

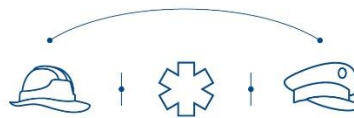
ASSISTANCE

Adapted situation awareneSS tools and tallored training curricula for increaSing capabiliTie and enhANcing the proteCtion of first respondErs



European Commission

Project co-funded by the European Union within the Horizon 2020 Programme



assistance

Project Ref. N°	ASSISTANCE H2020 - 832576
Start Date / Duration	May 1, 2019 (36 months)
Dissemination Level¹	PU (Public)
Author / Organisation	Viasat

Deliverable D4.6

Mission Planner

31/12/2020

¹ PU: Public; PP: Restricted to other programme participants (including the EC services); RE: Restricted to a group specified by the Consortium (including the EC services); CO: Confidential, only for members of the Consortium (including the EC services).

ASSISTANCE

Nowadays different first responder (FR) organizations cooperate together to face large and complex disasters that in some cases can be amplified due to new threats such as climate change in case of natural disasters (e.g. larger and more frequent floods and wild fires, etc.) or the increase of radicalization in case of man-made disasters (e.g. arsonists that burn European forests, terrorist attacks coordinated across multiple European cities).

The impact of large disasters like these could have disastrous consequences for the European Member States and affect social well-being on a global level. Each type of FR organization (e.g. medical emergency services, fire and rescue services, law enforcement teams, civil protection professionals, etc.) that mitigate these kinds of events are exposed to unexpected dangers and new threats that can severely affect their personal safety.

ASSISTANCE proposes a holistic solution that will adapt a well-tested situation awareness (SA) application as the core of a wider SA platform. The new ASSISTANCE platform is capable of offering different configuration modes for providing the tailored information needed by each FR organization while they work together to mitigate the disaster (e.g. real time video and resources location for firefighters, evacuation route status for emergency health services and so on).

With this solution ASSISTANCE will enhance the SA of the responding organisations during their mitigation activities through the integration of new paradigms, tools and technologies (e.g. drones/robots equipped with a range of sensors, robust communications capabilities, etc.) with the main objective of increasing both their protection and their efficiency.

ASSISTANCE will also improve the skills and capabilities of the FRs through the establishment of a European advanced training network that will provide tailored training based on new learning approaches (e.g. virtual, mixed and/or augmented reality) adapted to each type of FR organizational need and the possibility of sharing virtual training environments, exchanging experiences and actuation procedures.

ASSISTANCE is funded by the Horizon 2020 Programme of the European Commission, in the topic of Critical Infrastructure Protection, grant agreement 832576.

Disclaimer

This document contains material, which is the copyright of certain ASSISTANCE consortium parties, and may not be reproduced or copied without permission.

The information contained in this document is the proprietary confidential information of the ASSISTANCE consortium (including the Commission Services) and may not be disclosed except in accordance with the consortium agreement.

The commercial use of any information contained in this document may require a license from the proprietor of that information.

Neither the project consortium as a whole nor a certain party of the consortium warrant that the information contained in this document is capable of use, nor that use of the information is free from risk, and accepts no liability for loss or damage suffered by any person using this information.

The information in this document is subject to change without notice.

Executive Summary

This deliverable presents the software for gathering on field data and mission planning.

The purpose of this document is to describe the detailed Implementation of the modules involved in the task T4.6. These modules contribute to the two following modules: Mission Planner Management (MPM) and Optimized Mission Computation (OMC):

- Modules related to Mission Planner Management (MPM) module: they are related to the definition of the context (e.g. available drones/robots and related capacities, forbidden zones, working altitude...) and tasks of the mission (camera shoot, temperature measure) before resources plans computation by optimized mission software.
- Modules related to Optimized Mission Computation (OMC) module: OMC can distribute tasks between drones/robots, taking into account their specific capabilities (functions performed) and autonomy. It provides a mission plan for each of the selected drones/robots.

The resulting flight plans of each resources are then sent to the Ground Station for execution.

List of Authors

Organisation	Authors
VIASAT	Stéphane Michaud
THALES	Lionel Gayraud

Change control datasheet

Version	Changes	Chapters	Pages	Date
0.1	First draft	All	45	16/10/20
0.2	Reviewed by PIAP	All	45	11/12/20
0.3	Reviewed by CNBOP	All	45	21/01/21
0.4	Reviewed by CATEC	All	45	25/01/21

Content

Executive Summary	4
List of Authors	4
Change control datasheet	5
Content	6
List of Figures	7
Acronyms	8
1. Introduction	9
1.1. Purpose of the document	9
1.2. Scope of the document.....	9
2. Description of the task T4.6	9
3. Architecture and data model	10
3.1. T4.6 Architecture	11
3.2. Data model.....	12
3.2.1. Internal T4.6	12
3.2.2. In relation to SAS	17
3.3. Sequence diagrams	19
3.3.1. Sequence “Mission Preparation”	19
4. Modules	22
4.1. Mission Planner Management (MPM) component	22
4.1.1. Implementation	22
4.1.2. Data Model	30
4.1.3. Module Tests	30
4.2. Optimized Mission Computation (OMC) module.....	33
4.2.1. Implementation	33
4.2.2. Data Model.....	42
4.2.3. Module Tests	43
5. Use cases	49
5.1. Mission Preparation	49
6. Conclusion	51
7. Bibliography	52

List of Figures

Figure 1 – ASSISTANCE Architecture Schema	10
Figure 2 – T4.6 Architecture Schema	11
Figure 3 – MissionOrderRequest message	14
Figure 4 – MissionPreparationRequest message	15
Figure 5 – Mission message.....	16
Figure 6 – Mission message.....	19
Figure 7 – Mission Preparation sequence	21
Figure 8 – Target definition	23
Figure 9 – Cost Penalized areas for flying.....	23
Figure 10 – Ground obstacles for driving	24
Figure 11 – Mission Definition.....	24
Figure 12 – Overview resources locations.....	25
Figure 13 – Example list of resource involved in the mission	25
Figure 14 – List of actions to perform	26
Figure 15 – Sensor used.....	27
Figure 16 – Resource path	28
Figure 17 – GCS Situation awareness	29
Figure 18 – Waypoints vs clusters planning	37
Figure 19 – Task ordering with respect to context.....	38
Figure 20 – Collapse of Genoa bridge in 2018.....	39
Figure 21 – Landslide in Taiïwan in 2010.....	39
Figure 22 – Smooth trajectories vs Dijkstra like trajectories	41
Figure 23 – Trajectory pruning	41
Figure 24 – Example of KML scenario file.....	43
Figure 25 – Test scenario edited on Google Earth	44
Figure 26 – Comparison of scenarii 1, 2 and 3	45
Figure 27 – Results on scenario 1	46
Figure 28 – Results on scenario 2	47
Figure 29 – Results on scenario 3	48

Acronyms

ASSISTANCE	Adapted situation awareneSS tools and tallored training curricula for increaSing capabiliTie and enhANcing the proteCtion of first respondErs
PC	Project Coordinator
D#.#	Deliverable number #.# (D1.1 deliverable 1 of work package 1)
DoA	Description of Action of the project
EC	European Commission
EU	European Union
GA	Grant Agreement
H2020	Horizon 2020 Programme for Research and Innovation
IPR	Intellectual Property Rights
M#	#th month of the project (M1=May 2017)
WP	Work Package
IPR	Intellectual Property Rights
PSC	Project Steering Committee
PIC	Project Implementation Committee
PSB	Project Security Board
AB	Advisory Board
TL	Task Leader
WPL	Work Package Leader
CMS	Content Management System
OMC	Optimized Mission Computation
MPM	Mission Planner Management
SAS	Sensor Abstraction Service
UxVs	Unmanned Vehicle
MPM	Mission Preparation Request
GCS	Ground Control Station
MMM	Mission Manager Module
HMI	Human-Machine Interface
JSON	JavaScript Object Notation
KML	Keyhole Markup Language
GUI	Graphical User Interface

1. Introduction

This document represents the deliverable D4.6 “Mission Planner” related to the task T4.6 of the Assistance H2020 project. This document contains several information proving that all software modules of this task are well implemented and tested.

1.1. Purpose of the document

The purpose of this document is explaining deep in detail how “Mission Management Module” has been developed.

This document contains information to prove that all software modules of this task are well implemented and tested. They are now ready for the integration phase.

All modules are described with the corresponding architecture, data models and software implementation. All software modules are tested in three different steps:

- Component standalone.
- Component with one other component.
- Mission Management Module with the SAS.

These tests with results (obtained during testing session) are also described in this document.

1.2. Scope of the document

The document is organized as follow. Chapter 2 recalls the objectives of task 4.6 and places the Mission Management Module within the ASSISTANCE project. Chapter 3 presents the overall architecture of the Mission Management Module and associated data models. Chapter 4 provides a detailed description of the two modules making up the Mission Management Module, namely MPM (mission planner management) and OMC (optimized mission computation). Finally, chapter 5 presents the overall sequence diagram in which the mission management module is involved in the ASSISTANCE project.

2. Description of the task T4.6

Task 4.6 is to develop the Mission Management Module. The latter must allow an operator to easily plan a multi-drone mission according to objectives (temperature measurements or shots), constraints (autonomy and carrying of available drones/robots, geographical areas to be avoided), and given initial conditions (initial position of the drones/robots). This planning takes into account other factors, such as the load and autonomy of each drone/robot.

Mission Management Module is broken down into 2 layers:

- The Optimized Mission Computation module computes all paths.
- The Mission Planner Management module defines the context and organize the workflow in order to achieve the computation. After computation, it delivers the different mission plans (one per selected resource) to the Ground Station via SAS middleware.

UGV path will avoid obstacles based on georeferenced footprints. These footprints must be provided by end user using User interface to define the corresponding polygons.

3. Architecture and data model

This figure below depicts the main ASSISTANCE components and their interactions and information flows. The following section describes these high-level information flows in detail. Task 4.6 is to develop the Mission Management Module and its HMI, which is aimed to be used by Tool Operator.

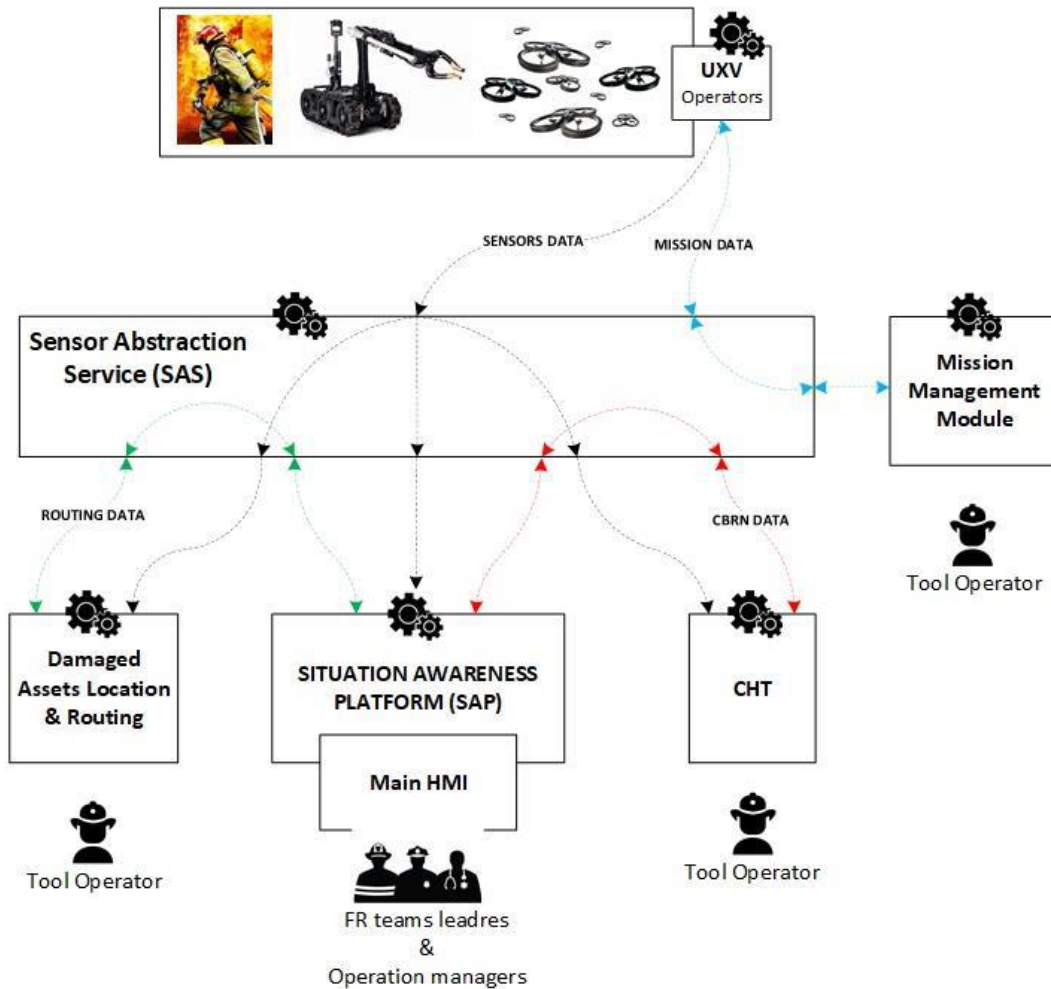


Figure 1 – ASSISTANCE Architecture Schema

3.1. T4.6 Architecture

The architecture of the task T4.6 and links between components can be summarized in the following diagram.

