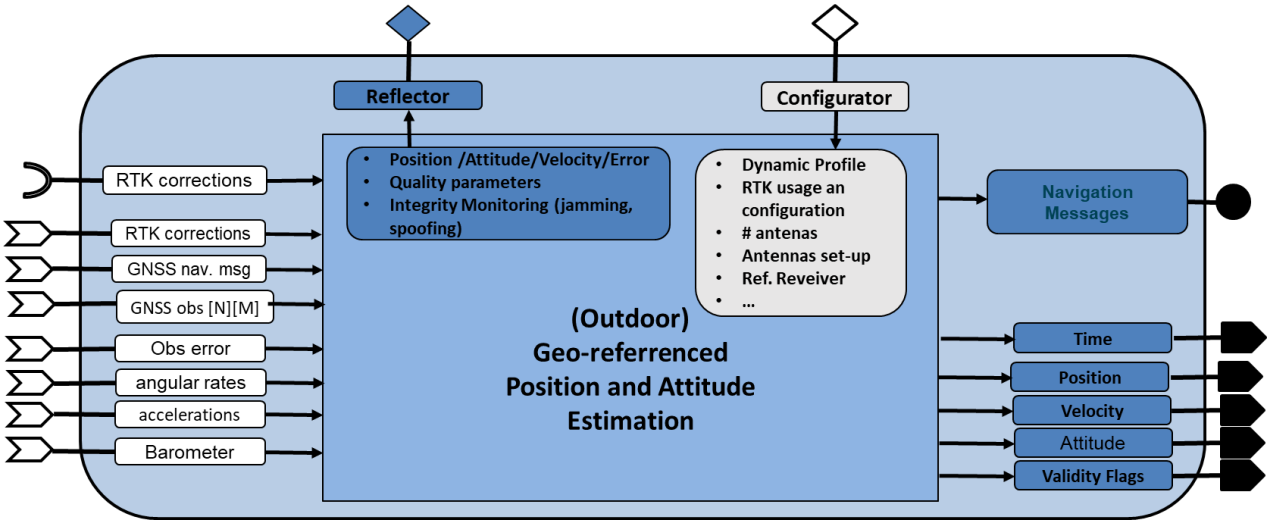

Figure 32: Outdoor positioning system block

While this component is oriented in UC2 of the project to support de digitization task, it can also serve the autopilot accurate and robust navigation data in the outdoor environment, serving for position, attitude stabilization and speed control. Therefore, it provides an essential information for an autonomous trajectory track and for geo-fencing in the outdoor scenarios.

6.3.3 Simultaneous Localization and Mapping Algorithms – MODIS

MODIS SLAM component provides positioning capabilities without relying on the GPS signal. Instead, the component relies on inertial measurements, the geo-magnetic earth field and the distances from fixed points within the map to estimate the position of the drone.

In the context of the reference architecture, this component is part of the Perception block. The component relies on the following blocks in the reference architecture: GPS information (Flight Guidance), sensors' driver (Perception). In turn the SLAM component provides inputs to the following blocks in the reference architecture: trajectory planning and validation (Flight Planning), and also to the position control (Flight Control).

6.4 Data Processing and Analytics

Most of drone missions require data capturing, processing, and analysis. Thus, in the context of the project, a number of components are developed to support that. In the following, we describe some of them.

6.4.1 Computer Vision Component for Drones – HIB

The Computer Vision Component for Drones is a post-processing computer vision system based on previously-trained CNN algorithms which enables the auto-detection and geo-referencing of different

objects from RGB images captured by the UAV's on-board camera. Particularly, this computer vision system intends to improve the digitalization of the state of the constructive process of a civil infrastructure by auto-detecting and geo-referencing different road elements which can be found in any constructive process.

In the **C4D** reference architecture context, the computer vision component for drones supports the payload data analytics building block in the data management block by performing an offline data analysis over the RGB images (that is, payload data) captured by the UAV in order to provide an auto-detection and geo-referencing of different objects. Concretely, the Computer Vision Component provides to the business domain an inventory of the road elements detected and their corresponding position in the terrain to serve as input for the creation of the Building Information Modeling (BIM) model. Then, this Computer Vision Component could be extended for other business functions following the generic building block defined for the payload data analytics.

6.4.2 Sensor Data Processing Algorithms – BUT

BUT is implementing and improving sensor data processing algorithms which include software and firmware for FPGA. This involves video processing algorithms (for example HDR algorithms). HDR multi-exposure fusion algorithm to be implemented in the drone, possibly also implementing tone mapping and/or ghost removal in order to "feed" further image and video processing subsystems in the drone by image information with high dynamic range. The design is based on programmable hardware tightly connected to an "embedded" processor (FPGA SoC Xilinx Zynq, but may also be implemented on other platforms with FPGA). It covers all functionality: reading data from a camera sensor, merging multiple images with alternating exposures into HDR images/HDR video and applying HDR tone mapping. The system can be extended with other functions (software, hardware or FPGA IP core) such as HDR video compression, image pre-processing, exposure control, and the "ghost-free" function removes possible artifacts caused by the movement of objects in the programmable hardware. This block provides acquired data in HDR or tone mapped format and can be extended with other data analytics tools/algorithms (e.g., detectors).

In the **C4D** reference architecture context, this block supports data acquisition for further processing either in the FPGA or in the following systems. Sensor information algorithms is part of payload management – data acquisition block, and component provides inputs to the data management block (i.e., payload data analytics).

6.4.3 Hyperspectral Imaging (HSI) Processing Pipeline - IMEC-BG

The hyperspectral imaging (HSI) pipeline is a system that processes and analyses the hyperspectral data originating from the hyperspectral payload developed by IMEC. It supports a drone system in many aspects. First, it provides an API to easily access radiometric- and reflectance-corrected hyperspectral images. Second, it implements tools to compress the data in a lossless manner in order to limit storage needs. Third, it provides a fully automated way of stitching the images together into a georeferenced orthomosaic. Finally, the HSI pipeline also accommodates an API to easily interpret the data by semantically labelling the data (i.e., assigning a class label to each pixel of the image) also known as classification. The latter is done in a semi-supervised way and supports many different use cases, including applications on surveillance and inspection.

In the context of the reference architecture, the hyperspectral imaging (HSI) pipeline supports payload data analytics block.

6.4.4 AI Drone System Modules – UDANET

The objective of UDANET on **C4D** project is to design, train, and test AI algorithms with camera imaging system to detect and identify parasite animals and to classify leaf diseases. The "AI Drone System Modules" supports data analytics block. The inputs for methods of this component are images from cameras, and the output is the plant health status classification. Artificial intelligence algorithms with

different characteristics will be designed and implemented with the aim of considering the computational cost of each, as well as the over-fitting problem and the dataset that is not always adequate, and the diagnostic performance is drastically decreased when used on test datasets from new environments (i.e., generalization problem).

The improvement targeted by this component is to develop and test the algorithms, increase the reference dataset using basic image manipulation and deep learning-based image augmentation techniques such as image flipping, cropping, rotation, colour transformation, PCA colour augmentation, noise injection, Generative Adversarial Networks (GANs) and Neural Style Transfer (NST) techniques. Performance of the data augmentation techniques will be studied using state of the art transfer learning techniques both in terms of accuracy and computational cost.

6.4.5 Video and Data Analysis Algorithms – AI

The Video and Data Analysis Algorithms is a software module that implements Video Content Analysis (VCA) algorithms. The algorithms are based on Deep Learning methodologies and their goal is to process different type of images (e.g., RGB, thermal, etc.) acquired by onboard cameras.

In more detail, VCA goals is to perform different tasks like target detection, localization and classification. The component is currently under development. It has been designed to be versatile and flexible enough to carry out different tasks, in heterogeneous applicative domains. As a matter of fact, after a proper training phase, the component will be able to detect and classify different objects and targets.

During the next phase of this project, this component will be used in the Smart Agriculture use case. In this use-case, it will be used to monitor crop and to detect relevant information concerning plants status.

Images and eventually other data collected by onboard camera(s) and sensors are the input for this block. A set of information extracted by processing such images correlated with the acquired data will represent the output of this component. This component will support the payload data analytics block that belongs to the data management block.

6.5 Generic API for Trusted Communication

IFAT is working different aspects of defining a standardized lower-level API for the easy and modular integration of a hardware security module (HSM) into any drone architecture. To access the functionality of a HSM, an API and command library needs to be designed, which is described below.

The multi-threaded architecture is split into two main parts, the transport driver and the command library, as depicted in Figure 33. The transport driver is communicating with the HSM, whereas the command library exposes the functionality to the higher-level user application.

Splitting the architecture in transport driver and command library allows being independent of the hardware, since the provided features of the hardware are different, and the command set can easily be extended. Executing a specific API function results in a serialized command which is added into the command queue by the command library. The transport driver processes the command and relays it via the I^2C driver to the HSM, where it is executed and passed back via the transport driver to the command library and subsequently to the user application.

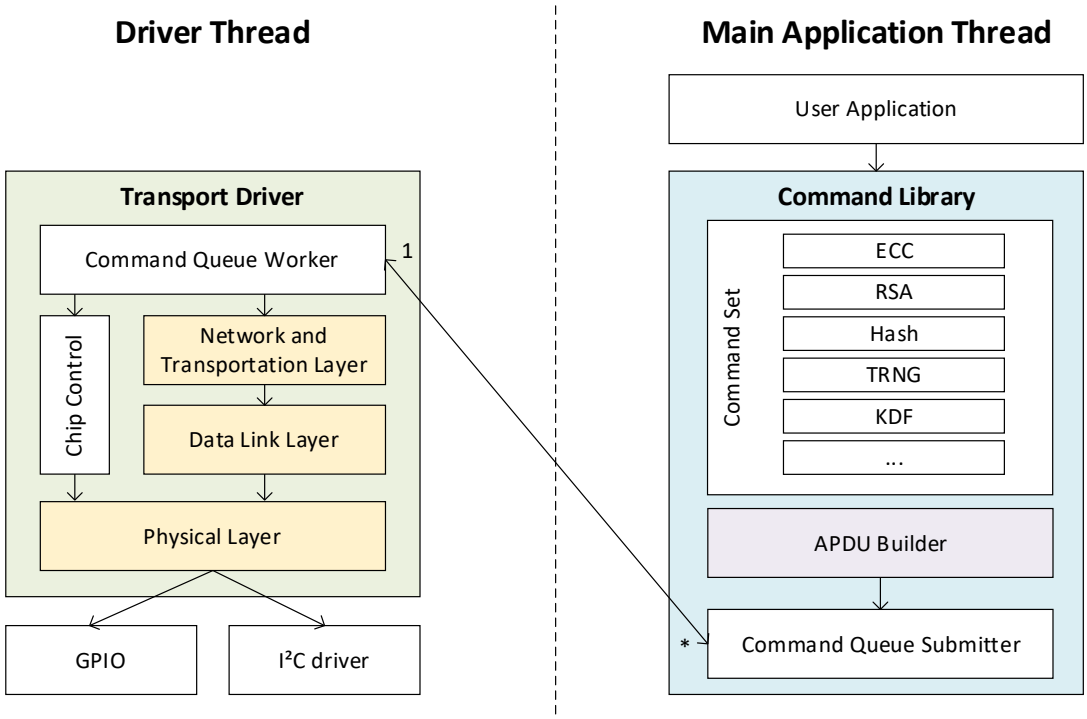

Figure 33: Design of generic architecture and SW-API for integrating various HSM modules into a drone

The transport driver reflects similar layers (depicted in yellow). The physical layer mainly consists of a set of registers to communicate with the two surrounding modules, the I²C driver and the data link layer. The data link layer is providing reliability to the communication channel by adding checksums and sequence numbers. Network and transport layer are combined in one module for simplification, since they share the same header structure. The main task of the transport layer is the packet fragmentation. The command queue worker is the direct interface to the command library and handles the state machine to communicate with the HSM. The chip control is out of scope for this deliverable, but it handles the power management of the HSM.

7 Conclusions

In this deliverable, we have provided the specification for the reference architecture of drone systems. This architecture is going to be adopted in the project. To specify the architecture, first, we listed different requirements that drive a drone system architecture. The requirements are either related to the overall architecture, or to specific architecture elements/blocks.

Second, a set of concepts are presented. These concepts include: (1) a strategy for deriving the reference architecture from its requirements and existing drone system architectures, (2) a decomposition of the drone system, and (3) a template for defining the system blocks.

Third, following the architecture specification strategy, the different building blocks of the system and their interaction patterns are identified and described. These blocks are then clustered and formed together (guided by the interaction patterns) to create the reference architecture of the drone systems.

Finally, in addition to building blocks of drone systems, a number of supporting technologies are described. These technologies either provide a block capability, or serve as a support for the execution of the different system blocks.

8 Annex A: Architecture Blocks Specification

In this section, we list and provide the specification of the different building blocks of the drone architecture.

8.1 Generic Mission Controller

Figure 34 and Table 1 depicts the abstracted description for the Generic Mission Controller (GMC). It receives requests in the form of mission update (mission starts, mission status) change operations update (changes in geo-fence) and GCS orders (human request overriding mission orders). The GMC requests the activation of the different components to achieve the mission, but also provides the mission updates to the Knowledge Base (KB) as the agents complete it.

