Technological and Economic Feasibility Report

Deliverable ID	D5.2
Project Acronym	IMPETUS
Grant:	763807
Call:	H2020-SESAR-2016-1
Topic:	RPAS 02 - Drone Information Management
Consortium coordinator:	CRIDA A.I.E.
Edition date:	18 12 2019
Edition:	00.01.00
Template Edition:	02.00.00

Authoring & Approval

Authors of the document

Name/Beneficiary	Position/Title	Date
Michael Christian Büddefeld	TUDA PMST representative	12/12/2019
Dominik Janisch	CRIDA participant	12/12/2019
Pablo Sánchez-Escalonilla	CRIDA PMST representative and manager	12/12/2019
Marta Sánchez	CRIDA participant	12/12/2019
Anna-Lisa Mautes	JEPPESEN PMST representative	12/12/2019
Hugo Eduardo	JEPPESEN participant	12/12/2019
Javier Espinosa	INECO PMST representative	12/12/2019
Chris Foster	ALTITUDE ANGEL PMST representative	12/12/2019
Miguel Vilaplana	BR&TE PMST representative	12/12/2019
Nicolás Pena Ortiz	BR&TE participant	12/12/2019

Reviewers internal to the project

Name/Beneficiary	Position/Title	Date
Pablo Sánchez-Escalonilla	CRIDA PMST representative and manager	17/12/2019
Michael Christian Büddefeld	TUDA PMST representative	17/12/2019
Dominik Janisch	CRIDA participant	17/12/2019
Marta Sánchez	CRIDA participant	17/12/2019

Approved for submission to the SJU By - Representatives of beneficiaries involved in the project

Name/Beneficiary	Position/Title	Date
Pablo Sánchez-Escalonilla	CRIDA PMST representative and manager	18/12/2019
Michael Christian Büddefeld	TUDA PMST representative	18/12/2019
Anna-Lisa Mautes	JEPPESEN PMST representative	18/12/2019
Rafael Gallego	INECO PMST representative	18/12/2019
Chris Foster	ALTITUDE ANGEL PMST representative	18/12/2019
Miguel Vilaplana	BR&TE PMST representative	18/12/2019

Rejected By - Representatives of beneficiaries involved in the project

Name/Beneficiary	Position/Title	Date

Document History

Edition	Date	Status	Author	Justification
00.00.01	15/12/2018	Draft	H. Eduardo	Structure Proposal
00.00.02	09/05/2019	Draft	M. Büddefeld	Structure Proposal
00.00.03	16/05/2019	Draft	TUDA/CRIDA/JEPP	Final structure / division
00.00.04	19/07/2019	Draft	ALL	First contributions to §2 and restructure the sections after comments in F2F meeting
00.00.05	25/08/2019	Draft	BRTE/INECO/AA/JEPP	Final contributions to §2
00.00.06	24/10/2019	Draft	BRTE/INECO/AA/JEPP	Contributions to §3
00.00.07	25/11/2019	Draft	BRTE/INECO/AA/JEPP	Contributions to §3 and §4 after inputs from external workshop
00.00.08	29/11/2019	Draft	BRTE/AA/JEPP	Contributions to §3 and §4
00.00.09	12/12/2019	Draft	BRTE/JEPP	Final contributions to §3 and §4 and consolidated version
00.01.00	18/12/2019	Final	CRIDA/TUDA/JEPP	Overall consistency and peer review

IMPETUS

INFORMATION MANAGEMENT PORTAL TO ENABLE THE INTEGRATION OF UNMANNED SYSTEMS

This deliverable is part of a project that has received funding from the SESAR Joint Undertaking under grant agreement No 763807 under European Union's Horizon 2020 research and innovation programme.

Abstract

IMPETUS D5.2 presents the results of the experimental testing performed by the consortium. The experiments deal with a set of crucial U-space services and are addressing the previously defined use cases. All services have been prototyped and implemented based on a microservice-based architecture. The analysis reveals that, although there are challenges in designing a technical architecture for U-space, the microservice implementation approach is able to provide useful mechanisms and tools to address these challenges.

Table of Contents

Abstract 4

Executive Summary *11*

1 Introduction *12*

1.1 Scope of the document 12

1.2 Intended readership 12

1.3 Acronyms and terminology 12

1.4 IMPETUS approach 15

1.5 Document structure 16

2 Context of the experiments *17*

2.1 U-space services and capabilities: relation to the experiments 17

2.2 Experiment Purpose and approach 18

2.3 Summary of the Experimental Plan 19

2.3.1 Objectives and success criteria to validate the tested services 20
2.3.2 Objectives and success criteria to validate the federated microservice-based architecture 20
2.3.3 Assumptions 34

2.4 Expected deviations of planned experiments 35

2.4.1 Drone-specific Weather Service (EXE 1) 35
2.4.2 Mission and Flight Planning Management services (EXE 2) 43
2.4.3 Traffic Information and Monitoring Services (EXE 3) 44
2.4.4 Traffic Management Services dealing with dynamic airspace management (EXE 4) 50

3 Experimental results *55*

3.1 Summary of experimental results 55

3.2 Detailed analysis of results per service being tested 56

3.2.1 Drone-specific Weather Service (EXE 1) 56
3.2.2 Mission and Flight Planning Management Services (EXE 2) 72
3.2.3 Monitoring and Traffic Information Services (EXE 3) 83
3.2.4 Traffic management services dealing with dynamic airspace management (EXE 4) 91

3.3 Detailed analysis of results per research area of the microservice-based architecture 98

3.3.1 Safety and Failure Modes 98
3.3.2 Data Management 107
3.3.3 Scalability 111
3.3.4 Flexibility 118
3.3.5 Commercial Implications 121

4 Conclusions and recommendations *130*

4.1 Conclusions 130

4.1.1 Technological Feasibility 130
4.1.2 Economic Feasibility 136

4.2 Recommendations 136

5 References 139
Appendix A Objectives and success criteria to validate the tested services.................140
Appendix B Objectives and success criteria to validate the architecture.....................144
Appendix C Assumptions for the exercises 147

List of Tables

Table 1: Acronyms 12
Table 2: Relation of exercises with U-space service and capabilities 18
Table 3: Scenarios and scope of the exercises considered 20
Table 4: Relation of the exercises with the considered research areas.. 21
Table 5: Validation objective description of test SFM-1 23
Table 6: Validation objective description of test SFM-2 24
Table 7: Validation objective description of test SFM-3 25
Table 8: Validation objective description of tests DM-1 -DM-4.. 27
Table 9: Validation objective description of test DM-1 28
Table 10: Validation objective description of test DM-2 29
Table 11: Validation objective description of test S-1 31
Table 12: Validation objective description of test S-2 32
Table 13: Validation objective description of test S-3 33
Table 14: Validation objective description of test F-1 34
Table 15: Galicia high-resolution model implemented in the Core Weather Computation Service 39
Table 16: Dynamic Separation Criteria... 51
Table 17: Relation of the exercises with the validation areas.. 55
Table 18. Results for the East-West pattern 64
Table 19. Results for the North-South pattern 64
Table 20. Detailed results for the East-West pattern... 67
Table 21: Results of entry and exit time predictions 69
Table 22: Flight plan data components... 73
Table 23: Flight plan data components completed by the FPM service 74

Table 24: Models and data samples used in experiment FPM-1 76
Table 25: Summarized results in experiment FPM-1 76
Table 26: Models and data samples used in experiment FPM-2 80
Table 27: Summarized results in experiment FPM-2 80
Table 28: Models and data samples used in experiment FPM-3 82
Table 29: Summarized results in experiment FPM-3 83
Table 30: Metrics of MTI-Test 1: Latency 84
Table 31: Metrics of MTI-Test 2: Scalability 85
Table 32: Metrics of MTI-Test 5: Maximum workload 86
Table 33: Metrics of MTI-Test 2: Scalability 88
Table 34: Metrics of MTI-Test 2: Maximum workload (without filter) 90
Table 35: Number of instances and maximum workload for the considered microservices 90
Table 36: Summarized results of experiment related to resilience 101
Table 37: Summarized results of experiment related to delays and recovery time 102
Table 38: Summarized results of experiment related to mechanisms against faults 104
Table 39: Summarized results of experiment related to the processing of user requests 108
Table 40: Summarized results of experiment related to data consistency 109
Table 41: Summarized results of experiment related to the assessment of processing time 110
Table 42: Summarized results of experiment related to data processing 110
Table 43: Summarized results of experiment related to the scalability of the Weather Service 114
Table 44: Summarized results of experiment related to the deployment of additional services 115
Table 45: Summarized results of experiment related to the adaptability of the architecture 119
Table 46: Processing time in relation to certain transactions 122
Table 47: Summarized results for the estimation of flight plan requests 123
Table 48: Cost of exercise 3 in support systems 125
Table 49: Cost of the microservice architecture from exercise 3 per day 126

List of Figures

Figure 1: Architecture of the FPM Service and interaction with the MPM Service 26

Figure 2: Micro-Service Prototype high-level architecture and main information flows 36

Figure 3: Model structure for the Core Weather Computation Model and geographical coverage 38

Figure 4: Simulation environment put together to conduct the exercise EXE1................................. 40

Figure 5: Schematics of the mission pattern 41

Figure 6: Depiction of the platform used to conduct the missions in the exercise 41

Figure 7. Architecture of the Core Weather Service 42

Figure 8: Functional components of the MPM and FPM Services 43

Figure 9: Information flow of the services detailed in the microservice architecture......................... 45

Figure 10: Microservice environment (EXE 3) 46

Figure 11: Information flow in exercise 3... 48

Figure 12: Simulation scenario 52

Figure 13: Confliction resolution zones.. 54

Figure 14: Scheme for mission trajectory prediction analysis for different look ahead times 57

Figure 15. Mission Trajectory for scenario WX-1: East-West surveillance pattern 58

Figure 16. Flight time prediction results for East-West pattern (WX-1) 58

Figure 17. Fuel burn prediction results for East-West pattern (WX-1) 59

Figure 18. Predicted climb profile with Galicia forecast and 3 hours look ahead (WX-1) 60

Figure 19. Predicted descent profile with Galicia forecast and 3 hours look ahead (WX-1) 61

Figure 20. Predicted cruise altitude with Galicia forecast and 3 hours look ahead (WX-1)................. 62

Figure 21. Mission Trajectory for scenario WX-2: North-South surveillance pattern......................... 63

Figure 22. Flight time prediction results for North-South pattern (WX-2) 63

Figure 23. Fuel burn prediction results for North-South pattern (WX-2) 64

Figure 24. Maximum range prediction results for East-West pattern (WX-3)..................................... 66

Figure 25. Maximum time prediction results for East-West pattern (WX-3)...................................... 66

Figure 26: Gliding path from a point in cruise of the East-West pattern for 3 different descent speeds (WX-3).. 68

Figure 27. Mission Trajectory for scenario WX-4, showing a reference airspace volume. 69

Figure 28. Prediction results for WX-4 using probabilistic micro-weather 70

Figure 29. Prediction results for WX-4 using static weather forecast.. 70

Figure 30: Probabilistic precipitation forecast using the IMPETUS Micro-Weather Service (Galicia model) 71

Figure 31: Flight trajectory in the spatial domain 74

Figure 32: Exemplification of the grid model used in experiment FPM-2 78

Figure 33: Conflict resolution applied in experiment FPM-2 79

Figure 34: Representation of a NOTAM publication 81

Figure 35: HMI of the microservice platform showing the congested airspace and alerts related to operations (in red).. 83

Figure 36: Latency of messages processed in experiment MTI-1 for 10 and 20 drone operations...... 84

Figure 37: Latency of messages processed in experiment MTI-1 for 30 and 100 drone operations.... 85

Figure 38: Latency of messages processed in experiment MTI-2 (1).. 85

Figure 39: Latency of messages processed in experiment MTI-2 (2).. 86

Figure 40: Latency for a maximum overload scenario in experiment MTI-5 87

Figure 41: Latency of messages processed in experiment MTI-2 for 10, 100 and 200 drone operations 89

Figure 42: Latency of messages processed in experiment MTI-1 250 drone operations 89

Figure 43: Latency for a maximum overload scenario in experiment MTI-5 90

Figure 44: Graph shows no loss of separation as the number of drone operations increase 91

Figure 45: Graph shows average drone operations efficiency as number of drones managed increases 92

Figure 46: A new flight restriction and all airborne drones within this region exiting the restricted zone 93

Figure 47: Sample of Audit Log 96

Figure 48: Example Flight Plan 97

Figure 49: Temporary Flight Restriction Data Example... 98

Figure 50: Load handling of implemented architecture... 99

Figure 51: Latency of messages processed in experiment MTI-6 103

Figure 52: Workload of different microservices in relation the number of processed messages 104

Figure 53: Console output demonstrating recovery time 107

Figure 54: RAM usage as a function of the load... 112

Figure 55: CPU usage as a function of the load.. 113

Figure 56: Latency of messages processed in experiment MTI-2 116

Figure 57: Latency of messages processed in experiment MTI-5 116

Figure 58: Shows Number of Drones in Conflict against CPU Usage 118

Figure 59: Comparison of the number of transaction against the transaction price......................... 124

Figure 60: Cost distribution per AWS module.. 126

Figure 61: Summary of the cost analysis for the Weather Core Service.. 128

Figure 62: Example of messages used in Exercise 3.. 133

Executive Summary

The IMPETUS deliverable D5.2 presents the results of a series of experiments performed to validate the proposed information management solutions, exploring if the micro-service technology can be employed to design a technical architecture, which copes with the key challenges of U-space.

In its first deliverable (D2.1 Drone Information User´s Requirements [1]) IMPETUS identified the drone user's information needs and characterized this information using predefined use cases as reference. The second deliverable (D2.2 Drone Information Services [2]) aimed to detect invariant information management functions that a conceptual drone operation lifecycle shall deal with. From this comprehensive groundwork, IMPETUS selected a certain set of crucial U-space services for further exploration under a federated architecture framework.

The third deliverable (D5.1 Experimental Plan [3]) focused on describing the operational scope, assumptions and tools used for conducting the experiments of the selected services. Complementary, the fourth deliverable (D3.1 IMPETUS Architecture and Technical Requirements [4]) identified the main functional requirements for those services and contributed with the description of the technical architecture based on the micro-service technology approach. To validate the technical and economic feasibility of this solution, IMPETUS focused on specific research areas:

- Research on how to deal with the possibility of failure modes, which are affecting safety-critical services and thus, safe drone operations;
- Analysis of the impact on data management of an ecosystem of microservices having their own private database in safe drone operations, and in particular the challenges of dealing with data quality and integrity;
- Research on the scalability of the solution based on microservices taking into account the expected growth of drone operations;
- Research on the flexibility of the solution to assign computing resources as needed optimising the cost implications to the final U-space users;
- Analyse the commercial feasibility of a U-space microservice architecture, with special focus on the business model and how to proportionally bill the services provided.

Following the experimental plan defined in D5.1 and considering the functional requirements so as the architecture framework from D3.1, this deliverable shows to what extent the defined validation objectives and success criteria have been fulfilled. A great value has been placed in validating if the benefits of a microservice implementation (efficient workload management, inter-process communication, robust self-management) can leverage the information management implementation.

Finally, this deliverable concludes with the technical and economic feasibility assessment of the simulated set of microservices and provides a series of recommendations.

1 Introduction

1.1 Scope of the document

This document summarizes the experiment results gathered by the IMPETUS Consortium. The experiments follow a detailed plan previously described in IMPETUS deliverable D5.1 (Experimental Plan) [3], and adhere to identified service requirements covered in IMPETUS deliverable D3.1 (IMPETUS Architecture and Technical Requirements) [4] as well.

On the one hand, the presented results reveal to what extent the implementation of selected service prototypes can be supported by well-established micro-service patterns. On the other hand, the service prototypes themselves are validated based on objectives previously established. Subsequently, the results undergo a critical evaluation so that ultimately it is possible to make an assessment about the technical and commercial feasibility of the proposed solutions.

1.2 Intended readership

This document is intended to be used by IMPETUS members, the SJU (included the Commission Services) and the community of drone stakeholders in general.

1.3 Acronyms and terminology

UTM acronym is used in this document for the general notion of a drone traffic management system and not for the specific system, which will be designed in the USA.

Table 1: Acronyms

Abbreviation	Description
AD	Aerodrome
AGL	Above ground level
AIBT	Actual In-Block Time
AOBT	Actual Off-Block Time
AIP	Aeronautical Information Publication
AIS	Aeronautical Information Service
AIXM	Aeronautical Information Exchange Model
AU	Airspace User
AMQP	Advanced Message Queuing Protocol
ANSP	Air Navigation Service Provider

Abbreviation	Description
API	Application Programming Interface
BD	Big Data
BI	Business Intelligence
BVLOS	Beyond Visual Line of Sight
CAA	Civil Aviation Authority
CAGR	Compound Annual Growth Rate
CAP	Common Agriculture Policy
CNPCL	Control and Non-Payload Communication
DAA	Detect and avoid
DEM	Digital Elevation Model
DIA	Drones in Activity
DSM	Digital Surface Model
DTM	Digital Terrain Model
EASA	European Aviation Safety Agency
EIBT	Estimated In-Block Time
EGNOS	European Geostationary Navigation Overlay Service
ENR	En-route
EOBT	Estimated Off-Block Time
EUSCG	European UAS Standards Coordination Group
FAA	Federal Aviation Administration
GBAS	Ground-Based Augmentation System
GEN	General Aeronautical Information
GNSS	Global Navigation Satellite System
ICAO	International Civil Aviation Organization
ICD	Interface Control Documents
IFR	Instrument flight rules

Abbreviation	Description
ILS	Instrument Landing System
INS	Inertial Navigation System
LTE	Long term evolution
LUC	Light UA operator certificate
MC	Multicopter
NAA	National Aviation Authority
NASA	National Aeronautics and Space Administration
NDB	Non-Directional Beacon
NM	Network Manager
OGC	Open Geospatial Consortium
PA	Precision Agriculture
PIC	Pilot in Command
PSS	Public Safety and Security
RDP	European UAS Standardisation Rolling Development Plan
RMSTE	Region Management Simulation Test Environment
RPAS	Remotely-piloted aircraft systems
RTT	Research Transition Team
SAA	Special Activity Airspace
SARPs	Standards and Recommended Practices
SOP	Signal of Opportunity
SORA	Specific Operations Risk Assessment
SWIM	System Wide Information Management
TSA	Temporary Segregated Area
UAS	Unmanned aerial system
UA	Unmanned aircraft
UTM	Unmanned traffic management (general term)

Abbreviation	Description
UTM	Unmanned Aircraft System Traffic Management (USA)
USS	UAS Service Supplier
VLL	Very Low Level
VLOS	Visual Line of Sight
VOR/DME	Very high frequency Omnidirectional Range and Distance Measuring Equipment
WXXM	Weather Information Exchange Model

1.4 IMPETUS approach

IMPETUS deliverable D5.1 (Experimental Plan) [3] is a comprehensive elucidation of how selected set of U-space services can be technically implemented and interact with others under a common micro-service-based service architecture framework. The operational context of the experiments is detailed and individual validation objectives are specified. Scenarios for the experiments are defined and the different implementation architectures are described. All the conceived architectures are designed under the micro-service paradigm.

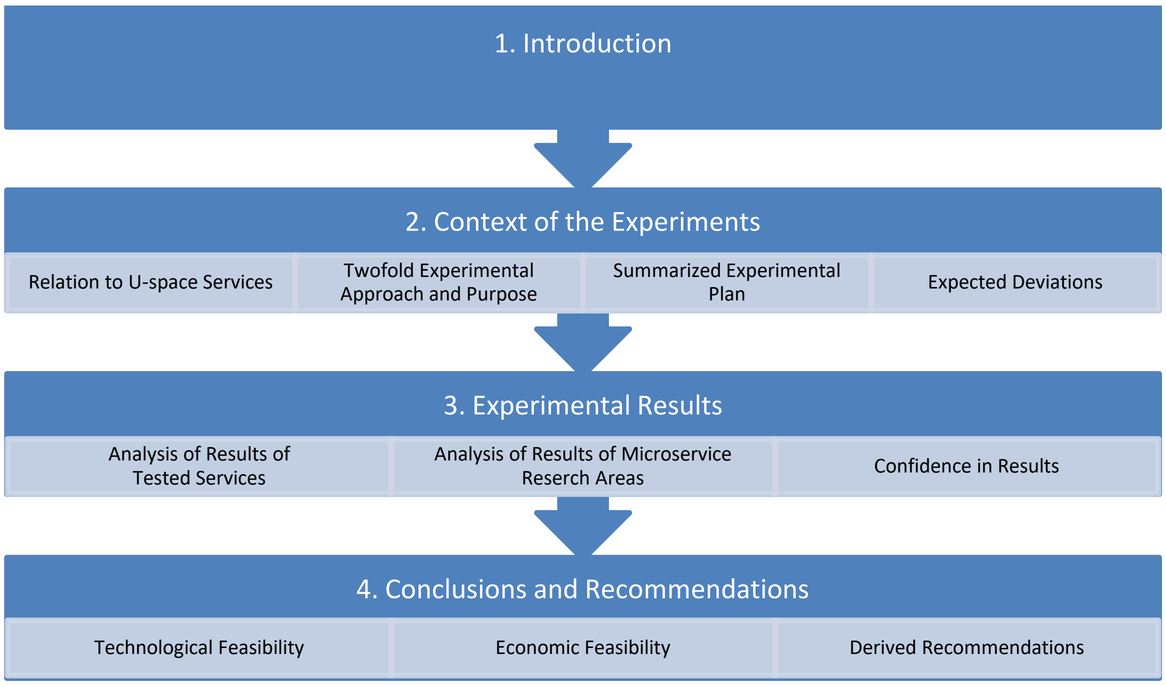
Additionally, IMPETUS deliverable D3.1 (IMPETUS Architecture and Technical Requirements) [4] presents a number of functional requirements for the U-space services in consideration. These requirements focus on the internal functions of the services and the mechanisms necessary for the interaction with further functionalities of the drone information management system. Furthermore, an extended description of the micro-service architecture and its common characteristics is provided.

This deliverable takes the two previous deliverables as a groundwork for setting the framework of the experiments and compiles the main findings of the examinations. Although an initial validation strategy is devised in D5.1, this is refined in this document in order to provide more precise conclusions and recommendations, especially those concerning the feasibility of the implemented solutions. In principle, the IMPETUS validation strategy has twofold:

- First, to **validate the requirements of the selected U-space services** together with the IMPETUS solutions to address the challenges of their implementation;
- Second, to research on how these services interact with each other within **the context of a federated architecture in which the microservices paradigm is implemented** as a good choice to address the challenges of the future U-space architecture.

1.5 Document structure

The following diagram provides a compact view of the document structure, showing the main document components:


1. Introduction
2. Context of the Experiments
Relation to U-space Services
Twofold Experimental Approach and Purpose
Summarized Experimental Plan
Expected Deviations
3. Experimental Results
Analysis of Results of Tested Services
Analysis of Results of Microservice Reserch Areas
Confidence in Results
4. Conclusions and Recommendations
Technological Feasibility
Economic Feasibility
Derived Recommendations

2 Context of the experiments

2.1 U-space services and capabilities: relation to the experiments

Following table shows how the exercises (description summarized in § 2.3) relate to the services and capabilities defined in the current U-space Roadmap [5]. Moreover, the level of relation is detailed as the exercises address services and capabilities to a different extent.

U-space services and capabilities		Exercise 1	Exercise 2	Exercise 2	Exercise 3
U1 services	E-registration				Dependency
	E-identification				Dependency
	Pre-tactical geofencing				
U1 capabilities	E-identification			Dependency	Dependency
	Geofencing				
	Security				Dependency
	Telemetry				Dependency
	Tracking			Dependency	Dependency
	Command & control				Dependency
	Emergency recovery				
	Communication, navigation and surveillance			Dependency	Dependency
	Operations management	Moderate	**Strong**		
U2 services	Tactical geofencing				
	Emergency Management			**Strong**	
	Strategic de-confliction		Moderate		Dependency
	Weather information	**Strong**	Moderate		Dependency
	Tracking			Dependency	Dependency

U-space services and capabilities		Exercise 1	Exercise 2	Exercise 2	Exercise 3
	Flight planning management	Moderate	**Strong**		
	Monitoring			**Strong**	Dependency
	Traffic information			**Strong**	Dependency
	Drone AIM		Moderate		Dependency
	Procedural interfaces ATC				
U2 capabilities	Tracking			Dependency	Dependency
	Emergency recovery				
U3 services	Dynamic geofencing				
	Tactical de-confliction				**Strong**
	Collaborative interfaces ATC				
	Dynamic capacity management				Moderate
U3 capabilities	V2V				
	Detect & avoid				
	V2I				

Table 2: Relation of exercises with U-space service and capabilities

2.2 Experiment Purpose and approach

The experiments conducted in IMPETUS served to address two main purposes: First, to **validate the requirements of the selected U-space services** elaborated in D3.1 which addressed the challenges of their implementation; Second, to research on how these services interact with each other within **the context of a federated architecture in which the microservices paradigm is implemented**, to evaluate if such an architecture is applicable.

In order to cover both aspects in the experiments, specific validation objectives and associated success criteria were established.

To assess the requirements of the selected U-space services, validation objectives were tailored to each individual exercise. No overlap between results related to individual services occurred between the experiments, as each exercise addressed different services. Each objective covered at least one of the identified requirements per scenario. Success criteria were established based on available metrics to be analysed and necessities of the validation objectives.

Validation objectives related to the testing of the federated microservice architecture were distributed over several experiments, as it was assumed that all aspects of the architecture would fall under the same architectural concept. Each exercise leader addressed those aspects of the microservice architecture which best fit the areas of expertise of his company, and proposed relevant success criteria and metrics. This approach was seen as the most suitable to cover the wide range of research questions that the consortium aimed to address in this project – namely the safety & failure modes of the system, impact on data management, scalability and flexibility of the solution as well as its commercial feasibility.

Once experiments were executed and results were gathered, the assessment of the success criteria was performed following SESAR Joint Undertaking guidelines for validation [6].

The evaluation of the defined success criteria served to summarize conclusions on the technical aspects of the tested U-space services as well as the findings on the applicability of the microservice-based architecture.

Requirements related to U-space services were then updated accordingly. Requirements related to the architectural solution were elaborated based on the conclusions gathered from the experiments.