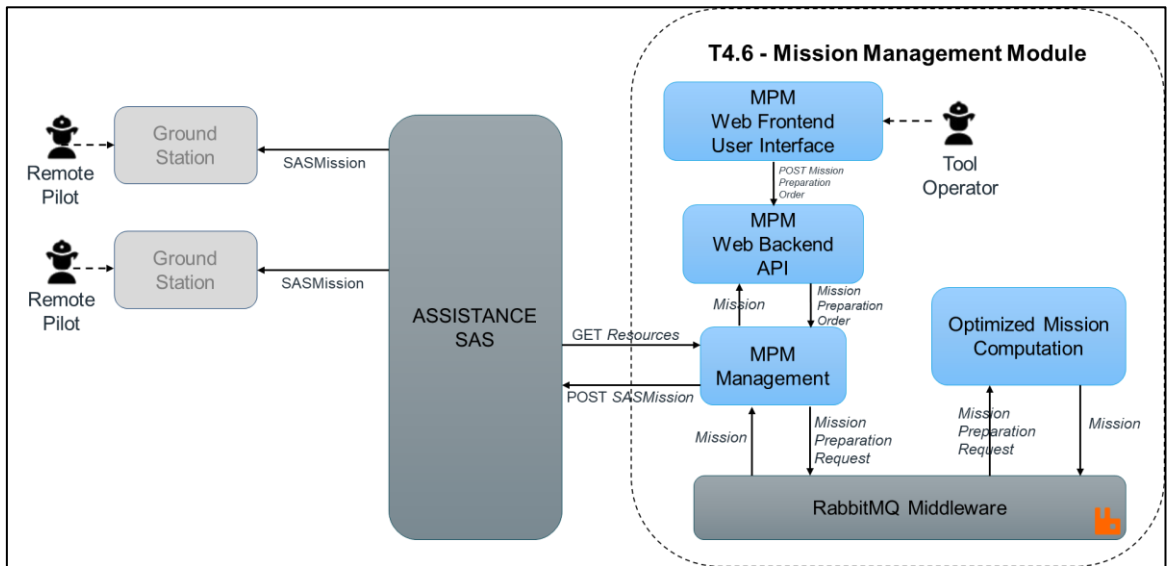


Figure 2 – T4.6 Architecture Schema

In this diagram, plain links represents communication between component through messages. Dash links represents communications between user and components.

The detailed message used in the context of the task T4.6 will be described in the following chapters.

The main goal of this architecture is to avoid direct links between components (inside or outside T4.6). We are using a middleware in order to clarify communication and to be able to improve the number of components that can be used.

MPM is managing the workflow to compute the mission of UxVs. For this case, MPM collects information about the context and send the *MissionPreparationRequest* to the middleware. OMC will use this information to compute the whole mission. It is an asynchronous call because OMC response time is not fixed (it depends on the number of tasks, the number of resources etc.). Once the mission is computed, OMC publish a *Mission* message on the middleware which contains a detailed description of the mission that each UxV must perform (waypoints and sensor actions the Pilot/UxV will have to perform).

MPM will then build a dedicated message *SASMission* in order to reach each involved GCS/UxV. This message is sent using Web API REST provided by SAS.

3.2. Data model

3.2.1. Internal T4.6

As described above, inside T4.6, messages are exchanged between components by using a middleware (based on RabbitMQ). RabbitMQ is an open source message-oriented middleware that allows software to exchange messages without any low-level network considerations. This middleware is available for any type of software development language. It is also an opportunity to validate this principle.

Each message has been described using YAML format (see the detailed specification here: <https://en.wikipedia.org/wiki/YAML>). This format allows developers to define the message in a human readable language.

The content of messages transmitted is using JSON format (see the detailed specification <https://en.wikipedia.org/wiki/JSON>). At Viasat, we have developed a tool that allows us to transform the YAML message into a C# class. In our component, we are basically using C# class and a serialization/deserialization library. With this compilation chain, any modification/improvement is easily considered at the development level.

For the task T4.6, we are using the following structures:

- Area.schema.yaml
- Drone.schema.yaml
- FlightAction.schema.yaml
- FlightActionGoto.schema.yaml
- FlightActionLand.schema.yaml
- FlightActionPrepare.schema.yaml
- FlightActionType.schema.yaml
- FlightActionWait.schema.yaml
- GeographicLocation.schema.yaml
- Mission.schema.yaml
- MissionOrderRequest.schema.yaml
- MissionPreparationRequest.schema.yaml
- Plan.schema.yaml
- Sensor.schema.yaml
- SensorAction.schema.yaml
- SensorActionCameraShoot.schema.yaml
- SensorActionParameter.schema.yaml
- SensorActionTemperatureRead.schema.yaml
- SensorActionType.schema.yaml
- SensorCameraIR.schema.yaml
- SensorCameraVisible.schema.yaml

D4.6 Mission Planner

- SensorTemperature.schema.yaml
- TargetType.schema.yaml

The table below gives a short description of each structure

Message Name	Description
Area	Define the area of the mission.
Drone	Define the characteristic of available resource
FlightAction	It is the base structure for all defined flight action
FlightActionGoto	Describe the action 'Goto waypoint'
FlightActionLand	Describe the action 'Land'
FlightActionPrepare	Describe the action 'Prepare' before takeoff.
FlightActionType	Enumeration which define all type of action
FlightActionWait	Describe the action 'Wait'
GeographicLocation	Describe a geographic location (latitude, longitude, heightAboveEllipsoid)
Mission	Define the mission
MissionOrderRequest	Define the request for a mission
MissionPreparationRequest	Define the Mission preparation information
Plan	Define a Plan for a resource
Sensor	Define the characteristic of a sensor (T4.6 need)
SensorAction	Base structure for sensor action
SensorActionCameraShoot	Define the action 'Camera shoot'
SensorActionParameter	Define the parameter for an action
SensorActionTemperatureRead	Define the action 'Temperature read'
SensorActionType	Enumeration who define the list of sensor action
SensorCameraIR	Define the characteristics for an IR Camera
SensorCameraVisible	Define the characteristics for a Visible camera
SensorTemperature	Define the characteristics for a temperature sensor
TargetType	Define the type of the target requested for the mission

Table 1 Summary of internal structures

The diagram below describes the link (and their type) between all structures. The top-level structure defines the message that is exchanged between components.