Table 1: Description of the mission planning block

Block Name		**Multi-Drone Formation**
Short Description		The Generic Mission Controller offers three main features: task planning to allow the agent to decide on the next actions to perform, task monitoring to execute the selected actions and a system watchdog ensuring that all the components and systems are healthy during execution.
Functional Requirements		• The GMC shall receive mission updates and orders, plan the actions and trigger the different components to achieve the mission. • The GMC shall ensure, at all time, the health of the system. • When a component is unhealthy, the GMC shall take an appropriate action to ensure the system remains safe. • The GMC shall react as soon as possible to the GCS orders, since they must override the mission
Data	**Input**	• Periodic update
	Output	• Periodic update
	Configuration	• Agent type
	Status	• Heartbeat • Process status
Services	**Import**	• Mission update • Operations update • GCS orders
	Export	• Component requests: trigger for the components processes • Mission update: progress on the mission achieved by the agent

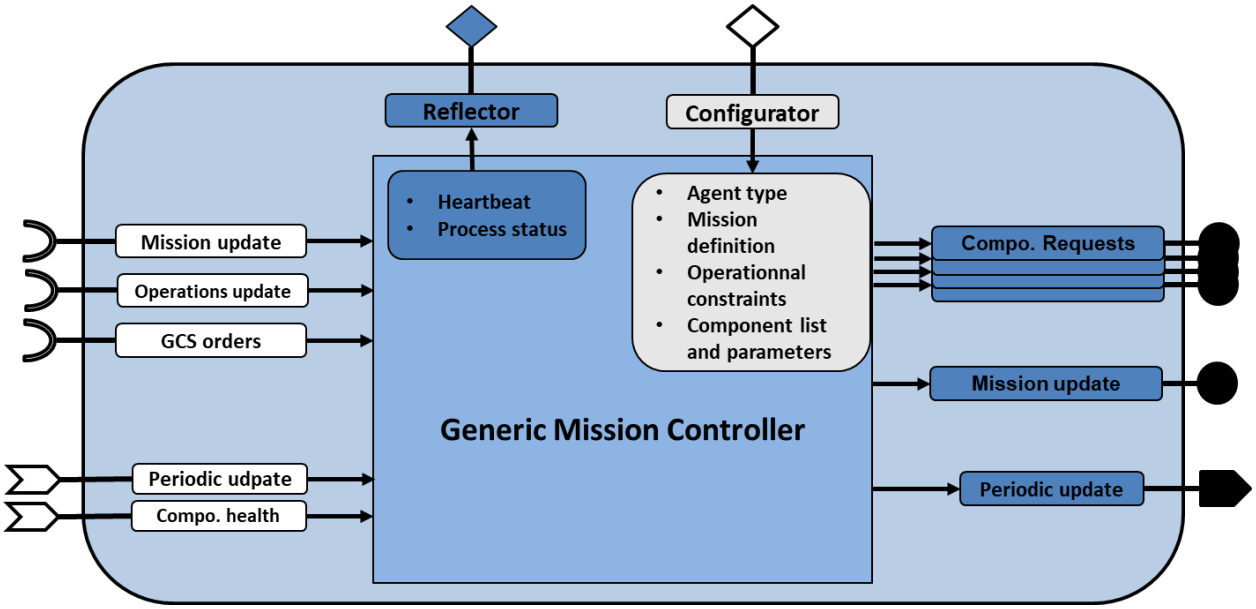

Figure 34: Generic mission controller

8.2 Multi-Drone Formation

In Table 2 and Figure 35, a description of planning a mission with multiple drones is given, which can be considered as a specific type of mission controller.

Table 2: Description of the multi-drone formation block

Block Name		Multi-Drone Formation
Short Description		Formation pattern are produced by multiple drones working in cooperative manner to accomplish various tasks. Any formation shape can be described by a formation vector.
Functional Requirements		• Target x, y, z coordinates based on formation shape shall be calculated. • Target roll, pitch, yaw to achieve target position shall be calculated and enforced. • Target thrust shall be calculated and enforced.
Data	Input	• Drone position • Drone rotation • Position of neighbouring drones • Formation vector
	Output	• Goal roll, pitch, yaw (target angular rate/torque) • Goal thrust • Goal position
	Configuration	• Drone type
	Status	• Formation error
Services	Import	• Communication service to obtain neighbours' position • Control services for angular rate
	Export	• Control attitude • Control position

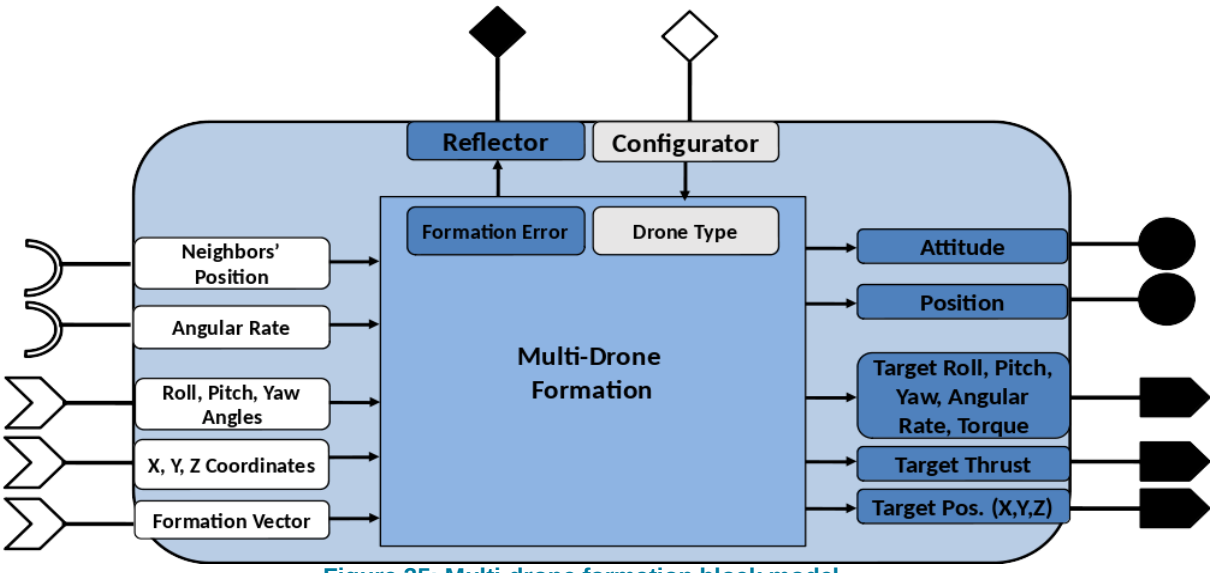

Figure 35: Multi-drone formation block model

8.3 Trajectory Planning and Validation

The description of the trajectory planning and validation is in Figure 36 and Table 3.

Table 3: Description of the trajectory planning and validation block

Block Name	Trajectory Planning and Validation
Short Description	The trajectory validation is the process of authorization of the flight trajectory and flight plan. The trajectory validation service can process drone operation plans and check for compliance with the applicable national regulations. Depending on the customized workflow, the systems can authorize the operations automatically, reducing response time and automating most of the workload. If the drone operation cannot be authorized automatically, the drone operator can request authorization from a supervisor such as Air Traffic Control (ATC), local government, or law enforcement.
Functional Requirements	• The trajectory validation system shall evaluate the validity of the flight plans requested. The possible outputs shall be planned, manual, or denied status. • The trajectory validation system shall provide alternative flight plans when possible if the original flight plan requested was denied. • The trajectory validation system shall be able to receive the authorization request for an alternative flight plan proposed. The complete authorization process shall be performed. • The trajectory validation system shall provide the state/authority with each manual flight plan for its manual authorization. • The trajectory validation system shall modify airspace allocation when a Flight Manager or Operation Manager becomes unavailable. • The trajectory validation system shall detect trajectory conflicts between different agents. • Flight plans in PLANNED status are authorized and ready to flight once its start time is reached.

		• The trajectory validation system shall send to the drone operator a notification message with the complete output of the authorization process.
Data	**Input**	• Flight plan • Drone and drone operator ID
	Output	• Flight planning status & updates
	Configuration	• Database credentials
	Status	• Validation status
Services	**Import**	• Flight plan management • Ground service • Geo-fence service
	Export	• Flight plan authorization

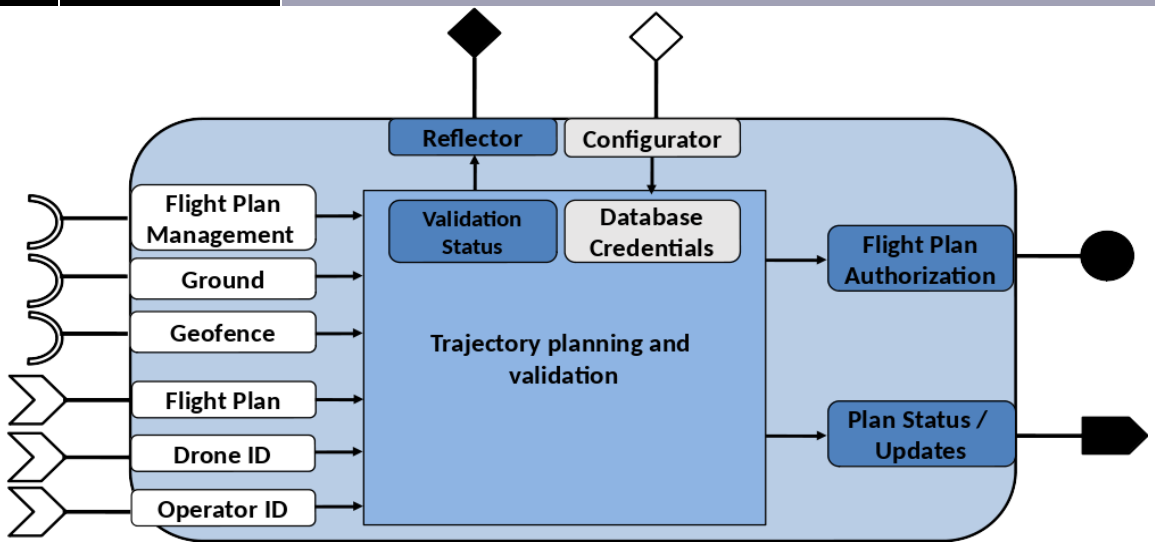

Figure 36: Trajectory planning and validation block model

8.4 Trajectory Execution

The description of the trajectory execution is in Figure 37 and Table 4.

Table 4: Description of the trajectory execution block

Block Name		**Trajectory Execution**
Short Description		This block translates the drone trajectory vector set by the mission planner and/or obstacle avoidance to set-points for the flight controller blocks (position and attitude control), so the planned mission is executed.
Functional Requirements		• The Trajectory Execution shall obtain the desired drone trajectory as position, velocity and/or acceleration, and time-duration input. • The Trajectory Execution shall obtain the actual drone orientation status position, velocity and/or acceleration. • The Trajectory Execution shall calculate the set-points for position, velocity and attitude control blocks. • The Trajectory Execution should validate the new trajectory to see if its execution is doable.
Data	**Input**	• Drone estimated state • Drone goal trajectory vector with timestamp

	Output	• Goal position coordinates of the drone • Goal velocity • Goal roll, pitch, and yaw
	Configuration	• Drone physics restrictions • Time synchronization
	Status	• Error diagnosis
Services	**Import**	• Time synchronization • Localization
	Export	N/A

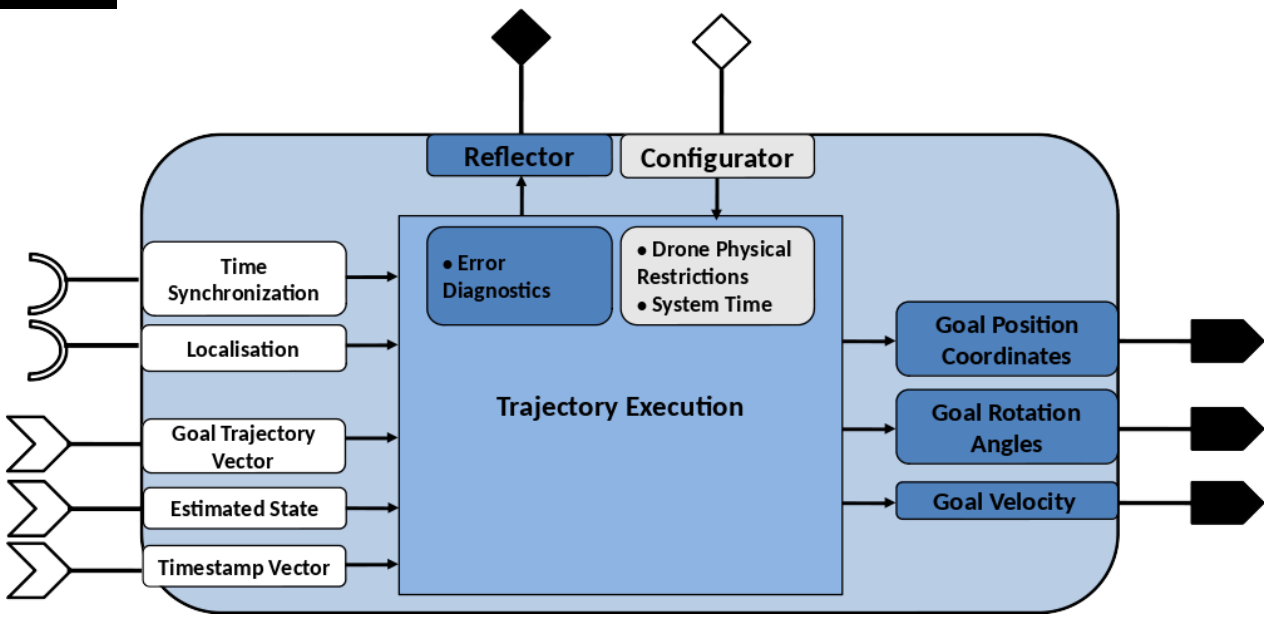

Figure 37: Trajectory execution block model

8.5 Obstacle Detection

The description of the obstacle detection is in Figure 38 and Figure 5.