2.3 Summary of the Experimental Plan

IMPETUS has selected crucial U-space services to be further explored via specific experiments, to not only address the requirements and challenges of them, but also to transversally explore the benefits of a U-space implementation based on microservices. Microservices – self-contained, de-coupled and individually deployed units of very specific services will allow for a fully automated drone operations lifecycle. All exercises will provide inputs to better quantify the benefits of this implementation, by proving that the microservice solution is scalable, efficient, secure and cost-effective enough to meet the U-space operational requirements, whilst meeting all the safety requirements from an ATM perspective.

- Exercise#1 illustrates how a **better knowledge of the uncertainty in the meteorological prediction** – as this is the most important novelty with respect to the state of the art, will improve the robustness of trajectory-based decision making processes underlying Mission Plan, Flight Planning and Traffic Management services;
- Exercise#2 investigates how the Flight Planning Management Service can be used as a controlling entity for flight plan submission and strategic deconfliction. The exercise will show how to guarantee **the successful and safe adaptation of the initial submitted flight plan** when the Flight Planning Management Service interacts with the Local Weather service, the Aeronautical Information Management service and the Mission Plan Management Service;
- Exercise#3 is designed to explore how dynamic information, especially surveillance data and the positions of obstacles, is gathered, integrated and provided to all the actors involved in the operation. However, in this document an alternative exercise will be considered, focused on testing the performances of a microservice architecture (delays, failure modes, capacity…) considering the capabilities to develop **automated services to process traffic information**

from multiple users and sources, offering situational awareness and alert messages in case of detecting non-conformances;

- Exercise#4 explores the services, which are needed to dynamically manage the airspace in the execution phase, considering both the standard separation criteria proposed by CORUS, and new separation criteria that consider diverse drone capabilities. Furthermore, the exercise will assess how to deal with **dynamic changes in airspace restrictions** by determining the safest path for the affected drones to take.

The table below lists the different experiments performed within IMPETUS. Different criteria were defined to be addressed within the experiments. Categories represented in the columns are the environment, EASA operation category, traffic density, the drone automation level, the operation altitude as well as if the experiment is considering operations in controlled or uncontrolled airspace.

EXE#	Name	Environment - Rural (R) - Suburban (S) - Urban (U)	EASA Operation category - Open (O) - Specific (S) - Certified (C)	Density - Low (L) - Medium (M) - High (H)	Drone Automation level [0;5]	Altitude - VLL - Above VLL (AVLL)	Controlled airspace
EXE1	Drone-specific weather service	R/S	O/S/C	L/M/H	0-5	VLL/AVLL	BOTH
EXE2	Mission and flight planning management	S/U	O/S/C	L/M	4	VLL	BOTH
EXE3	Monitoring and traffic information	R/S/U	S/C	L/M/H	4	VLL/AVLL	BOTH
EXE4	Tactical deconfliction and airspace capacity	R/S/U	O/S/C	L/M/H	4	VLL	BOTH

Table 3: Scenarios and scope of the exercises considered

2.3.1 Objectives and success criteria to validate the tested services

The objectives to validate the tested set of microservices have been described in detail in the Experimental Plan document [3]. For most of the experiments these objectives and associated success criteria and metrics have remained the same, although for other experiments the metrics have slightly changed (see § 2.4). Therefore, a summarized overview is provided in Appendix A.

2.3.2 Objectives and success criteria to validate the federated microservice-based architecture

The objectives and associated success criteria presented in this section were elaborated for the validation of the federated microservice-based architecture taking into account the research areas identified in D3.1 [4]. The following table provides a traceability of the examined research areas to the different services tested. The areas address research questions relevant to conclude on the benefits of the microservice architecture implementation. In similar fashion as in the previous section, a summarized description is given in Appendix B.

Research area	EXE 1	EXE 2	EXE 3	EXE 4
Safety & Failure modes	X		X	X
Data management		X	X	X
Scalability	X		X	X
Flexibility			X	
Commercial implications	X	X	X	

Table 4: Relation of the exercises with the considered research areas

The next subsections detail the tests performed in each exercise to cover the research areas in the table.

2.3.2.1 Safety and Failure Modes

In this area, we researched on how to deal with the possibility of having failure modes, which are affecting safety-critical services and thus, safe drones' operations. We also identied mitigation actions such as services' duplication, real-time detection mechanisms or restores of services automatically. In addition, dedicated resilience tests for analyzing failure propagation and the inherent advantages of a microservices oriented architecture in this regard are explored.

Resilience tests for the Weather Service (EXE 1)

The weather service is expected to produce a large amount of data that will flow to the different clients through a publish/subscribe middleware and will consume real-time weather reports from many UAVs. Not only the number of clients consuming and reporting data will be highly dynamic but it would also vary in usage patterns. This requires an architecture for data management that allows each client to set its requirements, receive the service according to it, and guaranting that the continuous data flow that each client requires is not disrupted by the actions of other clients. Thus, user' actions should only affect its own data flows. The data management system has to take into account that some clients and reporting UAVs will misbehave in different ways (consume too slow from queues, report corrupted data, make malformed petitions, or simply disconnect abruptly). Ideally, data flows of the others should remain mostly unaffected.

In this exercise, one hundred different clients requesting weather information of partially overlapping areas have been simulated using sintethic data. Concurrently, another hundred are reporting real-time weather data over that area. Three runs are performed: ideal, realistic and pessimistic scenarios.

In the ideal one, every client behaves according to canon. There are also three different pessimistic scenarios with different rates of clients misbehaving. The measured metrics relate to the well-behaved clients:

- Success rate in completing service of well-behaved clients with zero percent misbehaving actors (Ideal scenario);
- Success rate in completing service of well-behaved clients with 10% percent misbehaving actors (normal client`s error rate scenario);
- Success rate in completing service of well-behaved clients with 25% percent misbehaving actors (high client`s error rate scenario);
- Success rate in completing service of well-behaved clients with 50% percent misbehaving actors (very high client`s error rate scenario);
- Success rate in completing service of well-behaved clients with 90% percent misbehaving actors (extreme client`s error rate scenario).

In order to tests the benefits of a micro-service approach and data flow the functionality of the micro-weather service has been packaged with different approaches, from purely monolithic to modular following the microservices architectures principles.

These approaches pack the functionality differently, from a single service instance offering all interfaces to clients to a fully compartmentalized one, running a dynamic amount of Docker containers:

1. Fully monolithic: Single service running all clients' interactions plus a single broker for clients;
2. Separation in two services: Gateway for service configuration and Single client agent for managing all clients (threaded);
3. Fully micro: Gateway for service configuration and an independent container with a single agent for one client;
4. Fully micro plus exclusive broker per client: Same as before but adding a separate light AMQP broker to every agent container instead of external broker.

For testing the resilience of the different architectures, a synthetic test scenario has been designed.

In the test, 100 processes will simulate clients asking for weather data within a certain area concurrently.

They follow the essential steps described above (request to gateway, request to agent, and consume from their queue). A client that is able to read for 5 minutes and requests to its agent the end of the service is considered a success.

Every thread is continuously executing petitions so that there are always 100 concurrent clients in total. That is, right after finishing one simulation of a client (or failing), a new one will start.

The computational core of the weather service is simulated by a series of dummy containers that fill their corresponding queues with synthetic data that simulates the prediction data at a rate of 10 packages per minute of 500KB each.

Different runs of this setup are executed varying the percentage of faulty clients. Every run lasts 20 minutes and takes place in a server running Ubuntu18.04 with 28 Intel Xeon cores (HT activated), 256 GB of RAM. All docker images are also built over the basic Ubuntu 18.04 image.

The four different approaches where tested at 0%, 10%, 25%, 50% and 90% of faulty clients.

Validation Objective id:	**SFM-1: Robust Weather service data management**
Validation Objective Description	Demonstrate capability to correctly manage and supervise data flows even in the presence of misbehaving actors.
Success Criteria 1	Ideal scenario should have over 99% of clients correctly served in proposed architecture.
Success Criteria 2	Normal scenario should have over 95% of clients correctly served in proposed architecture.
Success Criteria 3	High scenario should have over 95% of clients correctly served in proposed architecture.
Success Criteria 4	Very High scenario should have over 90% of clients correctly served in proposed architecture.
Success Criteria 5	Extreme scenario should have over 90% of clients correctly served in proposed architecture.
Success Criteria 6	In all scenarios, the proposed architecture should be more resilient than the monolithic approach.

Table 5: Validation objective description of test SFM-1

Service availability in congested scenarios for the Traffic Information Service (EXE 3)

This exercise 3 analyses the performance of data management in certain congested scenarios in which the interactions (separation) between drones must be detected, providing an answer to users notifying about dangerous situations. On the other hand, the architecture must be self-monitored to detect failure modules and a replacement (duplicated functionalities) shall be provided to ensure the availability of these services in this real-time environment.

In this case, a microservice-based architecture has been tested by means of a set of predefined trials that will consider the overload, the failure of one of the services, the recovery of the platform (imitating the failure modes expected in U-space and ensuring the availability of certain services) and the behaviour of the platform due to increases in demand. In addition to this, Traffic Information Service and Emergency Service have been chosen as critical services envisioned in a congested airspace, which will be reproduced in this platform.

The following metrics have been measured:

- (AD) Performance over time/delay of the alert messages (seconds) considering the interval between the positions being processed and the alert shown to the U-space user;
- (ID) Delay of information due to service failures (seconds);

- (FD) Time to detect the failure of the service (seconds);
- (RT) Recovery time to execute another instance of the service (seconds);
- (AM) Alert messages shown (% alerts/incidents);
- (IC) Instances created per service (number of services);
- (ML) Maximum workload admitted per service (%);
- (AV) Availability of the information (%).

Validation Objective id:	**SFM-2: Service availability due to failure module implementation**
Validation Objective Description	Assure high service availability by a self-monitoring failure detection module to identify and replace failed services in the architecture.
Success Criteria 1	In critical services, the delay of information due to failures in the execution of a service shall be less than 2s. (ID)
Success Criteria 2	A platform focused on information management for U-space shall ensure that, in case of a service failure, a backup microservice for critical services shall be deployed in less than 1s. (RT)
Success Criteria 3	A platform focused on information management for U-space shall ensure that the failures in services must be detected within 0.1s since the issue happened. (FD)
Success Criteria 4	A platform focused on information management for U-space shall be capable of dealing with increases in demand creating, at least, an additional instance of the same functionality to load the balance and prevent its overload (IC).
Success Criteria 5	The maximum workload of a service shall be less than 80% of its capacity, providing that a new instance will be created when overloads above this percentage are detected.

Table 6: Validation objective description of test SFM-2

A more detailed description of this exercise will be provided in § 2.4.3 addressing all these validation objectives and the ones defined that deal with the Traffic Information Service.

Failure mode mechanisms in the Traffic Management Service dealing with dynamic airspace management (EXE 4)

In the context of building scalable, distributed aviation systems, particularly on cloud technology, it becomes essential to provide deterministic failover during fault conditions, and to design the tactical deconfliction service in such a manner that modes of failure are largely deterministic, so that appropriate mitigations can therefore be built in.

For distributed systems such as the ones described by IMPETUS, it is important to consider geographic separation and the effects that this will have on replication, disaster recovery, and latency during failover. Modern public cloud platforms make it relatively simple to deploy services within multiple geographies, but it is also important to consider the service design "in-datacentre", such that individual nodes handling compute and storage can fail in a deterministic manner.

Distributed systems therefore require special considerations for both application and infrastructure level monitoring that go beyond traditional "load" monitoring (although this aspect is still a requirement), and the ability for applications and services to emit key performance metrics that are service-based so that proactive measures to correct early signs of degraded performance can be taken.

Based on these considerations, this exercise will test the following failure modes:

- Failure mode of dependent services (disable the hyper-local weather service);
- Failure of the core tactical deconfliction services (disable service in active datacentre);
- Failure of the situational picture;
- Failure of the transformation;
- Failure of the C2 link.

The following metrics are tested:

- % Separation maintained;
- Time (in seconds) to resolve updated dependent services (e.g. change in weather provider);
- Alerts Shown;
- Time (in seconds) for automatic failover of core tactical deconfliction services.

Validation Objective id:	**SFM-3: Robust micro-service architecture against different failure modes**
Validation Objective Description	To test a failure in any services that Tactical Deconfliction is dependent on, as well as how failures are handled in components of the deconfliction service.
Success Criteria 1	Ability of the system to switch provider if a service Tactical Deconfliction is dependent upon fails, or increase separation to the maximum level while it recovers. Both approaches ensuing that separation is maintained.
Success Criteria 2	Ability to recover from an internal service failure in line in 10 seconds.

Table 7: Validation objective description of test SFM-3

2.3.2.2 Data Management

This area analyses the impact on data management of an ecosystem of microservices having their own private database in safe drones' operations, and in particular the challenges dealing with data quality and integrity.

Adequate processing of user requests and data consistency and assessment of the processing time in the Flight Planning Management Service (EXE 2)

The strong link between the Mission and Flight Planning Management services may represent a performance bottleneck in terms of data flow. Therefore, it is paramount to examine carefully the data management in both services. For instance, the data management within the FPM service (owning a centralized flight plan database) is an important object of investigation that shall ultimately corroborate the choice of a federated software architecture approach of the FPM service. Own data management is a well-known pattern in micro-service based applications and it carries a number of benefits at running time. However, special care must be placed in tracing and monitoring the data flow performance in order to assure that the flight plan data integrity is given at any time.

Apart from examining the data flow performance, it is important to demonstrate that the chosen structure for the data management has cost benefits for all the stakeholders involved. Specifically concerning the integrated mission planning, the coordination of the services involved shall support and maximize the competition in a marketplace where different DTM providers can offer specific capabilities suitable for a different number of business models.

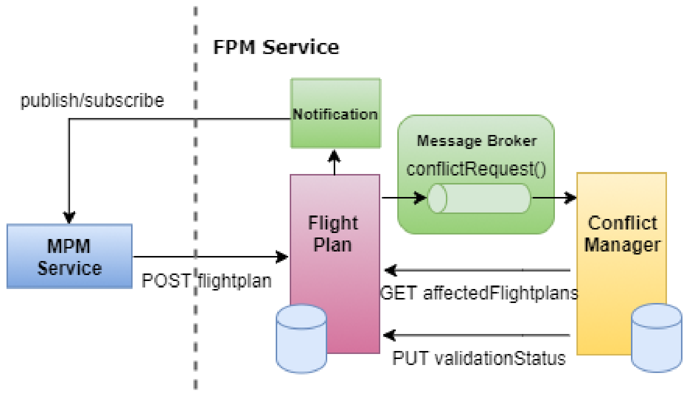


Figure 1: Architecture of the FPM Service and interaction with the MPM Service

Figure 1 represents the data flow within FPM service, so as the methods that the microservices use for communicating with each other. Monitoring the MPM and FPM service interaction through the simulations performed for the test scenario, the following data management aspects are examined in this exercise:

- Robust data exchange between the MPM and the FPM services: The data flow that takes place after a flight plan is initially submitted to the FPM until the MPM receives a notification is complex and involves the coordinated collaboration of various microservices. In this exercise the proper transmission of the flight plan data across the microservices is traced back from the logged data and consecutively analysed;

- Consistent flight plan data in the centralized database: The database in the Flight Plan service can be considered as single point of truth for validated and officially approved flight plans. Therefore, it is key to assure consistent data across the available datacenter. Additionally, it will be monitored in this exercise that the data has a high level of availability;

- Reliable data delivery in the cloud messaging capability: The Conflict Manager microservice employs an algorithm for detecting flight trajectory intersections in the spatial and temporal domain. This is a process that is highly time-variant and the implementation of a synchronous communication mechanism is not practicable in this case. Therefore, a message broker queuing requests has been implemented. Through the monitoring of the message broker it can be supervised that the messages are delivered to the Conflict Manager microservice;

- Equitable distribution of data management in the enhanced mission planning: Within the proposed architecture, the computing resources costs for planning a drone mission shall be distributed in both the Mission Planning Management and Flight Planning Management services. The MPM service assumes the role of provider of mission-relevant information and ensures therefore a high level of safety and awareness to the operation. On the other side, the FPM guarantees the coordination of all submitted flight plans to the U-space environment and holds the single point of truth of the approved flight plans.

Valuable metrics supporting the validation objectives are the following:

- Request processes simulated (RP) – (number/exercise);
- Success rate in the requests to the FPM service (FPM) - (% requests);
- Uniqueness in the flight plan data sets (UQ) – (number of records in the database for each flight plan);
- Availability of the service in uptime (AVS) - (%);
- Success rate in the message queue requests (MQ) - (number/operation);
- Ratio in the request duration between the MPM and the FPM service (RRD) - (% of processing time distribution).

Validation Objective id:	**DM-1: Robust flight plan data exchange**
Validation Objective Description	Demonstrate that all flight plan data requests are processed by the FPM service and all the data is correctly transmitted.
Success Criteria 1	99% success rate in the data exchange between the MPM and FPM services (RP) (FPM)
Success Criteria 2	The message queue delivers over 99% of transmitted requests for all the scenarios considered.
Validation Objective id:	**DM-2: Consistent and available flight plan data**
Validation Objective Description	Demonstrate that the flight plan data samples remain consistent and available through the complete validation process.
Success Criteria 1	No duplicate flight plan records exist in the database (UQ)
Success Criteria 2	Uptime of the service is at least 99% (AVS)
Validation Objective id:	**DM-3: Equitable distribution of data management**
Validation Objective Description	Quantitatively measure the computing time distribution required for the mission planning and validation process.
Success Criteria 1	The Mission Planning Management service has a ratio of at least 50% in the complete processing time (RRD)

Table 8: Validation objective description of tests DM-1 to DM-3

Efficient data processing in the Traffic and Monitoring Services (EXE 3)

Considering the number of sources (drone operations) to be included in the information management process sending their position it is expected that a platform that will process these datasets will be impacted by increases and decreases in demand, affecting its performances. On the other hand, the internal processes to exchange the information between all the modules of the architecture and to provide a valuable answer to the user are also a key part of the experiment, as they will add delays. Moreover, the system shall be prepared to react to overloads that excess a certain percentage of the capacity of the service, offering new modules with the same functionality and balancing the load.

These aspects of information management are vital in the services addressed, as the availability requirements, flexibility in sudden increases and effective use of resources, together with a monitoring system shall be integrated to provide high-quality service even in the direst of circumstances. The integration of parallel processing and duplication of key services are also assessed, as a way to compensate these rises in the number of drones (and information datasets) without adding hard processes and complex links to the architecture.

For these specific topics, the metrics tested are:

- (MP) Percentage of messages processed (%messages);
- (PD) Delay of a position message since it is acquired by the system and is processed with the other flights in the same area (second);
- (PR) Delay of a position message since it is acquired by the system, processed with the other flights in the same area and provided to users (second);
- (AD) Performance over time/delay of the alert messages (seconds) considering the interval between the positions being processed and the alert shown to the U-space user;
- (MD) Maximum delay of the information considering the number of flights processed (seconds).

Validation Objective id:	**DM-4: Data capacity and performance in traffic information service**
Validation Objective Description	Demonstrate that a micro-service based architecture has enough capacity to process the huge amount of inputs expected in congested environments.
Success Criteria 1	The platform shall be able to process, at least, 99% of the messages collected. (MP)
Success Criteria 2	The maximum delay that can be admitted by a U-space information management system shall be 5s when considering traffic information service (processing the data, integrate it with the rest of the operations and provide the view to the users). (MD)

Table 9: Validation objective description of test DM-4

Up-to-date situational representation of the data in the Traffic Management Service dealing with dynamic airspace management (EXE 4)

Closely linked to performance and reliability, it is common in micro-services architectures that multiple redundant nodes store their own copies of state data. Thus, the primary concern is often one of consistency: data replicated across 'n' nodes is often considered too complex, or even unnecessary, to ensure all independent copies are up-to-date and globally consistent within a single operation transaction (i.e. instantaneous consistency).

Instead, commonly, massive-scale systems consisting of diverse micro-services typically embrace "eventual consistency": the fundamental concept is that at some point throughout a defined period, all copies will be consistent. To make this work in practice, however, services to 'affinitize' users with nodes that store data logically consistent with their use of the service is important. Facebook famously employs such mechanisms as so do most major service providers. In this cases for example, a service deconflicting traffic in and around London, would need to have the most up to date version of information for its area, with for example backup nodes based in Amsterdam being able to reach the consistent state at a higher latency.

Based on these considerations, this exercise tests "eventual consistency" in the following manner:

- By running a tactical deconfliction test and utilising audit logs to confirm that the most up to data was used in relevant calculations.

The following metric is tested:

- The availability of current data in the tactical deconfliction process at the point where active decisions are calculated.

Validation Objective id:	**DM-5: Data consistency in tactical deconfliction service**
Validation Objective Description	Ensure data set is up-to-date when internal service decisions are made from it.
Success Criteria 1	An up to date situational picture at all times for the area being monitored.

Table 10: Validation objective description of test DM-5

2.3.2.3 Scalability

This area will research on the scalability of the solution based on microservices taking into account the expected growth in the drones' operations and the utilization areas. It will be also analysed how well the services can be automatically scaled to manage maximum possible capacities.

Scalability of the Weather Service in dense utilization areas (EXE 1)

The scalability of any architecture of a service that aims to be offered to a potentially high number of clients has two distinct aspects: First, the basic one, the service should be able to serve the expected maximum of concurrent clients, that is, the service should be able to scale so that demand is always met; Second, to be economically viable, the cost should scale as much as possible with the present workload of the service at any time, that is, ideally, the service architecture should be such that the resources are instantiated as required as any part that is running constantly regardless of the load would be an extra cost.

Of course, there are parts of a service that, due to its nature, cannot be scaled. Some can be shared between many clients and others require exclusive use. Identifying all these parts and when possible, instantiating them just when required, depending on the present demand, is key in the design of any service architecture to make it economically viable. As we will try to test with the experiments described in this section, the micro-service approach in conjunction with the protocols that are used to coordinate the different building blocks, usually allow for good scalability.

The micro weather service can potentially serve to very large number of clients, even reporting weather during flight to UAVs. This means that the need to be able to scale, and to do it in a cost sensitive way, is essential for the service. The same can be said about its capability to ingest real-time data incoming from UAVs.

The architecture of the service has been designed to be able to accommodate this by a set of micro-services that are created dynamically and that sit between the core weather model nodes and the clients. These instances accommodate the peculiarities of every client, isolate them from each other and process in and out dataflows of the core instances.

The core instances requirements are only related with spatial volume of the area to cover and the frequency predictions. On the other hand, the ability to scale to high number of users in a particular area is mostly mandated by the capability of the service gateway to create these instances following demand from the users. Consequently, the critical case would be for areas with a high number of UAVs. Of course, the same capability should exist to increase the number of instances dynamically that are able to process incoming real-time weather sensor readings.

In this exercise it is used synthetic data to serve an increasing number of clients making partially overlapping petitions of weather information and it is measured the computing and bandwidth resources used by the service to accommodate the demand. The maximum number of concurrent clients will be increased from zero to one thousand, or until our testing hardware infrastructure is able to support it. The testing platform mimics the typical servers offered by cloud providers, having 28 cores, 56 concurrent threads, 256 gigabytes of RAM. The use of these resources is expected to be close to linear with the number of concurrent users.

The measured metrics and success criteria for this experiment are:

- RAM usage at different number of users from zero until one thousand;
- CPU utilization at different number of users from zero until one thousand;
- Bandwidth utilization at different number of users from zero until one thousand.

Validation Objective id:	**S-1: Scalability of number of users for the weather service in dense utilization areas.**
Validation Objective Description	Sufficient RAM per server, sufficient CPU per server and sufficiency of bandwidth.
Success Criteria 1	At least 500 users served concurrently with 256 GB installed memory.

Validation Objective id:	S-1: Scalability of number of users for the weather service in dense utilization areas.
Success Criteria 2	At least 500 users served concurrently with two Xeon Gold processors (56 concurrent threads total).
Success Criteria 3	At least 500 users served concurrently with 1 gigabit/s of bandwidth.

Table 11: Validation objective description of test S-1

Self-management in the Traffic Information Service (EXE 3)

This exercise increases the demand and the necessity of replicated modules (with the same functionality, balancing the workload of the system) will be provided by the architecture automatically. This test analyses how this prototype provides a certain level of service depending on the number of data sources considered and how these changes affect the whole system.

This experiment provides a predefined architecture, consisting on a maximum of five virtual machines that will be activated/deactivated in terms of the demand. These virtual machines are capable of deploying containers and instances of the services based on the workload of the existing ones and is completely adaptable to demand. Moreover, the microservices developed are atomic. This implies that the processing load should not be significant, so huge amounts of positions sent by simulated drones will be collected by the platform and its behaviour analysed. It will be considered the maximum workload admitted and predefined per service (to ensure that the creation of new instances is executed before its failure), the duplication of modules to process the information, the reaction time to these events and the management of possible failures in the service execution that will lead to rebalancing the load while a backup system is launched.

The metrics that are measured to highlight this are:

- (FD) Time to detect the failure of the service (seconds);
- (ST) Setup time to execute another instance of the service (seconds);
- (IC) Instances created per service (number of services);
- (ML) Maximum workload admitted per service (%);
- (LR) Reduction of delays due to workload balance.

Validation Objective id:	S-2: Self-management of the system in response to increases in demand
Validation Objective Description	Creation of microservices related to U-space Traffic Information service due to increases in demand.
Success Criteria 1	The platform shall create at least 1 microservice when workloads close to the maximum limit are detected. (IC)
Success Criteria 2	The delays of information must be reduced to, at least half of its value when another microservice is deployed. (LR)
Success Criteria 3	Additional microservices shall be developed within 5s after the workload is detected. (ST)

Validation Objective id:	S-2: Self-management of the system in response to increases in demand
Validation Objective Description	Instantiation of a backup microservice instead of one that has failed due to overload.
Success Criteria 4	The failure must be detected within 0.1s since the issue happened. (FD)
Success Criteria 5	There will be at least 1 extra microservice instead of the one that has failed. (IC)
Success Criteria 6	Backup microservices shall be deployed within 3s since the failure was detected. (ST)

Table 12: Validation objective description of test S-2

Automatic scaling of the Traffic Management Service dealing with dynamic airspace management (EXE 4)

Most cloud-scale systems employing microservices architectures are designed around individual "units" of performance or "compute". Each compute unit is designed to contribute a known amount of capacity of the overall solution. Thus, when designed in this way, the solution is able to achieve 'linear scale': the ability to simply turn on additional compute units or nodes when load increases, and to turn them back off again when the load decreases.