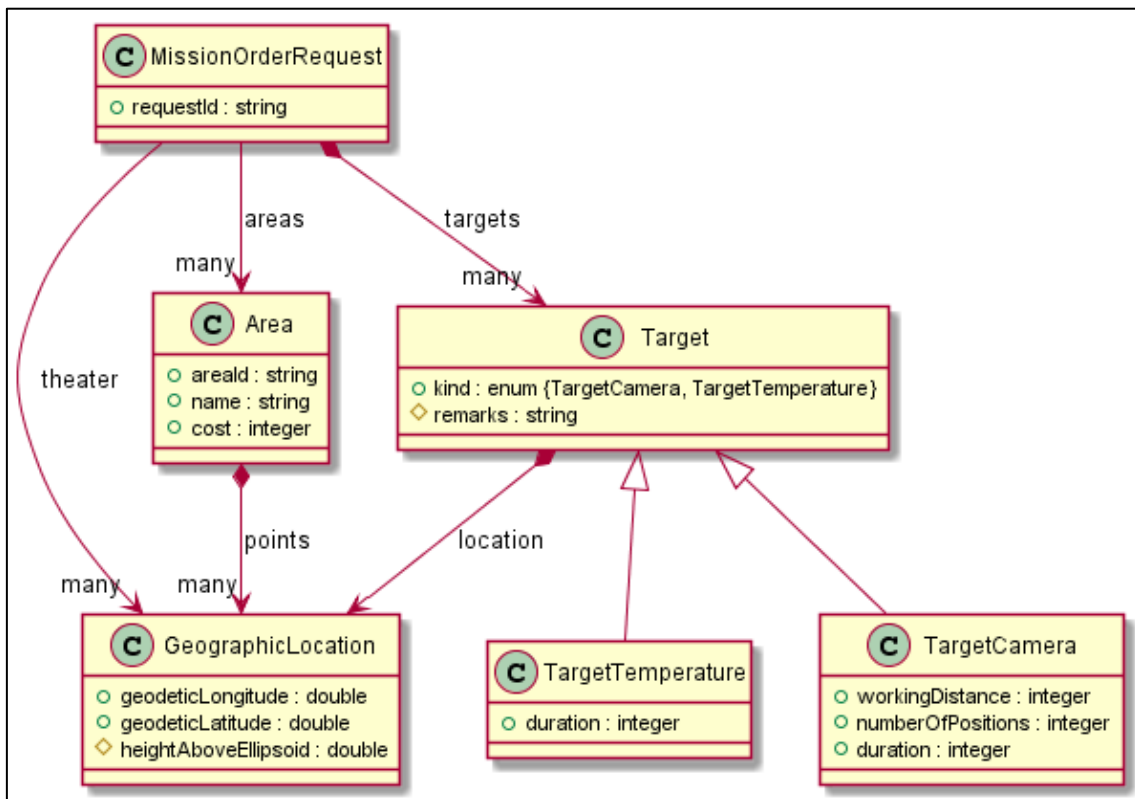


Figure 3 – MissionOrderRequest message

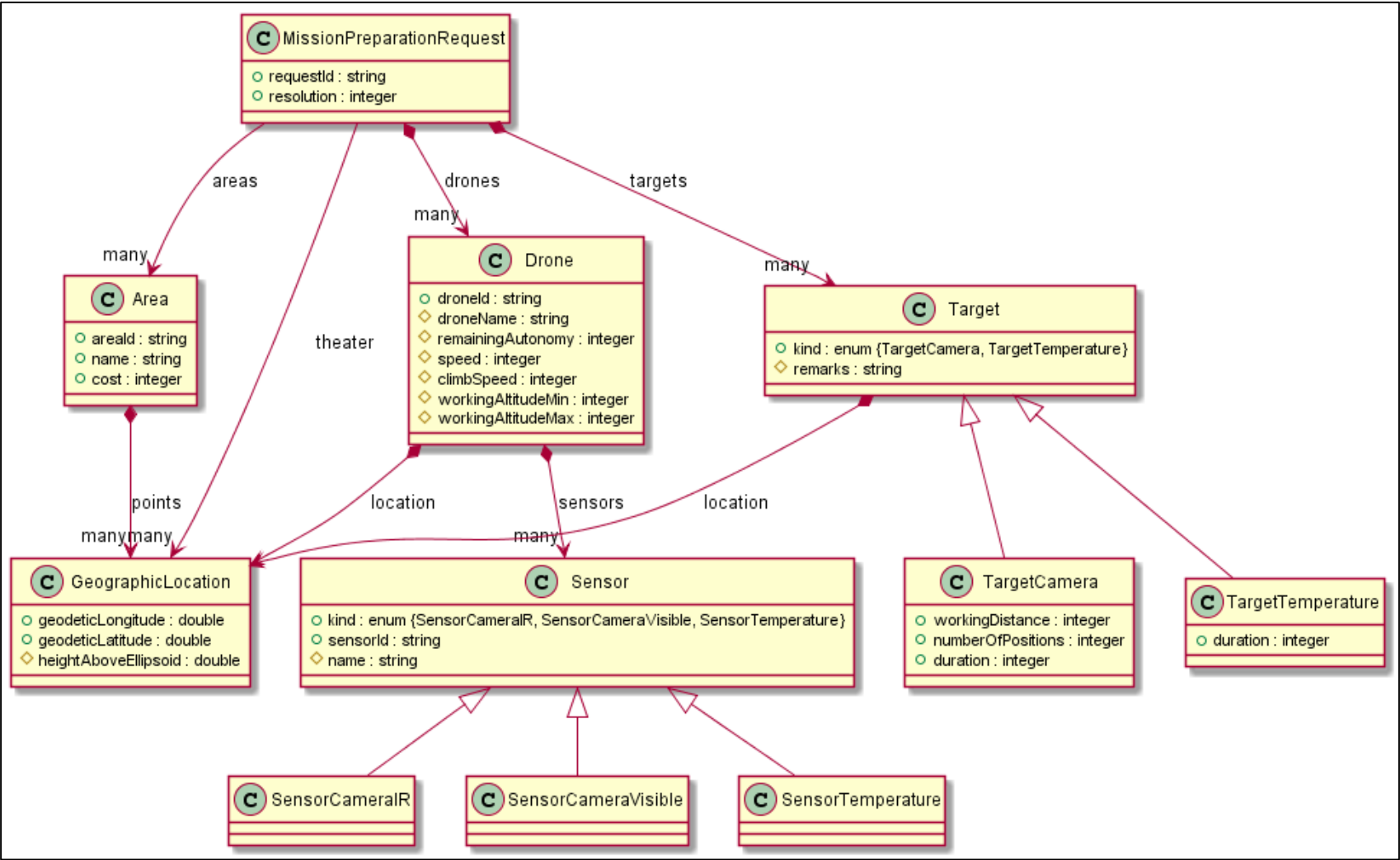


Figure 4 – MissionPreparationRequest message

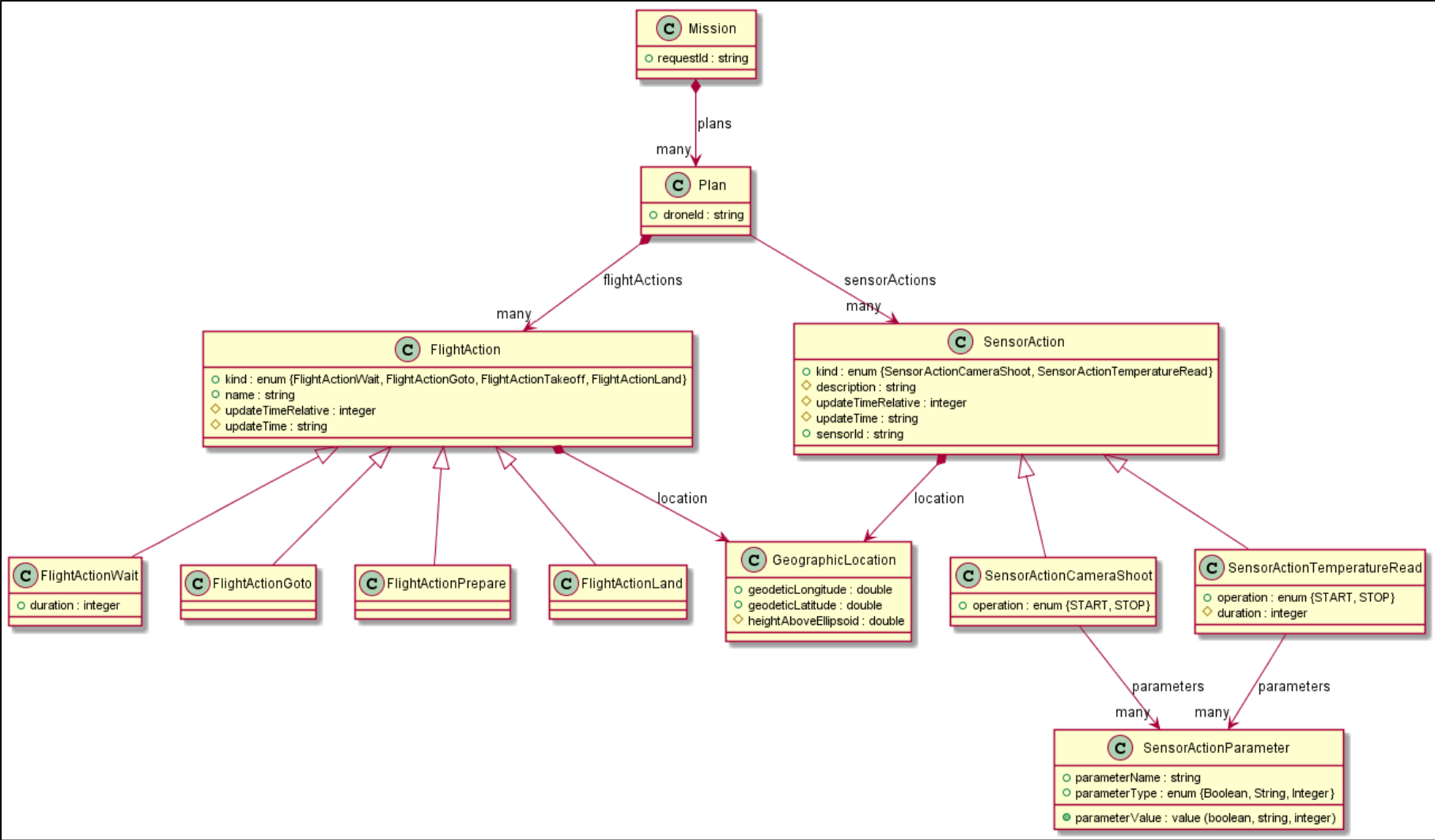


Figure 5 – Mission message

3.2.2. In relation to SAS

As it is described in the diagram T4.6 Architecture, Mission Management Module has links with SAS (Sensor Abstraction Service). This chapter describes messages used to communicate with SAS. Document D3.1 and D3.2 describes the list of messages that SAS is able to manage.

In order to be compliant with the T4.6 principles, each message has been translated into YAML format.

T4.6 (MPM) used this list of messages:

- SASContext.schema.yaml
- SASGeometry.schema.yaml
- SASMapping.schema.yaml
- SASMission.schema.yaml
- SASPointOfInterest.schema.yaml
- SASPosition.schema.yaml
- SASReply.schema.yaml
- SASResource.schema.yaml
- SASResourceSubType.schema.yaml
- SASResourceType.schema.yaml
- SASRoute.schema.yaml
- SASSensor.schema.yaml
- SASWaypoint.schema.yaml
- SASZone.schema.yaml

The table below gives a short description of each structure

Message Name	Description
SASContext	Define the mission context
SASGeometry	Define the GeoJSON geometry
SASMission	Define the structure for a mission (sent to the GCS)
SASReply	Define the structure received by each call to the REST API
SASResource	Define a resource (from ASSISTANCE point of view)
SASResourceSubType	Enum that define the subtype of a resource

SASResourceType	Enum that define the type of a resource
SASSensor	Define a sensor
SASWaypoint	Define a waypoint (where a resource has to go)
SASZone	Define an area in term of GeoJSON

- Table 2 Summary of the SAS structures

The diagram below describes the link (and their type) between all structures. The top-level structure defines messages that are exchanged with SAS.

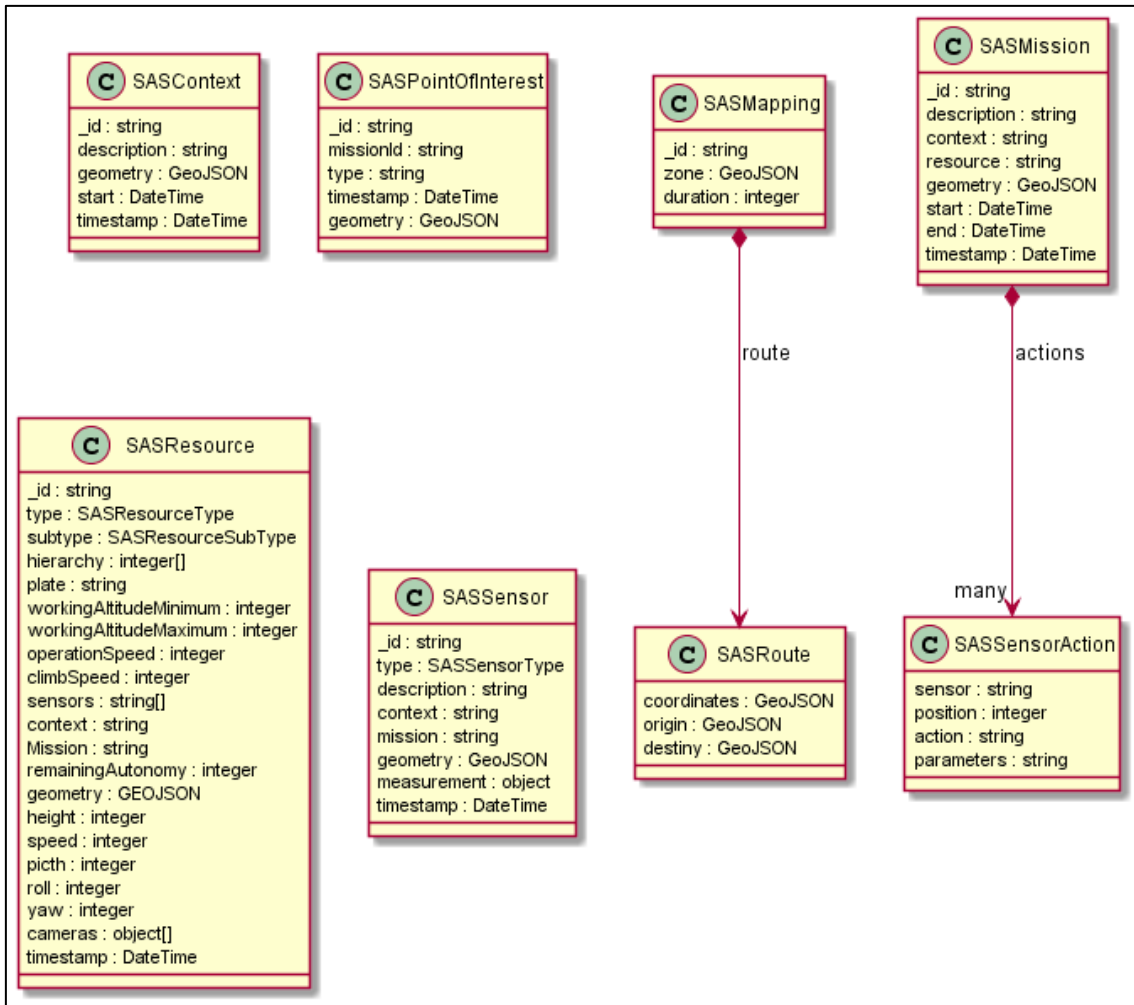


Figure 6 – Mission message

3.3. Sequence diagrams

This chapter describes the sequence diagram for this task.

3.3.1. Sequence “Mission Preparation”

This chapter describes the sequence diagram implemented to compute the flight path associated to the sensor configuration in order to execute a mission. The sequence diagram uses the messages exchanged between components as well. It allows users to prepare the mission for resources (Unmanned type).

The sequence diagram below describes the whole process from the Tool Operator to the Remote Pilot:

- **Block 1 (Prepare Mission Context):** Tool Operator prepares the context of the mission by defining the mission area, targets and the cost penalized area. Forbidden areas (with penalization level) will be used by OMC to compute the best path to follow.
- **Block 2 (Create Mission):** Remote Pilot asks for the mission to be created. It is basically the step where action is sent from the Frontend to the MPM Management.

D4.6 Mission Planner

- **Block 3 (Resource Management):** MPM asks SAS to inventory all available UxVs for this new mission, and for each UxV, the list of sensors hosted by this resource. MPM Management builds and sends a *MissionPreparationRequest* message.
- **Block 4 (Computation):** OMC computes the mission and produces a flight plan per selected UxV. After computation, OMC sends back the message *Mission* containing all actions (transit to waypoint and sensor actions).
- **Block 5 (Deployment):** deployment of this new mission to the SAS via Web REST API. MPM Management posts the new mission to the SAS. MPM Management notifies Tool Operator via MPM Frontend.
- **Block 6 (group):** taking into account by Remote Pilot of the new mission via SAS connection to its Ground Control Station

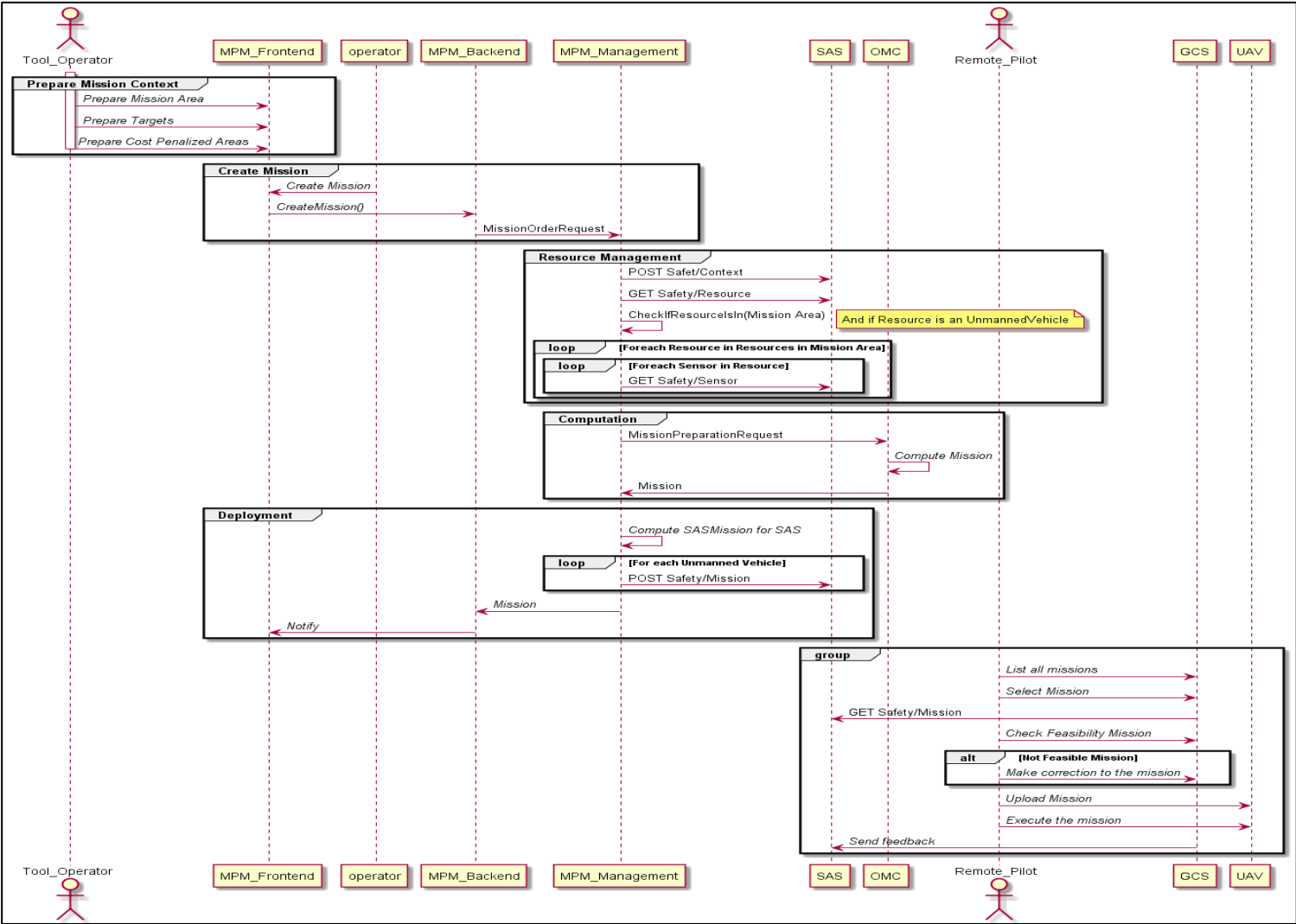


Figure 7 – Mission Preparation sequence

4. Modules

The following chapters describe the detailed architecture, implementation data model and unit tests performed for modules MPM (Mission planner management) and OMC (Optimized Mission Computation) in order to achieve the final T4.6.