Table 5: Description of the obstacle detection block

Block Name		**Obstacle Detection**
Short Description		This block fetches sensor data and processes it to generate a map that represents the detected obstacle, so the obstacle avoidance block can calculate new trajectory to avoid the hazard/obstacle.
Functional Requirements		• The Obstacle Detection shall have connected obstacle measurement sensors (2D or 3D) • The Obstacle Detection shall have the drivers to obtain the sensor data
Data	**Input**	• Drone estimated state
	Output	• Obstacle map
	Configuration	• System time • Sensor configuration • Map type • Sensor selection

		• Data fusing/filter algorithm selection • Data publish rate • Drone mech setup
	Status	• Error diagnosis
Services	**Import**	N/A
	Export	N/A

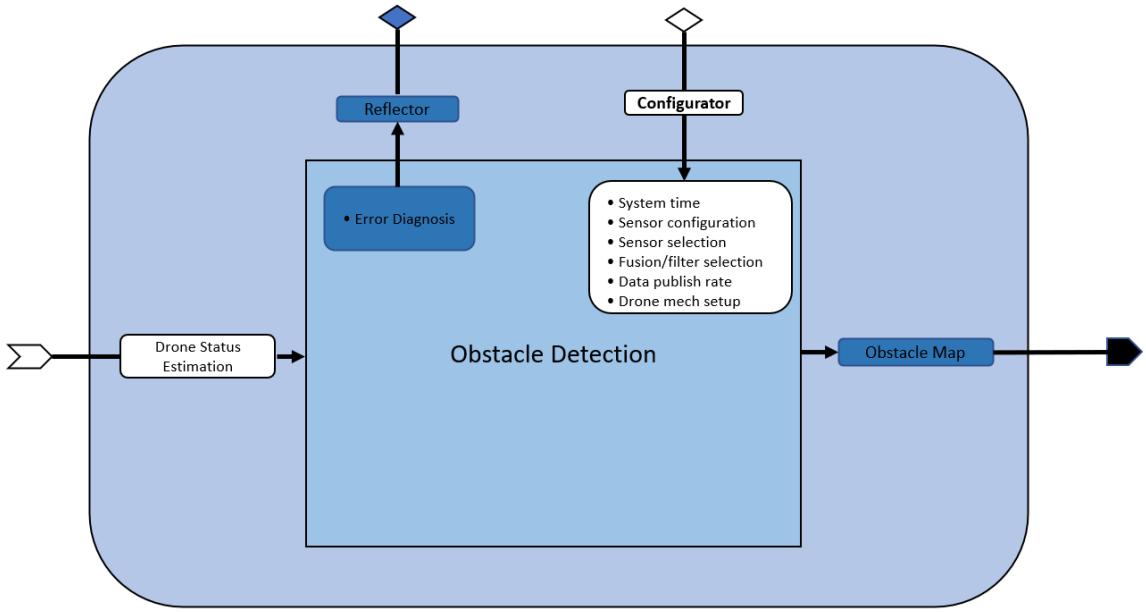

Figure 38: Obstacle detection block model

8.6 Obstacle Avoidance

The description of the obstacle avoidance is in Figure 39 and Table 6.

Table 6: Description of the obstacle avoidance block

Block Name		**Obstacle Avoidance**
Short Description		This block fetches obstacle map and, depending on the physical limitations of the drone, calculates a new trajectory vector to avoid the hazards
Functional Requirements		• This block shall have an obstacle map as input • This block shall detect an obstacle from the input map • This block shall know the drone type and its manoeuvring capabilities. • This block shall calculate a valid new trajectory to avoid the obstacle
Data	**Input**	• Obstacle map
	Output	• Trajectory vector with timestamp
	Configuration	• System time • Mechanical capabilities • Trajectory type/configuration
	Status	• Error diagnosis
Services	**Import**	N/A
	Export	N/A

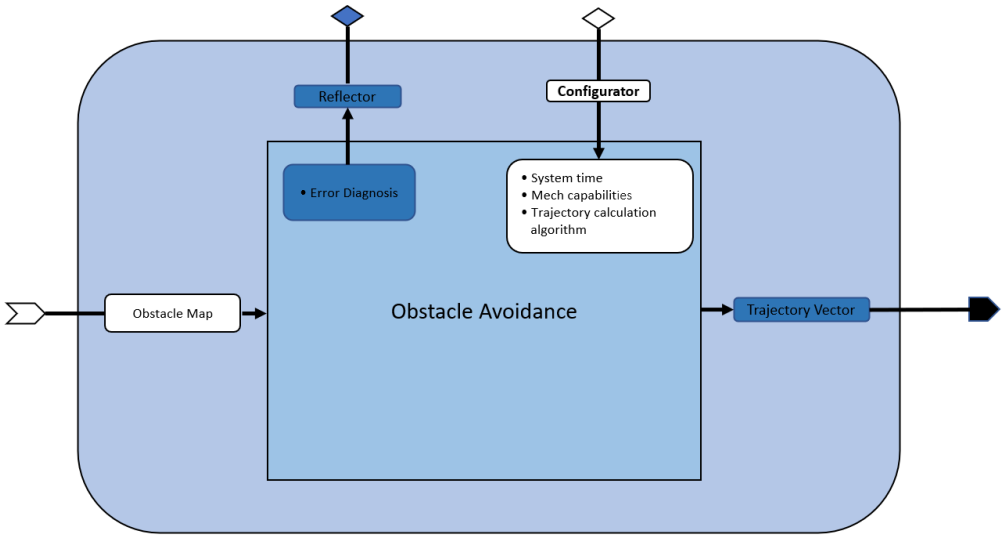

Figure 39: Obstacle avoidance block model

8.7 Geo-awareness and Geo-fencing

The description of the geo-fencing and geo-awareness is in Table 7 and Figure 40.

Table 7: Description of the geo-awareness and geo-fencing block

Block Name		**Geo awareness and Geo-fencing**
Short Description		Geo-fencing block keeps the drone flying without violating the geo-fence area defined by the regulation authorities.
Functional Requirements		• The trajectory of the drone and the distance between the drone and the geo fence shall be monitored • A warning shall be generated if the distance becomes less than a certain threshold • Repulsive force shall be generated and enforced to fly away the drone from the fence • Alter the trajectory of drone based on the repulsive force to avoid the violation of geo-fence by controlling the position and attitude
Data	**Input**	• Position • Coordinates of the geo fence • Velocity
	Output	• Distance to geo fence • Geo fencing violation alert • Goal roll, pitch and yaw • Goal speed
	Configuration	• Maximum speed • Drone type
	Status	• Formation error
Services	**Import**	• Proximity sensor service
	Export	• Geo awareness • Geo-fence preservation • Control attitude • Control position

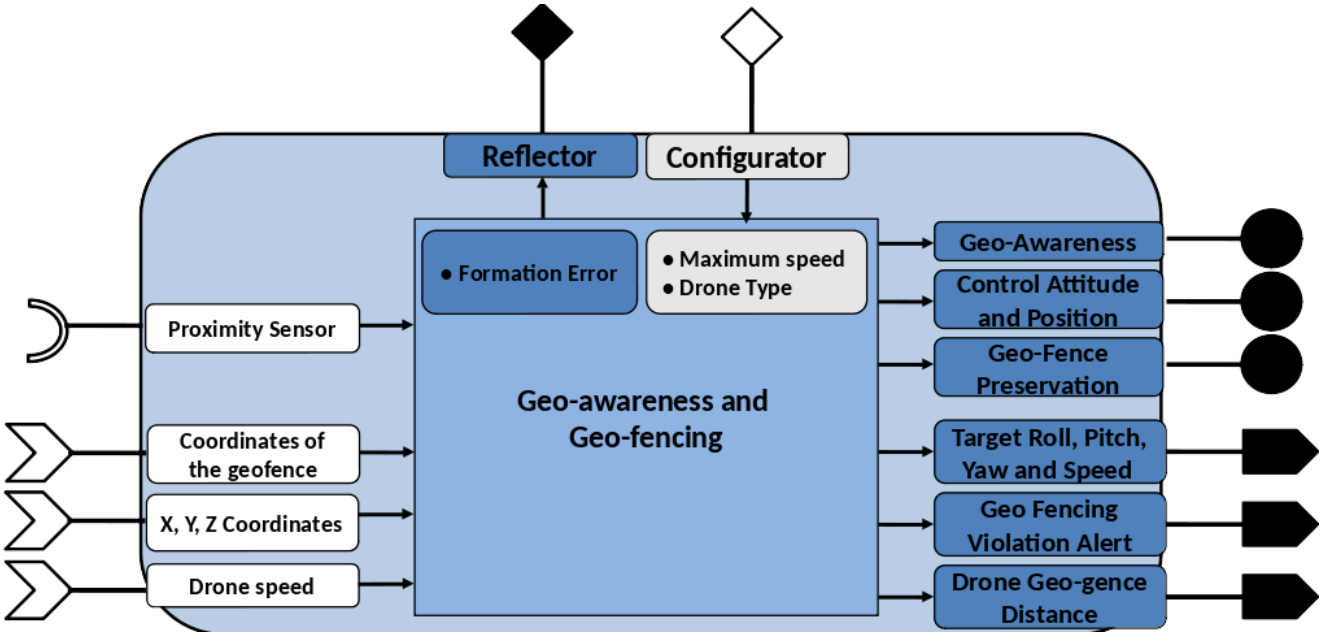

Figure 40: Geo-awareness and geo-fencing block model

8.8 Position and Velocity Control

The description of the position and velocity control is in Figure 41 and Table 8.

Table 8: Description of the position and velocity control block

Block Name		Position and Velocity Control
Short Description		The position and velocity controller takes the inputs from the trajectory generation and compute the desired attitude and desired thrust
Functional Requirements		• The position controller shall calculate the desired attitude and thrust, taking into account constraints on speed • When used with speed inputs, the controller shall calculate the desired attitude and thrust • If applicable, the speed to control and constrain is the airspeed (fixed-wing) or the ground-speed (rotorcraft)
Data	Input	• Goal position • Goal velocity • Drone estimated state
	Output	• Goal attitude • Goal thrust
	Configuration	• Control loops gains • Saturation on velocity, acceleration and outputs
	Status	• Position and velocity errors
Services	Import	• Control services for position and speed
	Export	• Position control • Speed control • Speed or airspeed constraints

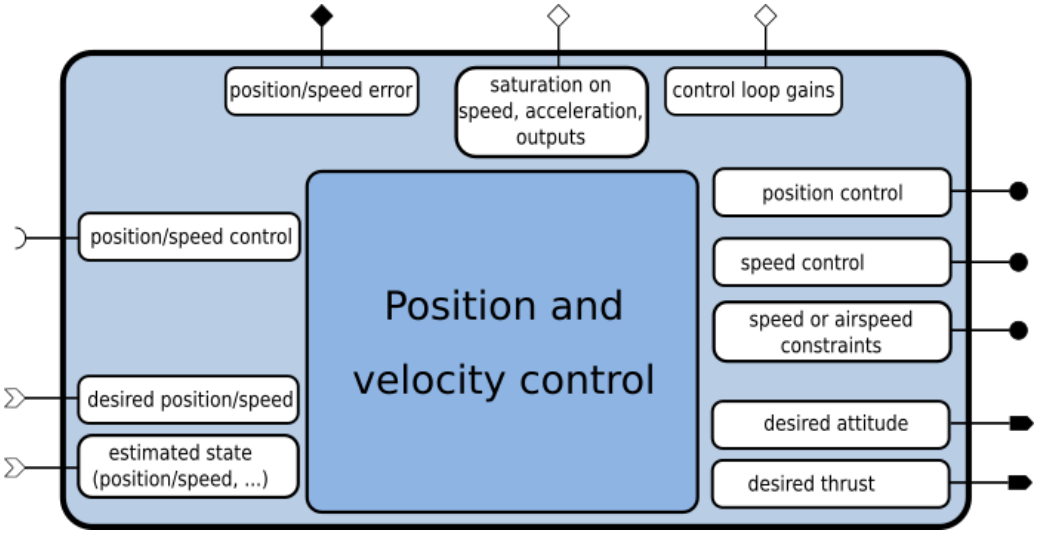

Figure 41: Position and velocity control block model

8.9 Attitude and Rate Control

The description of the attitude controller is presented in Table 9 and is visualized in Figure 42.

Table 9: Description of the attitude and rate control block

Block Name		**Attitude and Rate Control**
Short Description		The attitude controller takes the desired attitude (or body rates) and computes the desired command, most of the time around the body axis (roll, pitch, yaw)
Functional Requirements		• The attitude control shall calculate the command vector to stabilize the orientation of the UAV, taking into account constraints on rotation speeds • When used with rates inputs, the rate control shall calculate the command vector to stabilize the body rates of the UAV
Data	**Input**	• Goal attitude • Goal thrust • Estimated state
	Output	• Command vector (usually around roll, pitch and yaw axis)
	Configuration	• Control loop gains • Saturation on rates, acceleration and outputs
	Status	• Attitude and rates errors
Services	**Import**	• Control services for attitude and rotation rates
	Export	• Attitude control • Rotation rate control • Rotation rates and acceleration constraints

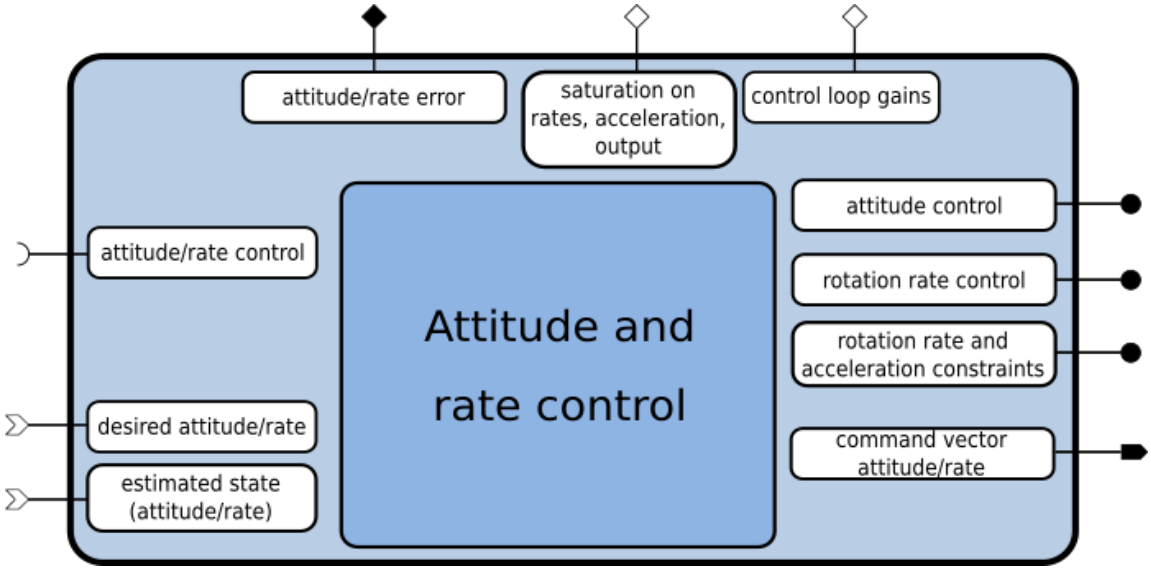

Figure 42: Attitude and control block model

8.10 Mixing

The description of the mixing is in Table 10 and Figure 43.