This ability to self-size the cluster enables the system to also be run more efficiently than most typically designed enterprise IT systems, which are often built to specifications of anticipated "peak" load, at significantly higher cost, when in reality they very rarely have to handle such peaks.

For the drone industry, some basic conclusions can be drawn to help inform the scalability decisions up-front. For example, the UK handles approximately 5-6k flights per day, for manned aviation. It is reasonable to assume that there will be at least one order of magnitude greater flights taking place within the lifetime of an initial deployment for unmanned aircraft, thus the system should start with an approximate loading of between 50-60k flights per day, or 41 flights per minute.

A typical resilient cloud system is distributed across at least two data centres, with a minimum of two compute nodes in each. If we assume these data centres are configured in active/passive modes (one is the primary, one is a failover), then 41 flights per minute need to be distributed across two nodes in a single datacentre, thus each node in the cluster should be able to contribute to the overall performance by handling about 20 flights per minute.

There are many factors that will affect this calculation, including necessarily the services actually being delivered to the drone. For example, "simple" navigation services are much less bandwidth-intensive than video encoding services. Thus, the microservices architecture enables compute nodes to handle different jobs, and therefore be scaled independently of each other based on the requirements of the overall system at the time.

Based on these considerations, this exercise tests service scalability by utilizing "compute units" in the following manner:

- Breaking down the simulated environment and continually increasing the number of drones in the system to the point of saturation.

The following metric is tested:

- Maintaining the rate of the de-conflictor analysis (in Hz) against an increasing number of drone flights in the experimental area (specific metrics will be captured during the experiments as a base line.

Validation Objective id:	**S-3: Automatic scaling of the cloud system microservices**
Validation Objective Description	To ensure the system automatically scales to manage the UAV load within the maximum possible capacity of the test area.
Success Criteria 1	Ensure the system is never constrained by processing load of UAVs being managed but instead only by the environmental conditions being faced.
Success Criteria 2	Ensure the system is not over provisioned (to ensure commercial efficiency).

Table 13: Validation objective description of test S-3

2.3.2.4 Flexibility

This area will research on the flexibility of the solution to assign computing resources as needed optimising the cost implications to the final U-space users.

Self-monitoring in the Traffic Information Service (EXE 3)

This test will research into a self-monitoring architecture that is constantly measuring the demand of resources of each one of the microservices deployed to adapt the number and workload to a minimum use of resources with the minimum effect on the performance, creating the proper set of microservices to respond to sudden increases and permanent flow of information.

Using synthetic information, fluctuations in data stream are executed and the expected reactions of the platform are detected, analysed and evaluated. Series of 10-20-30-20-10 operations are executed, increasing and decreasing the workload of the infrastructure and making the platform to create new microservices, close them and, in summary, adapting its own architecture to fulfil the requirements of a changing environment.

The metrics that address these topics are:

- (IC) Instances created per service (number of services);
- (IR) Instances reduced per service (number of services);
- (ML) Maximum workload supported per microservice (% of the capacity).

Validation Objective id:	F-1: Self-management of microservices.
Validation Objective Description	Address the processes of self-management in a microservice architecture to respond to fluctuations in demand.
Success Criteria 1	The maximum workload of a microservice shall be less than 80% of its capacity, to avoid peak increases that will lead it to a failure. (ML)
Success Criteria 2	Once the workload of a microservice reaches its maximum rate, another instance of the service shall be created. (IC)
Success Criteria 3	In off-peak periods, the number of microservices shall decrease in, at least, one instance. (IR)

Table 14: Validation objective description of test F-1

As it was mentioned, more details about this exercise are described in section § 2.4.3 of this document.

2.3.2.5 Commercial Implications

Lastly, this area will analyse the commercial implications of a U-space microservice architecture with special focus on the business model and how to proportionally bill the services provided e.g. according to the computing resources required to satisfy the requests. Since this is an analysis using different approaches, the results are directly shown in § 3.3.5.

2.3.3 Assumptions

The main assumptions adopted for the conducted experiments have been summarized. These assumptions cover the following areas:

1. CNS technologies are available and provide the required performances in support of operations, including the availability of technologies that should be in place to implement the set of services to be tested;
2. Service pre-requisites are in place, including those features that should exist to ensure that the proposed service can be implemented, e.g. enough local weather sensors are necessary to put in place the drone-specific weather service;
3. Operational environment is defined, including the description of the operational context in which the service will be implemented, e.g. separation criteria to implement tactical deconfliction service;
4. Other U-space services (other than the ones tested in IMPETUS experiments) are in place, including other services, functionalities, databases that should exist to properly implement the service to be tested, e.g. consistent database with UAVs performance models;
5. Regulatory framework is defined, including relevant regulations that should be in place for the implementation of the services;
6. Implementation architecture is compatible with the micro-service-based approach, e.g. direct relation of the service with the core modules of the microservice architecture (Service Discovery, API Gateway, etc.).

The detailed description of the main assumptions can be found in the Appendix C.

2.4 Expected deviations of planned experiments

Additional to the experimental plans summarized in the previous section, further tools and tests mechanisms are described in this section, which are necessary to understand the coverage of the experiments. On the other hand, deviations from the initial planning in D5.1 are also provided in this section.

2.4.1 Drone-specific Weather Service (EXE 1)

The main objectives of this exercise are the following:

1. *Test a prototype probabilistic micro-weather forecasting service implemented in accordance with the IMPETUS microservices architecture*. We have deployed the service in a cloud environment and run it in different conditions simulating a variety of relevant operational scenarios. We have also conducted experiments to assess the scalability and resilience of the service;

2. *Assess the potential benefits of using the probabilistic micro-weather forecasting service to support drone mission planning in U-Space.* We have run experiments where an emulated mission planner interacts with the service to request micro-weather forecasts that are used as inputs to predict the trajectories that will be flown by the drones in order to perform their required missions. We have then compared the resulting predicted mission trajectories with those that would be obtained using a deterministic low-resolution weather forecast, which represents current aviation weather forecasts.

The information flow process that has been tested in this exercise remains as the one described in D5.1 [3]. In order to conduct the exercise, a prototype micro-weather service was deployed following the microservice-architecture concept proposed in IMPETUS. The service was tested in a simulation environment that allowed it to interact with an emulated mission planner and provided the required data to represent a plausible future drone mission-planning scenario.

The details of this test environment are provided in this section, which is structured as follows: First, the architecture and main features of the prototype micro-weather service tested in the exercise are presented; Second, the simulation environment used to conduct the exercise is described, including the type of vehicle considered, the missions to be assessed and the main assumptions of the analysis; Finally, the different scenarios that were studied as part of this exercise are presented.

Micro-weather Service Prototype

Figure 2 below shows a schematic representation of the high-level architecture and main data flows of the micro-weather service prototype that was implemented and deployed to conduct the exercise. Following the IMPETUS microservices architecture and in accordance with the design principles outlined in Deliverable 4.1, the service decouples the computation of weather forecasts from the provision of information extracted from those forecasts to the service's clients.

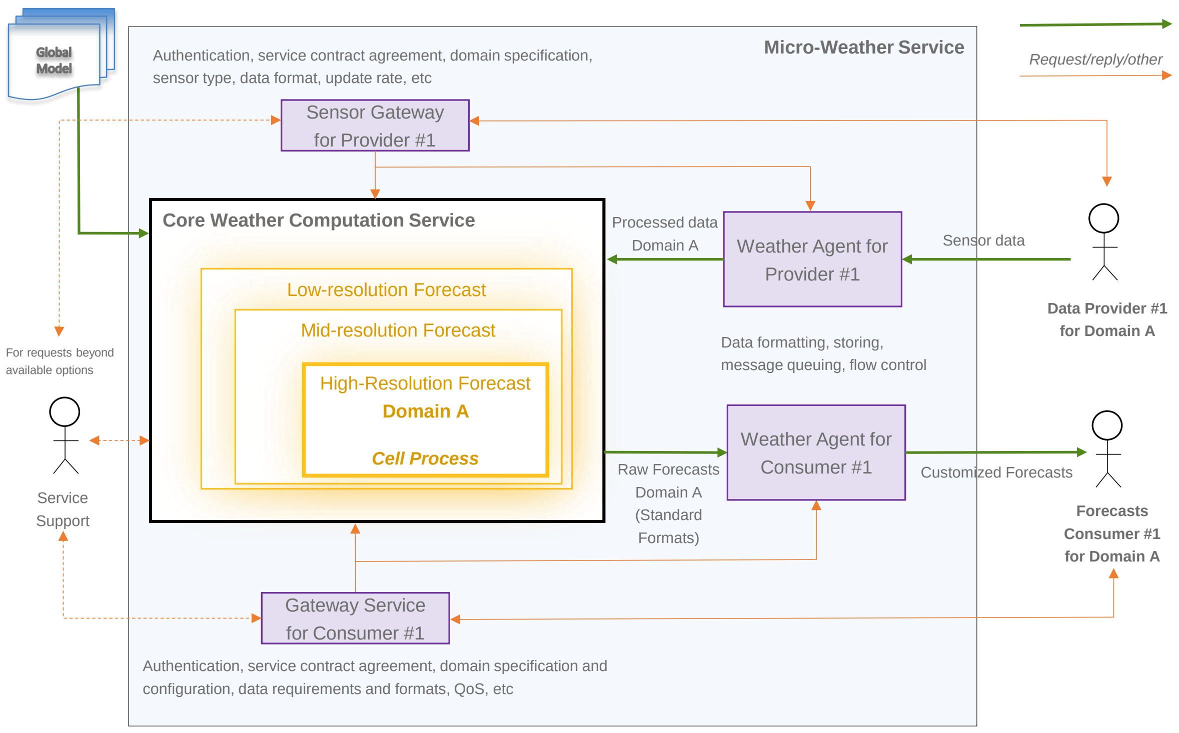

Global Model
Micro-Weather Service
Request/reply/other
Authentication, service contract agreement, domain specification, sensor type, data format, update rate, etc
Sensor Gateway for Provider #1
Core Weather Computation Service
Low-resolution Forecast
Mid-resolution Forecast
High-Resolution Forecast
Domain A
Cell Process
Processed data Domain A
Weather Agent for Provider #1
Sensor data
Data Provider #1 for Domain A
Data formatting, storing, message queuing, flow control
For requests beyond available options
Service Support
Weather Agent for Consumer #1
Raw Forecasts Domain A (Standard Formats)
Customized Forecasts
Forecasts Consumer #1 for Domain A
Gateway Service for Consumer #1
Authentication, service contract agreement, domain specification and configuration, data requirements and formats, QoS, etc

Figure 2: Micro-Service Prototype high-level architecture and main information flows

The Micro-Weather Service has two types of clients: Forecasts Consumers and Data Providers. Forecasts Consumers are clients that request meteorological forecast information from the service. Data Providers are clients that interact with the service to deliver real-time data from meteorological sensors. The data is to be used by the service to improve the accuracy of its forecasts. A Forecasts Consumer can also be a Data Provider.

When a Consumer connects to the service in order to obtain meteorological data, a Gateway Service assigned to that specific Consumer is instantiated by the micro-weather service. The Gateway Service manages all the interactions of its Consumer with the micro-weather service and is in charge of marshalling the required resources to fulfil the Consumer's requests. Among other tasks, the Gateway Service is in charge of establishing and managing a service agreement with the Consumer, including data requirements, domain of interest, Quality of Service, security and authentication. Thus, the Gateway Service provides the tools to enable the Consumer to define the domain for which meteorological information is required. This domain includes a geographical area of interest together with the time interval for which the data is required. In addition, the Gateway Service allows the Consumer to define the specifics of the data it requires, including the format and granularity of the meteorological variables requested from the service. The Gateway Service may provide the Consumer with pre-established options to define the domain and the data requirements or may require human intervention, i.e. Service Support, to customise them according to the Consumer's requirements.

Once the Gateway Service has established an agreement with the Consumer, a Weather Agent service is instantiated for that Consumer. The Weather Agent is in charge of interacting with the Consumer to notify the availability of the required data, instantiating the message broker required to deliver the data to the Consumer and managing the corresponding message queues. The Weather Agent retrieves the weather forecasts provided by the Core Weather Service, processes those forecasts to extract the data required by the Consumer, transforms the data into the specified format and delivers the data to the Consumer as per the agreed contract. Thus, the Weather Agent effectively decouples the computation of weather forecasts, performed by the Core Weather Computation Service, from the provision of weather data to Consumers.

The Core Weather Computation Service at the heart of the Micro-Weather Service is responsible for generating numerical weather forecasts for the domains of interest of the Consumers. The forecasts are delivered to the Weather Agents in standard formats and, as indicated above, the Agents are responsible for processing those raw forecasts, distilling the information required by the Consumers and delivering it to them in the agreed format. As we will see below, the Core Weather Computation Service requires as input a global weather forecast, such as the one produced by the Global Forecasting System (GFS) from NOAA (National Oceanic and Atmospheric Administration) [7]. GFS provides a global weather forecast that is used to initialise the numerical weather predictions processes carried out by the Core Weather Computation Service. In addition, the Core Weather Computation Service can leverage real-time meteorological observations to improve the accuracy of the forecasts it produces. These observations can be obtained from meteorological sensors deployed in the domain of interest or from other means such as satellites or indirect measurements such as using frequency variations in 4G/5G signals. For example, drones flying in the domain of interest may provide meteorological data measured during flight back to the service.

An entity providing meteorological data to the micro-weather service is considered a Data Provider client to the service and its interactions with the service are analogous to those of Consumers, as it is shown in Figure 2. Thus, the Gateway Service for each Data Provider instantiates a Weather Agent for the provider Agent that manages the data ingestion process, decoupling the data delivery to the Core Weather Computation Service from the data stream fed to the service by the Provider. The data is delivered to the Core Weather Computation Service in a standard format and at the expected rates, while the Providers may deliver data in different formats and at different rates in accordance to their measurement and communication capabilities.

A description of the Core Weather Computation Service implementation used in the exercise is provided below.

Core Weather Computation Service

The Core Weather Computation Service implemented in the IMPETUS Micro-Weather Service prototype is based on an advanced probabilistic numerical weather prediction system capable of computing accurate high-resolution meteorological forecasts over rural/suburban regions with surface areas of up to 60,000 square kilometres. Besides, the service is able to produce probabilistic weather predictions in the form of ensemble forecasts. An ensemble forecast includes a set of probable predictions of the weather instead of just a single deterministic prediction. Each of the predictions in an ensemble is called an ensemble member and describes a potential evolution of the state of the atmosphere over time, including all the variables required to describe that state, such as pressure, temperature, wind, precipitation, visibility, etc.

The different members of an ensemble are the result of computing a forecast, i.e. resolving a set of equations describing the evolution of the atmosphere into the future, for different sets of initial conditions and considering different assumptions in the equations used to model the atmospheric physics. The initial conditions of the model are given by the variables describing the state of the atmosphere at the time from which the model equations are propagated forward to compute a forecast, i.e. the initial time from which the weather prediction is available. A nominal set of initial conditions are obtained from the outputs of a global weather prediction model, e.g. GFS, and from measured meteorological data. In order to create the different ensemble members, the nominal set of initial conditions is perturbed according to mathematical techniques that ensure an appropriate sampling of the set of all possible initial conditions, while ensuring the physical consistency of the atmospheric state. It is assumed that the members of the ensemble are equiprobable and capture the full range of possible ways in which the atmosphere can evolve into the future. Thus, an ensemble forecast provides a measure of the uncertainty of the weather prediction by means of the spread between its different members. In addition, an ensemble forecast can be used to assess the probability of a weather-related event happening in the future by means of the percentage of members that predict that event. For example, consider a 10-member ensemble forecast used to predict surface temperature. If 8 out of the 10 members predict that the highest temperature tomorrow will be greater than 30 degrees, we can estimate that there is an 80% probability of the temperature tomorrow will be higher than 30 degrees.

In order to compute high-resolution weather predictions in a computationally efficient manner, it is common to use nested meteorological models with different resolutions and levels of complexity, with lower resolution models covering larger areas than the region of interest. The resolution is gradually increased as the target geographical area is reduced, while introducing more complex physics. The model with the highest resolution is run over the geographical domain of interest, considering the effects of terrain and including detailed models of the boundary layer and other physical phenomena associated with small-scale atmospheric effects.

The nested model structure adopted for the IMPETUS Core Weather Computation service is schematically depicted in Figure 3, including the specific geographical areas covered by each of the models and their grid resolution in kilometres. In order to obtain a prediction with the highest resolution model (Galicia model), it is necessary to run a prediction with lower resolution models (Europe and Spain).

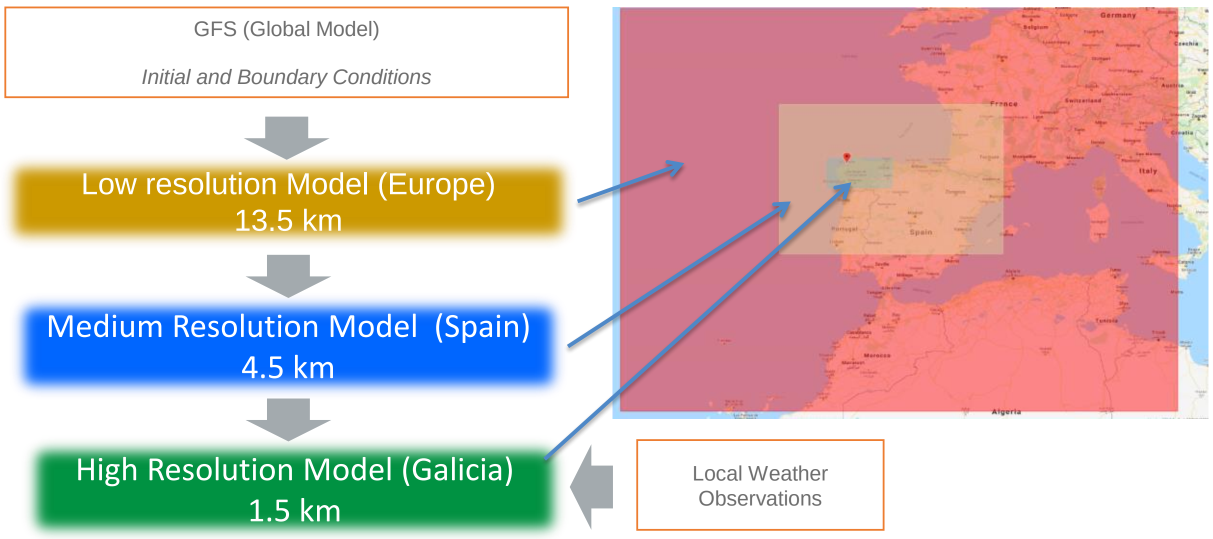

GFS (Global Model)
Initial and Boundary Conditions
Low resolution Model (Europe)
13.5 km
Medium Resolution Model (Spain)
4.5 km
High Resolution Model (Galicia)
1.5 km
Local Weather Observations

Figure 3: Model structure for the Core Weather Computation Model and geographical coverage

The resolution is the size of the horizontal square grid at whose vertices weather predictions are provided. The horizontal grid covers the region of interest at ground level and is replicated along the vertical dimension at a set of pressure levels. The model provides a forecast for each vertex of the resulting three-dimensional mesh at different pre-defined times into the future. In order to obtain the values of meteorological variables at points that are not vertices of the mesh or at times other than the pre-defined forecast times, an interpolation process is carried out by the Weather Agent.

The three nested models depicted in Figure 3 have been implemented using WRF (Weather Research and Forecasting Model) [8]. WRF is a state-of-the-art open source numerical weather prediction system designed for both atmospheric research and operational forecasting applications. The system includes an extensible atmospheric model as well as tools and code to implement the model. The model serves a wide range of meteorological conditions across scales from tens of meters to thousands of kilometres. The WRF system features two main components: a data assimilation system and a software architecture supporting parallel computation for model implementation and system scalability. The effort to develop WRF began in the latter 1990's and was a collaborative partnership of several US public institutions, including the National Center for Atmospheric Research (NCAR), the National Oceanic and Atmospheric Administration, the University of Oklahoma, and the Federal Aviation Administration (FAA).

In addition to the standard WRF functionality, the Core Weather Computation Service includes an ensemble forecasting scheme to produce ensemble forecast based on running the WRF model for different initial conditions and physics assumptions, as indicated above. In addition, the high-resolution model includes advanced boundary layer features as well as specifics data assimilation and post processing capabilities.

Table 15 below shows a summary of the main characteristics of the high-resolution model implemented in the Core Weather Computation Service (1,5 km horizontal grid resolution). The geographical coverage of this model, an area in northwestern Spain, corresponds to the domain of interest defined by the mission planner (a Forecast Consumer client) intending to plan the missions considered in the exercise. The computation process required to produce the output of this model is referred to as *Cell Process* in the architecture diagram in Figure 2.

The model produces an ensemble forecast containing six members. Each member predicts the atmospheric variables in the region of interest (pressure, temperature, wind, etc.) up to 21 hours into the future, with outputs every hour, i.e. atmospheric values at each vertex of a mesh with a 1,5 kilometer grid and 38 vertical pressure levels are provided with 1 hour intervals.

Geometry	
Grid resolution	1.5 km
Vertical levels	38 levels
Domain coverage	Longitude: 9.5-W 5.4 W Latitude: 42.1N-43.7N
Timeframe	
Lead time	21h
Output forecast interval	1h
Simulation	
Members in ensemble	6 (multi-physic and perturbed initial conditions)
Main Variables	Pressure, temperature, altitude, wind velocity (U,V,W) and TKE (turbulence)

Table 15: Galicia high-resolution model implemented in the Core Weather Computation Service

Simulation Environment

The simulation infrastructure that has been used to conduct the exercise is schematically depicted in Figure 4. The key elements of the simulation are the prototype Micro-Weather Service, described above, and a Mission Planner emulator that interacts with the service. This emulator essentially consists of a trajectory predictor that calculates the intended drone mission trajectory considering the following elements: a) a description of the mission, including constraints from the airspace and the terrain; b) the performance of the vehicle and c) the predicted weather conditions along the trajectory. The predictor integrates a set of 3 degrees-of-freedom equations of motion to anticipate the trajectory that the drone would fly to carry out its mission assuming the weather conditions are those provided by the Micro-Weather Service.

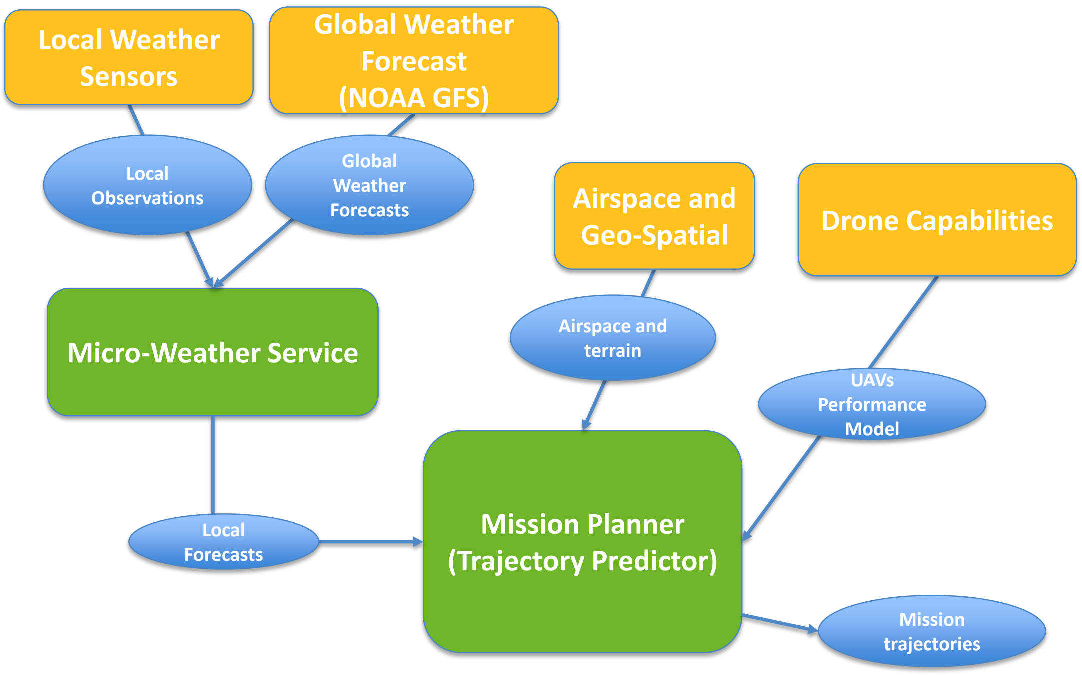


Figure 4: Simulation environment put together to conduct the exercise EXE1

As described in D5.1 [3], the missions considered in the exercise are forest fire surveillance operations conducted by fixed wing drones. A schematic description of a typical mission is shown in Figure 5 below.

The vehicle considered to conduct the missions is a drone with the configuration depicted in Figure 6. The platform has a 3-meter wingspan, a Maximum Take-Of Weight of 25 kilograms and is powered by a piston engine driving a pusher propeller. A performance model of this vehicle is used by the Mission Planner to predict the mission trajectories.