4.1. Mission Planner Management (MPM) component

The objective of the whole component MPM is to organize the computation of Mission Preparation by following different steps that will allow to send a detailed plan to each unmanned resource via the GCS. This component allows the user to define the context of the mission by defining:

- Mission area.
- Target point (temperature/camera, working distance, duration of observation).
- Cost penalized areas (with associated magnitude).

4.1.1. Implementation

The implementation of this task is composed by the frontend user interface, the backend process and the workflow management. This implementation is based on asynchronous call that allows this task to process several missions at the same time (scalability). One of the major goals for us is the ability to build a mission with many UxVs.

Viasat implements a user interface to define the Mission Context. The following paragraph shows several screenshots who will explain the process to define the *MissionContext*.

The following pictures show how user can define threats for this mission. It is possible to define:

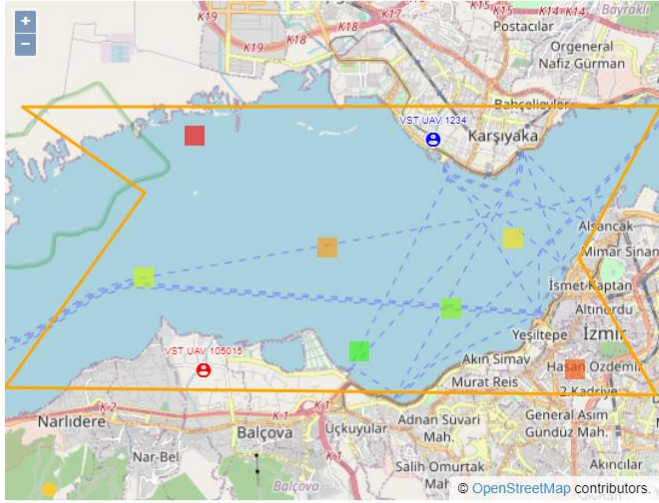
- Coloured squares which represent the target
- Field "Altitude target" defines the altitude of the target
- Field "Duration for observation" defines the duration in seconds to observe the target
- Field "Working distance" defines the distance to the target

Remark: colour of target can be modified by the user.

D4.6 Mission Planner

Edit an existing mission order

General
Mission area
Forbidden areas
Targets



Details

Observation

■ Camera Target1 ✎ ✖

Working distance	600	✎
Altitude	200	✎
Duration for observation	180	✎

■ Temperature Target2 ✎ ✖

Working distance		✎
Altitude	300	✎
Duration for observation	90	✎

■ Temperature Target4 ✎ ✖

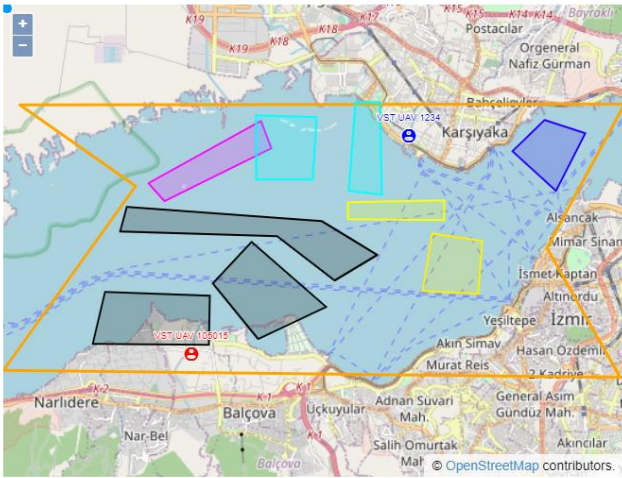
Update
Can

Figure 8 – Target definition

This user interface describes how to define the different cost penalized areas of the mission. Each polygon defines an area, and then the users associate a cost to each of these areas by selecting a value in the combo box (cost penalized level).

Edit an existing mission order

General
Mission area
Forbidden areas
Targets



Details

Type	Name	Actions
DoNotFly-100%	DNF100-3	✎ ✖
DoNotFly-80%	DNF80-2	✎ ✖
DoNotFly-60%	DNF60-1	✎ ✖
DoNotFly-20%	DNF20-1	✎ ✖
DoNotFly-80%	DNF80-1	✎ ✖
DoNotFly-100%	DNF100-1	✎ ✖
DoNotFly-100%	DNF100-2	✎ ✖
DoNotFly-40%	DNF40-1	✎ ✖
DoNotFly-80%	DNF80-1	✎ ✖

Update
Cancel

Figure 9 – Cost Penalized areas for flying

D4.6 Mission Planner

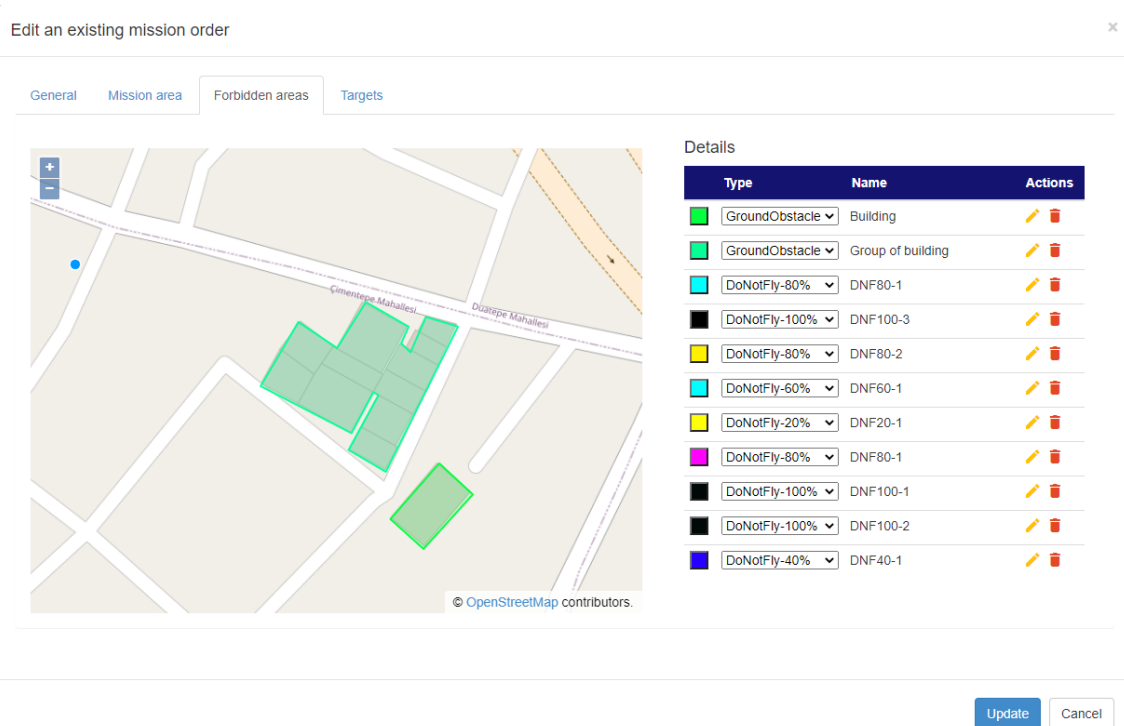


Figure 10 – Ground obstacles for driving

And finally, the “Mission area” tab summarizes the whole information in one screen. All previous information defined by user are shown in this dialog box. In this user interface, users can create/update the whole mission area. It allows the component MPM to discover resources that can be involved in the mission. In this picture, resources (defined by their type and subtype) are displayed with several icons.



Figure 11 – Mission Definition

D4.6 Mission Planner

The following pictures shows all resources available on SAS and their location.

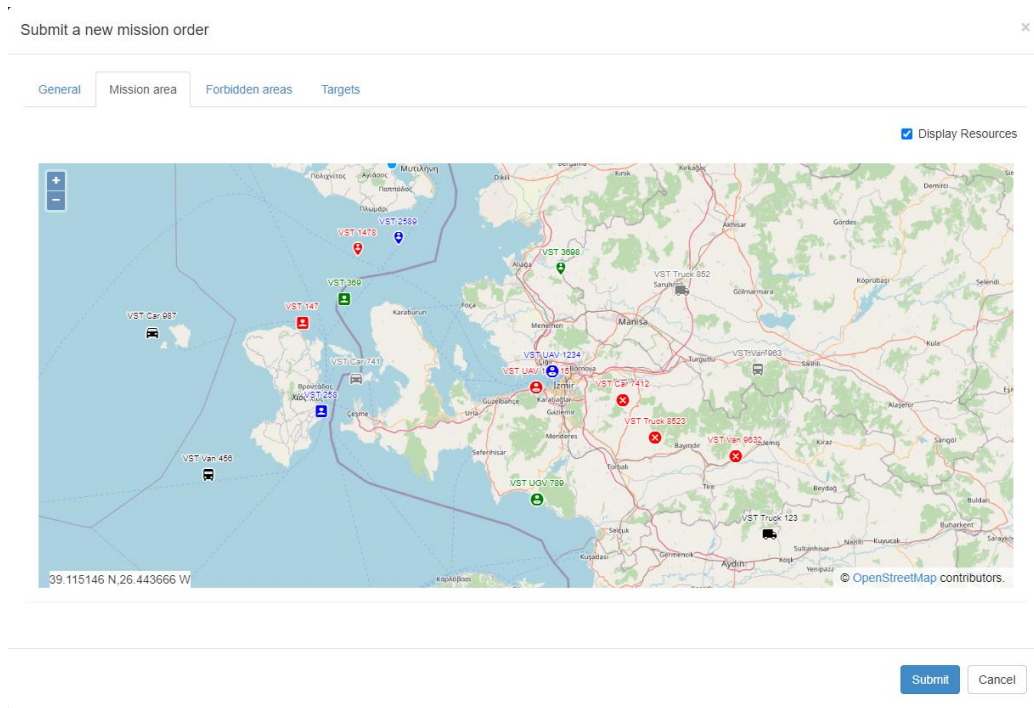


Figure 12 – Overview resources locations

And finally, the following picture shows the computation of the list of action after the computation by OMC. In this picture, the user can see the list of resource involved in this mission (after computation). It is the most relevant information coming from this part of the screen. Each resource is displayed with a dedicated colour, which is also used to display the associated path.