Table 10: Description of the mixing block

Block Name		**Mixing**
Short Description		Command mixing takes as inputs the command vector from the attitude controller and the thrust command from the position controller to compute the actuators set-points according to the structure of the aircraft
Functional Requirements		• The mixing shall compute the command for each actuator from the command vector (output of attitude and position controllers) • If applicable, the mixing shall prevent saturation of actuators that would reduce the stability of the flight (priorities between attitude, heading and position can be defined)
Data	**Input**	• Command vector (usually around roll, pitch, yaw axis) • Goal thrust • Additional commands (such as flaps) • Flight status (flight mode, fail-safe, stop motors, ...)
	Output	• Actuators setpoint
	Configuration	• Mixing parameters (equations or matrices giving the relation between commands and actuators) • Saturation constraints and priorities is applicable
	Status	• Actuator status
Services	**Import**	• Mixing services between command vector and actuators
	Export	• Command mixing

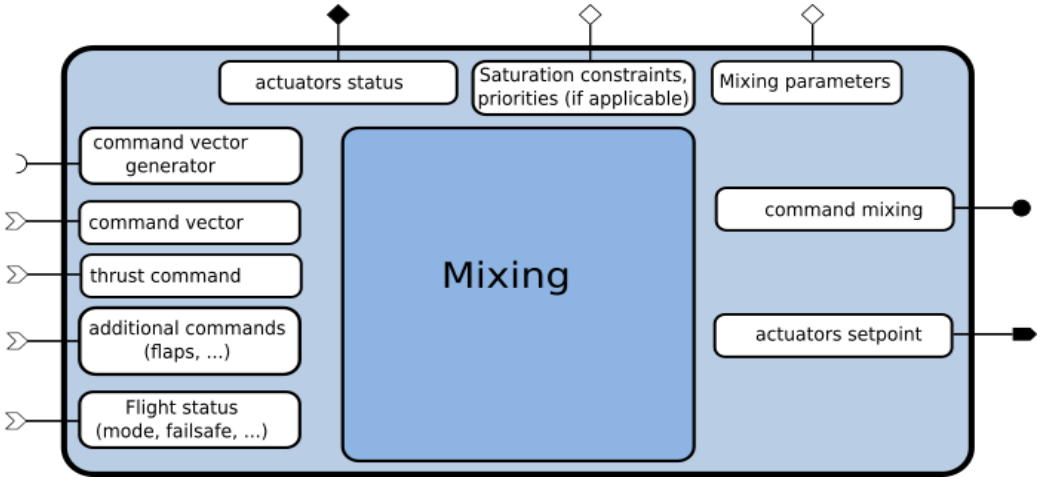

Figure 43: Mixing block model

8.11 Geo-Referenced Position and Attitude Estimation System

The description of the position and attitude estimation is in Figure 44 and Table 11.

Table 11: Description of the (outdoor) geo-referencing positioning block

Block Name		Geo-referenced positioning, attitude and velocity estimation
Short Description		The geo-referenced Positioning and Attitude Estimation System provides accurate, precise, continuous and robust estimation of the position and attitude (pose) of the drone with regard to a geographic coordinate system.
Functional Requirements		• Shall provide geo-referenced position, attitude and velocity upon a global time scale (e.g. GPS and UTC)
Data	**Input**	• GNSS navigation data (ephemeris and ionospheric corrections) • GNSS observables (pseudoranges, doppler, carrier-phase), DOPs, range variances, flags, etc.) • GNSS corrections (RTK) • Anchor positions and ranges to them • IMU data (angular rates and acceleration) • Barometer data (pressure and temperature) • Others: depending on sensors (Radars, laser, and visual odometry)
	Output	• Global Time • Position • Velocity • Attitude
	Configuration	• Profile (slow, dynamic) • RTK usage and configuration • Antenna(s) usage positioning (relative among them, i.e., the body) • GNSS receiver's model
	Status	• Error estimations • Quality parameters (C/N, observable flags, etc.) • Integrity monitoring (RAIM, spoofing, jamming)
Services	**Import**	• Status • RTK corrections
	Export	• Initialization • Navigation messages

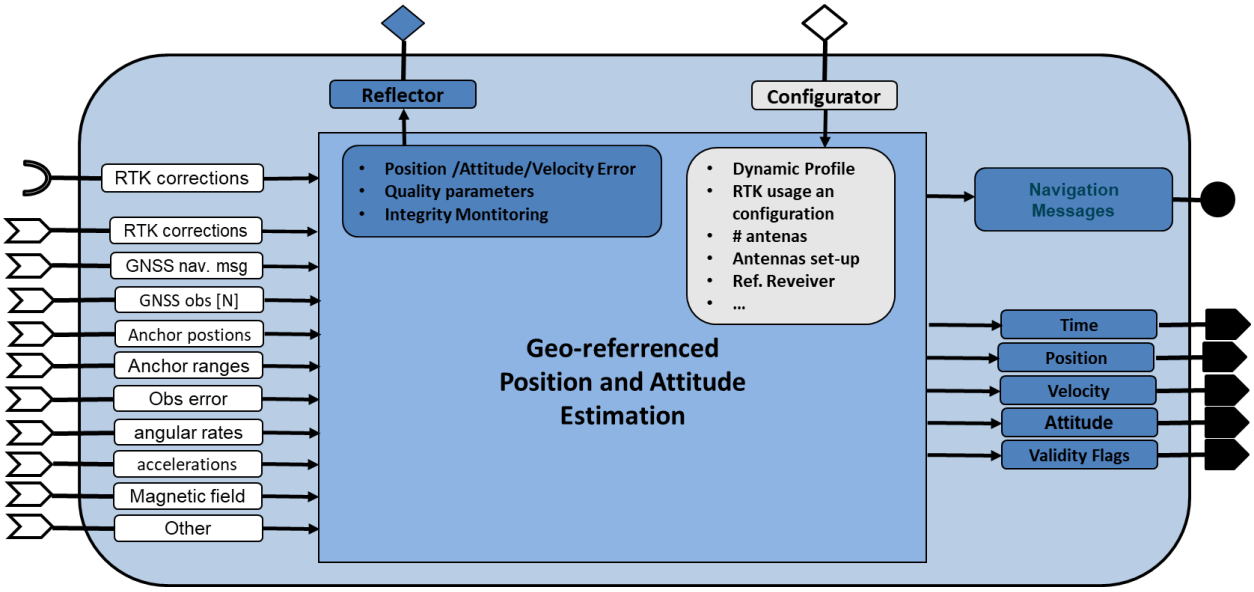

Figure 44: Geo-referenced position, attitude and velocity estimation block

8.12 UAV Surrounding

The description of the UAV surrounding is in Table 12 and Figure 45.

Table 12: Description of the UAV surrounding block

Block Name		**UAV Surrounding**
Short Description		UAV Surrounding block makes use of different sensor data and preferably fuses them while creating and maintaining a coherent map of the surrounding obstacles and objects.
Functional Requirements		• The UAV Surrounding component shall maintain a coherent map of surrounding obstacles and objects. • The UAV Surrounding component shall expose mechanism for exploiting object and obstacle detection confidence.
Data	**Input**	• Geometry-oriented data (e.g. RGB/infra-red, LIDARs, RADARs, and ultra sound) • High-definition map
	Output	• Obstacle map (list of obstacle)
	Configuration	• Coefficients for input data filtering • Sensor tractability metric for fusion (covariance matrix)
	Status	• Real-time performance metrics
Services	**Import**	• Detected objects • Navigation
	Export	• Coherent surrounding map

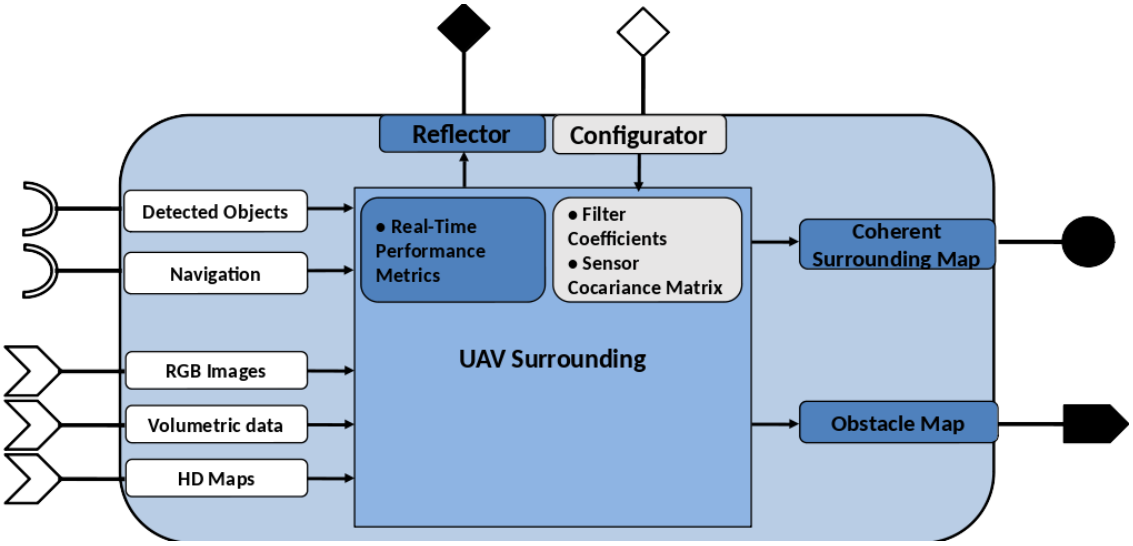

Figure 45: UAV surrounding block model

8.13 Power Management

The description of the power management is in Figure 46 and Table 13.

Table 13: Description of the power management block

Block Name		Power Management
Short Description		This building block provides management of batteries or other power units. It takes care of the estimation of the current status and provides communication for higher-level diagnostic systems.
Functional Requirements		• The building block shall estimate state of the available power management and provides the information about the actual state. • The block informs and constrains the operating time.
Data	Input	N/A
	Output	• Power info • Power fault
	Configuration	• Power type • Power function
	Status	• Power management status
Services	Import	N/A
	Export	N/A

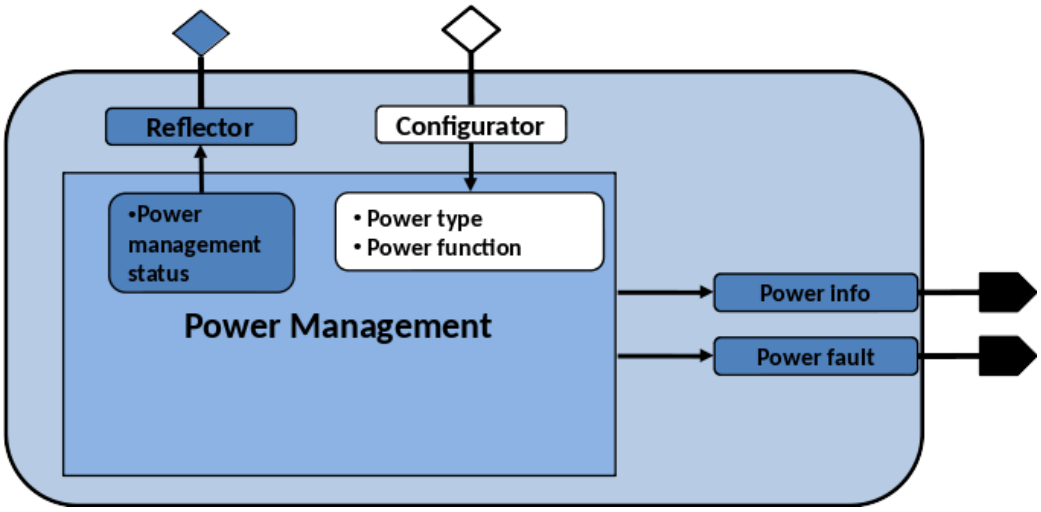

Figure 46: Power management block model

8.14 Fault Management

The description of the fault management is in Table 14 and Figure 47.