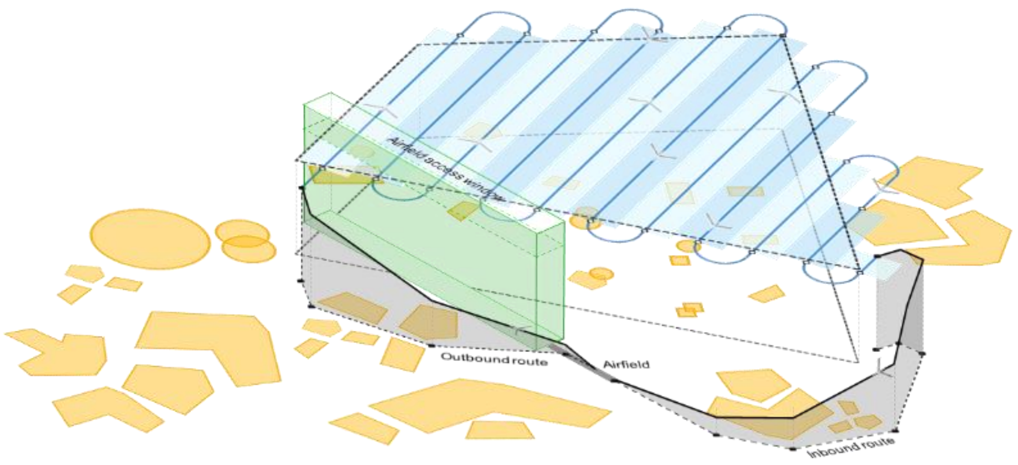


Figure 5: Schematics of the mission pattern

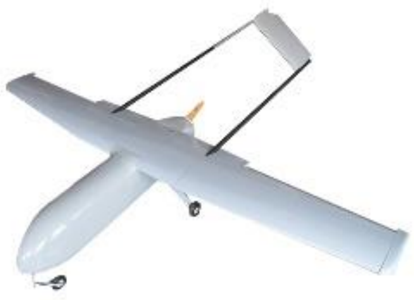


Figure 6: Depiction of the platform used to conduct the missions in the exercise

In addition to 4-dimensional trajectory of the flight along the planned route, i.e. the evolution of 3-dimensional position of the aircraft (latitude, longitude and altitude) with time, the Mission Planner also calculates other important variables of the vehicle motion, such as thrust and drag, fuel burn, airspeed and vertical velocity. Thus, the Mission Planner's produces a detailed predicted trajectory that allows analysing a variety of aspects of the flight.

Description of the Architecture

One of the traditional problems of any service architecture is that they are subject to failure propagation. That is, errors caused by errant clients can cause anomalies in the server leading to a degradation of the service offered, as higher latencies or even leading to fatal service failures while serving other clients. The root reason for this is that, by design, a service is a shared resource for the clients.

This disadvantage of traditional monolithic architectures can be mitigated through the micro-service paradigm, which can be leveraged to minimize the effect of errant (or malicious) clients over the Quality of Service offered to others. This is done dividing the functional blocks of the service and designing a data flow between them that is designed to contain the propagation of errors. This may require some duplication of the elements of the micro-service architecture that are interacting with the clients.

This can be taken into account during the design so that the elements that face the clients are independent lightweight micro-services while fully protecting the more computationally heavy ones from clients' unpredictable behaviour.

In the case of the micro-weather service, this was taken into account from the very start and is fully reflected in the functional diagram shown in Figure 7:

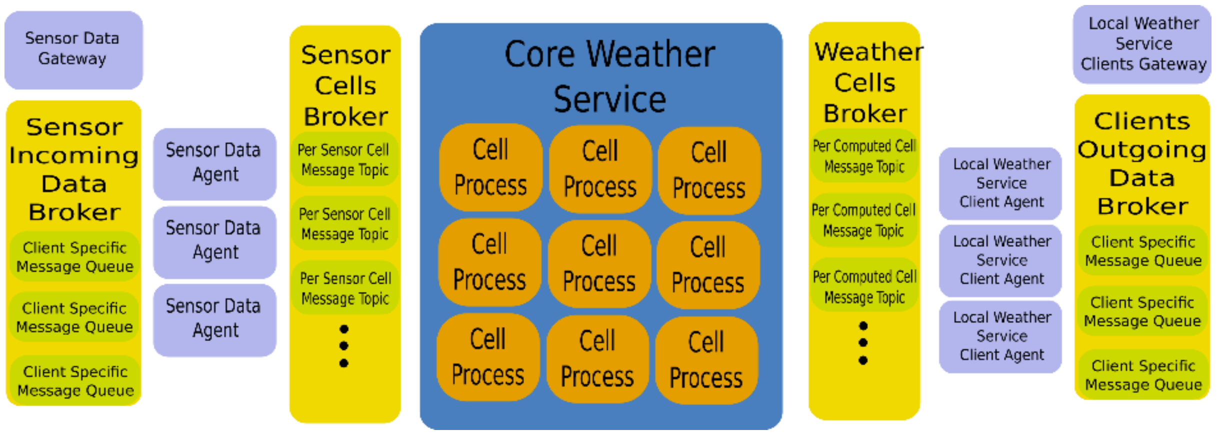

Sensor Data Gateway
Sensor Incoming Data Broker
Client Specific Message Queue
Client Specific Message Queue
Client Specific Message Queue
Sensor Data Agent
Sensor Data Agent
Sensor Data Agent
Sensor Cells Broker
Per Sensor Cell Message Topic
Per Sensor Cell Message Topic
Per Sensor Cell Message Topic
Core Weather Service
Cell Process
Weather Cells Broker
Per Computed Cell Message Topic
Per Computed Cell Message Topic
Per Computed Cell Message Topic
Local Weather Service Client Agent
Local Weather Service Client Agent
Local Weather Service Client Agent
Local Weather Service Clients Gateway
Clients Outgoing Data Broker
Client Specific Message Queue
Client Specific Message Queue
Client Specific Message Queue

Figure 7. Architecture of the Core Weather Service

The core service, which occupies more than 90% of the computational cost, is completely separated from the clients. In addition, they interact only with their respective agents, which are dynamically instantiated by the Gateway.

The clients' use of the service follow a series of steps:

1. A client sends a REST petition with all parameters of the requested service to the Local Weather Service Gateway (Gateway);
2. The Gateway parses the petition and adjusts the core weather job queue to make sure that the cells required to cover the requested area during the time requested are active. Overlapping;
3. In the case of valid petition, the Gateway instantiates a Local Weather Service Client Agent (Agent) and returns the entry point to this agent to the client. At the same time, it forwards the petition of the client to the agent for further processing. From this point on, all interaction between client and the service goes through the agent;
4. Agents subscribe to the AMQP queues that are coming from the cell that contains data of interest for their clients;
5. Agents parse the rest of the petition and establish a topic in the client broker;
6. The client makes a REST petition to the agent and receives the queue end-point. He starts consuming data.

7. The Agent can control flow and format details through interactions with the client. The agent will be listening for the live of the contract.

2.4.2 Mission and Flight Planning Management services (EXE 2)

A number of adjustments has been made to the simulation platform described in D05.01 §8.2.1 [3]. The updated high-level architecture is given in Figure 8.

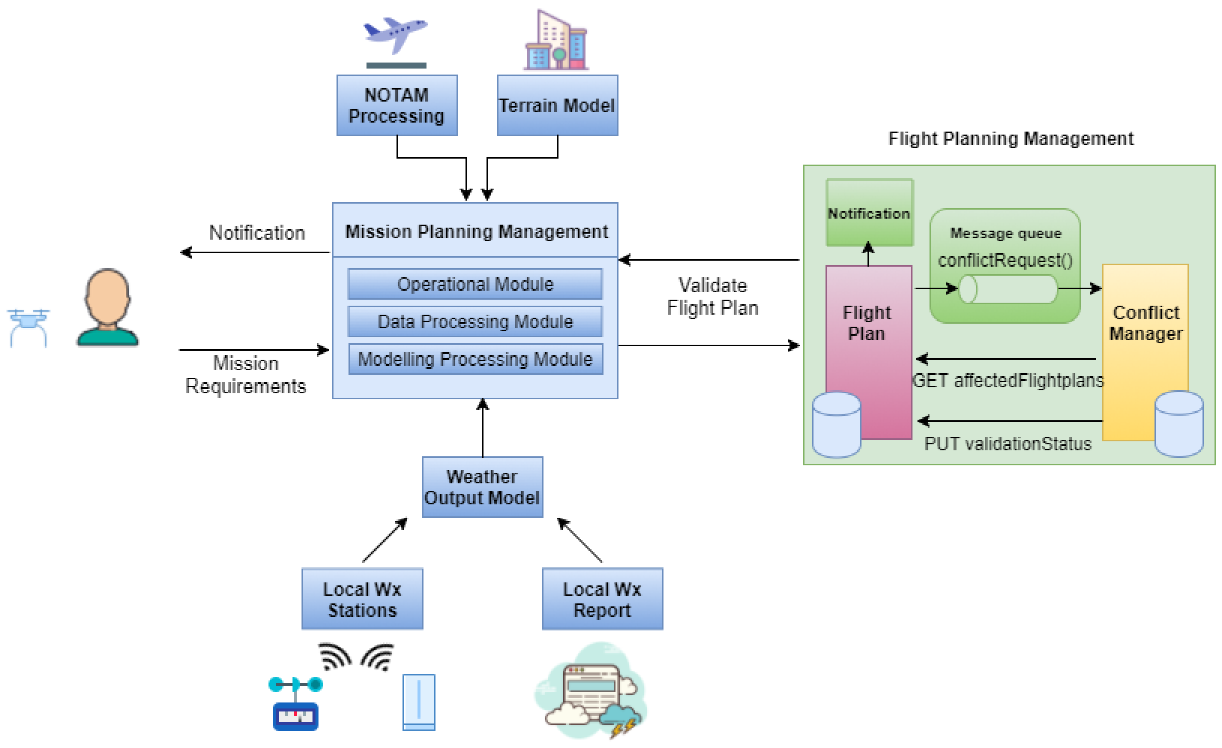


Figure 8: Functional components of the MPM and FPM Services

The main modifications of the platform are the following:

- The weather output model combines local weather measurements with a standardized local weather report (in this case METAR report);
- A terrain model is used for the trajectory modelling by the MPM service in order to model a feasible trajectory within the considered urban environment;
- For the validation of the flight trajectory, a conflict assessment process is carried out by the FPM service, where an intersection detection in the legs of the modelled trajectories is performed. Based on the result of this assessment the service will approve or reject the submitted flight plan and consecutively notify the client.

The specific characteristics of the employed models and implemented processes are detailed and summarized in the according sections dealing with the presentation of the results in § 3.2.2.

2.4.3 Traffic Information and Monitoring Services (EXE 3)

The scope of this exercise has substantially changed. The main reason of this change is to put more focus on the performances and benefits of the proposed architecture. This experiment will aim to demonstrate the strengths of a microservice infrastructure that, thanks to its nature, can be fully customized and, consequently, analysed to obtain all the valuable information needed.

Reformulation of the exercise: performance test in a microservice-based architecture for U-space

The framework that will be tested is a microservice-based architecture that allows for the execution of a highly available platform while making it a resilient, fault-tolerant and scalable system. It also allows new functionalities to be integrated without affecting the correct execution of the rest of the microservices.

This architecture has been developed trying to offer several functions for different U-space services that are simulated. The exercise tested the data exchange processes necessary to provide:

- Monitoring Service: using synthetic data from real operations in a certain area, the platform was fed with information that can be provided by real drones. The platform was capable of filtering them to process only the messages related to the position, battery status and attitude of these flights (orientation, speed, flight mode…). This module was based on MAVLink protocol, highly compatible with most of the flight controllers in the market;

- Traffic Information Service: all positions were gathered by the platform and processed, obtaining the complete view of all drone operations. Their updates are shown in an HMI, a web server in which each drone can be represented and the most representative data can be checked;

- Emergency Service: using the information provided by the Traffic Information Service, this module will compare the relative position of each pair of drones and, when the distance between them reaches less than 50 m (customizable), it raises an alarm message that is shown in the user interface, advising to the user about possible issues in the scenario.

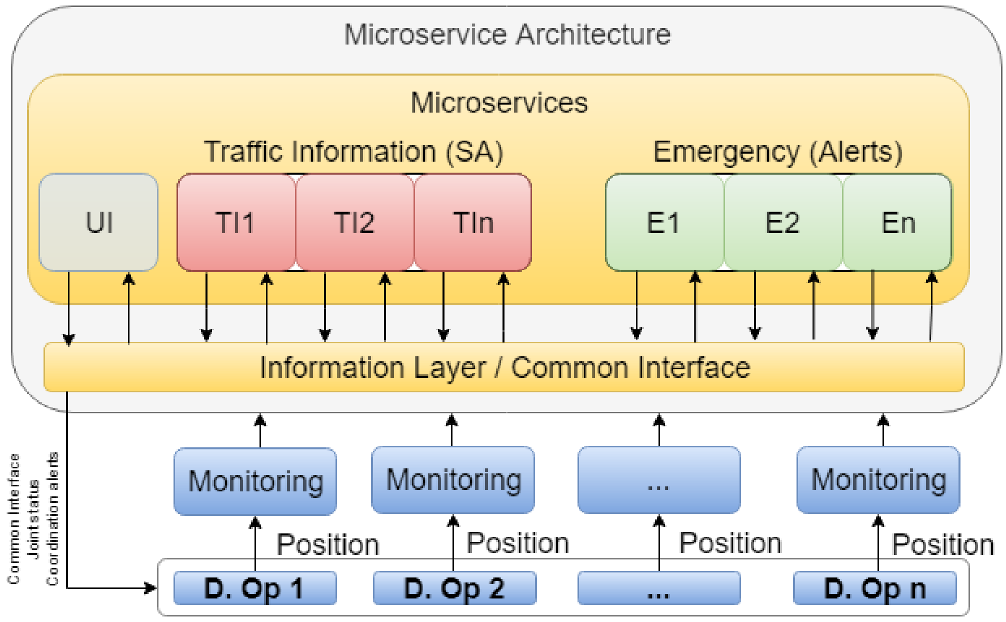


Figure 9: Information flow of the services detailed in the microservice architecture

Technologies Implemented

This platform has been developed using some existing products that provide the basic capabilities for information management:

- NGNIX: an asynchronous high-performance web server configured to receive all the messages generated by the drone operations using MAVLink protocol, redistributing them to the microservices that show these positions in a web interface and store the datasets in distributed databases ready to be queried;
- MongoDB: storing the historical information of the platform and, in this specific case, the messages filtered by the MAVlink microservice (this module is capable of skipping messages defined by the user, allowing to focus only on the topics of interest). The use of a NoSQL and distributed database allows this platform to store large amounts of information and provides access to these datasets to exploit them in an efficient, quick way;
- ActiveMQ: an open source message broker that implements the Java Message Service (JMS) specification (a communication framework in which the integration of different applications that accept the same messaging protocol/service is easily deployable). The use of a messaging system allows decoupling the functionalities of these microservices and allows their scaling. The messages are processed using two different methodologies:
  - P2P (Point to Point, Sender/Receiver): it is guaranteed that the message is processed once, regardless the number of possible processors or instances of the same microservices. If a receiver is not available or overloaded, the message is stored until it can be delivered to a defined receiver. To do this, message queues are available;
  - Publisher/Subscriber: the message is broadcasted and received by all receivers subscribed to a Topic. In this case, two topics are used: POS and ALARM. The sender

and receiver can be even more decoupled, as the sender does not need to know which one of the receivers processes the message.

- PostgresXL/PostGIS: consists of a distributed relational database management system based on PostgreSQL (object-oriented and open source relational database management system). The main objective of this product is to provide the same functionality of PostgreSQL while distributing the workload in clusters. On the other hand, PostGIS is a plugin that provides the necessary capabilities to process spatial information in an efficient way and store, analyse and transform geographic information. It can also be executed in a distributed way, allowing scaling and skipping the limitations of a monolithic architecture;
- IGNITE: cache-memory database whose main objective is to store the incidents generated by the Alert microservice. It generates the reports to users and delivers them using MQTT system;
- Kubernetes: consisting on an open source framework that is capable of automating deployment and scaling the systems that have been previously containerized. In other words, it allows to activate a certain set of predefined functionalities while monitoring the status of the entire system, providing automated roll-outs and roll-backs and self-healing mechanisms;
- AWS: elastic cloud computing resources are also integrated in this environment, providing the necessary infrastructure to deploy all the modules described and implemented for this architecture.

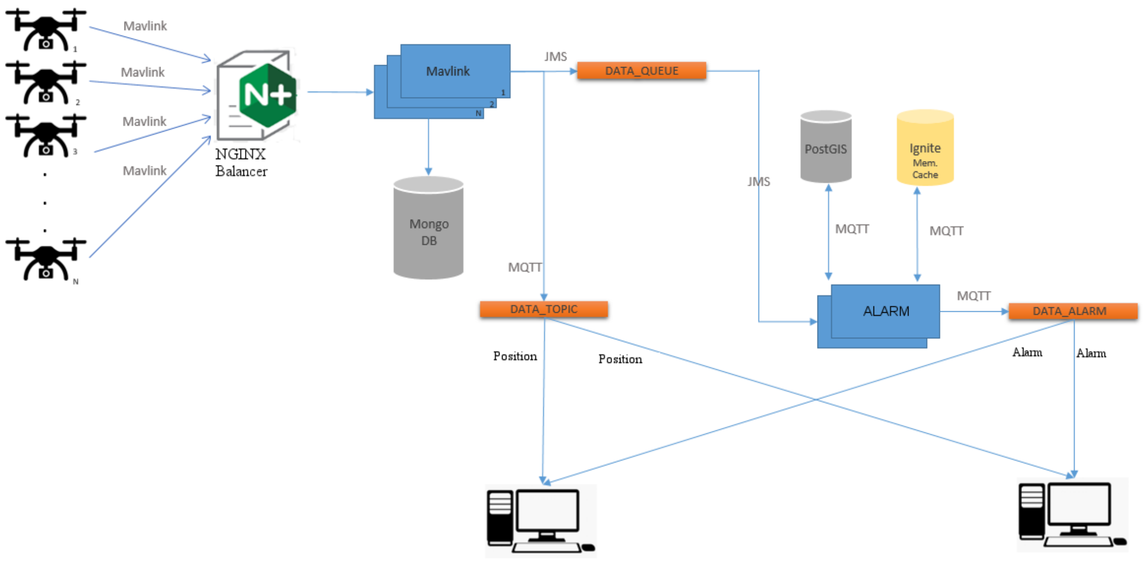


Figure 10: Microservice environment (EXE 3)

Microservices developed and integrated

Using these products, the microservices that have been developed to be attached to this infrastructure are described below:

- MAVLink/Traffic Information microservice: this service is in charge of managing the data that is sent by the drones using the MAVLink protocol. It is configured to listen to the messages through a customizable UDP port (that can also be reached using MAVProxy protocol). Once the message is received, it can also be filtered (customizable by the users, leaving the possibility of deactivating it to users) collecting only the messages of interest, such as 3D position, attitude, battery and drone status. This filtered information can be sent to a web browser to be represented and stored in the message queue, waiting for another microservice that will use it;

- Alarm microservice: performs the generation of alarms when a defined proximity between drones is detected (50 m in this case, but can be configured by the user depending on the desired risk margins). It sends the message to the web browser to be represented. Ignite and PostgresXL/PostGIS are used to query the position data;

- Listener microservice: this service receives all the messages received by the MAVlink and Alarm microservices and redirects the messages to the services that will use them. To do this, it asks the existing services for the topics they need while it stores the datasets in a MongoDB database, forwarding the expected information to the proper service when it is available.

Finally, as an added value of this platform, it is possible to attach other U-space services to this architecture. The only condition is to send the message in a predefined format that the Listener shall be able to understand. Currently, pre-tactical geofencing, tactical geofencing and tracking can also be part of this environment.

As part of the configuration of the architecture, the architecture is design to detect a maximum workload of a service above 80% of its capacity. To avoid the complete failure of the microservice, it will react to this overload, deploying another microservice before it is completely congested and allowing to balance its load.

Test plan of a microservice-based architecture

The expected behaviour of the deployed architecture is described below:

- Once the architecture is executed for the first time, there is at least one microservice per type implemented working without overloads and waiting for inputs;

- The messages from a small amount of drones are processed. The information is not affected by delay and it is not expected that the microservice fails;

- The demand suddenly increases and so delays in information processing can be increased. The platform is capable of detecting the 80% of the capacity until reaching overload. To avoid its failure, the process is:

    - Generation of a number of instances of the same service, as a result of a fast reaction of this architecture. During its initialization, workload in recently created microservices can be high, so ensuring that several of them are created will allow to balance the load between them;

    - Once they are completely initialized and the initial overload is controlled, the surplus microservices are deactivated and the minimum amount of them (necessary to absorb the sudden increase in resources) are still working to adapt to the demand in a certain instant.

- The demand decreases and the workload in active microservices decline. The platform detects it and destroys the excess of microservices, adapting the minimum amount of resources to ensure the high quality of information management. The latency also decreases;
- Suddenly, one of the microservices fails and must be removed from the architecture:
    - The platform is capable of detecting this failure and stops listening to this service while it is being closed.
    - At the same time, it is ready to deploy several instances of the same service, allowing to load the balance between them and absorb the initialization workload.
    - Once it is stabilized, the platform only keeps the necessary services to process the level of demand at that time.

If the functions of each microservice were even more simple (atomic) theses initialization issues would be of no importance and less microservices will be deployed because of the reaction of the system.

The following diagram describes the complete information flow between services that is being tested:

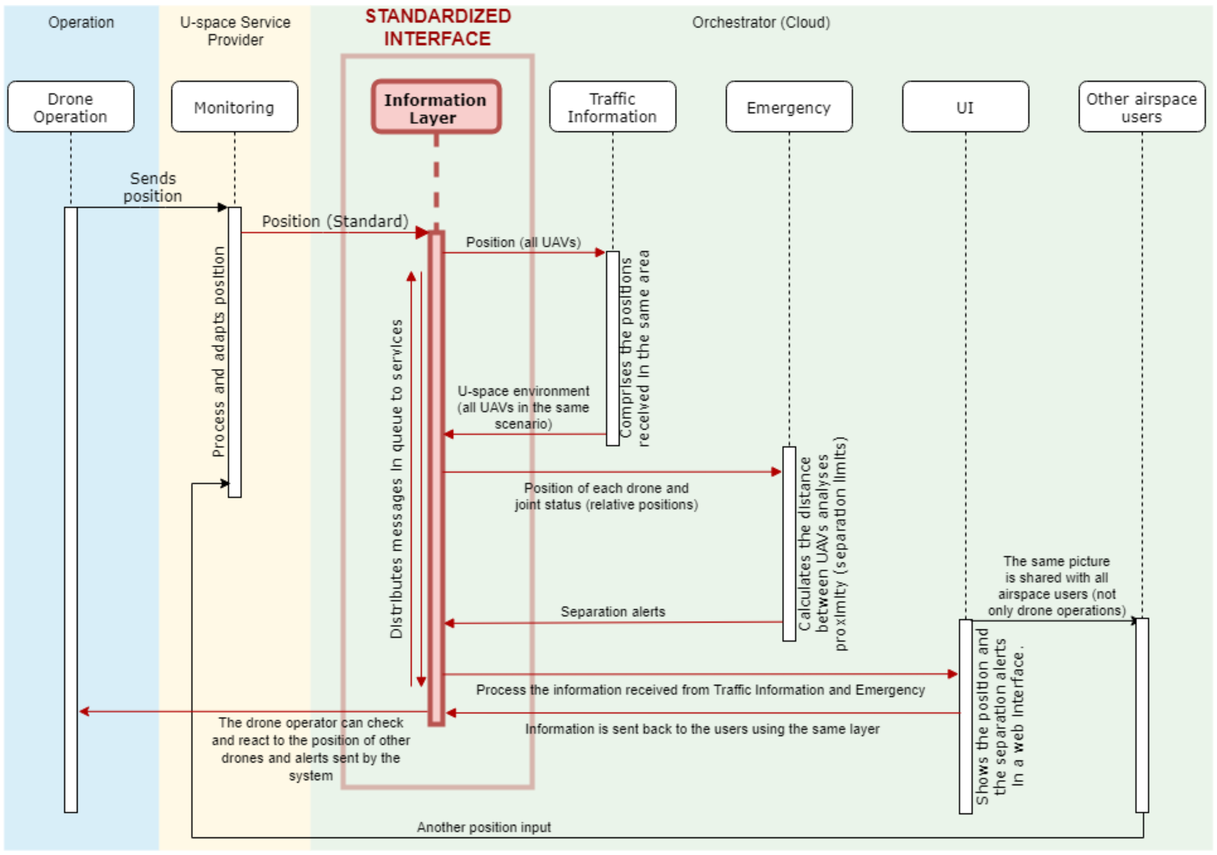

Operation
U-space Service Provider
STANDARDIZED INTERFACE
Orchestrator (Cloud)
Drone Operation
Monitoring
Information Layer
Traffic Information
Emergency
UI
Other airspace users
Sends position
Position (Standard)
Position (all UAVs)
Process and adapts position
Distributes messages in queue to services
Comprises the positions received in the same area
U-space environment (all UAVs in the same scenario)
Position of each drone and joint status (relative positions)
Calculates the distance between UAVs analyses proximity (separation limits)
Separation alerts
The same picture is shared with all airspace users (not only drone operations)
Process the information received from Traffic Information and Emergency
Information is sent back to the users using the same layer
Shows the position and the separation alerts in a web interface.
The drone operator can check and react to the position of other drones and alerts sent by the system
Another position input

Figure 11: Information flow in exercise 3

It should be stressed the need of a standardized information layer that provides the necessary mechanisms (formats, flow rules, capacity…) to process the information provided by each input, which should adapt its data streams to these predefined rules.

The tests that were carried out have been designed to verify its correct operation and its compatibility with the U-space requirements (those related to the services themselves and the architecture requirements) defined in D03.01 [4].

MTI-TEST 1.Latency tests: increasing the number of drones flying at the same time (using series of 1-10-20-30-100 simultaneous operations), the delays are expected to grow and so the workload of the microservices involved. The main aim of this test is to measure the increase in the delays considering the number of drones in operation;

MTI-TEST 2.Scalability and elasticity: increasing the number of drones in a set of predefined series, this test aims to detect the creation of instances of these microservices and analyse the variations in delays caused by this behaviour. The number of drones are increased from 10 to more than 200 drones and the platform should absorb the increase in demand providing more execution capacity (creating services, loading the balance…);

MTI-TEST 3.Flexibility tests: increasing and decreasing the number of drones processed by the platform, the key objective is to analyse the adaptability of the platform to these changes, avoiding over expenditures and ensuring optimal use resources. Drones oscillates between 1 and 100 and it is expected that the platform creates and destroys services under demand;

MTI-TEST 4.Maximum workload tests: this test allows measuring the maximum amount of operations that can be managed simultaneously with the available resources (5 virtual machines) and the delays associated to this overload;

MTI-TEST 5.Maximum workload tests (without filters): analogous to the previous one, but without the filter that has been defined in the MAVLink microservice. This test also has to deal with an increase in the number of messages that can be or not used by the platform. It is estimated that there are more than 15 times the number of data streams and the platform needs to handle with their recognition and omission. This process adds workload to the system, imitating the operation in an environment with multiple and different U-space services deployed;

MTI-TEST 6.Resilience tests: this test focuses on the behaviour of the system during failures of the microservice, provoking the following issues:

- With 1 microservice per type in execution, MAVLink microservice is intentionally closed;
- With 1 microservice per type in execution, Alarm microservice is intentionally closed;
- With 1 microservice per type in execution and more than one MAVLink microservice running, one of the MAVLink microservices is intentionally closed;
- With 1 microservice per type in execution and more than one Alarm microservice running, one of the Alarm microservices is intentionally closed.