29	ASSISTANCE.c532c468-4894-4bf3-bcb9-da73c732042a.9-MM9-7	Assistance Test 2	Test 2	AAHD	10/1/2020	2	Details Resend
30	ASSISTANCE.30101022-9278-45b0-a5f9-a4b221a58167.9-MM9-9	Assistance Test 2	Test 2	AAHD	10/1/2020	2	Details Resend

Multi-Mission #30 - ASSISTANCE.30101022-9278-45b0-a5f9-a4b221a58167.9-MM9-9

Title	Description			
Assistance Test 2	Multi-Missions - Test 2			
Company name	Mission date	Missions	Mission UAV	Mission Pilot
AAHD	<ul style="list-style-type: none"> 10/30/2020 10/30/2020 	<ul style="list-style-type: none"> ASSISTANCE.30101022-9278-45b0-a5f9-a4b221a58167.9-105015 ASSISTANCE.30101022-9278-45b0-a5f9-a4b221a58167.9-1234 	<ul style="list-style-type: none"> 105015 1234 	<ul style="list-style-type: none"> Sample Pilot User Sample Pilot User

ID	Alarm type	Timestamp	Message	Position
Mission area map				

Figure 13 – Example list of resource involved in the mission

D4.6 Mission Planner

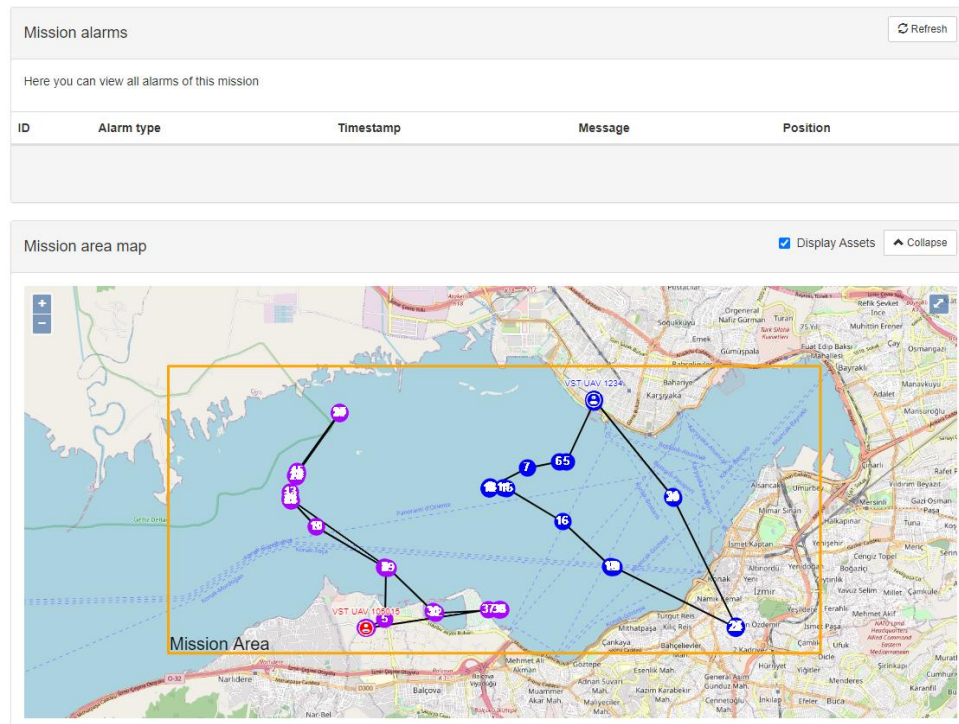


Figure 14 – List of actions to perform

A list of actions is represented by a path:

- Goto action is basically a straight line
- Wait action is just a point at the previous location (that is why sometimes, the numbering seems to skip a number).
- Prepare action is basically the first point.
- Land action is basically the last point.

Each circle shows waypoint to be performed. The number indicates the order to follow.

MPM sends also *SASMission* message to SAS. The next picture shows the Viasat Ground Control Station that allows Remote Pilot to control camera hosted in our Ground Vehicle. This GCS also allows us to validate the whole process of Mission Preparation as an end to end process.

Figures below shows the *SASMission* received by the Ground Control Station. Remote Pilot can see the detailed path (with all waypoints).

The figure below shows the selected sensor used for this mission.

D4.6 Mission Planner

Mission Management Tool (v2.0.0.0)

Missions **Assistance Test 2 (Mission ID: 58)**

System monitor

Pre-mission phase
 Mission phase
 Post-mission phase

Preparation
 Execution status
 Upload

Mission
Id: 58
Pilot Id: 2
Pilot name: Sample Pilot User
Date: 30/10/2020
Status: Preparation
Description: [{"Identifier":"ASSISTANCE.30101022-9a5f9-a4b221a58167.9","VersionNumber":1,"\":100-10.12.1234","\":1,"FlightActions":[{"kind":"FlightActionPrepare","name":"TakingOff","location":{"geodeticLongitude":27.09456,"geodeticLatitude":38.457652,"heightAboveEllipsoid":0},"updateTimeRelative":0,"updateTime":{"2020-10-30T16:07:14.3967623+01:00"},"kind":"FlightActionGoto","name":"waypoint_1","location":{"geodeticLongitude":27.09456,"geodeticLatitude":38.457652,"heightAboveEllipsoid":300},"updateTimeRelative":10,"updateTime":{"2020-10-30T16:07:24.3967623+01:00"},"kind":"FlightActionGoto","name":"waypoint_2","location":{"geodeticLongitude":27.094642608167383,"geodeticLatitude":38.457368767271426,"heightAboveEllipsoid":300},"updateTimeRelative":70,"updateTime":{"2020-10-30T16:08:24.3967623+01:00"},"kind":"FlightActionGoto","name":"waypoint_3","location":{"geodeticLongitude":27.094642608238637,"geodeticLatitude":38.457028261637319,"heightAboveEllipsoid":300},"updateTimeRelative":76,"updateTime":{"2020-10-30T16:08:30.3967623+01:00"},"kind":"FlightActionGoto","name":"waypoint_4","location":{"geodeticLongitude":27.086690978142943,"geodeticLatitude":38.443665396513367,"heightAboveEllipsoid":300},"updateTimeRelative":84,"updateTime":}]}]

Yellowscan-LIDAR-1
 Light Detection and Ranging

FLIR-A650-75000434
 Standard IR camera

GPS-ublox-1
 Global positioning system provided from UBLOX

GT6600_02-2631D-07127
 Allied vision camera (GT6600)

Axis QE6215 LE (VPN)
 AXIS PTZ Camera

FLIR IR Camera
 AXIS PTZ Camera

Axis QE6215 LE (Direct)
 AXIS PTZ Camera

Fixed Camera
 Only Video stream over RTSP

Autopilot
 Autopilot simulator

VSTSensorCamera1
 Camera for Assistance Project

VSTSensorCamera2
 Camera for Assistance Project

VSTSensorTemperature1
 Temperature sensor for Assistance Project

VSTSensorTemperature2
 Temperature sensor for Assistance Project

Sensors Algorithms Trigger points Logic rules

SensorInstanceDto
 Search

Api
 Api name
 SensorSensorApiF...

Overview
 CanControl True
 Description Camera for Assistance Project
 Label VSTSensorCamera1
 SensorDetectionTy...
 Type

Abort

Figure 15 – Sensor used

D4.6 Mission Planner

The next figure shows the path and the trigger point where the sensor has to execute a sensor action. Stars represents targets.

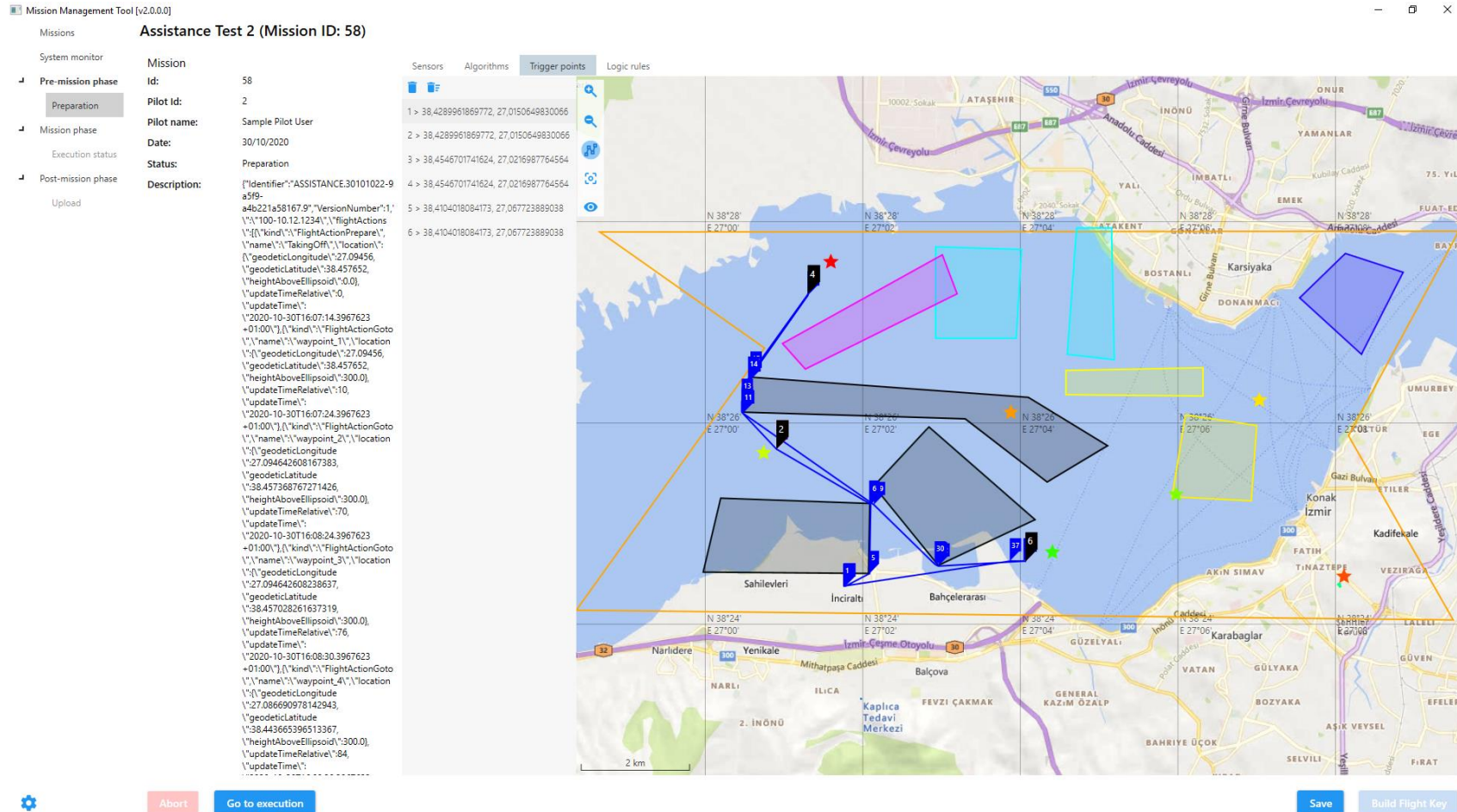


Figure 16 – Resource path

D4.6 Mission Planner

And finally, the figure below shows the screen used by the remote pilot during the mission.

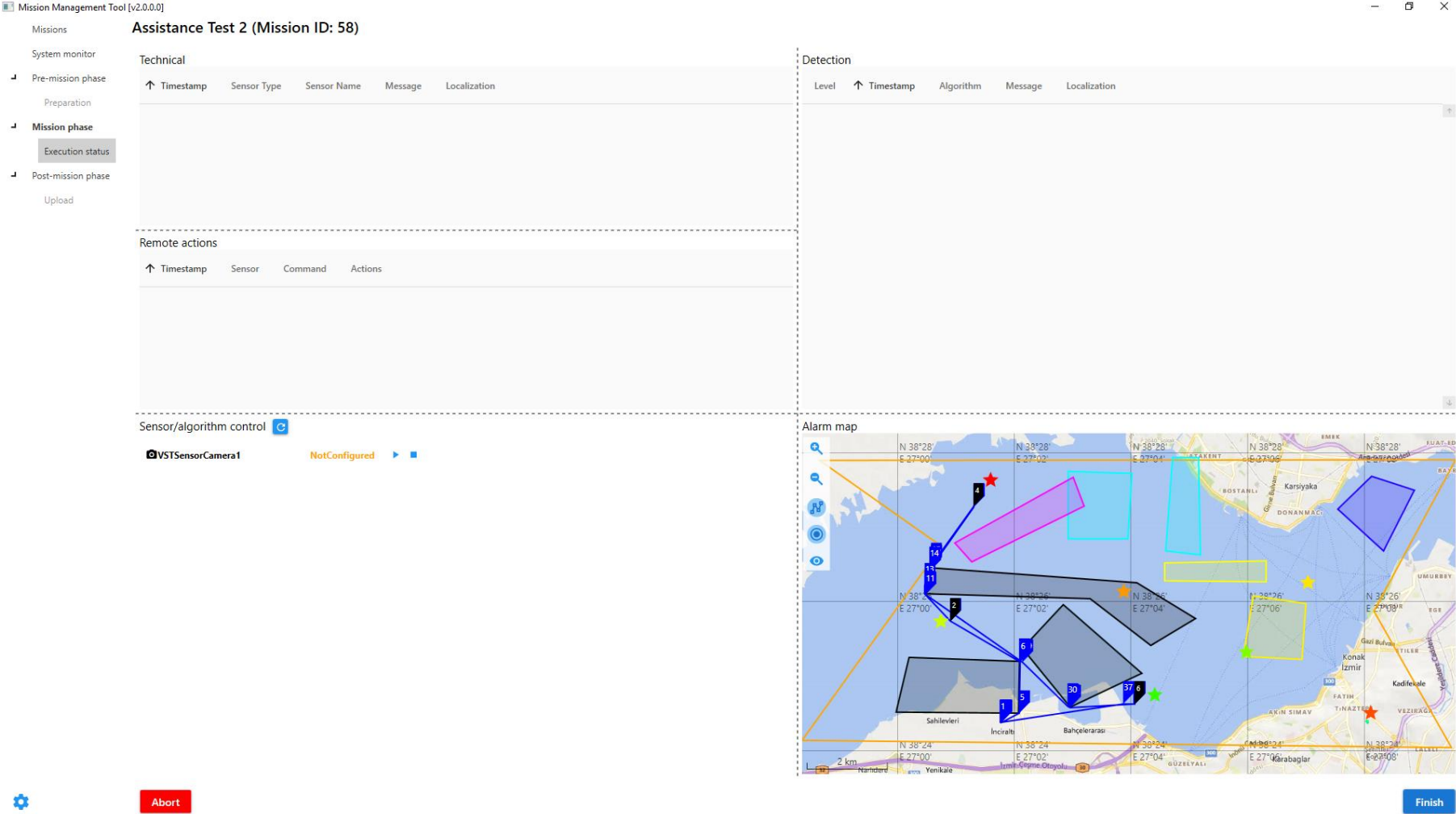


Figure 17 – GCS Situation awareness

4.1.2. Data Model

This chapter describes the messages used by the component MPM defined as inputs/outputs:

The table below lists messages used by component MPM and indicates whether the message is an input or an output of this task.