Table 14: Description of the fault management block

Block Name		Fault Management
Short Description		This building block monitors the current state of the UAS and decides to switch to emergency mode if an error condition is detected. Based on the available data, the emergency mode either performs a return to the home position or an emergency landing or another predefined movement.
Functional Requirements		• The building block shall provide faults management capabilities. • The building block shall implement triggers for failsafe actions.
Data	**Input**	• Building block faults
	Output	• Failsafe triggers
	Configuration	• Fault-failsafe table
	Status	• Fault Management status
Services	**Import**	N/A
	Export	N/A

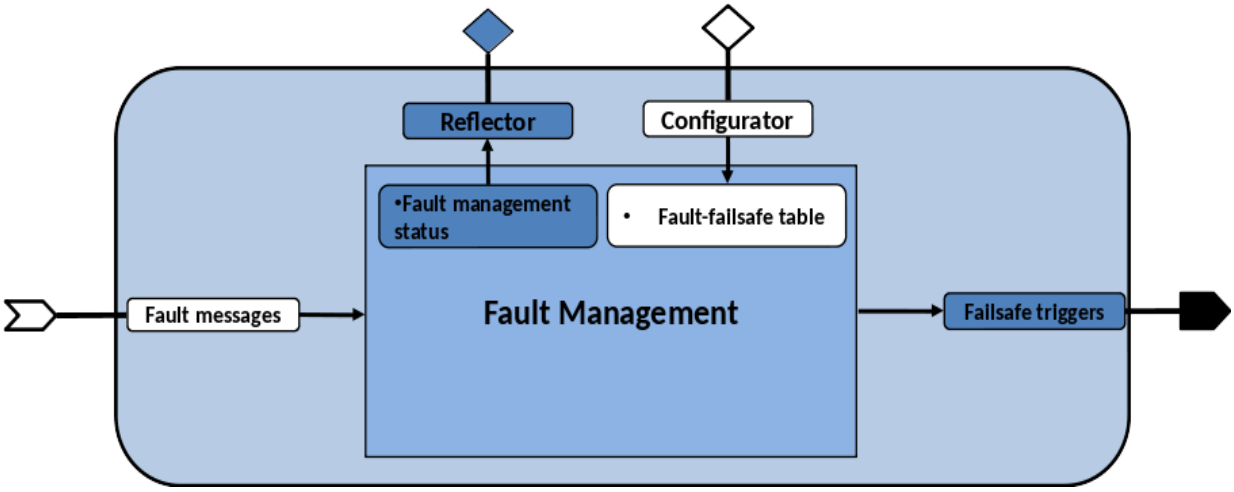

Figure 47: Fault Management block model

8.15 Diagnostics

The description of the diagnostics block is in Figure 48 and Table 15.

Table 15: Description of the diagnostics block

Block Name		Diagnostics
Short Description		Registering status of all UAV blocks and provides output for loggers and ground station
Functional Requirements		• The diagnostics block shall monitor status of every building block in real-time. • The diagnostics building block shall provide telemetry data.
Data	Input	• Array of status of each building block
	Output	• Telemetry data
	Configuration	• Flight log
	Status	• Diagnostics block status
Services	Import	N/A
	Export	N/A

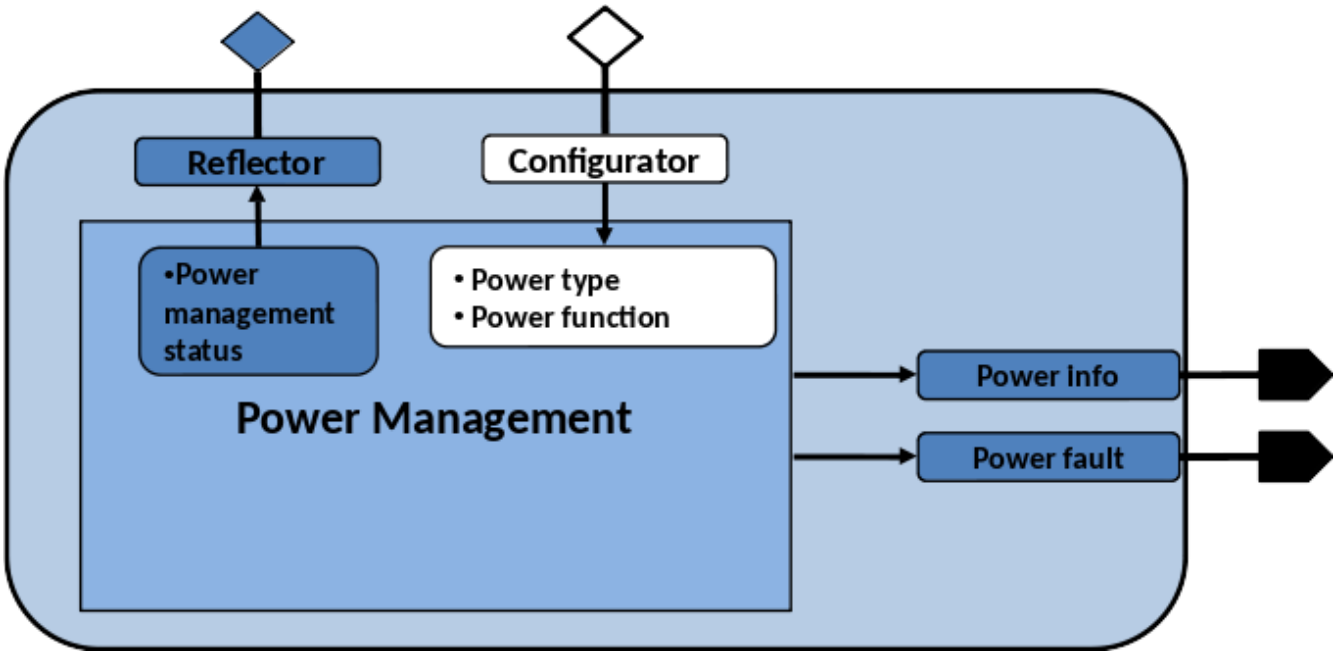

Figure 48: Diagnostics block model

8.16 Payload Manager

The description of the payload manager is in Table 16 and Figure 49.

Table 16: Description of the payload manager block

Block Name		Payload Manager
Short Description		The Payload Manager captures sensor data, tags it with GPS coordinates, stores it locally, processes it to provide certain analytics and sends the raw and processed sensor data to be further transmitted to the ground controller.
Functional Requirements		• The payload should capture sensor data. • The payload should allow user to set the parameters: exposure time, when to start capture and when to stop capture, orientation of the sensor. • The payload should provide processed and unprocessed images to the user/ground controller. • The payload should tag the captured sensor data with GPS coordinates.
Data	Input	• Capture control • Exposure time for the cameras • GNSS navigation data • System time
	Output	• Current captured frame (e.g. RGB image, hyperspectral image, lidar scan, etc. including metadata) • Current processed frame (classified, segmented, registered)
	Configuration	• Spectral range • Maximum speed • Number of spectral bands • Spectral range • Spectral resolution

		• Spatial resolution • Frame rate of acquisition • Degree of overlap at ground level • Ground sample distance • Range (in case of active sensors such as lidar, radar, sonar)
	Status	• Storage space, Temperature of cameras, Number of frames acquired, Current Frame
Services	**Import**	• Power supply • GPS coordinates • A data & control channel to ground controller
	Export	• Data analytics (processed sensor data)

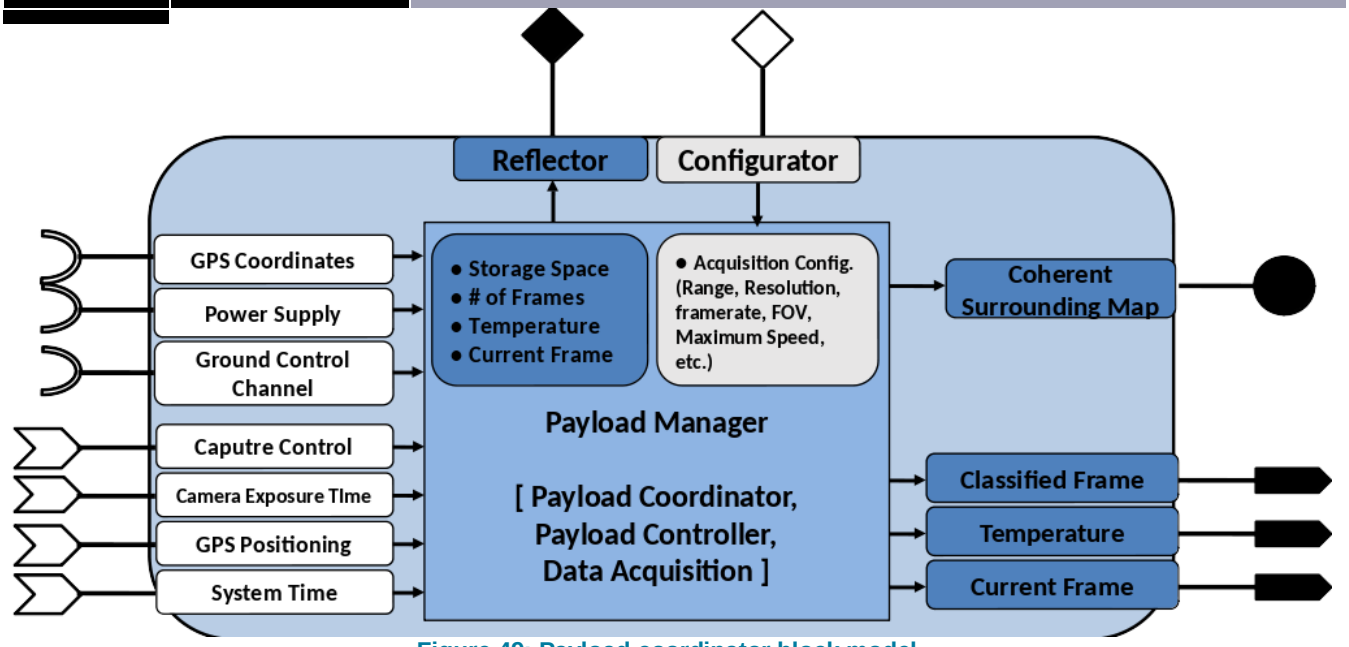

Figure 49: Payload coordinator block model

8.17 Flight Logger

The description of the flight logger is in Figure 50 and Table 17.

Table 17: Description of the flight logger block

Block Name	Flight Logger
Short Description	The flight logger keeps track of the UAV status and records it for latter analysis by the prognostics block. By default, logging is automatically started when UAV is arming, and stopped when disarming. A new log file is created for each drone mission.
Functional Requirements	• The flight logger block shall keep track of all flight-related state variables such as position in space, orientation, and speed along the three reference axes in the rigid body. • The flight logger block should store the status of individual components such as motors, battery, and processor operation. • The flight logger block should store the status of the sensors and their measurements, as well as the actuators and their implemented values. • The flight logger block should store any anomalies that have occurred in all systems present.

Data	**Input**	• Position • Roll • Pitch • Yaw • Velocity • Wind • Obstacle map
	Output	• A flight log file for each drone mission
	Configuration	N/A
	Status	• Writing error
Services	**Import**	• Data writing: UAV navigation insert the above mentioned inputs in the Flight Logger
	Export	• Flight log creation

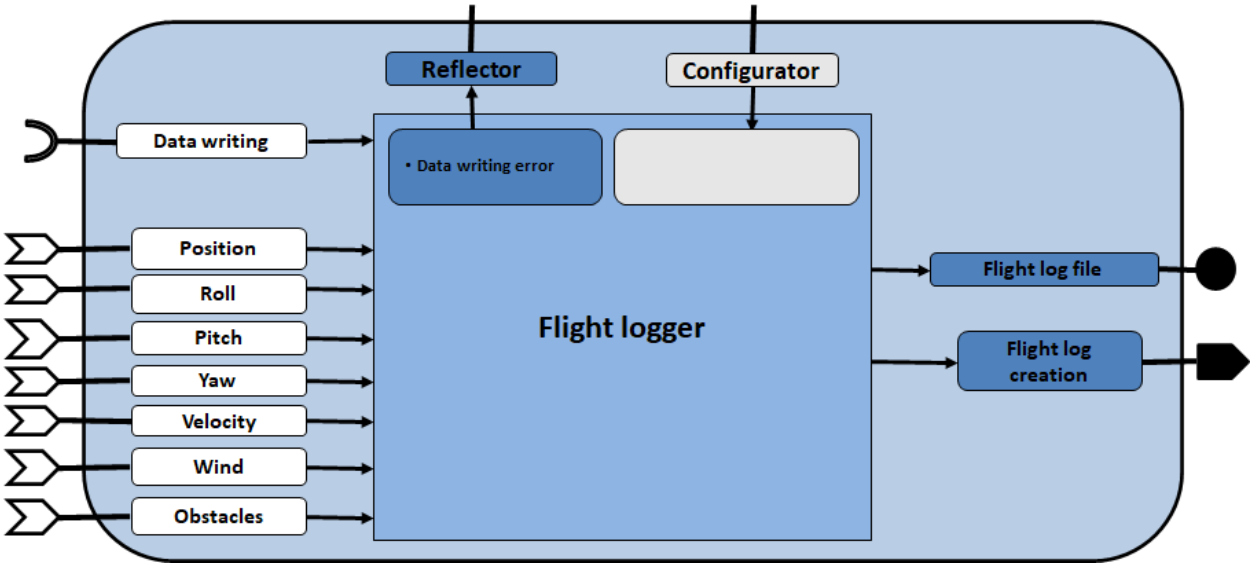

Figure 50: Flight logger block model

8.18 Prognostics

The description of the prognostics block is in Table 18 and Figure 51.

Table 18: Description of the prognostics block

Block Name	Prognostics
Short Description	Prognostics block consists of the set of functions that serves as a predictive analytics engine of the drone systems. It relies on historical data of drone operation and performance, engineering and physics properties of the drone, and modelling information. The main goal is to identify potential issues before they occur and provide recommendations on their mitigation.
Functional Requirements	• The Prognostics Block shall foresee problems related to the state of charge of the battery as insufficient/risk to carry out the mission with a certain safety.

		• The Prognostics Block shall foresee situations of possible collision danger based on the planned trajectory and of the data deriving from the sensors on the other agents/objects present in the visual field. • The Prognostics Block shall identify possible hardware failures based on anomalous signals.
Data	**Input**	• Battery charge status • Mission in act (desired X, Y and Z, roll, pitch, and yaw and all their velocities during the mission to be executed) over time • Normal hardware operating signal values
	Output	• Warning sign if the battery is insufficient or at the limit to perform the mission • Warning sign if there is a risk of possible collisions in the future • Warning sign due to the anomaly of some signals at the hardware level
	Configuration	• Drone Type • Battery parameters • Sensors Type and parameters
	Status	• Percentage of correctness of the danger reports and verification. • Confusion matrices respect the predicted output and real output.
Services	**Import**	• Monitoring of the most important variables.
	Export	• Provide potential issues before they occur like low state of charge respect to the mission, possible future collisions, hardware malfunctions.