The main objective is to analyse the time to relaunch another instance (which is analogous to the time of loss of communication and valuable outputs) and the load balance between duplicated services when one of them fails. For this test, 30 drones are considered and duplicated microservices are forced to be deployed.

2.4.4 Traffic Management Services dealing with dynamic airspace management (EXE 4)

For the basis of this series of experiments, we provisioned one of our Region Management nodes, which would serve as the simulation environment, and configured it with parameters representative of the future CORUS 'Type Zu' airspace.

A Region Management node can be thought of as the tactical deconfliction instance responsible for managing a unique airspace volume.

The area covered by the Region Management node was described as the area contained by a rectangular bounding box with the North West coordinate 51.46353 degrees North, -1.07561 degrees West and 51.40382 South, -0.94549 East respectively.

This equates to a surface area of approximately 60 km^2.

Simulation of Drone Traffic

To ensure repeatability against a common traffic picture, the experiment leveraged our simulation engine, which was configured to present into the Region Management node a linearly increasing number of drone operations, to a maximum of 1,000 concurrent operations.

New drones were introduced at a constant rate of 1 every 0.5 seconds; to reach full saturation would therefore take 500 seconds, or 8.3 minutes. This timeframe was useful because it (a) falls within the flight lifetime of our vehicle journeys, and (b) challenges the algorithms and test scenarios sufficiently by introducing changes at a high rate.

Each simulated drone was given a pre-determined origin and destination coordinate within the boundary, and had its characteristics configured in accordance with Table 16.

	Fixed wing				**Rotary**			
Drone operation	Autonomous	Automated	Semi-automated	Human controlled	Autonomous	Automated	Semi-automated	Human controlled
Standard separation	5	5	5	8	3	3	3	5
Drone speed: 10km^{-1}	3	3	3	5	3	3	3	5
30km^{-1}	5	5	5	8	5	5	5	8
Endurance	5	5	5	8				
Mission VLOS	N/A	N/A	5	8	N/A	N/A	3	5
BVLOS	5	5	N/A	N/A	3	3	N/A	N/A
Location Rural	5	5	5	8	3	3	3	5
Semi Urban	5	5	5	8	3	3	3	5
Urban	3	3	3	5	2	2	2	3

	Fixed wing				**Rotary**			
Drone operation	Autonomous	Automated	Semi-automated	Human controlled	Autonomous	Automated	Semi-automated	Human controlled
Mission Priority:								
Emergency service flight	10	10	10	10	10	10	10	10
Commercial flight	5	5	5	8	3	3	3	5
Recreational flight	N/A	N/A	5	8	N/A	N/A	3	5
Drone electronic conspicuity:								
Plan only								
ADS-B	10	10	10	12	10	10	10	12
LTE	5	5	5	8	3	3	3	5
combination	5	5	5	8	3	3	3	5
	3	3	3	5	2	2	2	3
Drone command:								
LTE	5	5	5	8	3	3	3	5
GCS	5	5	5	8	3	3	3	5
Human control	N/A	N/A	N/A	8	N/A	N/A	N/A	8
LTE Coverage								
Poor	8	8	8	10	5	5	5	8
Good	3	3	3	5	2	2	2	3
Latency (poor)	8	8	8	10	5	5	5	8
Weather data quality:								
National	8	8	8	10	5	5	5	8
Regional	5	5	5	8	3	3	3	5
Hyper local	3	3	3	5	2	2	2	3
Actual weather; wind speed								
Low	5	5	5	8	3	3	3	5
Medium	8	8	8	10	5	5	5	8
high	10	10	10	N/A	8	8	8	N/A

Table 16: Dynamic Separation Criteria

For the purposes of the test, all drones' flight paths were computationally generated through a separate mechanism in such a way that their origin and destination waypoints were all unique, but that at least one vehicle's shortest path (straight line) would come into conflict with at least one other simulated vehicle during the simulation timeframe. In this way, we are able to simulate an environment of increasing conflict.

In brief, the simulation test environment provided a relatively advanced capability to simulate the characteristics of a drone in flight – such as acceleration and turn rate – but for the purposes of this experiment did not progress into a full physics-based simulation environment. It was decided that this extra level of simulation would not materially alter the outcome of the initial suite of tests.

Experiment Graphical User Interface (eGUI)

The main purpose of the experiment was to yield test *data*, and evaluate the performance of the algorithms, however, for the purposes of this experiment we also created a sample 'visualiser', or Experiment Graphical User Interface (eGUI) which was limited (for time purposes) to show only 100 concurrent vehicles. This was purely to aid of providing a visual description of the digital environment.

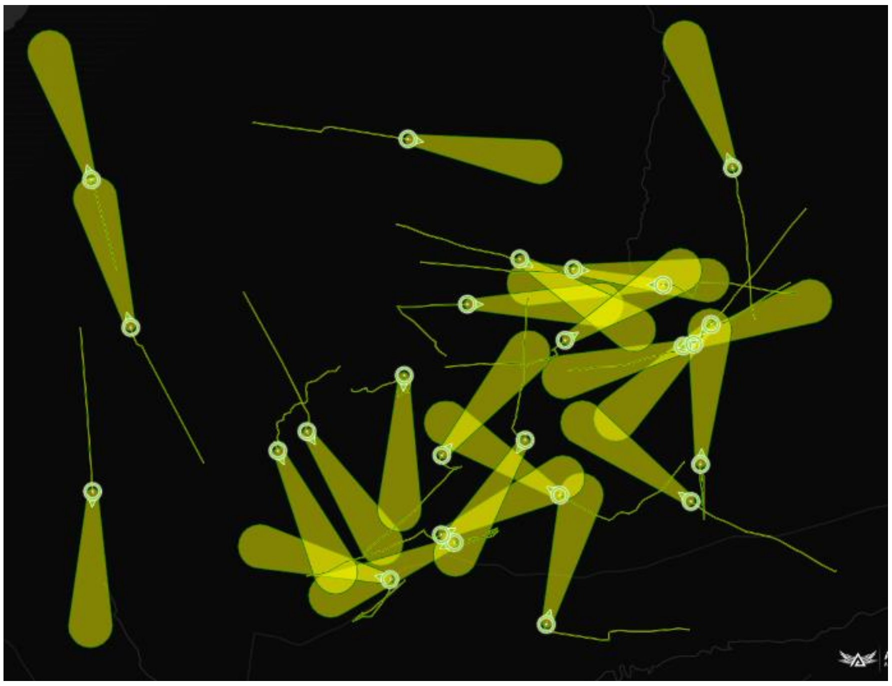


Figure 12: Simulation scenario

Summary of Test Methodology

Several predetermined scenarios, based on the common traffic simulation pattern described above, were loaded sequentially into the Region Management Simulation Test Environment (RMSTE).

At the start of a test scenario, RMSTE was pre-loaded with the vehicle simulation and dynamic separation criteria and then executed. RMSTE would evaluate the location of each object at a rate of 0.5Hz, and simulated varying latency as described in Table 16 in the communication mechanism between RMSTE and the remote vehicle.

RMSTE Configuration Parameters/Dynamics

1. To test the **Positive Field Theory** algorithm, and several adaptations of it (described later), as an effective digital, automated 'air traffic control' service. In RMSTE, the role of the algorithm was to provide minimal navigation assistance to simulation vehicles such that their flight paths – which are designed to conflict at the time of take-off – can be dynamically altered such that,

en-route, the vehicles then do not go on to collide. The algorithm generated one instruction per second, to each vehicle in conflict. Instructions were given repeatedly until the course change was observed, and then refreshed until the conflict was past, or, there was a loss of separation event;

   a. A table showing the Dynamic Separation Criteria we tested against is described later in this document;

   b. RMSTE's only mechanism of communication with simulated drones would be in the form of the 'transmission' of waypoints, which it was assumed the drone would follow faithfully and expediently with a transmission delay of 0ms, with a view to creating an alternative flight path that takes vehicles out of conflict;

   c. For further information on the positive field theory adaptation used in RMSTE, see the relevant section below.

2. To enable the inclusion of other types of aircraft that affect the simulation but cannot be controlled *by* the simulation, such as a GA aircraft, or a "priority drone" vehicle;

3. To enable customisable load pattern files to be consumed;

4. To be able to report on specific events, such as a "loss of separation", which is an event when one of the simulated vehicles comes into direct conflict with any other item in the RMSTE;

5. Each vehicle must reach its destination within a total of 10 minutes' departure from its origin. At the speeds predetermined, each vehicle could arrive at its destination within 5 minutes, however, a simple mechanism to evaluate whether course corrections to avoid a conflict were detrimental to the overall ability to complete a flight or not was required. Thus, RMSTE would record a failure if the avoiding action could not compute a path to complete the journey within the additional 5 minutes, simulating also the depletion of the vehicle's fuel;

6. RMSTE must not instruct any manoeuvre which the drone cannot safely perform. However, these instructions for the purposes of the evaluation were permitted to be given without respect to remaining 'fuel' capacity.

To ensure consistency of test results, the Region Management node had a full state clean at the end of each scenario and was put back into a "clean" state.

Positive Field Theory Algorithm used in RMSTE

Using positive field theory, a standard size and shape of 'field' was virtually established around individual drones within the scope of the algorithm. The size and shapes of the fields were changed to mimic differing types of operations, such as 'blue light' operations, which would have a higher priority than non-blue light traffic. The type of operation and the types of drones being used will need differing separation requirements, which will be modelled to assess the effects on airspace capacity.

There were several initial limitations in our earlier iterations with 'out of the box positive field theory' that meant we had to make some minor adaptations to suit the use case in traffic management:

Through the experimentation, we found that:

- Where there were multiple field aggregations, in cases where 3 or more objects required deconfliction, the algorithm did not "combine" well creating a much exaggerated confliction behaviour;
- Where we applied a "shaping" to the potential field we found several inefficient and unexpected avoidance actions;
- In complex deconfliction scenarios, it became possible for drones to become "stuck" in areas of low potential or between areas of equal potential.

Our modifications included specific work to introduce 'buffers' as illustrated below, which are constructed as follows:

A circle was drawn around the vehicle (an arbitrary numeric value, multiplied by velocity), with another circle projected out (distance was the same arbitrary value multiplied by velocity, with radius calculated similarly). These two circles are merged (with the space between them treated as a solid shape) to create a conflict resolution zone. This results in a cone shape projection out ahead of the vehicle's direction of travel.

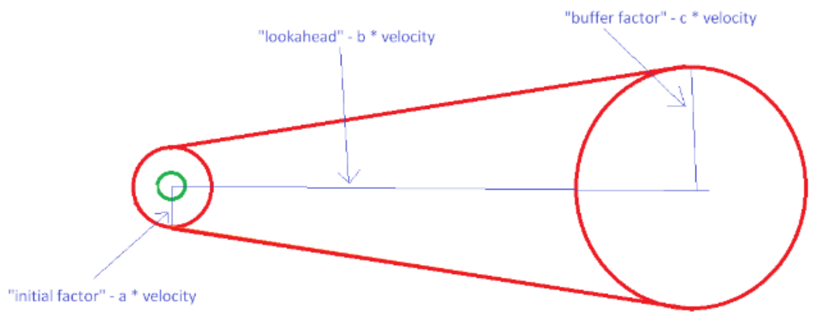

"buffer factor" - c * velocity
"lookahead" - b * velocity
"initial factor" - a * velocity

Figure 13: Confliction resolution zones

A conflict between two drones occurs when their 'cones' intersect.

Conflict resolution was achieved by having the drones turn to the *right* of the drone they conflict with; the amount they rotate/turn rate was determined by a "time to impact", where the sooner the objects will collide, the more they will turn.

These instructions were only sent for drones that are "approaching" others, for instance at an angle or head on. If they are not determined to be approaching the drone they conflict with, then they will take no action. For example, two drones in a line travelling in the same direction will conflict with each other, but only the one behind will receive instructions, since the one ahead has no conflict in front.

For instances where a drone conflicts with many other drones, the algorithm determines which other drone (that it was approaching) was *furthest* to its right, and then turns right around it to avoid it.

3 Experimental results

IMPETUS performed a series of experimental tests to validate the proposed solutions for a specific set of future U-space services and to challenge the implementation framework of these services, which is based on the microservice approach.

3.1 Summary of experimental results

The following table shows how the results address the validation objectives of the project and how the chapter is structured. As previously mentioned, the validation of the IMPETUS solution is twofold:

i) the validation of the service functionalities and solutions (§ 3.2), and

ii) The validation of the microservice-based architecture (§ 3.3).

Validation of Crucial Services	Weather Service	Flight and Mission Planning Management Services	Traffic Information and Monitoring Services	Traffic Management Services
Individual Requirement-related Criteria	X	X	X	X
Validation of Microservice-based architecture				
Safety & Failure modes	X		X	X
Data management		X	X	X
Scalability	X		X	X
Flexibility			X	
Commercial implications	X	X	X	

Table 17: Relation of the exercises with the validation areas

3.2 Detailed analysis of results per service being tested

3.2.1 Drone-specific Weather Service (EXE 1)

The main objective of the exercise is to assess the potential benefits of using probabilistic micro-weather forecasts in Mission Planning. To that aim, we will compare predicted mission trajectories calculated using the high-resolution ensemble forecasts produced by the IMPETUS Micro-Weather with the trajectories that would result using current low-resolution aviation forecasts.

A set of 4 validation cases are considered for the exercise, denoted as WX-1, WX-2, WX-3 and WX-4, and each associated to a Validation Objective as described in D5.1 [3] and summarized in § 2.3.1.

The analysis will focus on individual missions performed by a single vehicle. For each mission, the mission planner predicts the trajectory at 3 different times before the flight starts: 18 hours before take-off, 6 hours before take-off and 3 hours before take-off. For comparison purposes, the mission trajectory is predicted using 3 different weather forecasts:

- **Static deterministic forecast**, which represents currently available aviation forecasts and consists of an average wind velocity throughout the mission area and standard pressure and temperature profiles. It is based on the outputs of the low-resolution forecast of the Core Computation Service (Europe Model). The trajectory prediction results obtained using this forecast are considered as the benchmark against which to assess the improvements on mission planning accuracy and robustness that result from using probabilistic micro-weather forecasts;

- **Probabilistic low-resolution forecast**, an ensemble forecast containing 6 members, obtained from the low-resolution Europe model. This forecast is used to represent an intermediate step between currently available aviation forecasts and probabilistic micro-weather. The forecast has low resolution (leading to prediction errors associated to interpolation) but it captures forecast uncertainty through the ensemble members;

- **Probabilistic high-resolution forecast**, which is the output of the high-resolution Galicia model and consists of an ensemble forecast containing 6 members. This forecast is the output of the Micro-Weather service.

The Mission Planner always uses the most recent forecast available to predict the mission trajectory, which is assumed to be the forecast whose initial time is the current time, i.e. the time when the trajectory prediction process is launched.

For each prediction of the mission trajectory, we analyse a set of variables such as flight time, fuel burn and vertical profile. For each of the 3 different weather forecasts, we measure the difference between the predicted value of each variable and a reference value assumed to be the “truth”, i.e. the one associated to the actual trajectory flown by the drone. To determine these reference “truth” values we need to define the actual trajectory flown by the drone, which depends on the actual state of the atmosphere when the flight takes place. As an approximation to that atmospheric state, we consider the output of the high-resolution Galicia model considering the take-off time as the initial time of the forecast. It is assumed that this forecast is an accurate representation of the actual atmospheric conditions at the time of the flight, since its initial state, which is based on actual weather observations, coincides with the take-off time. To estimate the reference "truth" values for the variables of interest, the flight trajectory is calculated for each of the 6 ensemble members of the Galicia forecast. The reference "truth" values are assumed the average of the values obtained for each ensemble member. For example, the "truth" fuel burn will be calculated as the average of the 6 values of fuel burn obtained for the 6 trajectories that result from computing the flight trajectory for each of the 6 ensemble members.

Figure 14 schematically describes the trajectory analysis process described above. The trajectory prediction at Take-Off Time (TOT) uses the forecast with initial time (T0) equal to TOT and it is assumed to represent the actual trajectory flown with the aircraft.

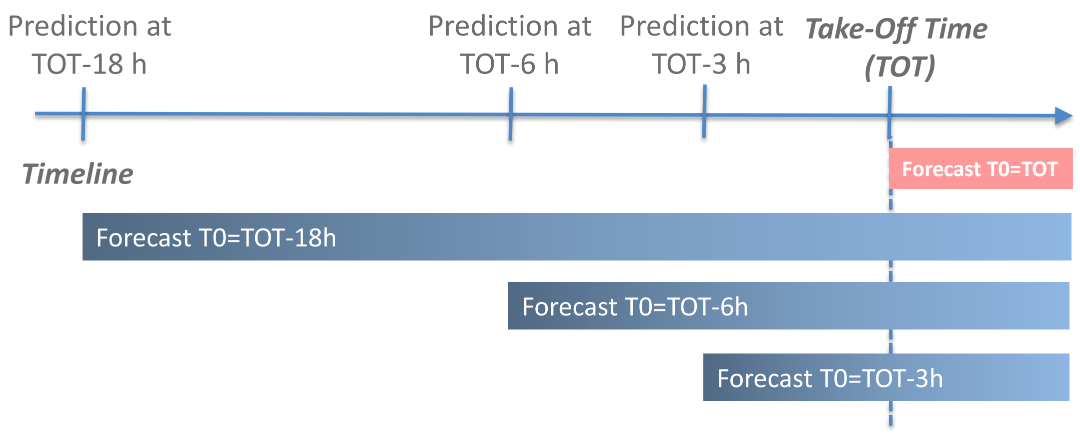


Figure 14: Scheme for mission trajectory prediction analysis for different look ahead times

In the following subsections, the results of simulating the mission planning process in the 4 validation cases described in D5.1 (WX-1, WX-2, WX-3 and WX-4) are presented and analysed following the scheme described above. Each case focuses on specific aspects of the mission planning process for the forest fire surveillance scenario described in D5.1 [3]. The overall objective is to establish whether the predicted mission trajectories calculated using the output of the Micro-Weather Service (Galicia high-resolution ensemble forecasts) allow for a more robust and efficient mission planning process than using lower resolution, less accurate weather forecast models, according to the Success Criteria defined in D5.1 [3].

3.2.1.1 WX-1

The validation objective is defined as: "Quantitatively measure how the uncertainty associated with the forecast of atmospheric conditions relevant to drone trajectory prediction (pressure, temperature and wind) translates into trajectory prediction uncertainty in terms of 3D position, timing and fuel/energy consumed."

Figure below depicts the mission profile considered in this case. We have measured the differences between the predicted values of the flight time and fuel consumption required to complete the mission for each the 3 different forecasts and at the 3 look-ahead times considered (18 hours, 6 hours and 3 hours). We have then compared the results with the assumed "true" values of flight time and fuel burn for the mission.

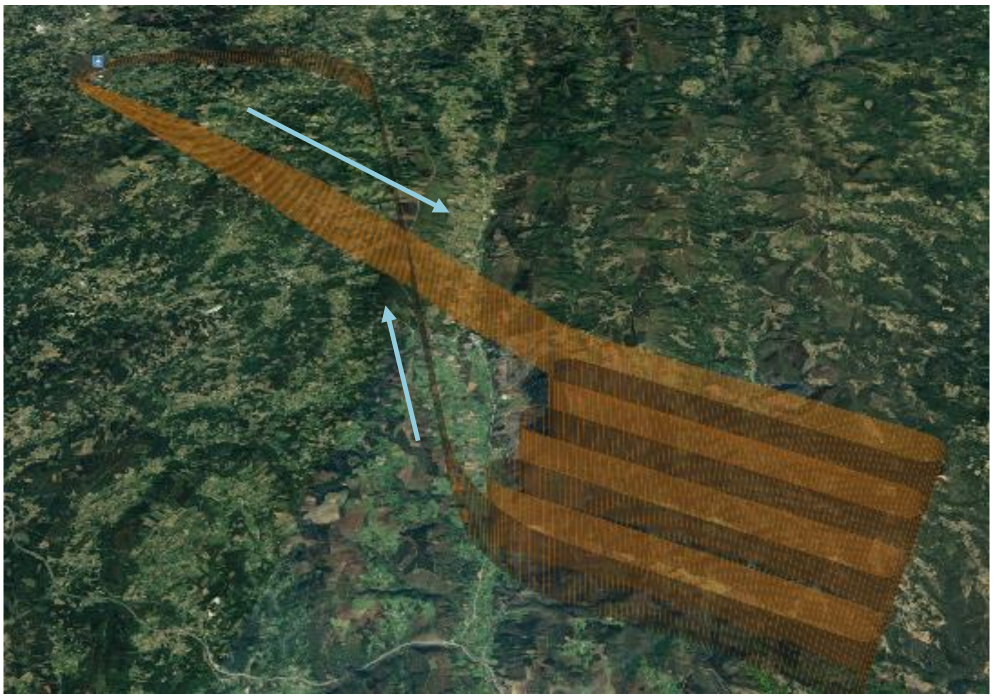


Figure 15. Mission Trajectory for scenario WX-1: East-West surveillance pattern

The results of the analysis are shown in Figure 16 and Figure 17 below

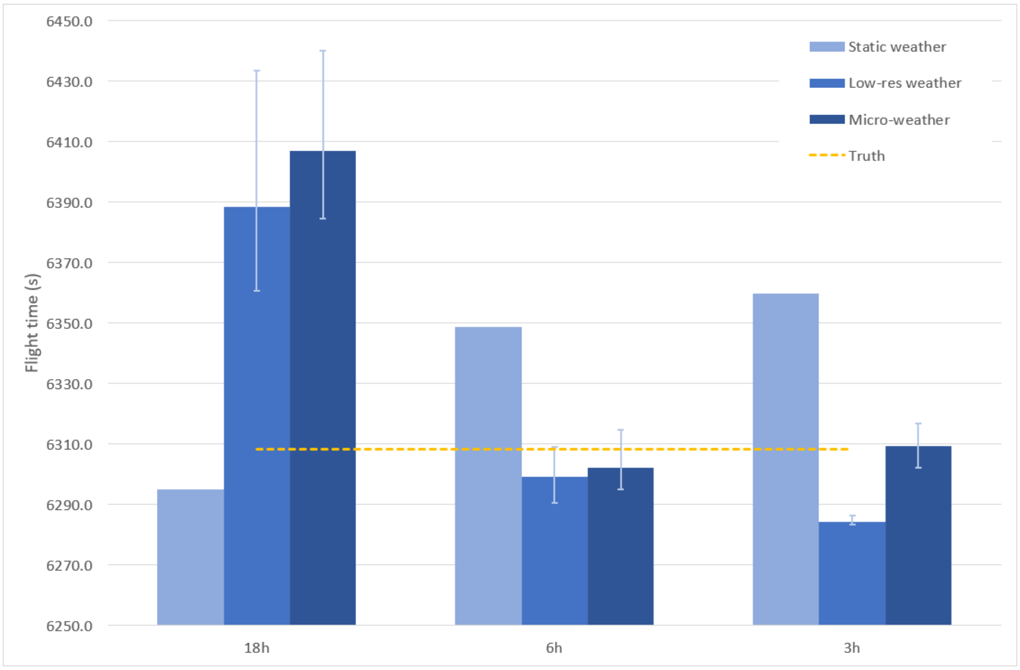


Figure 16. Flight time prediction results for East-West pattern (WX-1)

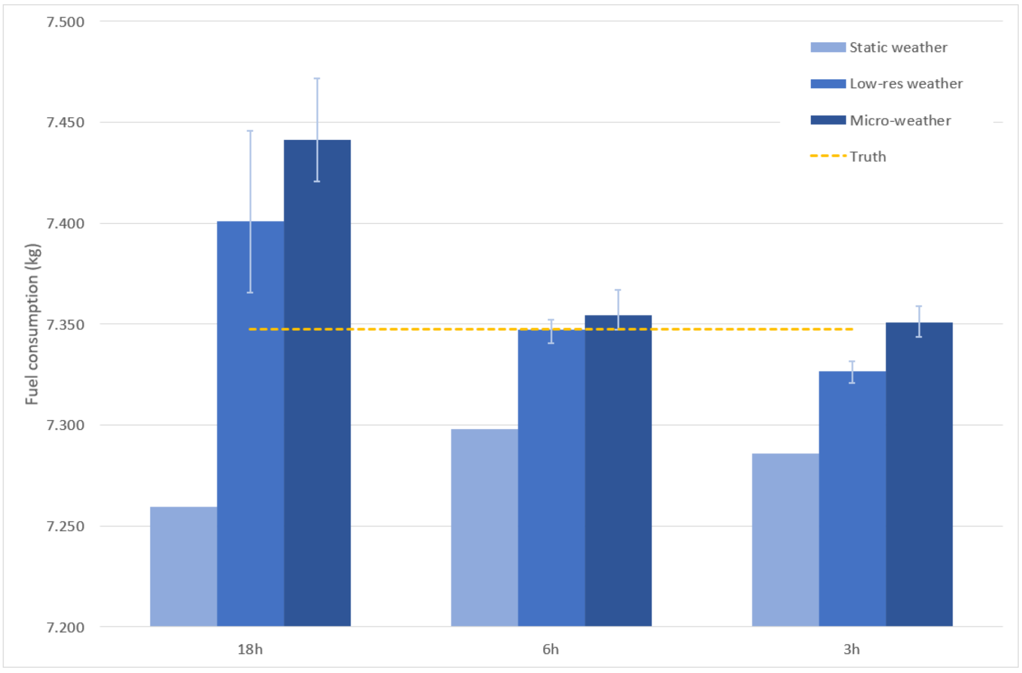


Figure 17. Fuel burn prediction results for East-West pattern (WX-1)

The convention in the figures, which will be followed to depict the results in all cases, is as follows: The dotted yellow line marks the "truth" value of the variable being analyzed, calculated as explained in the previous subsection. Predictions are shown for each of the 3 forecasts considered, each represented by a bar of a different shade of blue, and for each look-ahead time, which are shown in the horizontal axis. The lightest blue bar corresponds to the value of the variable obtained using the static forecast, the darkest blue bar corresponds to the value obtained with the high-resolution probabilistic forecast (output of the Core Weather Computation Service, i.e. the Galicia forecast) and the mid-blue bar corresponds to the value obtained with the low-resolution probability forecast. In the latter two, the bar indicates the mean of the values obtained for the different ensemble members and the deviation lines at the top of the bar indicates the minimum and maximum values, i.e. the spread across the ensemble.