Message name	Input/output
MissionOrderRequest	Input
MissionPreparationRequest	Output
Mission	Input
SASMission	Output
SASResource	input
SASSensor	Input
SASContext	Output

Table 3 structures used by MPM

Remark: see chapter 3.2 to get the detailed description of each message.

4.1.3. Module Tests

In order to validate the component MPM, Viasat has implemented several mockup applications for all the other component:

- OMC: Optimized Mission Computation.
- SAS: Sensor Abstraction Service.

The goal of these unit tests is to validate interfaces between tasks. It is not considered the algorithm behind.

To the extent that this was feasible, Viasat has also run several tests with the real implementation of the other components in order to start validation of our development. The following chapters describe deep in details stages that we perform to reach the end of the implementation of MPM.

Tests between MPM and OMC

This test aims to check *MissionPreparationRequest* and *Mission* messages. These messages are the main used between these two components.

Step	Description	Checks
1	MPM Web Frontend defines a Mission Context	Check user interface usage
2	MPM Web Frontend requests a computation	
3	MPM Management receive a MissionOrderRequest	Check log file with JSON content
4	MPM Management sends MissionPreparationRequest	Check log file with JSON content
5	OMC reply a Mission	Check log file with JSON content

Table 4 Unit test for MPM and OMC

This unit test has been performed with different contexts:

- 1 resource and 1 target (Camera) – 1 Cost Penalized area
- 1 resource and 1 target (Temperature) – 1 Cost Penalized area
- 1 resource and 2 targets (Camera and Temperature) – 2 Cost Penalized areas
- 2 resources and 2 targets (Camera and Temperature) – 2 Cost Penalized areas
- 2 resources and 4 targets (Camera and Temperature) – 4 Cost Penalized areas
- 4 resources and 2 targets (Camera and Temperature) – 4 Cost Penalized areas

Tests between MPM and SAS

The goal of this test is to validate messages *SASResource* and *SASSensor*. These messages are used to launch a mission computation. It allows to know where resources are and what are hosted by resources.

Step	Description	Verification
1	Clean resources in SAS	Check that database is empty
2	Clean sensors in SAS	Check that database is empty
3	Create a combination of each type/subtype of resources Type:	Check that database contains 18 resources

	<ul style="list-style-type: none"> - person - vehicle - unmannedVehicle Subtype: <ul style="list-style-type: none"> - car - van - truck - fireman - policeman - sanitary 	
4	Create a sensor for each resource of type “unmannedVehicle”	Check that database contains 6 sensors
5	MPM requests resources	Check log files JSON messages and MPM Dictionary
6	MPM requests sensors for each resources of type “unmannedVehicle”	Check log files JSON messages and MPM Dictionary

Table 5 Unit test for MPM with SAS

The goal of this test is to validate messages *SASContext* and *SASMission*. These messages are used to

Step	Description	Verification
1	Clean resources in SAS	Check that database is empty
2	Clean sensors in SAS	Check that database is empty
3	Create a combination of each type/subtype of resources Type: <ul style="list-style-type: none"> - person - vehicle - unmannedVehicle Subtype:	Check that database contains 18 resources

	<ul style="list-style-type: none"> - car - van - truck - fireman - policeman - sanitary 	
4	Create a sensor for each resource of type "unmannedVehicle"	Check that database contains 6 sensors
5	MPM requests resources	Check log files JSON messages and MPM Dictionary
6	MPM requests sensors for each resources of type "unmannedVehicle"	Check log files JSON messages and MPM Dictionary

4.2. Optimized Mission Computation (OMC) module

4.2.1. Implementation

OMC is a state-of-the-art technological brick developed for ASSISTANCE. It makes it possible to build a collaborative mission for a swarm of heterogeneous drones (different autonomies and functionalities). A patent is pending. We begin by presenting a state-of-the-art analysis of mission planning that will allow us to position and justify the technological concept upon which the OMC is based. We then present in a more specific way the mission solver we developed.

4.2.1.1 State of the art and justification of the technological concept used

In terms of task allocation, optimal solutions are rarely mentioned, due to the resulting combinatorial explosion [2] [3] [15]. There is an immense literature on meta-heuristic approaches, that is to say, producing solutions whose optimality is not guaranteed. For simple illustration, see for example [8] to [17]. This list is far from being exhaustive. In [2] authors report for example the case where the difference between the optimal solution and the solution produced by a heuristic approach is 50% on average, and 150% in the worst case.

Many other solutions have been devised in the event that the allocation must be decentralized for reasons related to various operational constraints. See for example [23]. However, this is not the case with the SAS developed for ASSISTANCE.

D4.6 Mission Planner

The optimal approaches are based on mathematical programming and therefore ultimately on generic solvers like CPLEX, like [1] [3] [4] [7]. These approaches make it possible to formulate very rich planning problems, going beyond the problem treated for ASSISTANCE (addition of various constraints, such as time windows to be respected).

When it comes to critical issues, such as the allocation of drones in a context where lives and critical infrastructure are at stake, we consider that OMC must strive to offer strictly optimal solutions and therefore we do not wish to rest on a meta heuristic approach. In addition, the approach consisting in proposing a global mathematical modelling of the problem then in calling upon a generic CPLEX-type solvers induces execution times which can quickly become prohibitive for many applications. These response times are often unpredictable, varying from a few milliseconds to an hour depending on the complexity of the problem.

In the context of ASSISTANCE, on the contrary, we want to obtain a solution whose reaction times are both short and predictable. Typically, we don't want a first responder to experience response times that could potentially be several minutes each time he presses a button. Indeed, the operational employment context of SAS is one of major crises involving permanent management of priorities and urgency.

In conclusion, we opted for an ad-hoc solution (i.e. not based on the use of a generic solver). In addition, we took care to adopt a modelling of the problem allowing to propose a layered architecture, which makes it possible to better manage the combinatorial complexity (see 4.2.1.4). Finally, we have taken particular care in the implementation of our planning solution, in order to guarantee short response times and to be able, if necessary, to rely on GPU type computers.

4.2.1.2 Presentation of the developed mission solver

The mission solver developed is based on a formulation of the mission planning problem encountered in the literature. The problem we solve is in fact more generic, since we have added the notion of mission cluster.

Problem formulation used by THALES

This mission planner relies on the problem formulation proposed by John Bellingham, Michael Tillerson, Arthur Richards and Jonathan P. How, from the MIT [1]. These authors demonstrate that the problem of optimal allocation of M tasks between V vehicles can be formulated as a multi-dimensional, multiple choice, knapsack problem (MMKP).

We solve the following problem:

$$\min J = \sum_{j=1}^{N_M} c_j x_j$$

Subject to the following constraints:

$$\sum_{j=1}^{N_M} V_{ij} x_j = 1$$

And:

$$\sum_{j=N_p}^{N_{p+1}-1} x_j = 1$$

Where the petals of vehicle p are numbered N_p to $N_{p+1} - 1$, with $N_1 = 1$ and $N_{N_v+1} = N_{M+1}$ and the indices have the ranges $i \in \{1, \dots, N_w\}$, $j \in \{1, \dots, N_M\}$, $p \in \{1, \dots, N_v\}$. c_j is a vector of costs (mission times) for each petal, x_j is a binary decision variable equal to one if petal j is selected, and 0 otherwise.

The first constraint enforces that task i is done exactly once. The second constraint prevents more than one petal being assigned to each vehicle.

The mission planner performs the following steps:

- **Step 1:** build the list of all possible petals. A petal is subset of L indexes in interval $\{1, \dots, N_w\}$,
- **Step 2:** evaluate the cost of each petal for each drone. This evaluation is based on trajectories provided by a transport layer (a path planning algorithm). This evaluation is made for all L possible orders of the indexes of the evaluated petal (this evaluation can be made using some heuristic approach). The best order cost is returned. The best order of execution of a petal may depend on the considered drone (due to different starting point or turning radius, for instance, see next paragraph for an illustration).
- **Step 3:** select N_v petals (one per drone) such that the union of these satisfy the two constraints explained above: each task is done exactly once.

Adaptation for ASSISTANCE use case (notion of cluster)

Step 2 involves associating a cost with a subset of L tasks, called a petal.

In the cited publication, evaluating the cost of a petal (it may for example be its duration or any other criterion to be optimized) requires considering several possible sequencing orders, or even all of the L possible web orders. We call sequence a possible sequence of L tasks of a petal. We must therefore enumerate the L possible sequences and calculate for each of them the best trajectory achieving this sequence. For higher dimension problems (> 9 or 10 waypoints), when it becomes intractable, we can use heuristic such as 2-opt instead [24].

We call path planning the step of building an optimal trajectory connecting the initial position of the drone, the waypoints associated with each of the tasks of the sequence, the desired final position of the drone (for example: its base, or its starting position).

D4.6 Mission Planner

Finally, this trajectory is evaluated according to any operational criterion to obtain a score (here it is a cost that we want to minimize).

We summarize the steps of phase 2 below. For illustration, suppose a petal made up of tasks {1, 2, 3, 4, 5}. There are $5! = 120$ possible sequences. For example: {5, 1, 2, 3, 4} and {5, 1, 4, 3, 2}. For each of these sequences of waypoints:

- Call for a path planning algorithm to determine the best trajectory connecting the starting position of the drone, waypoints 5, 1, 4, 3, 2 and the desired arrival position for the drone
- Use it to evaluate the score of this sequence
- If the score of this sequence is better than that of the other sequences, keep this score

Thus formulated, the problem addressed is not exactly the one we want to address in ASSISTANCE. In fact, in the aforementioned work, the authors consider that a task (for example: taking a photo) is associated with a single waypoint: this is the position from which the photo must be taken, and on which the drone must therefore surrender). By modelling the problem in this way, we assume that we are able to associate a unique position with each task. Concretely, the first responder must choose, for each action, a unique position of the drone at the time of the action.

However, it may happen that a first responder does not have a particular preference as to the position of the camera to take a shot on the ground (he wishes to observe a position on the ground, but a priori does not have a preference on the angle of view itself). The first responder will then prefer to provide a possible zone for the position of the camera (for example, a circle or a disc) and let the mission hover choose in this zone a position which decreases the overall duration of the mission and therefore improves the autonomy operational of the drone.

This point is illustrated below. A first responder wants to take pictures of four houses on which there is a fire. In figure 18-a, the first responder had to manually choose a shooting position per object of interest. In figure 18-b, the first responder only chose a shooting circle, specifying only the position to be observed, the radius of the circle and the altitude of the latter, and left the mission planner select a shooting position on this circle. By leaving an additional degree of freedom for the mission planner, the first responder has obtained a shorter and therefore more optimal planning solution.