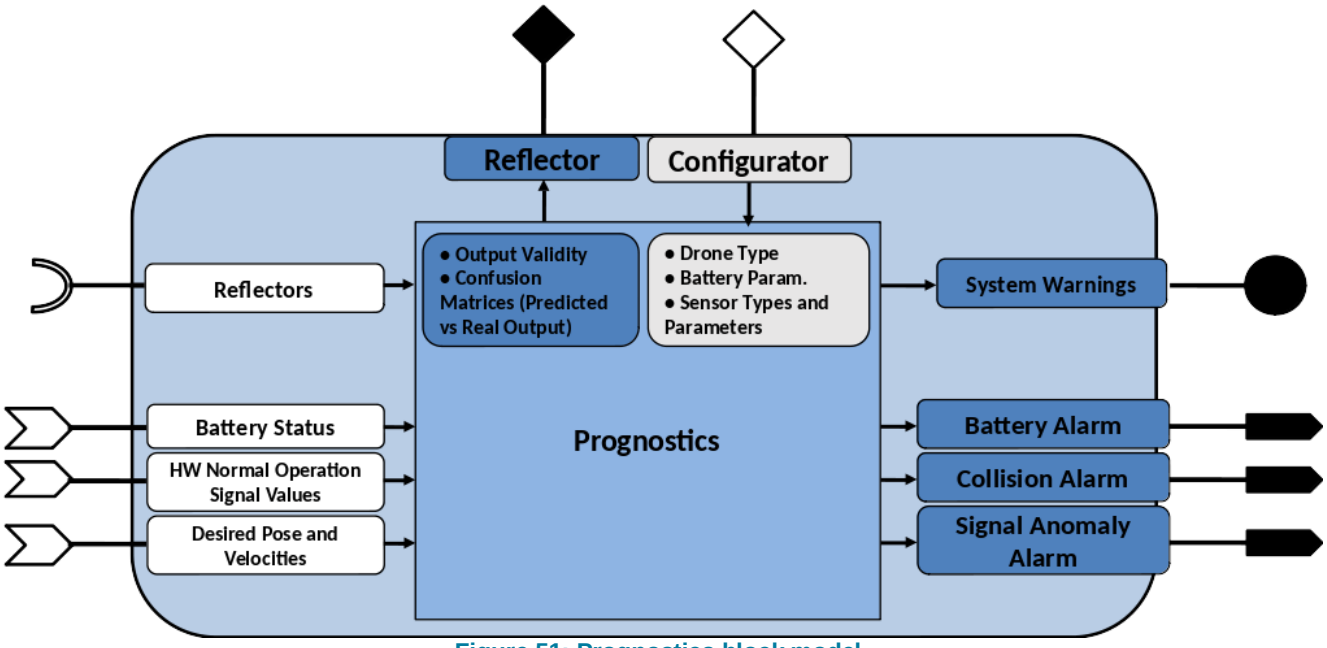

Figure 51: Prognostics block model

8.19 Storage

The description of the storage block is in Figure 52 and Table 19.

Table 19: Description of the storage block

Block Name		Storage
Short Description		The storage block is responsible for providing and for storing all the needed information to execute a mission. Example information needed for executing a flight includes flight plans, geo-fence limitation, UAV model (e.g., UAV power supply). During mission execution, it also stores the generated information about the vehicle health during the flight (e.g., power level, engine temperature, etc.), and information about the drone environment. The storage block also has access control, which is a method of allowing access to the drone's sensitive data only to those people who can access such data and to restrict access to unauthorized persons.
Functional Requirements		• The storage shall read and save data. • The storage shall enable other blocks to read data
Data	Input	• Heterogeneous type of data
	Output	• Data saved in the storage
	Configuration	• User Access control (Username, Password, MAC address)
	Status	• Percentage of the storage space available/occupied • Number of other BBs that are using simultaneously the storage BB to read/write • Reading / Writing error
Services	Import	• Data writing: Other blocks can write data in the storage
	Export	• Data Reading: Other blocks can read data saved in the storage

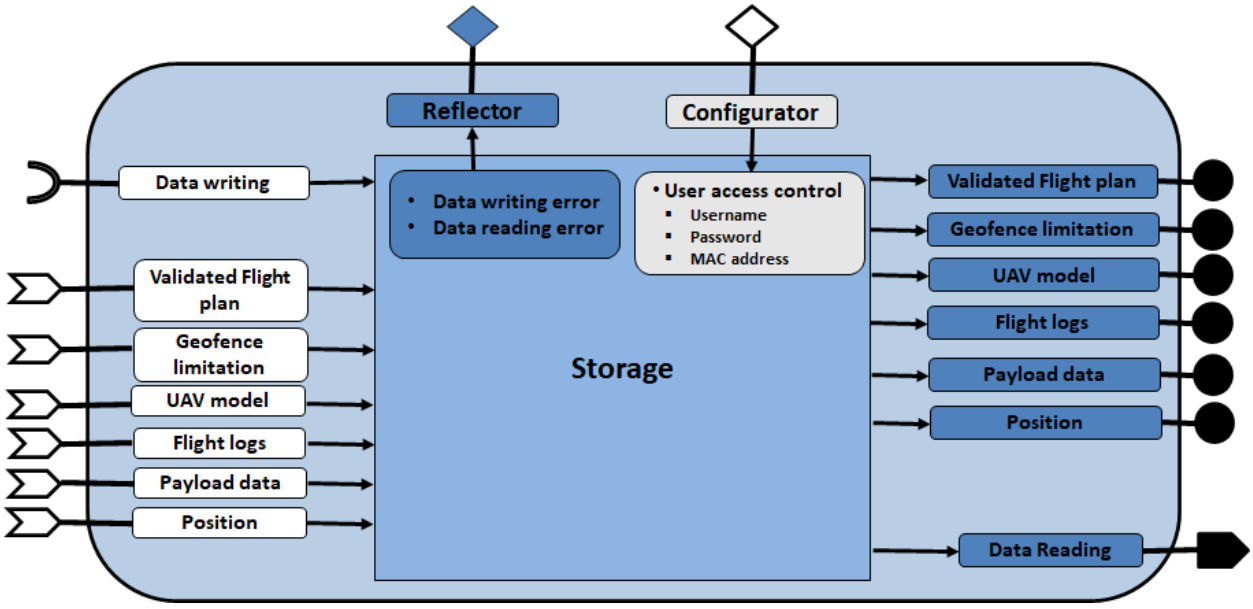

Figure 52: Storage block model

8.20 Fleet Knowledge Base (KB)

Table 20 and Figure 53 present the abstracted view of the Fleet Knowledge Base (a specific storage block for mission plans). It takes as inputs: the mission updates from the generic mission controller (see before), the transaction requests from the different components (e.g. the flight path from the path planner, air segment booking request from the flight controller …) and the file transfer requests from the components that need to send files. Its outputs are: the mission update, operations update and GCS orders, forwarded to the Generic Mission Controller (produced by the GCS and Mission supervisor). It also outputs a data warning the components when an external update has been processed. This allows to have idle components and prevent the components from doing periodic pulling (better performances).

Table 20: Fleet knowledge base description

Block Name		**Storage**
Short Description		The Fleet Knowledge Base ensures that a synchronized knowledge among all the agents of the system. It allows them to take safe decisions.
Functional Requirements		• The KB shall ensure a consistent and reliable knowledge among the agents.
Data	**Input**	• Periodic update
	Output	• KB updated
	Configuration	• Network configuration
	Status	• Percentage of the storage space available/occupied • Number of other BBs that are using simultaneously the storage BB to read/write • Reading / Writing error
Services	**Import**	• Transaction requests: message from the components that require a validation from the main KB • Mission update: message from the GMC containing the progress on the mission (functionally speaking, it is a transaction request, see above for format) File transfer request: a component wants to send a file to the fleet
	Export	• Mission update: update on the mission by the other agents (consumed by the GMC, see above for format) • Operations update: update on the operations parameters (consumed by the GMC, see above for format) • GCS orders: high-level (human) orders (consumed by the GMC, see above for format)

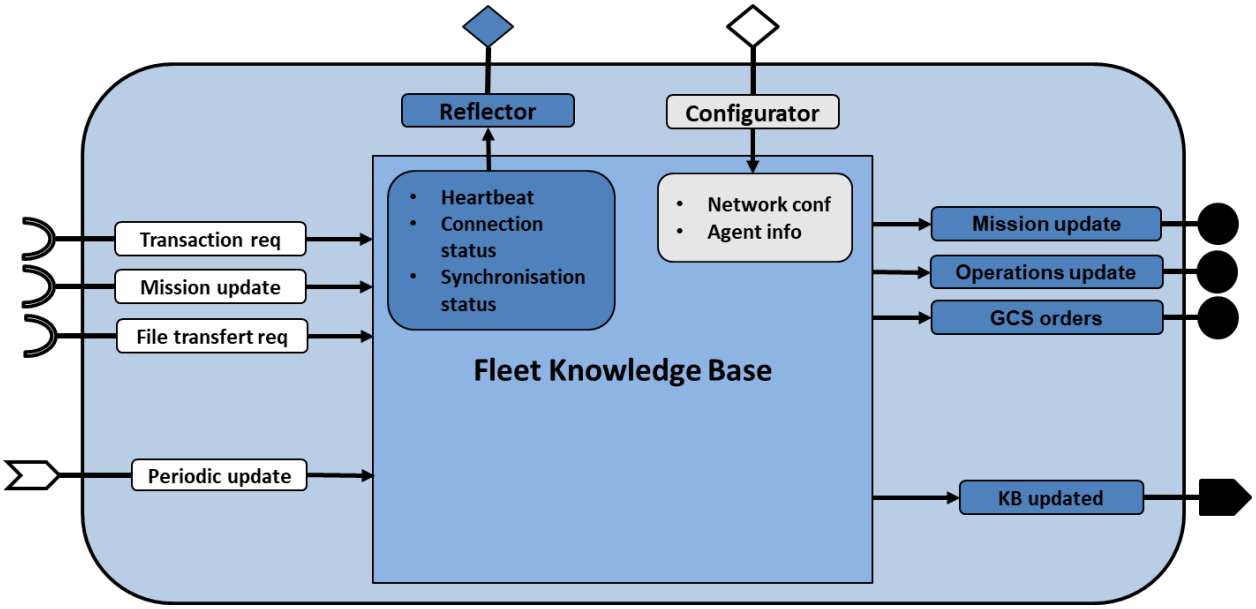

Figure 53: Fleet knowledge base

8.21 Data Analytics

The description of the data analytics block is in Figure 54 and Table 21.

Table 21: Description of the data analytics block

Block Name		Data Analytics
Short Description		This block is deputy to analyse data collected by onboard sensors in order to detected relevant information. According to the use of the drone, different type of data can be considered like images (multispectral), microphones, temperature, air quality, data from chemical sensors, etc.
Functional Requirements		• The component shall support different data analytics functions. • The component shall be able to extract high-level information from raw sensor data. • The component shall support predictive analysis functionality for the drone system and/or environment.
Data	Input	• Video flows • Audio flows • Onboard sensor data
	Output	• Analytics Metadata (e.g. JSON files which describe the detected information, alarms, notification., which can be stored on board or sent to the ground station)
	Configuration	• Data processing models • Algorithm parametrization settings
	Status	• Running / not running
Services	Import	• Storage • Onboard Sensors • Ground station communication
	Export	• Data processed and information detected analysing row data.

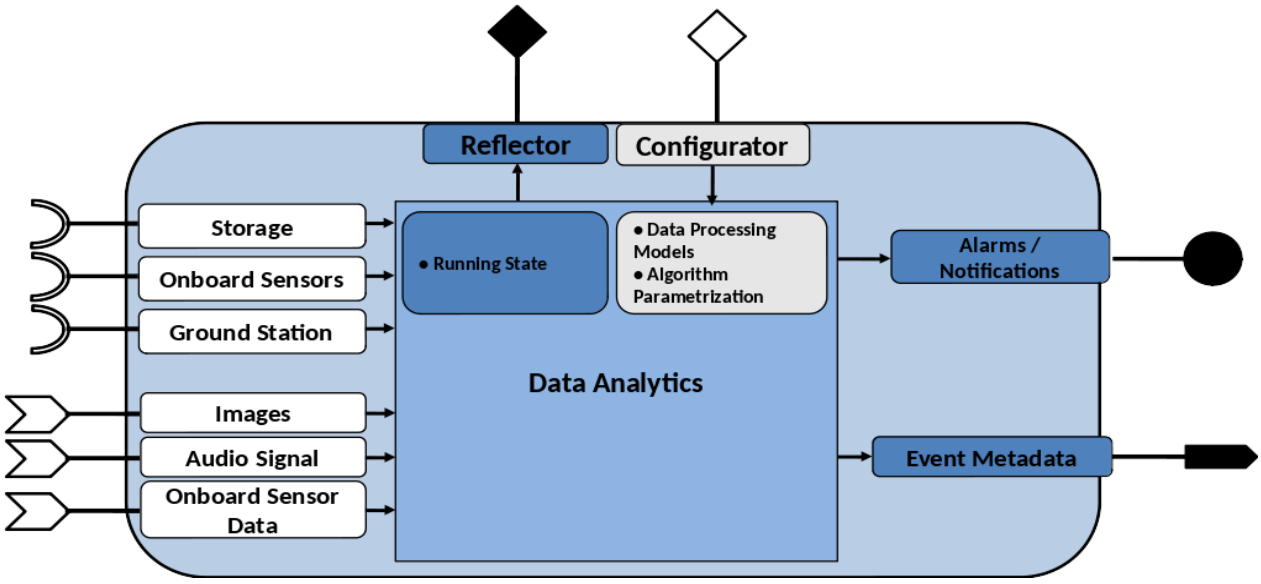

Figure 54: Data analytics block model

8.22 Video Analytics

This data analytics block is quite generic as it includes a quite large set of heterogeneous data processing applications. Thus, a specific video analytics block (as an example data analytics block) is presented in Table 22 and Figure 55.