As it can be observed in the figures above, the results obtained using the probabilistic high-resolution forecast accurately and robustly predict the flight time and fuel burn values 6 h and 3 h before the mission. For example, 3 hours before the mission we can estimate that the mission flight time will be within the interval 6310s ±10 s (the actual mission time is 6309 s). The spread in the flight time prediction error for the 6 ensembles, i.e. the uncertainty interval, is 20 seconds. This represents an 80% reduction in the prediction error with respect to the static forecast (from 50s to 10 s). The low-resolution forecast produces better results than the static one but it does not always guarantee that the true values are contained within the uncertainty intervals.

In conclusion, the use of probabilistic high-resolution forecasts to predict flight time and fuel burn 3 hours before take-off delivers a substantial error reduction when compared to current aviation forecasts, potentially enabling robust planning of fuel loading and mission time. These results address the Success Criteria for this case described in D5.1 [3].

However, it is important to highlight that the results, albeit encouraging, are limited in scope and cannot be generalised. The objective was to illustrate the potential benefits of the approach and we have focused only on a very specific case on a specific day with certain meteorological conditions. Further tests considering different weather scenarios, potentially with more uncertainty and a higher influence of micro-scale atmospheric effects, should be conducted to further validate the results.

It is also important to highlight the limitations of the benchmarking against the assumed true values, since those true values are estimated using the same simulation infrastructure employed to calculate the predictions. Thus, as it can be observed in the results, the prediction errors obtained are very small even with the static forecast, which results in errors in the order of 1 % for flight time and 1.5% for fuel burn. This can also be observed in the results obtained for the other cases below. Thus, further work is required to validate the reference truth values used for benchmarking. In any case, the results are still relevant since the main objective is to measure the prediction improvements obtained when using probabilistic high-resolution forecast instead of low-resolution deterministic forecasts.

To provide further details on how high-resolution probabilistic forecasts can support the mission planning process, Figure 18 and Figure 19 depict the climb and decent profiles of the mission trajectories predicted 3 hours before take-off using the Galicia forecast. The figures show the profiles obtained with each of the ensemble members as well as the reference true profile, which is shown to be contained within the spread of the results of the ensemble members.

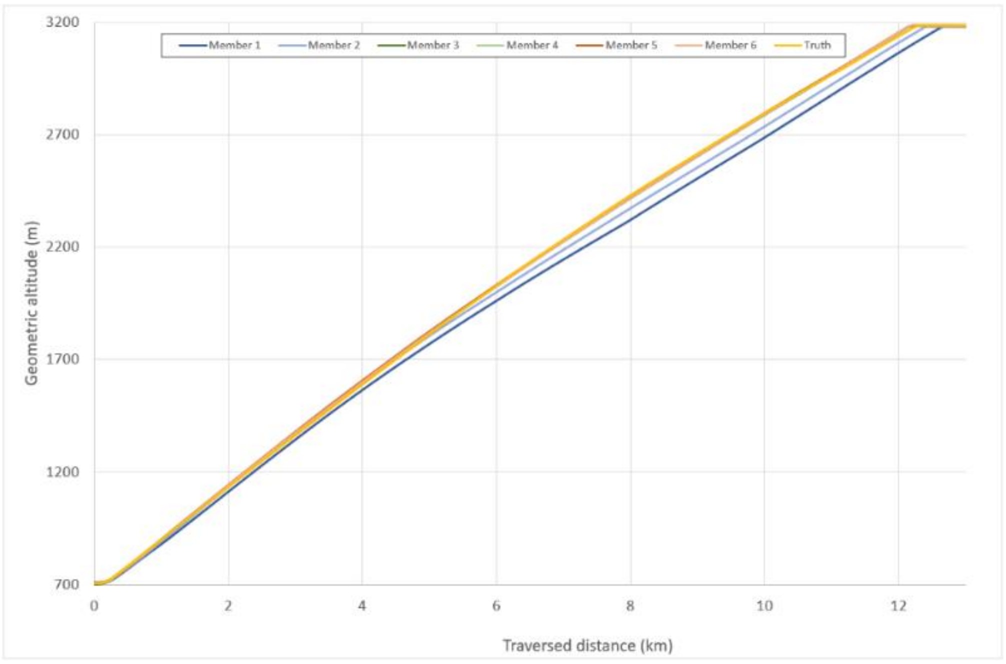


Figure 18. Predicted climb profile with Galicia forecast and 3 hours look ahead (WX-1)

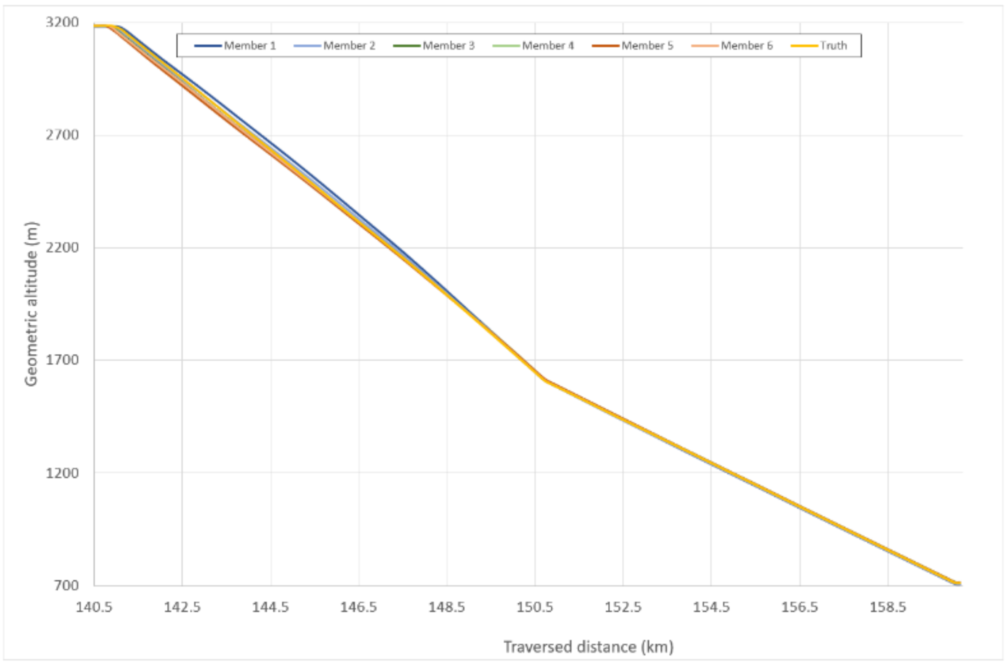


Figure 19. Predicted descent profile with Galicia forecast and 3 hours look ahead (WX-1)

Figure 20 depicts the cruise profile of the mission trajectory predicted 3 hours before take-off using the Galicia forecast. The figure shows the profiles obtained with each of the ensemble members as well as the reference true profile, which is shown to be contained within the spread of the results of the ensemble members. It is important to highlight that, since the cruise is assumed to be flown at constant pressure altitude, the resulting geometric altitude varies depending on the atmospheric conditions (pressure and temperature). Using the micro-weather probabilistic forecast, it is possible to have a robust prediction of the potential geometric altitude variation expected during the cruise. Thus, in the example, the altitude is expected to remain within a 13-meter uncertainty interval. **This is very valuable information to inform the determination of a safe vertical separation buffer between flights.**

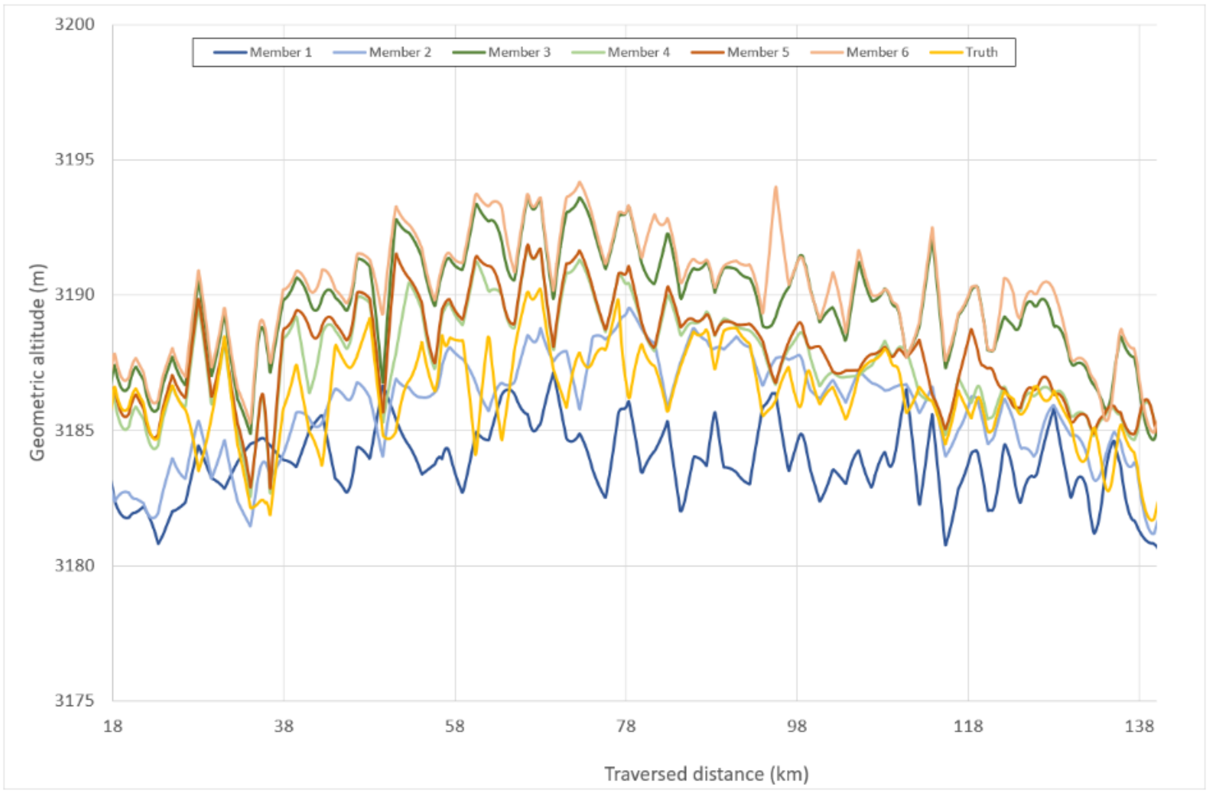


Figure 20. Predicted cruise altitude with Galicia forecast and 3 hours look ahead (WX-1)

3.2.1.2 WX-2

The validation objective is defined as: "Quantitatively measure how trajectory prediction uncertainty caused by the uncertainty associated with the weather forecast, used at mission planning time, impacts mission effectiveness and efficiency".

In this case, we compare the predicted flight time and fuel burn for two possible mission patterns aimed at fulfilling the same mission objectives in the forest fire surveillance scenario. The first pattern is the one analysed in WX-1, a lawnmower pattern where the predominant direction of movement is East-West, as shown in Figure 15. The second pattern is also a lawn mower pattern, but this time the predominant direction of movement is North-South, as shown in Figure 21.

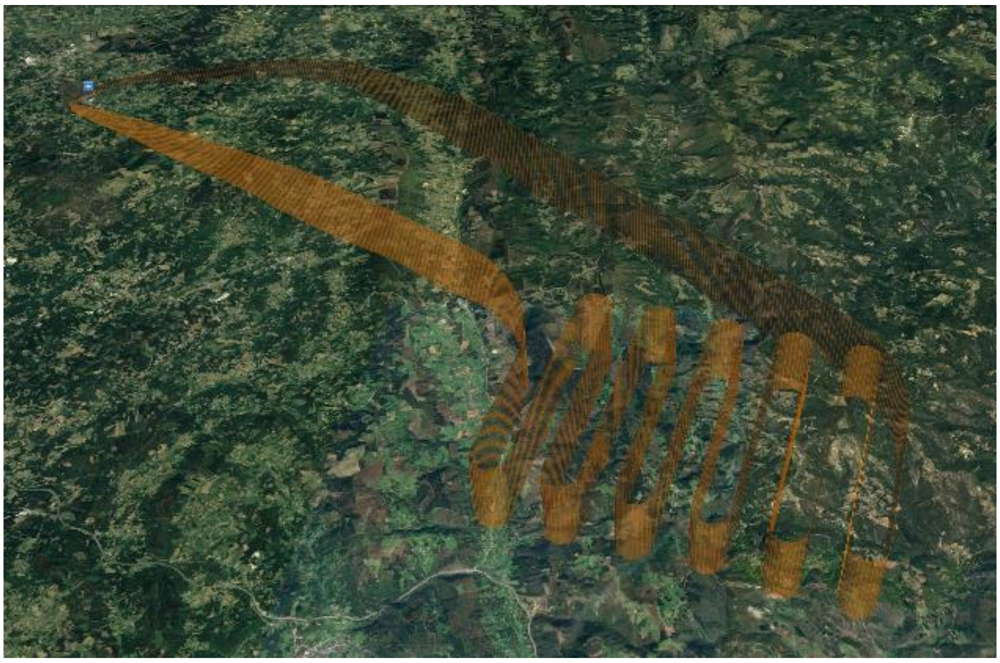


Figure 21. Mission Trajectory for scenario WX-2: North-South surveillance pattern.

The results of the analysis for the North-South pattern are shown in Figure 22 and Figure 23 below. Table 18 and Table 19 show the results obtained 3 hours before take-off for each of the two mission patterns. As it can be observed, the micro-weather probabilistic forecast improves the accuracy results for both patterns and allows capturing the differences in mission flight time and fuel burn for both patterns. This capability will allow to accurately and robustly plan the pattern for each mission to better adjust to the timing requirements as well as to optimize the timing of the successive take off times when a sequence of sorties is required to sustain a given level of surveillance continuity over the forest.

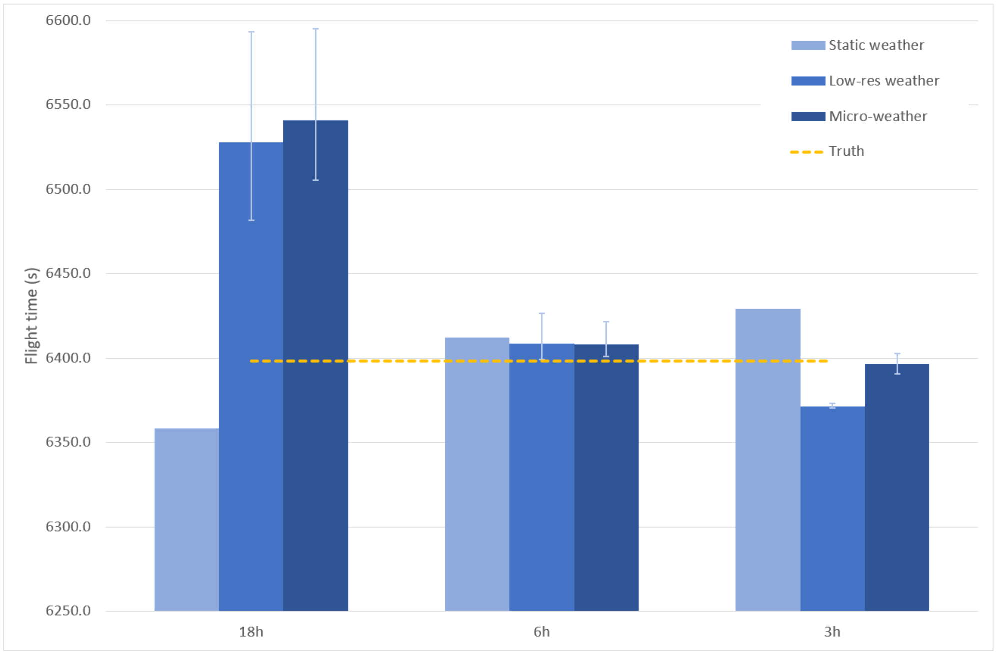


Figure 22. Flight time prediction results for North-South pattern (WX-2)

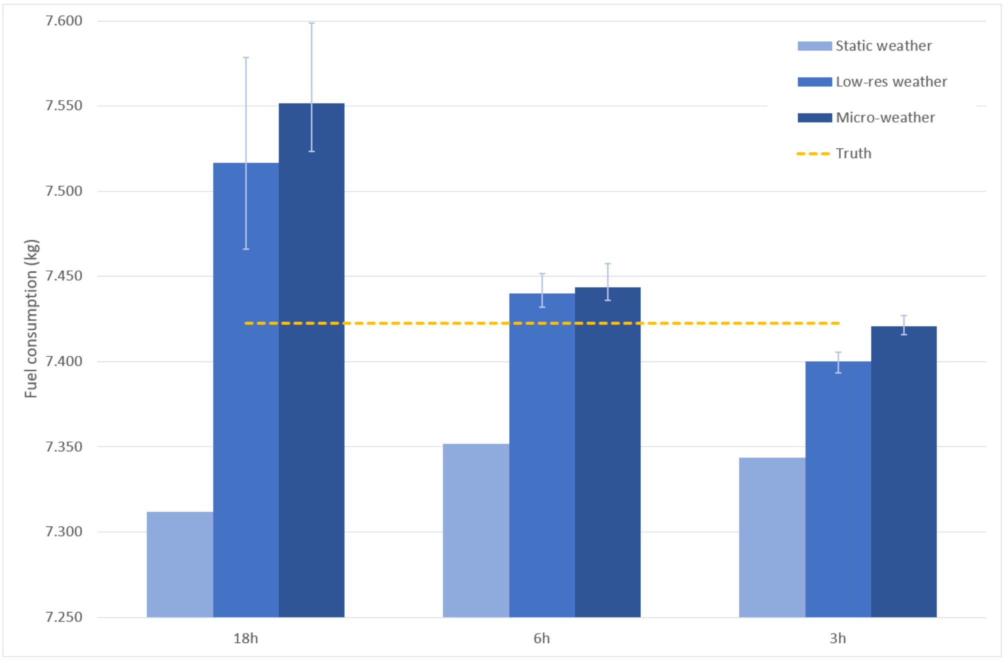


Figure 23. Fuel burn prediction results for North-South pattern (WX-2)

3h Forecasts	Avg. Time	Std. Dev. Time	Avg. Fuel	Std. Dev. Fuel
Static	6359.9	-	7.286	-
Low-res	6284.2	1.2	7.327	0.005
Micro	6309.5	7.6	7.351	0.008
Truth	6308.2	-	7.348	-

Table 18. Results for the East-West pattern

3h Forecasts	Avg. Time	Std. Dev. Time	Avg. Fuel	Std. Dev. Fuel
Static	6429.4	-	7.344	-
Low-res	6371.3	0.9	7.400	0.006
Micro	6396.5	6.2	7.421	0.006
Truth	6398.3	-	7.422	-

Table 19. Results for the North-South pattern

3.2.1.3 WX-3

The validation objective is defined as: "Quantitatively measure how trajectory prediction uncertainty caused by the uncertainty associated with the weather forecast, used at flight planning time, affects the feasibility of the flight plan – and therefore safety, in both nominal and off-nominal conditions".

In this section, we analyse the potential benefits of using the probabilistic micro-weather forecast in two use cases relevant to mission flight planning:

- Estimate the maximum range (traversed distance) and maximum flight time that can be achieved by the vehicle for a given set of mission requirements;
- Estimate whether the vehicle will be able to safely reach a suitable landing area in the event of an engine failure (contingency planning).

We have studied the above use cases for the mission profile considered in WX-1 (East-West surveillance pattern). The results of the maximum range and maximum flying time are respectively shown in Figure 24 and Figure 25. Table 20 shows the results in detail including the variations obtained with the probabilistic forecasts (uncertainty intervals).

For the 3 hours look ahead time, the probabilistic high-resolution forecast results in very accurate predictions of the actual values of maximum distance and flying time when compared with the results obtained with the other forecasts. For example, the micro-weather probabilistic forecasts reduce the maximum range prediction error from over 7 kilometres to approximately 300 meters.

In the case of the maximum traversed distance, the spread of the values obtained with the probabilistic micro-weather forecast is very small (0.02%) and the true value is marginally outside the uncertainty interval. It may therefore be necessary to add buffers to the uncertainty interval to ensure prediction robustness in situations where the uncertainty spread is excessively low (in these situations the ensemble members in the forecast may not be fully capturing the uncertainty in the weather conditions). It is interesting to highlight that the low-resolution probabilistic forecasts result in robust uncertainty intervals that include the true values of maximum traversed distance and maximum flying time with 3 hours look ahead.

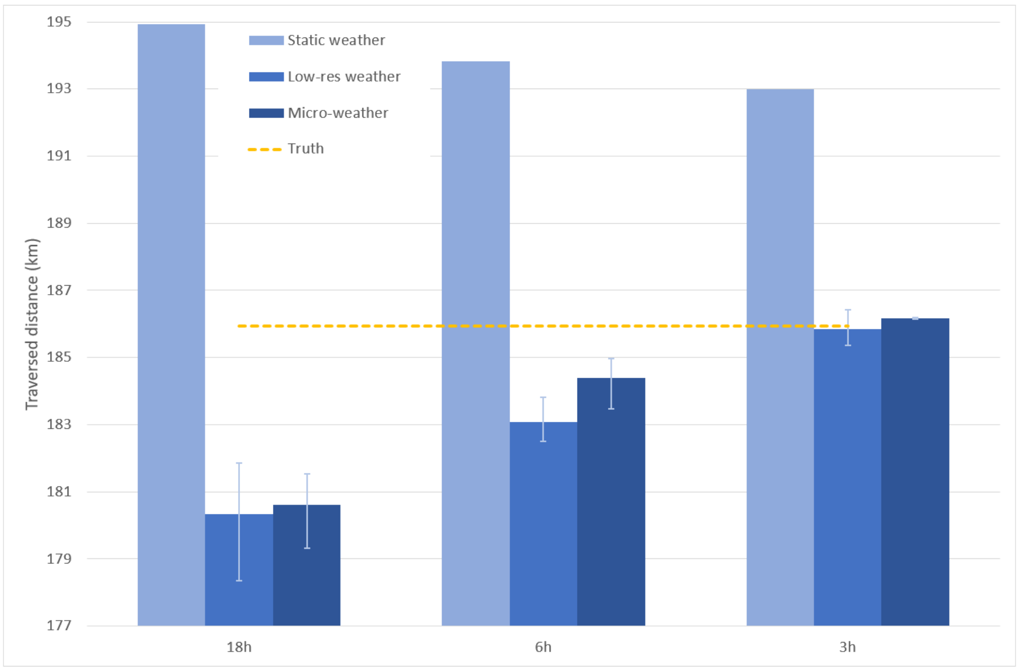


Figure 24. Maximum range prediction results for East-West pattern (WX-3)

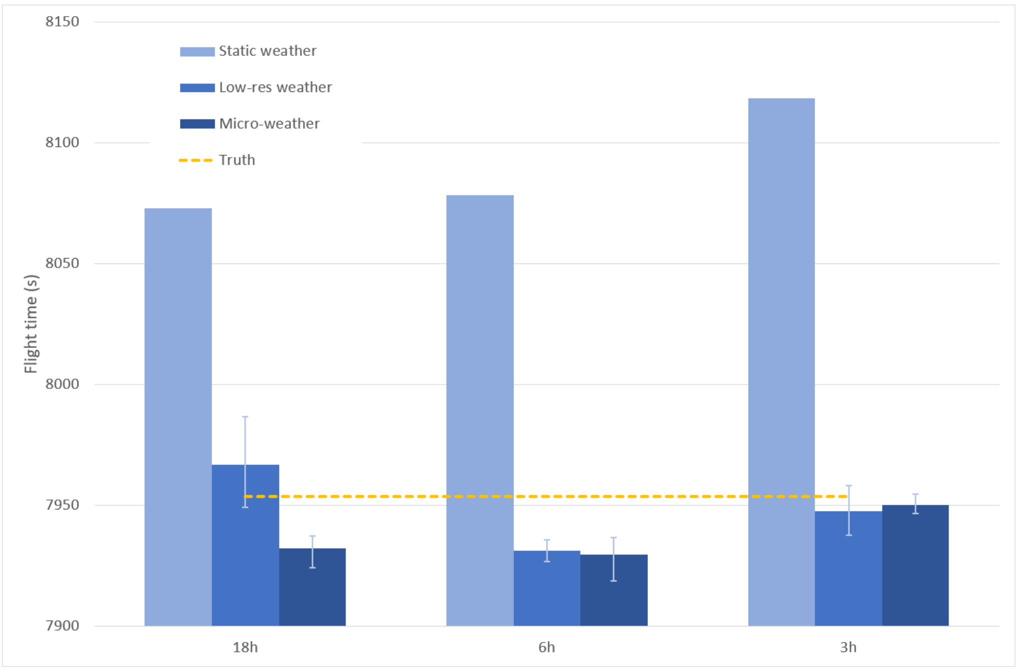


Figure 25. Maximum time prediction results for East-West pattern (WX-3)

Micro-weather	18 hr	6 hr	3 hr	Actual
Average flight time	7932 s	7930 s	7950 s	7953 s
Max. Diff. in time	13 s (0.2%)	18 s (0.2%)	8 s (0.1%)	-
Traversed distance	180.6 km	184.4 km	186.2 km	185.9 km
Max. Diff. in distance	2.2 km (1.2%)	1.5 km (0.8%)	0.05 km (0.02%)	-

Low-res weather	18 hr	6 hr	3 hr	Actual
Average flight time	7967 s	7931 s	7948 s	7953 s
Max. Diff. in time	37 s (0.5%)	9 s (0.1%)	20 s (0.3%)	-
Traversed distance	180.3 km	183.1 km	185.8 km	185.9 km
Max. Diff. in distance	3.5 mm (1.9%)	1.3 km (0.7%)	1.1 km (0.6%)	-

Static weather	18 hr	6 hr	3 hr	Actual
Average flight time	8073 s	8078 s	8118 s	7953 s
Traversed distance	194.9 km	193.8 km	193.0 km	185.9 km

Table 20. Detailed results for the East-West pattern

We have analysed the potential benefits of using the probabilistic micro-weather forecast to predict the descent trajectory in an engine failure scenario. The objective is to assess how robustly we can predict whether the vehicle would be able to glide to a safe landing location. The experiment consists of predicting the gliding trajectory of the aircraft from cruise level to the ground and compare the resulting ground distance for the 3 different weather forecasts considered. We have also compared the results with the hypothetical truth ground distance, which is the one obtained from flying the gliding trajectory in the "true" atmospheric conditions, i.e. those given by the forecast with T0=TOT. The experiment has been conducted for 3 different nominal gliding speeds with a 3 hours look ahead time. For each nominal landing speed, the resulting true ground distance is assumed to correspond to a safe landing location. Thus, if the predictions were close enough to the truth, the aircraft would reach the landing spot safely if an engine failure occurred. Figure 26 shows the descent profiles from cruise altitude in an engine failure scenario for 3 different speeds: 80 Km/h, 65 km/h and 50 km/h (slower speeds to the right of the figure). In the figure, the vertical dotted yellow lines mark the true ground distance, i.e. safe landing spot. The bundles of green, red and blue lines are the predicted gliding trajectories for 80 km/h, 65 km/h and 50 km/h descent speed, respectively. Each bundle contains 6 predictions, each corresponding to a member of the ensemble forecast. The dotted green, red and blue lines are the predictions calculated with the static forecast. As it can be observed, the predictions using the probabilistic micro-weather forecast are very close to the true distance, which is contained in the prediction uncertainty interval.