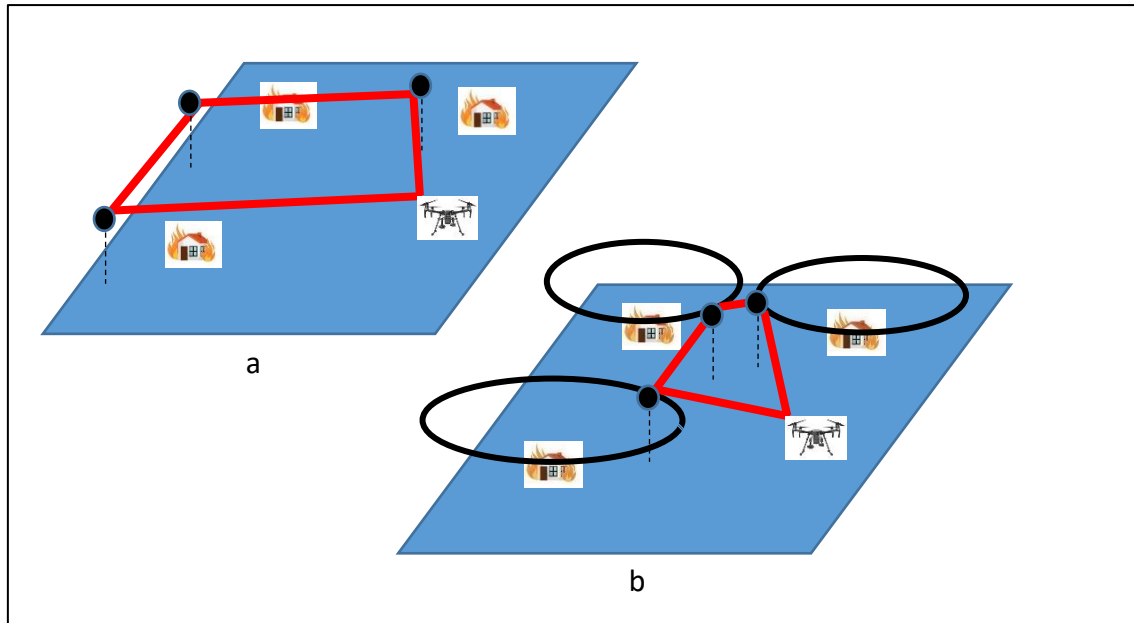


Figure 18 – Waypoints vs clusters planning

We have therefore modified the formulation of the problem as follows: we do not consider waypoints, but clusters of waypoints. A cluster is a set of N waypoints within which solver will choose a waypoint.

Thus, considering that a task is associated with a cluster and no longer with a single waypoint requires the introduction of an additional step, called cluster routing. We call cluster routing the process of selecting a waypoint within each of the clusters in a sequence. Choosing the best waypoint depends on the sequence of tasks (and therefore clusters). The cluster routing step is therefore to be carried out for each sequence.

We illustrate this below with the sequences $\{5, 1, 2, 3, 4\}$ and $\{5, 1, 4, 3, 2\}$, which are two possible realizations of the petal $[1, 2, 3, 4, 5]$ (i.e. 2 possible orders among the $!5$ that should be tested). Concretely, the evaluation of a petal consists not only in testing all the possible sequencing orders, and for each possible sequencing order (i.e. each sequence), performing cluster routing to select a waypoint within each cluster.

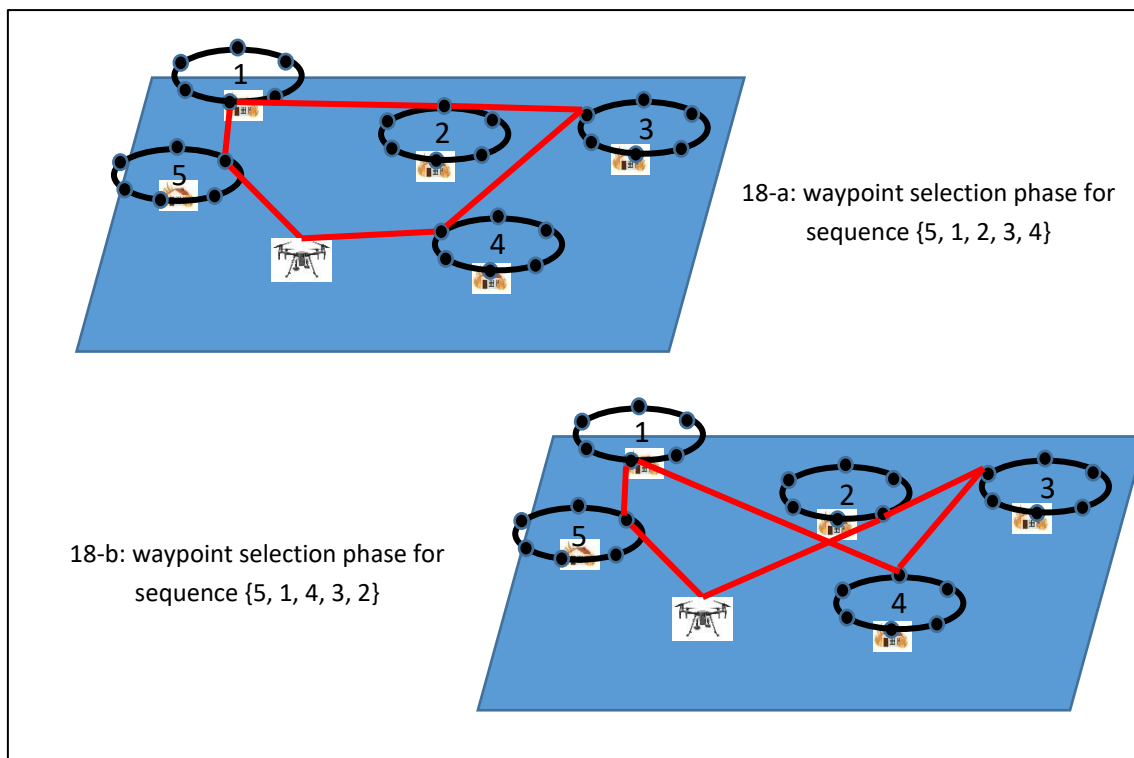


Figure 19 – Task ordering with respect to context

We summarize below the steps of phase 2 generalized to clusters instead of waypoints. Suppose by way of illustration a petal consisting of the spots {1, 2, 3, 4, 5}. There are $5! = 120$ possible sequences. For example: {5, 1, 2, 3, 4} and {5, 1, 4, 3, 2}. For each of these cluster sequences:

- Calling a cluster router algorithm to select the best waypoint within each cluster. This cluster routing phase can rely on a path planning algorithm (similar to the one used below), but could equally well rely on faster technologies, such as neural networks.
- Call to a path planning algorithm to determine the best trajectory connecting the starting position of the drone, the waypoint selected in cluster 5, the waypoint selected in 1, the waypoint selected in 4, the waypoint selected in 3, the waypoint selected in 2 and the desired arrival position for the drone
- Evaluation of the score of this sequence
- If the score of this sequence is better than that of the other sequences, keep this score

4.2.1.3 Path planning methodology

Taking UAV altitude into account

The realization of 3D trajectories for UAV would requires having a 3D model of the place of deployment of the drone system: height and footprint of buildings, vegetation (trees in particular), electrical infrastructures, landforms, etc. Such a model must also be precise enough to preserve the integrity of the drones.

However, it is unlikely that the users of the ASSISTANCE system will have such a 3D model in any doubt, whether it is a test campaign as part of the validation of ASSISTANCE, or even the deployment of a real operational system (at least in the near future). Moreover, in the event of an earthquake for example, or a landslide etc. such a 3D model may be wrong. This would have typically been the case in the appalling collapse of the Genoa bridge on August 14, 2018 (figure 20), or during a landslide such as the one that occurred in Taiwan on April 26, 2010 (figure 21).



Figure 20 – Collapse of Genoa bridge in 2018



Figure 21 – Landslide in Taiwan in 2010

For this reason, we have opted for 2D and a half trajectory, composed of three sequences: an altitude change phase (initial altitude \rightarrow cruising altitude), a 2D navigation phase at this cruising altitude (which will suffice) to take sufficiently large: for example 80m) and a new phase of change of altitude on arrival (cruising altitude \rightarrow final altitude). We only optimized the following case: initial and final altitudes both higher than the cruise altitude, in which case the drone cruises at the lower altitude of the two.

Note: the PIAP UGV has obstacle detection and avoidance capabilities that will allow it to adapt its trajectory to unplanned accidental obstacles, such as debris, vehicles, etc.

The trajectory will then be calculated based on information known to the first responders and the UGV will rely on its AI to adapt the trajectory obtained.

Obtaining 2D trajectories (for both UAV and UGV)

2D trajectories were obtained using James Sethian's fast marching algorithm [25]. The generation method is as follows:

- Sample the area with a regular pitch grid
- Associate a cost with each node of the grid. A pixel not belonging to any cost penalized area is assigned an arbitrarily low non-zero cost: 0.001^2 . A pixel belonging to an area penalized in cost is attributed the cost associated with this area (this is the cost entered by the user, as explained in paragraph 4.1.1).
- Solve the eikonal equation (for example: propagation of a mobile) by propagating the following initial conditions: the distance associated with the starting point is 0, the distance associated with any other point is infinite.
- To get the trajectory from the starting point to any position on the grid, go up the distance gradient (using any digital derivation process).
- **For each grid point considered, the trajectory thus obtained is the geodesic with the lowest cumulative user-defined cost, i.e. the best trajectory within the meaning of the used-defined cost function.**

The trajectories obtained in this way are continuous trajectories (they pass between the nodes), unlike the trajectories obtained with a graph-based approach (the trajectories would pass through the nodes via predefined transitions). The Figure 22 illustrates this phenomenon, on a trivial case (all the nodes have the same cost of 1). In blue, a trajectory obtained by applying a Dijkstra or A * type algorithm. In red, a trajectory obtained by applying a fast marching algorithm.

² The cost associated with a pixel cannot be strictly zero, otherwise the gradient ascent phase would fail at this point.

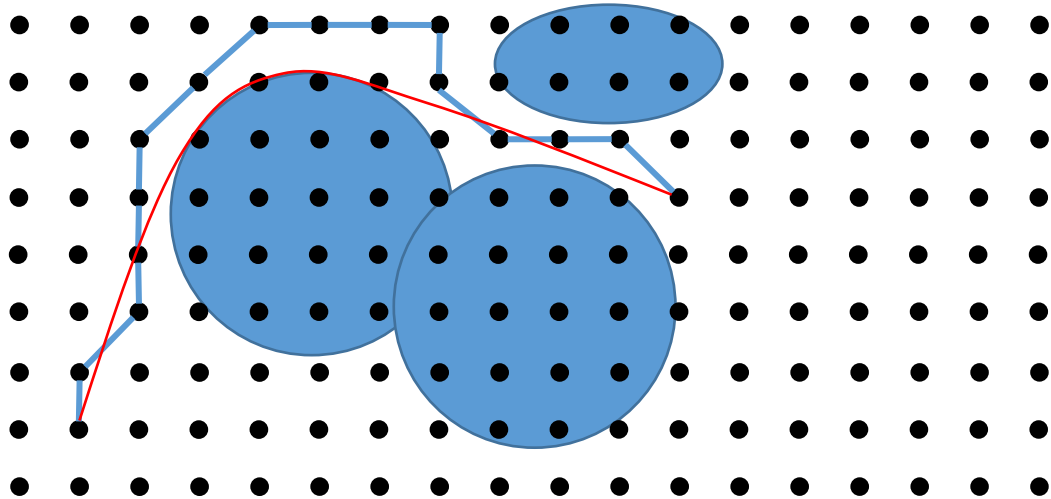


Figure 22 – Smooth trajectories vs Dijkstra like trajectories

These trajectories must then be filtered so as not to provide the UxV with waypoints that are too close together. The strategy devised within the framework of ASSISTANCE is based on segmenting the trajectories by portions of quasi straight lines using the variations in heading observed. Figure 23 illustrates the performance of this point filtering. In blue, the initial trajectory (from the trajectory solver). The red circles represent the positions that will be retained within the flight plan provided to the drone.

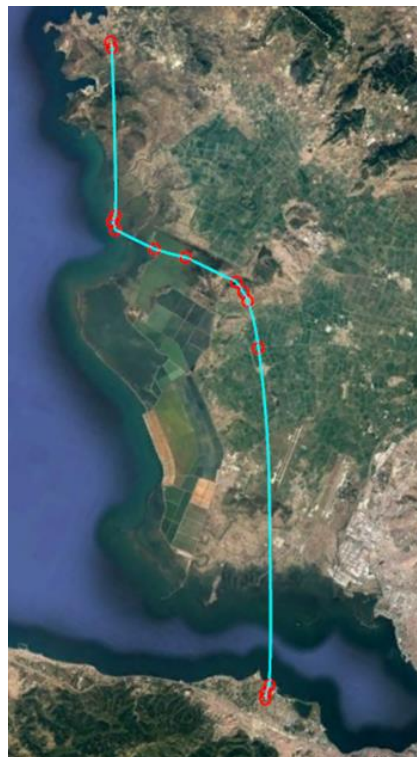


Figure 23 – Trajectory pruning

An essential parameter of the path planning layer is the grid step. Computing times increase when the grid pitch decreases.

D4.6 Mission Planner

An auto-adjustment mechanism is provided in order to guarantee a number of nodes close to 50,000, which seems to offer a satisfactory compromise between execution time and trajectory precision.

4.2.1.4 OMC architecture

This solution was instantiated as a C++/OpenCL library relying on a layered architecture:

- Task allocation layer: responsible for the repartition of the tasks between the drones.
- Task ordering layer (for a given task allocation on a given drone): responsible for the selection of the best order of execution of the tasks.
- Cluster routing layer (for a given sequence of tasks on a given drone): responsible for the selection of a waypoint in each cluster.
- Path planning layer (for a given sequence of waypoints on a given drone): responsible for determination of the best trajectory connecting the drone starting position, each of the selected waypoint and the desired final position for the drone.

4.2.2. Data Model

The OMC and MPM modules communicate through a request and response file mechanism based on the JSON standard. MPM communicates the context information (cost penalized areas formulated a polygons, performance, available function and current position of the drones, etc.) as well as the tasks to be carried out (camera shots and temperature readings). OMC distributes these tasks between the drones and provides, for each of them, a flight plan (waypoints and actions such as start shooting, stop shooting).