Table 22: Description of the data (video) analytics block

Block Name		**Data (Video) Analytics**
Short Description		This block is deputy to analyse data (images) collected by onboard camera in order to detected relevant information. It implements different type of video content analysis algorithms: traditional video analysis algorithms are based on **background estimation** methods (separation of static background from moving targets, the *foreground*). Innovative video analytics systems are based on **Artificial Intelligence** approaches and in particular on **Deep learning**. After an appropriated training phase, a neural network is able to perform a great variety of tasks, including the detection of different targets based on their shape.
Functional Requirements		• The video analytics block shall detect targets (the type of target will be defined by the application) - **Detection** • (Optional) The video analytics block shall detect targets' position - **Localization**
Data	**Input**	• Video flows.
	Output	• Images • Video flows
	Configuration	• Characteristics of the video flows received as input from the video analytics SW. They can be frame rate (i.e. number of frames per second) image resolution, frame quality, etc. • Processing resources available to run SW (i.e. memory, CPU, GPU/FPGA).
	Status	• Running / not running

Services	**Import**	• Images collected by onboard camera.
	Export	• Detection and localization of targets. They can be different types of object, people, animals etc. according to the specific applicative requirements. Moreover, detected targets can be counted, their position can be compared with specific areas on interests (e.g. inside/outside a specific area).

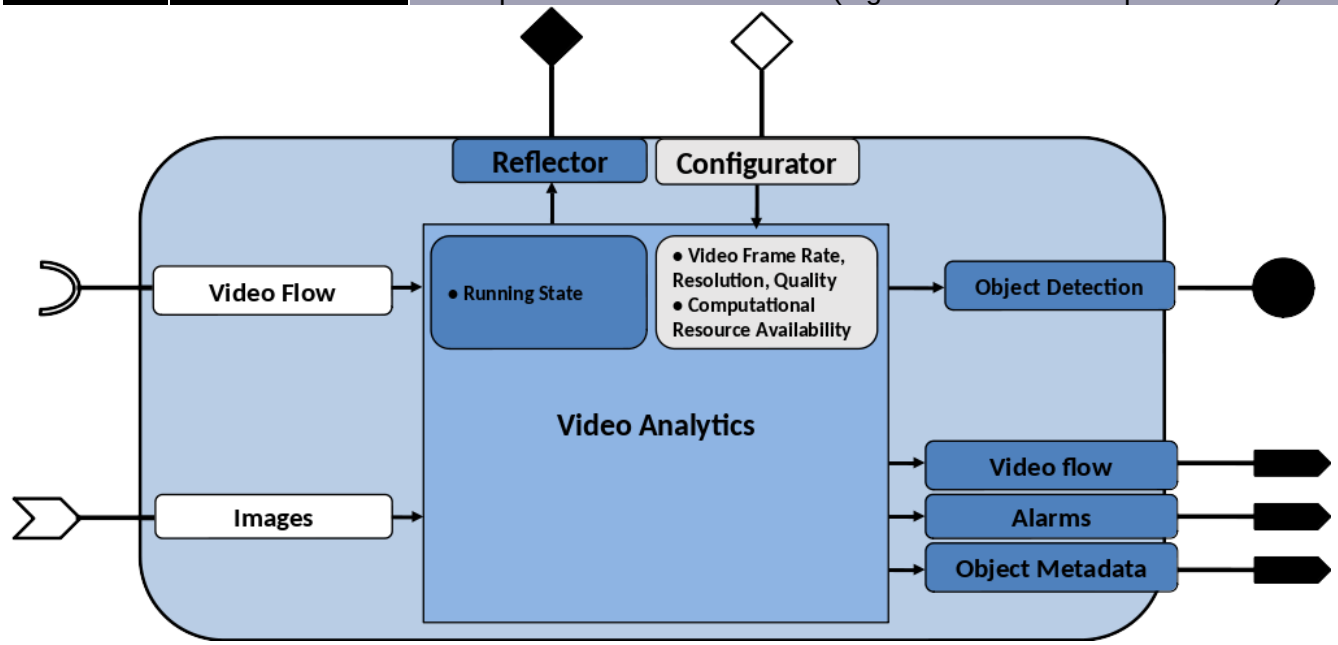

Figure 55: Data (video) analytics block model

9 Annex B: Interface (Data Flow) Structure

In this section, we list and describe the structure of the data that flows between the different building blocks of the drone architecture (described in annex A).

9.1 UAV Status

Attribute Name	Drone ID
Short Description	Unique identifier for each drone
Data Type	String
Usage	Trajectory Planning and Validation

Attribute Name	Pilot ID
Short Description	Unique identifier for each pilot
Data Type	String
Usage	Trajectory Planning and Validation

Attribute Name	Operator ID
Short Description	Unique identifier for the operator
Data Type	String
Usage	Trajectory Planning and Validation

Attribute Name	Flight Plan ID
Short Description	Unique identifier for each flight plan
Data Type	String
Usage	Trajectory Planning and Validation

Attribute Name	Authorization ID
Short Description	Unique identifier for each authorized flight plan
Data Type	String
Usage	Trajectory Planning and Validation

Attribute Name	GCS Authorized Person's Username
Short Description	This attribute corresponds to an authorized username that an authorized person in the GCS must have to access sensitive data in the drone storage.
Data Type	String
Usage	Storage, Mission Planning and Supervision

Attribute Name	GCS Authorized Person's password
Short Description	This attribute corresponds to an authorized password that an authorized person in the GCS must have to access sensitive data in the drone storage.
Data Type	String
Usage	Storage, Mission Planning and Supervision

Attribute Name	GCS Authorized Server's MAC Address
Short Description	This attribute corresponds with an authorized server/computer in the GCS from which an authorized person can access to sensitive data in the drone storage.
Data Type	String
Usage	Storage, Mission Planning and Supervision

Attribute Name	Drone Estimated State
Short Description	Drone current position and orientation state estimation
Data Type	Numeric
Usage	UAV Status, Trajectory Execution

Attribute Name	Flight Status
Short Description	Aggregation of flight descriptions, i.e. ,flight mode, fail-safe mode, stop motors status
Data Type	Structure of numeric and enumeration
Usage	Actuation, Command Mixing

Attribute Name	Power Info
Short Description	Structure of data containing: component ID, voltage, health status, temperature, current, remaining time, battery diagnosis
Data Type	Structure of numeric values
Usage	Diagnostics, Mission Supervision

Attribute Name	Position
Short Description	The position of the drone is described using X, Y, Z coordinates in the Cartesian space with respect to the global frame.
Data Type	Numeric (time stamped)
Usage	Position Control, Stabilizing, Trajectory Planning, Geo-fencing, Collision Avoidance

Attribute Name	Rotation
Short Description	Roll, pitch and yaw angles of the drone.
Data Type	Numeric (time stamped)
Usage	Position Control, Stabilizing, Trajectory Planning, Geo-Fencing, Collision Avoidance

Attribute Name	Velocity
Short Description	The speed vector of the drone.
Data Type	Numeric (time stamped)
Usage	Position Control, Stabilizing, Trajectory Planning, Geo-Fencing, Collision Avoidance

Attribute Name	Attitude
Short Description	Body angles in North, East, Down (NED) reference frame.
Data Type	Numeric (time stamped)
Usage	Position Control, Stabilizing, Trajectory Planning, Geo-Fencing, Collision Avoidance

Attribute Name	Trajectory Vector
Short Description	A trajectory vector for avoiding the obstacle. It considers the capabilities of the drone to manoeuvre. It contains timestamp to execute each vector.
Data Type	Numeric
Usage	Mission Planner, Trajectory Executor

Attribute Name	Position of Neighbouring Drones
Short Description	Position vectors of neighbouring drones in the formation.
Data Type	Numeric
Usage	Multi-drone Formation

Attribute Name	Telemetry Data
Short Description	Merged status of all blocks available in UAS.
Data Type	Structure
Usage	Communication, Telemetry, Data Management

Attribute Name	IMU data
Short Description	Inertial sensor data: angular rates, acceleration.
Data Type	Numeric
Usage	Geo-referencing, Collision Avoidance, Trajectory Execution

Attribute Name	Barometer data
Short Description	Pressure and temperature.
Data Type	Numeric
Usage	Geo-referencing

Attribute Name	Formation Vector
Short Description	Formation shapes can be defined through formation vector which contains position and velocity offset values with respect to the reference. Any constant and time-varying shape can be described using formation vector.
Data Type	Numeric
Usage	Formation Tracking

Attribute Name	Analytics Metadata
Short Description	JSON files which describe the detected information, alarms, notification. They can be stored on board or sent to the ground station.
Data Type	File
Usage	Data Analytics

Attribute Name	Building Block Status
Short Description	Array of numeric status of all building blocks of UAS.
Data Type	Array of numbers
Usage	Data Analytics

Attribute Name	Battery Charge Status
Short Description	Remaining battery charge percentage.
Data Type	Float
Usage	Mission Planning, Payload

Attribute Name	(Geo-referenced) Position
Short Description	Location of a specific body with regard to Earth, i.e., whose coordinates are referred to a geographic coordinate system.
Data Type	Numeric vector (3 coordinates)
Usage	Position Control, Payload

Attribute Name	(Geo-referenced) Velocity
Short Description	Velocity of a specific body with regard to Earth, i.e. taking a geographic coordinate system as a basis for the rotation.
Data Type	Numeric vector (3 coordinates)
Usage	Position Control

Attribute Name	(Geo-referenced) Attitude
Short Description	Pose of a specific body with regard to Earth, i.e. taking a geographic coordinate system as a basis for the rotation.
Data Type	Depends on the format: • Rotation matrix : 3x3 numeric matrix • Euler angles: 3x1 numeric vector • Quaternion: 4x1 numeric vector
Usage	Attitude control, Position Control, Also Payload purposes

Attribute Name	Power Type
Short Description	Enumeration of power type either battery powered or fuel powered: • 0 - Unknown • 1 - 4 Battery powered (LIPO,LIFE,LION,NIMH) • 5 - Fuel powered
Data Type	Numeric
Usage	Diagnostics

Attribute Name	Power Function
Short Description	Enumeration of battery function in system (system, avionics, payload, etc.)
Data Type	Numeric
Usage	Diagnostics

Attribute Name	Power Management Status
Short Description	Enumeration of the current building block status: • 0 – Building block is OK • 1-255 Various error states regarding estimation, power connection, etc.
Data Type	Numeric
Usage	Diagnostics

Attribute Name	Fault Management Status
Short Description	Building block status info
Data Type	Enumeration structure
Usage	Diagnostics

Attribute Name	Diagnostics Block Status
Short Description	Block status for fault management
Data Type	Enumeration structure
Usage	Diagnostics, Fault management

Attribute Name	State of Charge
Short Description	State of charge (SoC) is the level of charge of an electric battery relative to its capacity. The units of SoC are percentage points (0% = empty; 100% = full).
Data Type	Numeric
Usage	Mission Planning

9.2 UAV Surrounding

Attribute Name	Wind
Short Description	Vector representing the direction and speed of wind.
Data Type	Numeric
Usage	Mission Planning, Trajectory Execution

Attribute Name	Obstacle Map
Short Description	Array of obstacle representations in the environment, depending on the type of the sensor(s) used, different metadata can be available, i.e., type, size, location, pose, etc.
Data Type	Structure of numeric and enumeration
Usage	Obstacle Avoidance, Trajectory Planning

Attribute Name	GNSS Navigation Data
Short Description	GPS coordinate of the current location
Data Type	Numeric
Usage	Data Acquisition

Attribute Name	GNSS Observables
Short Description	Synchronization data coming from all the processing channels, e.g., traveling time of the signal to propagate from the phase center of the satellite antenna to the phase center of the receiver.
Data Type	Numeric
Usage	Mission Planning, Position Control

Attribute Name	GNSS Corrections (RTK)
Short Description	Data from single reference station or interpolated virtual station to providing real-time GNSS navigation data corrections.
Data Type	Numeric
Usage	Position Control

9.3 Flight Commands

Attribute Name	Goal Position
Short Description	Drone goal position coordinates.
Data Type	Numeric
Usage	Flight Controller

Attribute Name	Goal Velocity
Short Description	Velocity setpoint for controller.
Data Type	Numeric
Usage	Flight Controller

Attribute Name	Goal Roll
Short Description	Roll setpoint for controller.
Data Type	Numeric
Usage	Flight Controller

Attribute Name	Goal Pitch
Short Description	Pitch setpoint for controller.
Data Type	Numeric
Usage	Flight Controller

Attribute Name	Goal Yaw
Short Description	Yaw setpoint for controller.
Data Type	Numeric
Usage	Flight Controller

Attribute Name	Goal Attitude
Short Description	Attitude setpoint for controller.
Data Type	Numeric
Usage	Flight Controller

Attribute Name	Actuator setpoint
Short Description	Control values for the drone actuators.
Data Type	Numeric
Usage	Mixing, Attitude Control, Position Control

Attribute Name	Command Vector
Short Description	The command vector is the vector that groups the outputs of the different controllers and that is used to compute the actuators controls by the mixing block. The most common command vector is desired acceleration and moments around the body axis (roll, pitch, and yaw) and the thrust (the thrust axis depends on the type and structure of the drone).
Data Type	Numeric vector
Usage	Mixing, Attitude Control, Position Control

Attribute Name	Mission in Act
Short Description	Mission in act (desired X, Y and Z, roll, pitch, and yaw and all their velocities during the mission to be executed) over time.
Data Type	Array of numeric vectors
Usage	Mixing, Attitude Control, Position Control