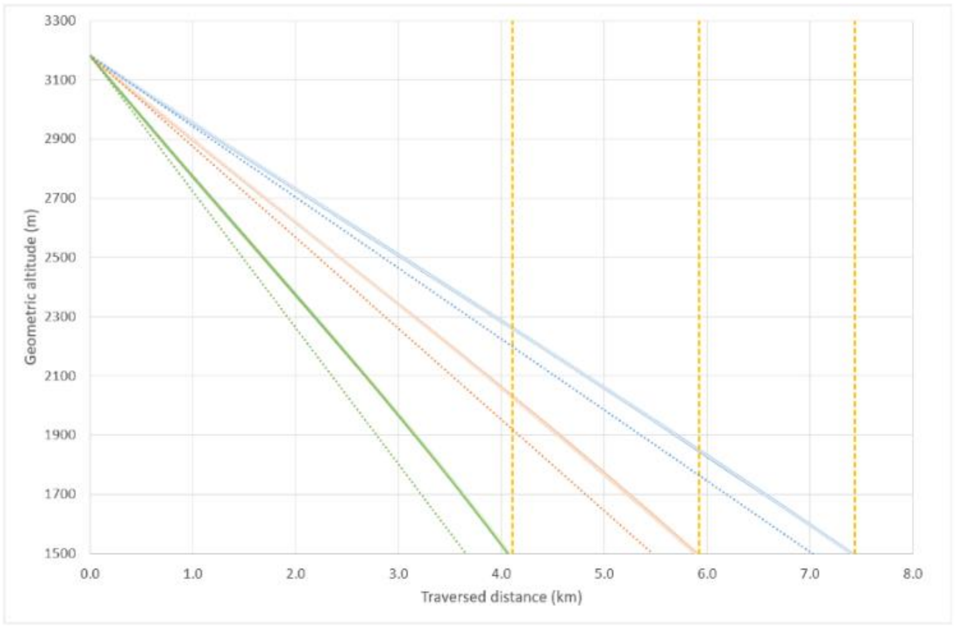


Figure 26: Gliding path from a point in cruise of the East-West pattern for 3 different descent speeds (WX-3)

3.2.1.4 WX-4

The validation objective is defined as: "Quantitatively measure how trajectory prediction uncertainty caused by the uncertainty associated with the weather forecast, used at traffic planning time (pre-tactical timeframe) impacts safe traffic separation".

In this case we analyse the benefits of using probabilistic micro-weather forecasts to predict the times when the aircraft will be at pre-defined points along the trajectory, such as the entry and exit points of a given airspace volume. This scenario is relevant to traffic management, since congestion/flow management involves adjusting the expected times at which aircraft will fly over specific reference points in or within a specific reference airspace region. Figure 27 depicts the scenario considered for the analysis, which involves a mission trajectory that traverses a reference airspace volume whose traffic density needs to be managed. Thus, we will focus on analysing the impact of using the 3 weather forecasts considered to predict the entry and exit times to the reference airspace volume.

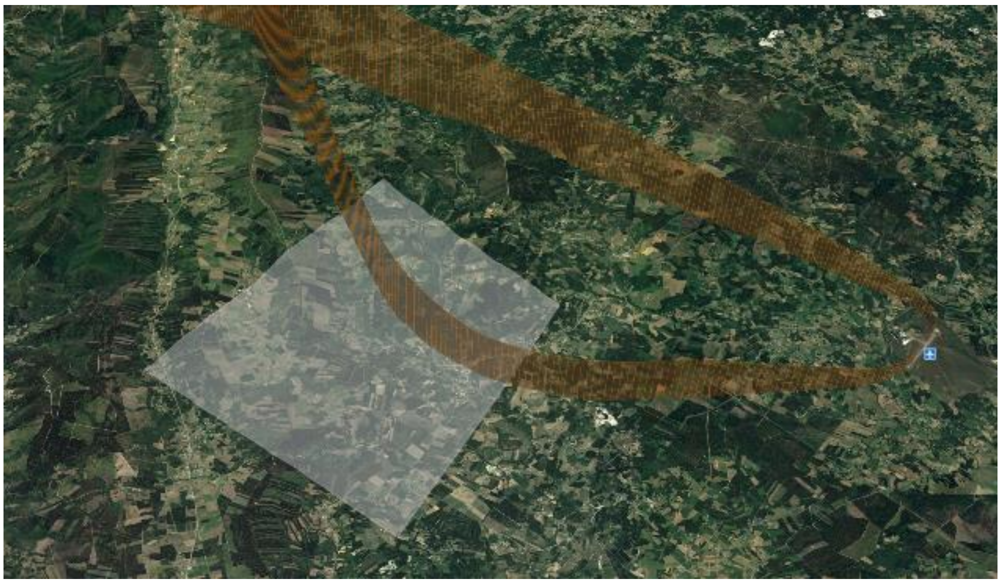


Figure 27. Mission Trajectory for scenario WX-4, showing a reference airspace volume.

Table 21 below shows a comparison of the results of predicting the entry and exit time to the reference airspace volume for a 3-hour look ahead with the 3 forecasts. The probabilistic micro-weather forecast enables a very accurate prediction of the entry (in-time) and exit (out-time) and the uncertainty interval includes the truth times. Figure 28 shows the prediction results for the probabilistic micro-weather forecast for the 3 look ahead times. For comparison purposes, Figure 29 shows the results obtained with the static forecast.

3 h Forecasts	Avg. in-time	Avg. out-time	Std. Dev. in-time	Std. Dev. out-time
Static weather	5724.0	5969.0	-	-
Low-res weather	5703.2	5928.7	1.5	1.6
Micro-weather	5712.2	5937.7	6.8	7.3
Truth	5716.2	5940.2	-	-

Table 21: Results of entry and exit time predictions

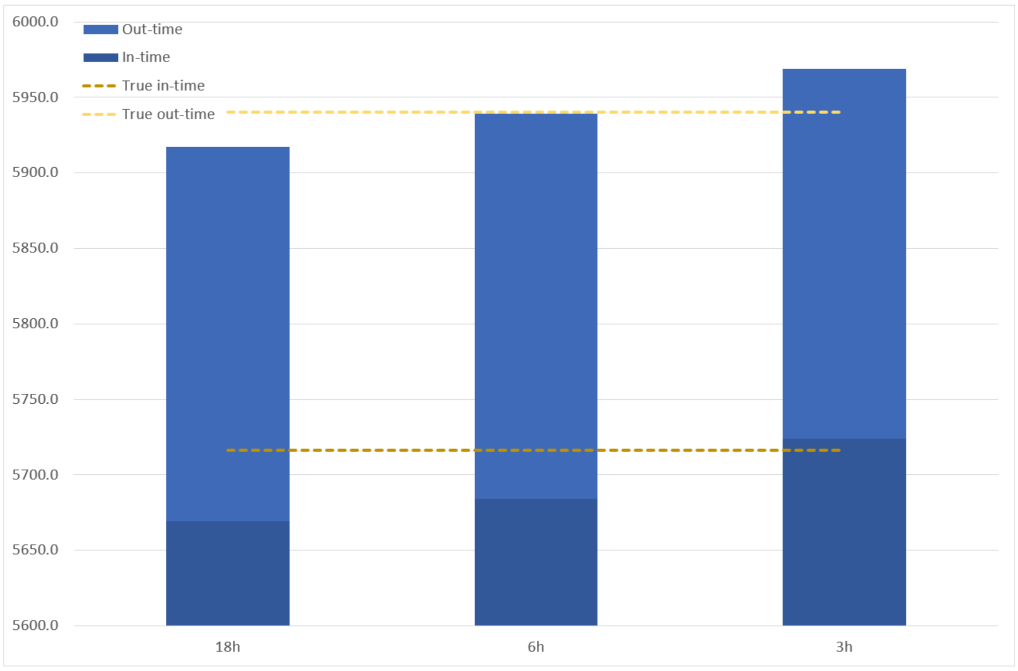


Figure 28. Prediction results for WX-4 using probabilistic micro-weather

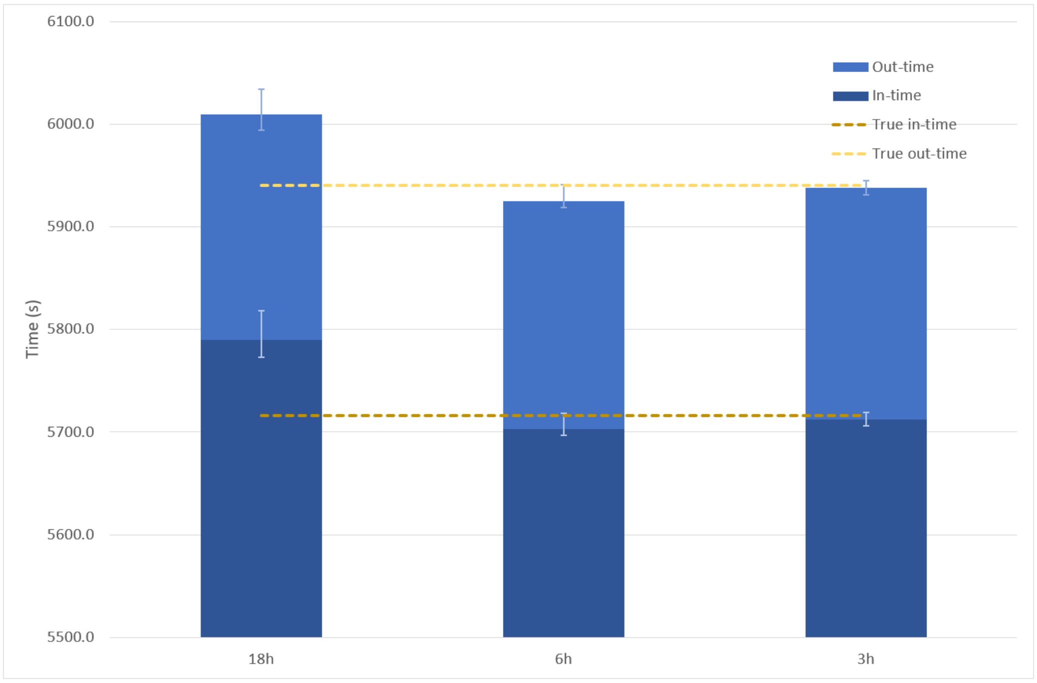


Figure 29. Prediction results for WX-4 using static weather forecast

3.2.1.5 Other probabilistic weather factors affecting mission planning

In addition to the cases described in the previous subsections, we have conducted a preliminary analysis of other potential applications of probabilistic micro-weather forecasting in mission planning beyond mission trajectory planning. Specifically, we have explored the use of probabilistic predictions of meteorological variables that can be relevant to the mission, such as visibility, precipitation and turbulence. These variables would need to be considered to assess whether the mission is feasible, safe to execute or economically viable before actually planning the flight trajectory. Thus, having a reliable probabilistic forecast of such variables could be of great value for mission planning.

Figure 30 below illustrates an example of a probabilistic prediction of one of such variables: precipitation. The prediction has been produced using the IMPETUS Micro-Weather Service (Galicia model). The initial time of the forecast (T0) is 0:00 UTC. The Weather Agent has processed the output of the ensemble forecast to calculate the probability of precipitation over a given area being greater than 15 litres/square metre and display the results on a map. It is assumed that such an amount of precipitation is the threshold that would prevent the mission from taking place. To calculate the probability, the Weather Agent checks how many of the ensemble members predict that the precipitation will be greater than the 15 litres/square meter threshold. Thus, if all the members predict that the amount of precipitation will be greater than the threshold, the probability would be 100% (red colour in the map of the Figure). If only 3 members predict it, then the probability would be 50% (green colour in the Figure), and so on. If none of the ensemble members predicts that the level of precipitation will be below 15 litres/square meter, then the probability would be zero (white colour in the map). Different shades of green correspond to probabilities between 16.7% and 83.3%.

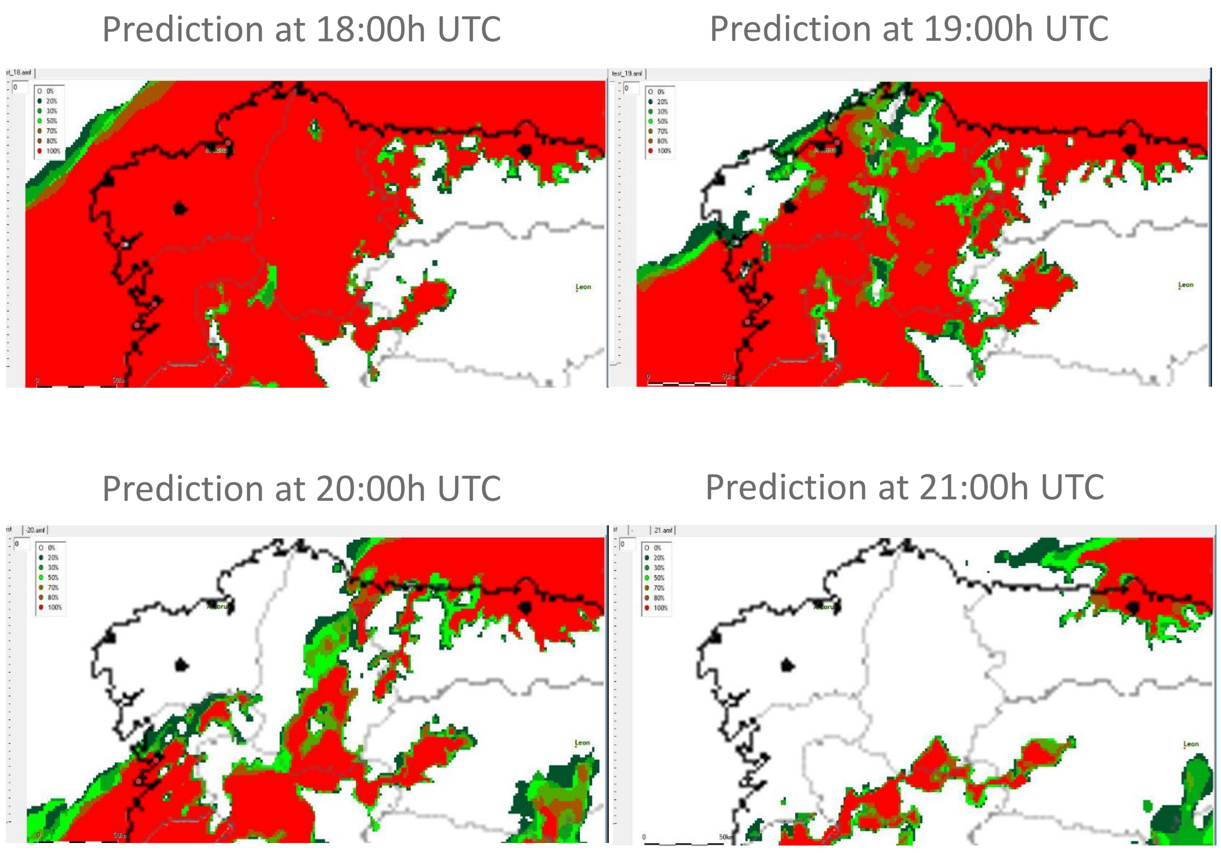


Figure 30: Probabilistic precipitation forecast using the IMPETUS Micro-Weather Service (Galicia model)

3.2.2 Mission and Flight Planning Management Services (EXE 2)

This exercise aims to validate if the implemented FPM service prototype and the interface with the MPM perform as expected for strategically planning a mission and applying deconfliction methods. The validation process is described in D5.1 §7.2. [3]. The implemented architecture deviates from the described one in D5.1, as already described in § 2.4.2.

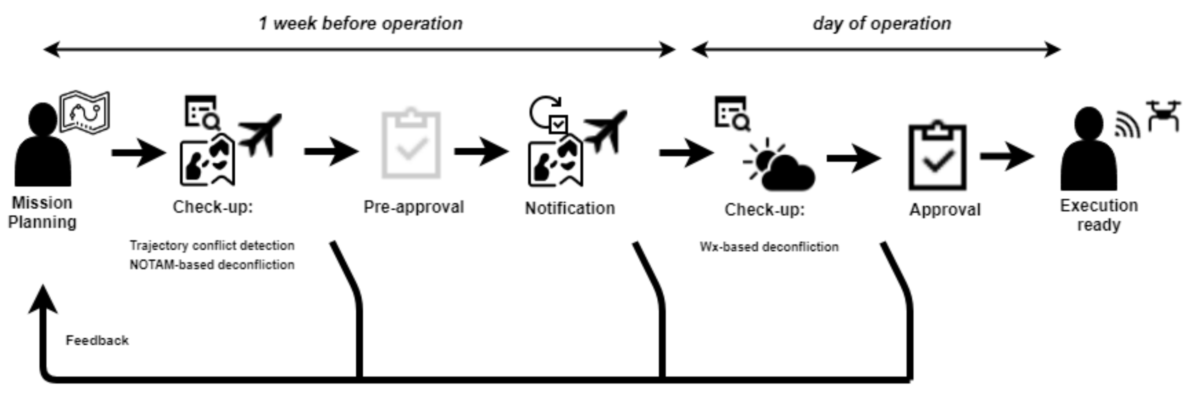
3.2.2.1 FPM-1

The validation objective is defined as: "Demonstrate that an enhanced mission planning capability can improve the overall level of safety of the approved missions as opposed to standard mission planning".

The enhanced mission planning does not only cover the achievement of mission requirements but it also incorporates safety-critical considerations. If the flight plans resulting from the mission planning are successfully validated by the centralized flight planning instance (FPM service), the whole integrated planning mechanism could be used as a feasible starting point for structuring mission and flight planning in a federated system architecture. Following table summarizes the storyboard considered in this scenario:

Use case: recurring delivery short distance (point-to-point) urban mission.
Simulation scope: day-to-day operation (2 times per day) over a 60 day timeframe.
Mission requirements: • Minimized power consumption; • Minimized flight distance; • Maximum time margin for in-flight phase: 5min; • Minimize flight time over high populated areas (parking lot, pedestrian walks) and streets; • Maximum drone speed: 4m/s (technical limitation).
Airspace and operational environment: • Compliance with restrictions in controlled airspace (e.g. max. flight altitude); • Avoidance of non-permanent airspace sectorisation restrictions (drone-relevant NOTAMs and other related TFR notifications); • Maintain a defined clearance distance to elevation of ground surface; • Consideration of maximum wind speed limitation (set by drone manufacturer).
Traffic picture: External operations taking place in the area and having different mission purposes as the use case mission (mapping, patrolling, etc.).

Temporal sequence of the complete planning process:



The enhanced mission planning capability encompasses the MPM and FPM service prototypes as described in § 2.4.2. Having the operation a regular character, the mission planning takes place a week before the operation. The FPM service is able to perform a validation process and compare the flight plan characteristics with other submitted flight plans in the database. Furthermore, a notification functionality in the FPM service informs the MPM service about updates and changes in the flight plan database. Finally, the FPM service is able to perform a check-up based on available weather information. This is considered as the last step before obtaining the approval from the centralized system.

An optimization technique is employed for generating a feasible 4D flight trajectory. The technique considers the modelling of the trajectory as an optimization problem where a cost function *J* is minimized:

$$\min_{M} J(M) = \min_{M} J(P, D, R, G, A)$$

J is a function of the power consumption (*P*), the overall distance (*D*), the flight time over safety-critical areas (*R*), clearance from the ground surface (*G*) and further restricted airspace volumes (*A*). An exemplary 4D trajectory model is shown in Figure 31. Furthermore, the MPM service provides an interactive user interface for modifying single waypoints in the trajectory if required. After confirmation from the user, a formal flight plan is elaborated by the MPM service. The following main information is included:

Flight plan data field	Description
Operator ID	Identifier assigned to the registered operator
Lateral, vertical path and timing	Sequence of waypoints and associated timestamps
Mission purpose	Description of the mission for priority assignment
Departure time	Date and time in standard format
Estimated time of arrival	Date and time in standard format

Table 22: Flight plan data components

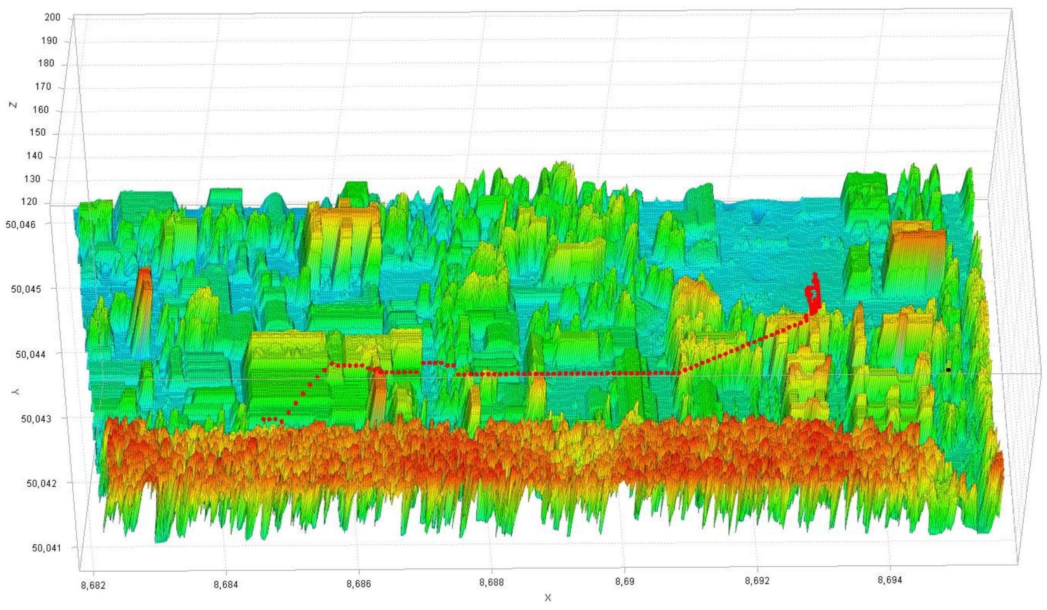


Figure 31: Flight trajectory in the spatial domain

The FPM service performs in turn a syntax and semantic check after the flight plan is submitted. Subsequently, the 4D trajectory is subjected to a validation check-up with other trajectories stored in the database (conflict detection). For this process, the conflict manager instance assigns bounding boxes in the temporal and spatial domain to the trajectory that can be used for a preliminary assessment with other trajectories. In case of preliminary conflict, the trajectory is analysed further and the conflicted legs between the waypoints are the output of the process. The conflicted leg is then notified to the MPM service. The key criteria in the conflict detection process is described as follows:

- A separation margin in the spatial and temporal domain is defined for all the trajectories, independently of the mission type (see Table 24 in next page);
- Prioritization criteria: the FPM service assigns a priority level to the flight plan according to the mission purpose (e.g. search and rescue missions have higher priority than commercial missions). In case that the priority levels are equal, the controlling instance follows a first-come-first-serve principle.

After processing the submitted flight plan, the FPM service stores the official flight plan including the following information:

Flight plan data field	Description
Flight ID	Unique Identifier assigned to the submitted flight plan
Priority level	Level assigned to the flight trajectory
Status	Requested, pending, valid/invalid, approved/rejected

Table 23: Flight plan data components completed by the FPM service

2 different types of criteria are considered for measuring the success of the validation objective:

- Criteria for determining if a planned mission is able to achieve the mission requirements:
    - All the trajectories (initial planned and re-planned after conflict detection) have a flight time lower than the maximum defined;
    - The maximum speed is not exceeded.
- Criteria for determining if a mission is safely planned:
    - 80% success rate in planning trajectories which avoid areas marked by standard NOTAMS and drone related NOTAMS. A conservative success rate value has been selected, as it is expected that the current size of NOTAM publications would impact the nominal trajectory considerably;
    - 60% success rate in planning trajectories which avoid areas of adverse weather and dangerous winds. Here, also a conservative success rate value has been chosen due to the potential large impact of adverse weather to a mission with relative short flight distance;
    - All the trajectories maintain a minimum clearance distance to the ground surface;
    - No conflicted flight trajectories appear in the database after the complete simulation. In case that a flight trajectory is initially detected, the MPM service is notified and is able to propose a new trajectory.

The following table summarizes the characteristics of the models and data samples used in the scope of the simulations:

DESCRIPTION	VALUE
Number of total flight plans for all operations. Use case flight plans and external traffic samples (FPD) (number).	500
Number of submitted flight plans for the test case mission (number).	250
Resolution of the digital surface model utilized in the optimization technique (meters).	1
Clearance margin of the trajectory from the ground surface model (meters).	20
Maximum flight altitude allowed (meters).	50
Number of official NOTAM publications analysed for all operations (number).	60
Maximum number of NOTAM publications in a single operation (number/operation).	4
Number of local weather sources in use (number/operation).	3

DESCRIPTION	VALUE
Density of drone operations in the area (drone operations at the same time / 1 square kilometre).	15
Separation margin in the spatial domain used in the conflict detection capability (meters).	50
Separation margin in the temporal domain used in the conflict detection capability (seconds).	30

Table 24: Models and data samples used in experiment FPM-1

The mission and flight planning process has been simulated for every of the test case flights within the defined timeframe. For every mission planned, the mission requirements have remained the same. Only the environmental conditions have dynamically changed according to the historical data and traffic samples available.