4.2.2.1 OMC request file content

JSON request file for OMC contains:

- Contour of the mission zone (within which the trajectories of the drones must be maintained)
- Current positions of drones
- Desired final position for drones
- Positions of temperature readings to be taken
- Shooting positions to be performed
- Non-zero cost zones (if any), associated with a variable amplitude weighting, allowing certain zones to be penalized more than others (an arbitrarily high weighting which can mean a prohibited zone)
- For each drone:
 - o List of functions performed
 - o Autonomy

4.2.2.2 OMC response file content

The JSON response file contains a flight plan for each selected drone. A flight is made up of the following information:

- A list of dated waypoints
- A list of dated actions (start a measurement, stop the measurement)

4.2.3. Module Tests

A scenario editor on KML support was produced in order to be able to generate a test set that we consider covering.

4.2.3.1 KML scenario editor

Google Earth software allows you to edit lines and positions of KML files and then save them as a file in KML format via a very user friendly GUI. The KML format is an overlay of the XML format. A KML file loading module has been developed to load the placemarks of a KML file.

By using the "description" and "name" fields of the edited placemarks (always thanks to the editing features offered by Google Earth), we can quickly edit scenarios.

Figure 24 shows the contents of a KML file. Each Placemark capsule corresponds to an object edited on the map. A Placemark capsule contains a <name> field and a <description> field. A clever use of these fields, associated with an algebra of key words (such as "camera", "temperature", "drone", "zone" or even "theater") then makes it possible to find all the elements of the scenario. This scenario is then transformed into a JSON request recognized by the solver.

```
<Placemark>
  <Placemark>
    <name>camera_1</name>
    <visibility>0</visibility>
    <description>radius:100
points:10
height_m:90
duration_s:5</description>
    <LookAt>
    <styleUrl>#msn_ylw-pushpin4</styleUrl>
    <Point>
      <gx:drawOrder>1</gx:drawOrder>
      <coordinates>27.16118162557414,38.43569503602544,0</coordinates>
    </Point>
  </Placemark>
</Placemark>
```

Figure 24 – Example of KML scenario file

4.2.3.2 Tests plan

We report here the results on 3 IZMIR scenarios each comprising 4 drones, 4 temperature readings and 4 shots.

We first present (figure 25) the scenario edited using Google Earth GUI. The blue outline marks the mission area. The red contours delimit areas at non-zero cost. The 4 green icons indicate the starting and finishing positions of the drones. The 4 yellow icons indicate the positions of the temperature readings to be taken. The 4 purple icons indicate the shooting positions.



Figure 25 – Test scenario edited on Google Earth

In scenario 1, the 4 drones each have a temperature sensor and a camera. In scenario 2, the swarm is heterogeneous (2 drones with only one temperature sensor, one drone with one temperature sensor and one camera, one drone with only one camera). Scenario 3 is identical to scenario 2 with non-zero weighted areas.

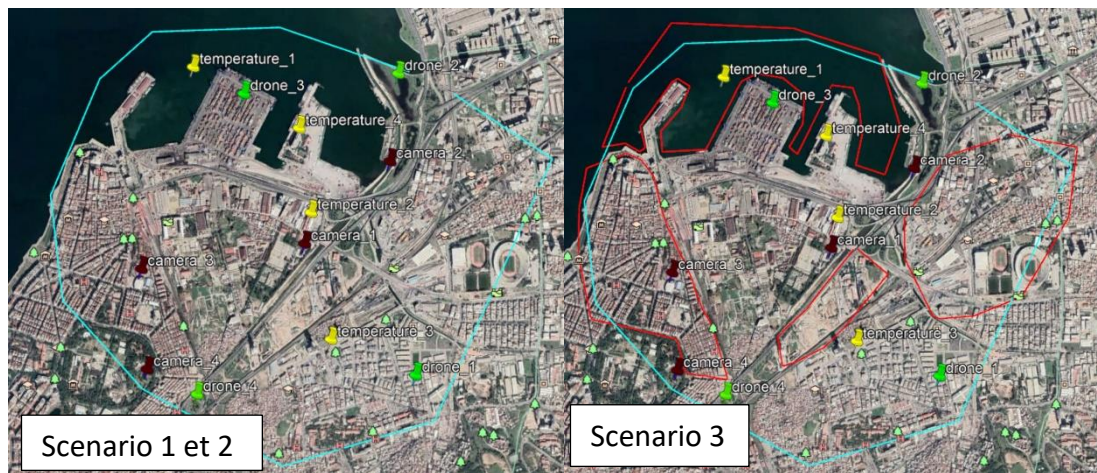


Figure 26 – Comparison of scenarii 1, 2 and 3

Scenario 1	drone_1	drone_2	drone_3	drone_4
Temperature				
Camera				

Scenario 2	drone_1	drone_2	drone_3	drone_4
Temperature				
Camera				

Scenario 3	drone_1	drone_2	drone_3	drone_4
Temperature				
Camera				

Scenario 1 results



Figure 27 – Results on scenario 1

As the 4 drones have the same functions, the optimal distribution is quite intuitive (each drone performs two tasks). In the absence of a non-zero-cost zone, drones fly in straight lines.

The assignments are as follows:

- Drone 1 (bottom right): tempMeas_2, camShoot_0
- Drone 2 (top right): camShoot1, tempMeas_1
- Drone 3 (top left): tempMeas_0, tempMeas_3
- Drone 4 (bottom left): camShoot_2, camShoot_3

Scenario 2 results



Figure 28 – Results on scenario 2

Since drones 2 (top right) only have a temperature sensor, it can no longer perform camShoot_1. Likewise, drone 1 (bottom right) can no longer perform camShoot_0. The drone 3 (top left), which still has 2 types of sensor, takes the two shots thus neglected.

The assignments are as follows:

- Drone 1 (bottom right): tempMeas_2
- Drone 2 (top right): tempMeas_0
- Drone 3 (top left): tempMeas_3, camShoot_1 and camShoot_0
- Drone 4 (bottom left): camShoot_2, camShoot_3

Scenario 3 results



Figure 29 – Results on scenario 3

The trajectories are now much more complex due to the presence of areas penalized in cost. The optimal distribution of tasks is no longer intuitive at all.

The assignments are as follows:

- Drone 1 (bottom right): tempMeas_2
- Drone 2 (top right): tempMeas_1, tempMeas_0, tempMeas_3,
- Drone 3 (top left): camShoot_1, camShoot_0, camShoot_3
- Drone 4 (bottom left): camShoot_2, camShoot_3

5. Use cases

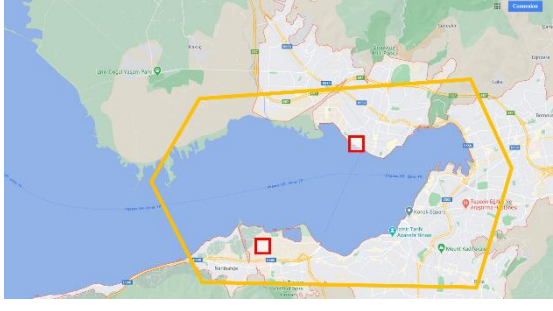
This section describes functional use cases that will help each partner to understand how their module is integrated into a functional scenario and validate interactions between modules as data exchanged.

5.1. Mission Preparation

Based on the sequence diagram described in the chapter 3.3.1, a detailed test case has been built. This test case is presented in this chapter.

It has been performed at different level to check that each task works as expected.

The table below shows the main steps of the test case (the most human readable things) that explain how the detailed plan is computed and send to all GCS. MPM will organize the whole workflow.

Step	Description
	Description of the resource available in the area of the mission. Resources are provided by SAS
	Define the context of the mission: <ul style="list-style-type: none"> - Mission area
	Define the context of the mission: <ul style="list-style-type: none"> - Targets



	<p>Define the context of the mission:</p> <ul style="list-style-type: none"> - Forbidden zones
	<p>After computation by OMC, list of actions (flight and sensor) are available</p>
	<p>MPM displays the whole mission for all resources</p>
	<p>MPM sends Mission to SAS</p>

Table 6 Mission computation use case

This module T4.6 will take place in :

- SC1- Use case 1: Deployment of Sensors and Mobile Platforms,
- SC2 – Use case 1 : Deployment of sensors and mobile platforms/Planning and preparation,
- SC2- Use case 6: Transformer building investigation,
- SC3- Use case 1: Deployment of sensors and mobile platforms.

All these use cases have been described in document D2.3

6. Conclusion

The main objective of this deliverable is to define the system architecture and the data model of the task T4.6.

Chapter 3 recalls the purposes and challenges of task 4.6, namely the development of the Mission Manager Module (MMM), as well as the role played by the latter in the ASSISTANCE architect. We detail the interfaces between MMM and the SAS, as well as the interfaces between the two modules constituting the MMM (MPM and OMC, namely Mission Planner Management and Optimized Mission Computation). Finally, we present the complete sequence diagram allowing the MMM operator to build a mission.

Chapter 4 provides some implementation details of the two modules that make up the MMM, namely MPM and OMC. For each of them, the data models are presented, as well as the unit test procedure used to validate them separately.

Finally, the chapter 5 describes the functional use cases where task T4.6 is involved. It helps the consortium to understand how this task works in the overall project. And it allows us to check that requirements have been considered by our components.

In this document, all components of the task T4.6 have been successfully described. All tests proving that components work as expected have also been defined and have been done.

7. Bibliography

- [1] Multi-task allocation and path planning for cooperating UAVs, J. Bellingham, M. Tillerson, A. Richards, J.P. How, 2003
- [2] Optimal vs heuristic assignment of cooperative autonomous unmanned air vehicles, S. Rasmussen, P. Chandler, J. W. Mitchell, C. Sschmuacher, A. Sparks, 2003
- [3] Flight planning for unmanned aerial vehicles, A. Fügenschuh, 2015
- [4] Mission planning for unmanned aerial vehicles, A. Fügenschuh, 2019
- [5] Complexity in UAV cooperative control, P. R. Chandler, M. Pachter, D. Swaroop, J. M. Fowler, J. K. Howlett, S. Rasmussen, C. Schumacher, K. Nygard, 2004
- [6] Cooperative path planning for multiple UAVs in dynamic and uncertain environments, J. S. Bellingham, M. Tillerson, M. Alighanbari, J. P. How, 2002
- [7] Military aircraft mission planning: a generalized vehicle routing with synchronization and precedence, N-H. Quttineh, T. Larsson, K. Lundberg, K. Holmberg, 2013
- [8] Military aircraft mission planning: efficient model based metaheuristics approaches, N-H Quttineh, T. Larsson, 2014
- [9] A memetic algorithm for path planning of curvature-constrained UAVs performing surveillance of multiple ground targets, Z. Wing, C. Jie, X. Bin, P. Zhihong, 2013
- [10] Cooperative task assignment of multiple heterogeneous unmanned aerial vehicles using a modified genetic algorithm with multi-type genes, D. Qibo, Y. Jianqiao, W. Ningfei, 2013
- [11] Cooperative task assignment and path planning of multiple UAVs using genetic algorithm, Y. Eun, H. Bang, 2007
- [12] Multi UAV reconnaissance task assignment for heterogeneous targets based on modified symbiotic organisms search algorithm, H-X. Chen, Y. Nan; Y. Yang, 2019
- [13] Genetic algorithm based decentralized task assignments for multiple unmanned aerial vehicles in dynamic environments, H. Choi, Y. Kim, H. Kim, 2011
- [14] Multiple task assignments for cooperating uninhabited aerial vehicles using genetic algorithms, T. Shima, S.J. Rasmussen, A.G. Sparks, K.M. Passimo, 2005
- [15] Assigning cooperating UAVs to simultaneous tasks on consecutive targets using genetic algorithms, T. Shima, C. Schumacher, 2009
- [16] Cooperative task allocation for unmanned combat aerial vehicles using improved ant colony algorithm, J. Tao, Y. Tian, X. Meng, 2008

- [17] Genetic algorithm for task allocation in UAV cooperative control, G. Chen, J.B.Cruz Jr, 2003
- [18] The Hungarian method for the assignment method, H.W. Khun, 1955
- [19] GPU-accelerated hungarian algorithms for the linear assignment problem, K. Date, R. Nagi, 2016
- [20] A new approach for solving the single objective unbalanced assignment problem, V. Yadaiah, V.V. Haragopal, 2016
- [21] The average sum method for the unbalanced assignment problems, S.K. Dubey, A. Kumar, V. Upadhyay, 2018
- [22] A new approach for getting optimality of assignment problems, B.M. Patel, M.J. Doshi, 2019
- [23] An Iterative Strategy for Task Assignment and Path Planning of Distributed Multiple Unmanned Aerial Vehicles, W. Yao, N. Qi, N. Wan, Y. Liu, 2019
- [24] A method for solving traveling salesman problems, G. A. Croes, 1958
- [25] Fast marching methods for robotic navigation with constraints, R. Kimmel, J.A. Sethian, 1996