9.4 Vehicle Dynamics

Attribute Name	Thrust
Short Description	Thrust is a reaction force described quantitatively by Newton's third law. When a system expels or accelerates mass in one direction, the accelerated mass will cause a force of equal magnitude but opposite direction, to be applied to that system. A fixed-wing aircraft generates forward thrust when air is pushed in the direction opposite to flight.
Data Type	Numeric
Usage	Stabilization, Attitude Control, Obstacle Avoidance, Geo-fence Preservation

Attribute Name	Yaw
Short Description	The yaw axis has its origin at the center of gravity and is directed towards the bottom of the UAV, perpendicular to the wings and to the fuselage reference line. Motion about this axis is called yaw. A positive yawing motion moves the aircraft to the right.
Data Type	Numeric
Usage	Stabilization, Attitude Control, Obstacle Avoidance, Geo-fence Preservation

Attribute Name	Pitch
Short Description	The pitch axis has its origin at the center of gravity and is directed to the right, parallel to a line drawn from wingtip to wingtip. Motion about this axis is called pitch. A positive pitching motion raises the head of the UAV and lowers the tail.
Data Type	Numeric
Usage	Stabilization, Attitude Control, Obstacle Avoidance, Geo-fence Preservation

Attribute Name	Roll
Short Description	The roll axis has its origin at the center of gravity and is directed forward, parallel to the fuselage reference line. Motion about this axis is called roll. An angular displacement about this axis is called bank. A positive rolling motion lifts the left wing and lowers the right wing.
Data Type	Numeric
Usage	Stabilization, Attitude Control, Obstacle Avoidance, Geo-fence Preservation

Attribute Name	Airspeed
Short Description	The airspeed is the relative speed of the airflow around to UAV body. It will generate drag and eventually lift forces depending on the type of drone. It can be used for stall protection on fixed-wing aircraft.
Data Type	Numeric
Usage	Position Control

Attribute Name	X, Y, Z, Roll, Pitch, Yaw
Short Description	These attributes identify the desired orientation and position, as well as their derivative values. The desired ones also indicate what you would like the drone to do and therefore can be useful for identifying both possible collisions and the length of the mission in question that remains.
Data Type	Numeric
Usage	Attitude Control, Position Control, Mission Planning

9.5 Plan

Attribute Name	Flight Plan
Short Description	Flight schedule, pattern, altitude, and image or video capture specifications, as well as any weather-related requirements.
Data Type	Numeric
Usage	Data Acquisition, Trajectory Planning and Execution

Attribute Name	Flight Plan Status
Short Description	Status of the flight plan (FP).
Data Type	Array of enumeration: • MANUAL: FP requires manual approval for a controller • PLANNED: FP is authorized • ACTIVE: FP is currently in flight • TERMINATED: FP is finished • REVOKED: FP was approved and later revoked • REMOVED: FP has been deleted • EDIT: FP saved as a draft but not submitted for approval • DENIED: FP has not been authorized • ABORT: FP aborted by pilot without finishing FP
Usage	Trajectory Planning and Validation

Attribute Name	Flight Priority
Short Description	Indicates whether flight plan is a priority or not.
Data Type	Array of enumeration: PRIORITARY, NORMAL
Usage	Trajectory Planning and Validation

Attribute Name	Mission category
Short Description	Category of the mission
Data Type	Array of enumeration: OPEN, EXPERIMENTAL, SPECIALIZED, CERTIFIED
Usage	Trajectory Planning and Validation

Attribute Name	Mission Type
Short Description	Type of the mission
Data Type	Array of enumeration: TRANSPORT, MAPPING, SURVEILLANCE, RECREATIONAL
Usage	Mission Planning

Attribute Name	Goal Trajectory Vector
Short Description	Trajectory vector to execute the planned mission or avoidance trajectory. Contains the timestamp to know the timing of each vector.
Data Type	Numeric
Usage	Flight Control

9.6 Payload

Attribute Name	System Time
Short Description	System time of the drone that can be used to match the system time of the payload, which in turn can be used to synchronize between GPS and acquired images.
Data Type	Numeric
Usage	Data Acquisition

Attribute Name	(Global) Time
Short Description	Time according to a global time scale (e.g., UTC, GPS, TAI).
Data Type	Depend on format and scale: Real Scalar or Vector: e.g. [YYYY, MM, DD, MIN, SEC, NS]
Usage	Overall UAV System

Attribute Name	Video Flows
Short Description	Video flows collected by onboard camera(s). They are characterized by different frame rates, resolutions, qualities etc.
Data Type	Sequence of images
Usage	Payload Management

Attribute Name	Audio Flows
Short Description	Audio signal collected by onboard microphones.
Data Type	Audio samples
Usage	Payload Management

Attribute Name	Onboard Sensor Data
Short Description	Data collected by onboard sensors(s). They are characterized by different characteristics according to the type of sensors used. They may be synchronous or asynchronous.
Data Type	Numeric
Usage	Data Analytics

Attribute Name	Start Capture Command
Short Description	A Boolean signal to indicate when to START capturing hyperspectral images.
Data Type	Boolean
Usage	Data Acquisition

Attribute Name	Stop Capture Command
Short Description	A Boolean signal to indicate when to STOP capturing hyperspectral images.
Data Type	Boolean
Usage	Data Acquisition

Attribute Name	Start Capture
Short Description	A Boolean signal to indicate when to START capturing hyperspectral images.
Data Type	Boolean
Usage	Data Acquisition

Attribute Name	Stop Capture
Short Description	A Boolean signal to indicate when to STOP capturing hyperspectral images.
Data Type	Boolean
Usage	Data Acquisition

Attribute Name	Exposure Time
Short Description	Exposure time for the camera.
Data Type	Numeric
Usage	Data capture

Attribute Name	Current Captured Frame
Short Description	The latest frame (an image) as acquired by the cameras in the payload.
Data Type	Numeric array
Usage	Data Analytics, Payload Control

Attribute Name	Current Classified (Processed) Frame
Short Description	The latest frame (sensor data, e.g. hyperspectral or RGB image, lidar scan) that is processed according to the data analytic algorithm used to process the hyperspectral image.
Data Type	Numeric array (Image)
Usage	Data Analytics, Payload Control

Attribute Name	Camera Temperature
Short Description	Shows the latest/current temperature of the camera in the payload.
Data Type	Numeric
Usage	Payload Control

9.7 Failures

Attribute Name	Power Fault
Short Description	Enumeration describing fault state of this building block: • 0 – Building block is OK • 1 - 255 – Multiple errors
Data Type	Numeric
Usage	Fault Management, Diagnostics

Attribute Name	Building Block Faults
Short Description	Status of all building blocks.
Data Type	Structure of enumeration
Usage	Fault Management, Diagnostics

Attribute Name	Battery Warning
Short Description	Warning sign if the battery is insufficient or at the limit to perform the mission.
Data Type	Boolean
Usage	Fault Management, Diagnostics, Mission planning

Attribute Name	Collisions Warning
Short Description	Warning sign if there is a risk of possible collisions in the future.
Data Type	Boolean
Usage	Collision avoidance

Attribute Name	Anomaly Warning
Short Description	Warning sign due to the anomaly of some signals at the hardware level.
Data Type	Boolean
Usage	Fault Management, Diagnostics

Attribute Name	Geo-fencing Violation Alert
Short Description	If the distance between the drone and restricted area becomes less than a certain threshold, a waning or alert notifies the operator that the drone is entering in a restricted, i.e., crossing the geo fence.
Data Type	Boolean
Usage	Geo-fencing, Collision Avoidance

Attribute Name	Failsafe Triggers
Short Description	Enumeration of failsafe triggers: • 0: Building Block is OK - No action. • 1 - 255: Various predefined failsafe triggers regarding the failsafe actions (return mode, mission hold, land mode, flight termination, parachute deploy, glide to safety, kill switch, etc.)
Data Type	Numeric
Usage	Flight Guidance

Attribute Name	Fault Messages
Short Description	Enumeration of fault messages: • 0: No action. • 1 - 255: Various predefined fault messages (power fault, flight guidance, etc.)
Data Type	Numeric
Usage	Fault Management

Attribute Name	Warning signal
Short Description	The output signal identifies normal operation of the drone or a possible anomaly that could occur. For each specific malfunction, a numerical reference code can be provided to uniquely identify the possible problem.
Data Type	Numeric
Usage	Mission Supervision

Attribute Name	Fault-failsafe Table
Short Description	Configuration of the connection among the faults and failsafe triggers.
Data Type	Numeric
Usage	Flight Guidance

Attribute Name	Error diagnosis
Short Description	System error diagnosis.
Data Type	Numeric
Usage	Diagnostics

9.8 Spatial information

Attribute Name	Anchor Positions and Ranges
Short Description	GNSS anchor positions and ranges between them.
Data Type	Array of numeric
Usage	Geo-referencing

Attribute Name	High Definition Map
Short Description	Highly accurate 3D map containing details not normally present on traditional maps.
Data Type	Map
Usage	Mission Planning

Attribute Name	Geo-fence Coordinates
Short Description	Coordinates and characterization of the virtual perimeter for a real-world geographic area.
Data Type	Array of vectors
Usage	Mission planning

Attribute Name	Distance to Geo-fence
Short Description	Distance to geo-fence.
Data Type	Numeric vector
Usage	Mission Planning, Flight Guidance

Attribute Name	Map Type
Short Description	Selection of the map type to created.
Data Type	Numeric
Usage	Mission Planning

9.9 Mission

Attribute Name	Mission Update
Short Description	Information on the mission: high-level orders (e.g. start/stop), addition/removal of an agent, modification of the objectives, progress on the mission, etc.
Data Type	Structure
Usage	Mission Controller

Attribute Name	Operations Update
Short Description	Information on the operations: addition/removal of a geo-fence, update on a dynamic-geo-fence position, update of the flight area, etc.
Data Type	Structure
Usage	Mission Controller

Attribute Name	GCS Orders
Short Description	High-level order from the GCS such as end of flight (nice stop of the mission), return to home (emergency stop of the mission), hold your position (emergency halt before further action), go to safety (emergency trajectory to go closer to the ground, freeing air space), emergency landing (last-resort emergency action).
Data Type	Structure
Usage	Mission Controller

Attribute Name	Component Requests
Short Description	Requests to trigger actions from the different components from the UAV. The format depends on the interface the components expose.
Data Type	Depends on the component in the system
Usage	Mission Controller

Attribute Name	Periodic Update
Short Description	Information on the agent status: current task, current position (telemetry), etc. This message serves as a heartbeat inside the fleet.
Data Type	Structure
Usage	Mission Controller

Attribute Name	Component Health
Short Description	Each component should provide a way for the mission controller to assess their health status. It can be a heartbeat or a more meaningful message (e.g., current activity). The mission controller will use this status to trigger corrective actions or emergency behaviour accordingly.
Data Type	Heartbeat or structure
Usage	Mission Controller

Attribute Name	Transaction Request
Short Description	A component of the system requires a message to be shared with the fleet, but this message must be validated by the main knowledge base (KB). For instance, the flight planner has computed a trajectory, and then it asks the main KB to validate that the air-space is still available since the beginning of the computation, if the segment is still valid, book it.
Data Type	Structure depending on the module
Usage	Mission Controller

Attribute Name	File Transfer Request
Short Description	A component needs to send a file to another agent (usually the GCS). For instance, the clearance module has detected an intruder, it sends the picture to warn the operators
Data Type	Download link to the agent file system
Usage	Mission Controller

Attribute Name	KB Updated
Short Description	Tell the components that an update occurred on the knowledge base (KB). For instance, when a new geo-fence is added, the flight planner must verify that all its trajectories are still valid.
Data Type	Structure
Usage	Mission Controller

9.10 Others

Attribute Name	Flight Log
Short Description	The flight log is a log file with recorded information about the UAV status and surroundings for each drone mission, it also includes all selected diagnostics data from other building blocks.
Data Type	XML or text file
Usage	Prognostics

Attribute Name	Hardware Signals
Short Description	Hardware signals can be very useful in predicting malfunctions, through the study of typical sequences of them.
Data Type	Numeric
Usage	Data Acquisition

Attribute Name	Operation Type
Short Description	Type of the operation
Data Type	Array of enumeration: VLOS, EVLOS, BVLOS, OTHER
Usage	Mission Planning

Attribute Name	Drone Physics Restriction
Short Description	Drone restrictions regarding its capabilities.
Data Type	Numeric
Usage	Trajectory Planning

Attribute Name	Validity/Quality Flags
Short Description	Information on the validity/quality of the output parameters.
Data Type	Practically, depends on the component. Some cases reflected: • Boolean • Array of numeric
Usage	Attitude Control, Position Control, Payload

Attribute Name	Coordinate Frame Type
Short Description	Coordinate frame configuration service.
Data Type	Numeric
Usage	UAV Status

Attribute Name	Localization Frame Type
Short Description	Localization frame configuration service.
Data Type	Numeric
Usage	UAV Status

Attribute Name	Sensor Configuration
Short Description	Sensor specific configuration, such as min/max angle, measurement distance or calibration.
Data Type	Numeric
Usage	Data Acquisition

Attribute Name	Sensor Selection
Short Description	Selection of the sensors to use.
Data Type	Numeric
Usage	Data Acquisition

Attribute Name	Data Fusing/Filter Algorithm Selection
Short Description	Selection of the algorithm
Data Type	Numeric
Usage	Data Analytics

Attribute Name	Trajectory Calculation Algorithm Selection
Short Description	Algorithm to use in trajectory calculation
Data Type	Numeric
Usage	Trajectory Planning

Attribute Name	Data Publish Rate
Short Description	Selection of rating when the map is published
Data Type	Numeric
Usage	Data Acquisition

Attribute Name	Drone Mech Setup
Short Description	Description of the placement of the sensors
Data Type	XML file
Usage	Data Acquisition

Attribute Name	Mechanical Capabilities
Short Description	Description of the mechanical capabilities of the drone
Data Type	Numeric
Usage	UAV Status

Attribute Name	Battery Parameters
Short Description	Some battery parameters that are used to determine how the state of charge discharge according to the torque requirements.
Data Type	Numeric
Usage	Attitude Control, Position Control