In many cases the FPM service detected a conflict in the submitted flight plans with external trajectories previously approved and stored in the flight plan database. Using the information provided by the FPM service that describes the conflict (conflicted leg with timing characteristics), the MPM service re-modelled the 4D trajectory still taking the mission requirements into consideration. **¡Error! No se encuentra el origen de la referencia.** summarizes the results from the simulations performed.

METRIC	VALUE
Maximum flight duration of the modelled 4D trajectory (min)	3:25
Response rate of FPM Service to flight plan submissions (flight plan requests/sec)	12.66
Update rate of FPM Service to flight plan modifications (flight plan requests/sec)	18.09
Success rate in the modelled 4D trajectories by the MPM in the 1st iteration (achievement of mission requirements) (%)	100
Number of conflicts per 100 flight plans detected by the FPM in the 1st submission iteration (number/100 flight plans)	26
Success rate in the modelled 4D trajectories by the MPM in the 2nd iteration and after a conflict has been detected by the FPM service(achievement of mission requirements) (%)	100
Maximum flight duration of the re-modelled 4D trajectory after conflict resolution by the MPM service (min)	3:35
Maximum deviation in the 4D trajectory from original in the spatial domain (%)	9%
Number of conflicts detected by the FPM in the 2nd submission iteration per 100 flight plans (number/100 flight plans)	0

Table 25: Summarized results in experiment FPM-1

In this simulation, **the provision of information about reason for conflict was sufficient for the MPM service to generate a new feasible trajectory that also meet the defined mission requirements**. The FPM did not restrict the proposed 4D trajectory by the MPM other than in the scenario where another trajectory had already previously occupied the airspace.

In any case the number of conflicts detected by the FPM is significantly high (26%) considering that the MPM service is directed to re-model the trajectory for getting approval, which ultimately carries additional computational and operational effort.

3.2.2.2 FPM-2

The validation objective is defined as: "Demonstrate that the stability of using strategic deconfliction based on trajectory models is not significantly affected by the uncertainty of predicted weather conditions and inaccuracies in local measurements".

This experiment analyses a weather-based trajectory deconfliction function. Differently than in the previous exercise, where a flight trajectory has been validated against other existing trajectories in the database, this experiment focuses solely on detecting a conflict of the trajectory with severe adverse weather conditions that could potentially represent a threat to the operation. Furthermore, after applying a *conflict resolution* mechanism, a new 4D trajectory is proposed.

Being this an initial exercise that explores the implications of deconflicting trajectories against weather models, the conflict resolution capability does not take into consideration other criteria for calculating a new trajectory other than weather model aspects. After modelling a deconflicted trajectory, it is verified whether this modified trajectory still can meet all the mission requirements described in the previous section. Moreover, and particularly in this experiment, the weather models are based on short-term reports and measurements from local stations.

The implications of applying this deconfliction mechanism with these type of models and measuring a low impact of the modified trajectories on mission requirements could facilitate a better assessment whether this process could be placed in the very pre-tactical phase of the mission (up to hours before operation) and thus not affecting significantly the high-level mission goals.

The utilized weather model includes one of the most important atmospheric variables that can compromise the safety of the mission, the wind component. A grid model has been created for the area of operation (see Figure 32) and wind measurements from local stations are fed in the model. For remaining cells where no stations are locally available, the data is interpolated with measurements from a larger scale model using a nearest neighbourhood and triangulation method. Weather reports in METAR format are generated from weather measurements coming normally from permanent weather stations in airports. Due to the very close proximity of the use case scenario to a large airport, the wind data from this type of reports is used for complementing the initial model.

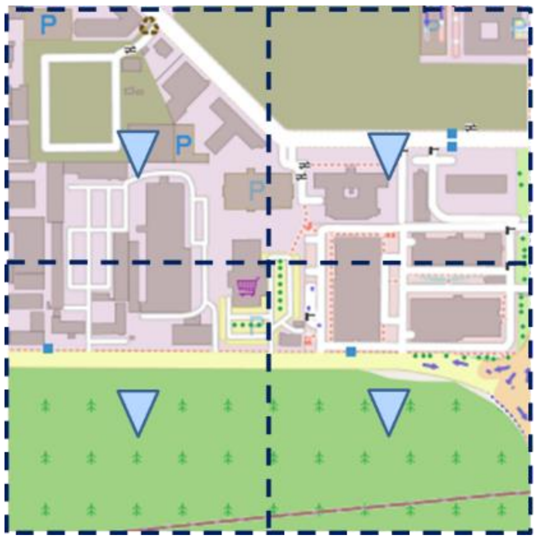


Figure 32: Exemplification of the grid model used in experiment FPM-2

The conflict detection function uses admissible wind speeds for operating the specific drone system as reference for detecting a conflict with the modelled trajectory. However, the maximum wind speed admissible is considered as limiting value for proposing a new feasible trajectory. Furthermore, the standard deviation between the different data sources within a grid cell is taken into account for the conflict detection analysis in order to consider the uncertainty in the utilized data samples. The key criteria for conflict detection are defined as follows:

- For wind speed measurements under an admissible threshold: condition regarded as not adverse and no conflict is detected;
- For wind speed measurements over an admissible threshold, but still under the maximum admissible value: condition regarded as adverse and conflict is detected;
- For wind speed measurements over the maximum admissible value: condition regarded as not feasible for proceeding with the flight planning.

In case that a conflict with the weather model has been initially detected, the conflict resolution algorithm generates a new trajectory that specifies a larger clearance value in the three dimensional spatial domain from the digital elevation model. Figure 33 exemplifies this mechanism. Ultimately the criteria for specifying the clearance value is the following:

- Small deviation (related to the standard deviation defined in Table 24) in measurements from different sources in grid cell (low weather uncertainty): small clearance value from elevation model;
- Large deviation (related to the standard deviation defined in Table 24) in measurements from different sources in grid cell (high weather uncertainty): large clearance value from elevation model.

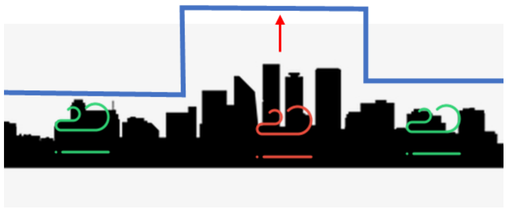


Figure 33: Conflict resolution applied in experiment FPM-2

3 different types of criteria are considered for measuring the success of the validation objective:

- Criteria for assessing the accuracy of the weather model applied for conflict detection:
    - Deviation (standard deviation) in measurements from different sources within a grid cell.
- Criteria for determining if the deconfliction solution performs as expected:
    - The resolution certainty of weather-based strategic deconfliction solutions is greater than 95%. This means that the deconfliction solution is able to find a feasible trajectory.
- Criteria for determining the impact of the deconfliction solution on the mission goals:
    - Less than 10% of the results of strategic deconfliction solutions which utilize weather models deviate by more than 5% from the total trajectory length and flight time from those that do not;
    - The trajectory has a flight time lower than the maximum defined (As in FPM-1 specified: 5 min).

The following table summarizes the characteristics of the models and data samples used in the scope of the simulations:

DESCRIPTION	VALUE
Grid size length of weather model (m)	250
Number of local weather sensor platforms in use (number)	2
Update rate of local weather sensor platforms (update/hour)	2
Update rate of official weather reports (update/hour)	2
Threshold for admissible wind speed values (drone-specific) (km/h)	30
Maximum admissible wind speed value (drone-specific) (km/h)	46
Small deviation (standard deviation) range in measurements from different sources within a grid cell (km/h)	3-7

DESCRIPTION	VALUE
Clearance value applied for small deviations (m)	5
Large deviation (standard deviation) range in measurements from different sources within a grid cell (km/h)	7-14
Increase of clearance value applied for large deviations (m)	10

Table 26: Models and data samples used in experiment FPM-2

In all the performed simulations, the deconfliction solution was able to produce a new 4D trajectory in the case of conflict detection. However, the deviation **in** the newly generated trajectories from the original is as large as 7% of the trajectory length in the spatial domain. However, only in less than 10% of the cases the adapted trajectory deviates more than 5% from the initially created trajectory in the spatial domain.

Table 27 summarizes the results from the simulations performed.

METRIC	VALUE
Number of small deviations in measurements from different weather sources per 100 data measurements (number/100 measurements)	36
Number of large deviations in measurements from different weather sources per 100 data measurements (number/100 measurements)	4
Number of conflicts detected per 100 simulated trajectories (number/100 flight trajectories)	12
Maximum flight duration in the initial generated 4D trajectories (min)	3:26
Maximum flight duration in the deconflicted 4D trajectories (min)	3:55
Maximum deviation in the 4D trajectory from original in the spatial domain (%)	7
Success rate in the deconfliction solution producing a trajectory that meets mission requirements (%)	100

Table 27: Summarized results in experiment FPM-2

The maximum 7% deviation in the deconflicted trajectory from the original did not affect the overall mission goals in this scenario (maximum flight duration of deconflicted trajectories was less than maximum accepted). However, this could not be the case given the situation that the clearance value is defined higher.

3.2.2.3 FPM-3

The validation objective is defined as: "Demonstrate that the strategic deconfliction based on trajectory models can provide feasible alternatives in the presence of large airspace sectorisation".

This experiment explores solely a trajectory deconfliction function that adapts trajectory models conflicting with temporal airspace sectorisation publications. In the scope of the experiments official standard NOTAM publications have been considered and all drone related publications have been filtered and further analysed.

Analog to the previous experiment, this one analyses a deconfliction capability that solely modifies trajectory models conflicting with temporal airspace sectorisation publications. In the scope of the experiments, official standard NOTAM publications have been considered and all drone related publications have been filtered and further analysed.

Temporal airspace sectorisation publications do not always include coordinates that facilitate the creation of a closed geographical area. In the normal case, only a radius from the geographical location is provided. Furthermore, the size of the radius is generally larger than 1km. This implies that **avoiding an area marked as restrictive can cause deviations that could compromise the achievement of mission requirements**. On the other side, these types of publications are generally published well in advance, so that the adaption of the mission can be also planned well in advance.

The deconfliction capability extracts the relevant characteristics from the NOTAM publication (coordinates, duration) and compare this information with the specifications of the modelled 4D trajectory. A relevant attribute that is extracted is the differentiation of warning and restriction publications, which can be extracted from the category and the text description. Figure 34 shows the modelling of a typical publication in the area where the simulations take place. If a conflict is detected at this stage with a NOTAM characterized as restricted area, the trajectory model is adapted for not violating the occupying NOTAM volume in the spatial domain.

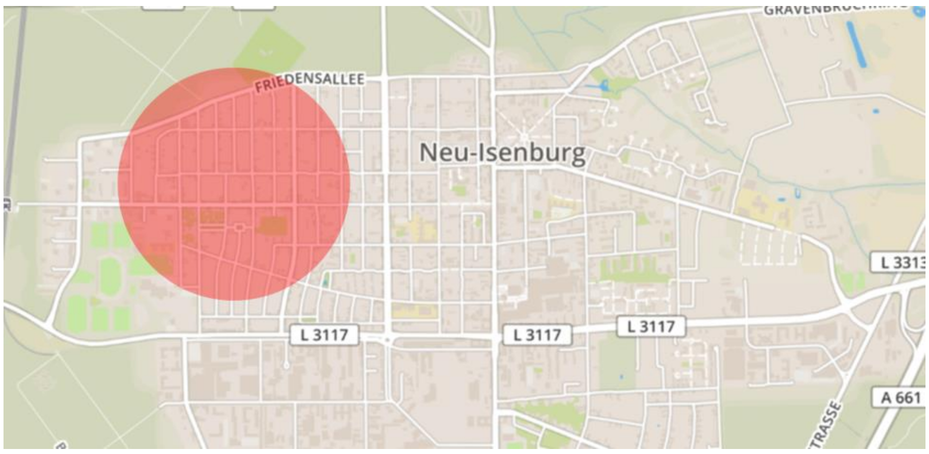


Figure 34: Representation of a NOTAM publication

2 different types of criteria are considered for measuring the success of the validation objective:

- Criteria for determining if the deconfliction solution performs as expected:
  - The resolution certainty of non-permanent airspace sectorisation information-based strategic deconfliction is greater than 95%. This means that the deconfliction solution is able to find a feasible trajectory.
- Criteria for determining the impact of the deconfliction solution on the mission goals:

- Less than 50% of the results of strategic deconfliction solutions which utilize specific aeronautical information deviate by more than 10% from the total trajectory length and flight time from those that do not;
- The trajectory has a flight time lower than the maximum defined. (As in FPM-1 specified: 5 min).

The following table summarizes the characteristics of data samples used in the scope of the simulations:

DESCRIPTION	VALUE
Number of drone-related NOTAMs characterized as warning type (number)	60
Number of drone-related NOTAMs characterized as restricted area (number)	38
Minimum radius specified in the drone-related NOTAM (km)	0.5
Maximum radius specified in the drone-related NOTAM (km)	2
Minimum activity duration of the drone-related NOTAM (hours)	1.5
Maximum activity duration of the drone-related NOTAM (days)	88

Table 28: Models and data samples used in experiment FPM-3

In 96% of simulations the deconfliction function has been able to produce an adapted trajectory model that does not conflict with the occupying 3D volume model of the NOTAM publication. For the rest of the times the conflict detection algorithm was not able to extract all the required characteristics for comparing the NOTAM volume with the 4D trajectory.

In regard to the second success criteria, the deviation in the trajectory from the original is as high as 39% in the spatial domain. Moreover, in less than 36% of the cases the adapted trajectory deviates more than 10% from the initially created trajectory in the spatial domain.

The following table summarizes the results from this experiment:

METRIC	VALUE
Number of trajectory models analyzed in the exercise	250
Number of conflicts detected with trajectory model (number)	25
Resolution certainty of the deconfliction solution(%)	96
Maximum flight duration in the initial generated 4D trajectories (min)	3:25
Maximum flight duration in the deconflicted 4D trajectories (min)	5:12
Maximum deviation in the 4D trajectory from original in the spatial domain (%)	39

METRIC	VALUE
Number of deconflicted trajectories that were not able to meet the mission requirements (number)	3

Table 29: Summarized results in experiment FPM-3

Differently as in the previous experiment, the resolution certainty of the deconfliction capability was not 100%. Moreover, here there were **deconflicted trajectories that were not able to meet one of the mission requirements** which was related to the maximum flight duration.

3.2.3 Monitoring and Traffic Information Services (EXE 3)

This section addresses the success criteria defined in section § 2.3.1, that describes the U-space services involved. This exercise focuses on the operational performance of these services in a congested airspace in which very close operations are being performed at the same time as it can be seen in the following figure.

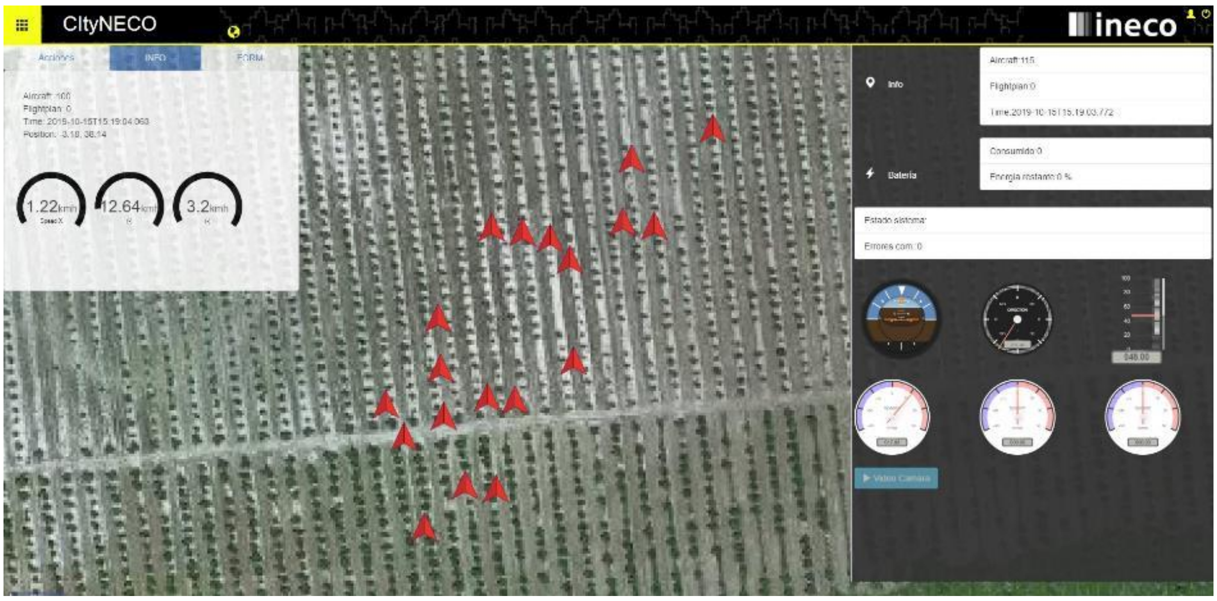


Figure 35: HMI of the microservice platform showing the congested airspace and alerts related to operations (in red)

3.2.3.1 MTI-1

This validation objective is defined as: "Quantitatively measure how many alerts about dangerous traffic situations are received by users in an appropriate amount of time".

Success Criteria 1 (SC1): All dangerous situation alerts are received by the end users with less than 5 second latency.

Success Criteria 2 (SC2): 99% of created alert messages are received by the end user (the Alarm Microservice).

The platform has been configured to measure the time that a position message is received, enters into the Alarm Microservice and is processed and finally shown in the user interface. These values have been extracted from some of the tests described in section 2.4[1].

The results are presented using the following structure:

- A table with the metrics and success criteria, in which the reached are marked with an "X". The reached value is included in the "VALUE" column;
- Diagrams containing information processed from the experiments.
- A short conclusion about the preliminary findings, that is more detailed in § 4.

MTI-Test 1: Latency

METRIC	VALUE	SC1	SC2
Performance over time/delay of the alert messages (seconds) considering the interval between the positions being processed and the alert shown to the U-space user.	0.25 s	X	
Alert messages shown (%alerts/incidents)	100%		X

Table 30: Metrics of MTI-Test 1: Latency

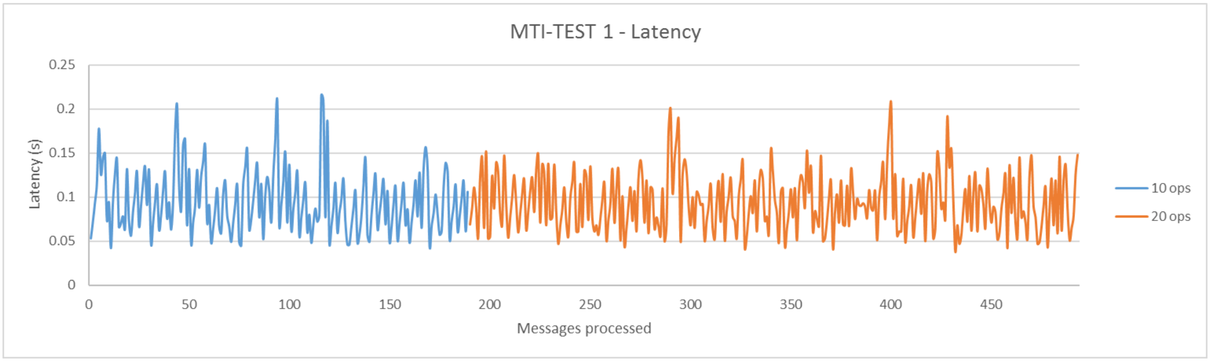


Figure 36: Latency of messages processed in experiment MTI-1 for 10 and 20 drone operations

[1] Not all the tests described in section 2.4 are valid for all the objectives and success criteria, which are defined to validate the covered services.

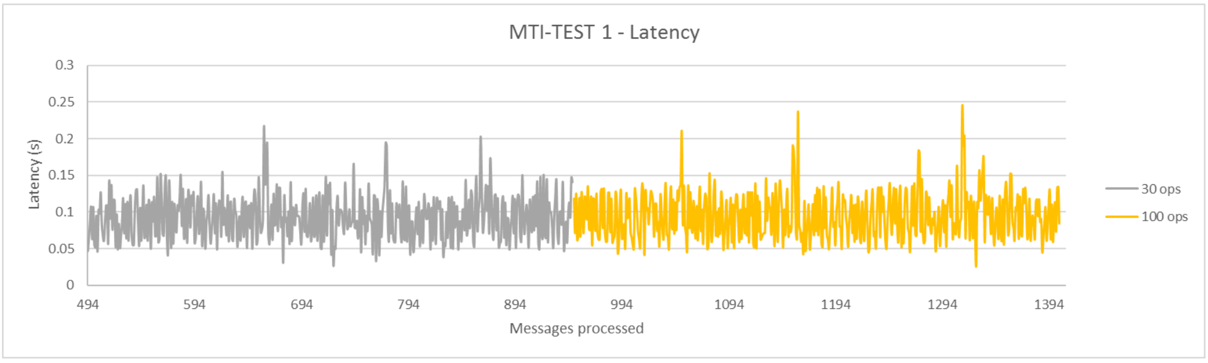


Figure 37: Latency of messages processed in experiment MTI-1 for 30 and 100 drone operations

The main conclusion is that the processes are simple enough to avoid significant delays while increasing the number of drones in operation, rising from a maximum of 0.243s to 0.25s between 10 and 100 drones. Each drone sends 1 message per 0.1 seconds and they can be received simultaneously, that means, it can even be collected at the same time as other messages. Moreover, additional instances have not been created, as the platform itself is capable of absorbing the workload. The identified peak is 41% of its capacity in the case of the MAVLink microservice.

On the other hand, comparing the number of alert messages generated and the ones shown in the HMI, all of them have appeared in the web browser. This ensures that all users have been alerted and no information has been lost during the information flow.

MTI-Test 2: Scalability

METRIC	VALUE	SC1	SC2
Performance over time/delay of the alert messages (seconds) considering the interval between the positions being processed and the alert shown to the U-space user.	0.28 s	X	
Alert messages shown (%alerts/incidents)	100%		X

Table 31: Metrics of MTI-Test 2: Scalability

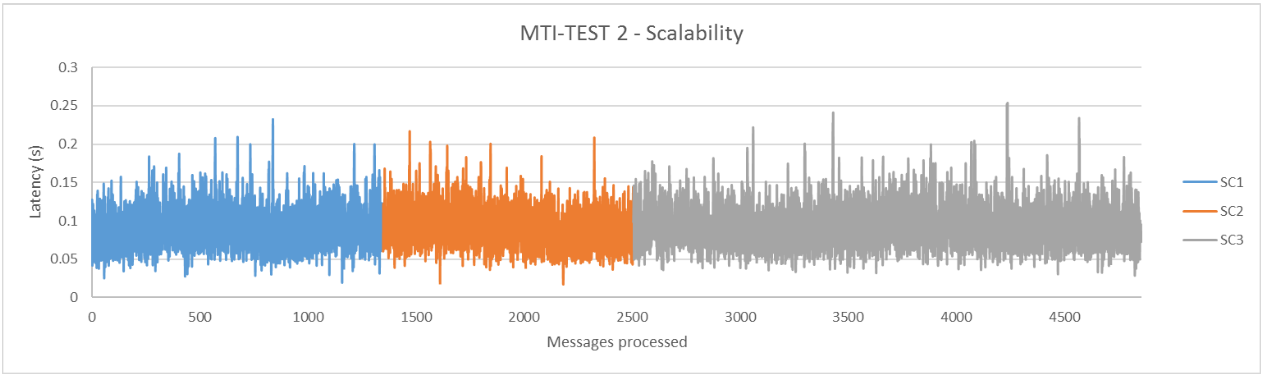


Figure 38: Latency of messages processed in experiment MTI-2 (1)

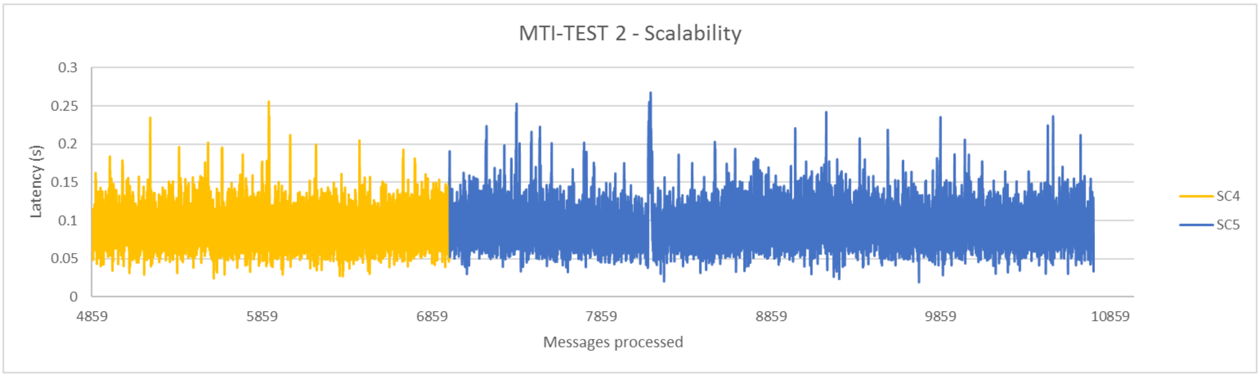


Figure 39: Latency of messages processed in experiment MTI-2 (2)

The previous diagrams show the slight increase in the use of resources when integrating more number of drone operations and, as it can also be shown, once the delay has a peak value (due to sudden increases), it is always absorbed by the system. Thus, the system adapts to reduce this number, balancing the load or creating instances if necessary. For this specific case, no additional instances were needed, as the platform was capable of stabilizing the bottleneck caused by the rises in the number of messages.

This case also reflects the power of the application, as it has been capable of processing and shown all the message alerts.

MTI-Test 5: Maximum workload (without filter)

METRIC	VALUE	SC1	SC2
Performance over time/delay of the alert messages (seconds) considering the interval between the positions being processed and the alert shown to the U-space user.	43.96 s		
Alert messages shown (%alerts/incidents)	100		X

Table 32: Metrics of MTI-Test 5: Maximum workload

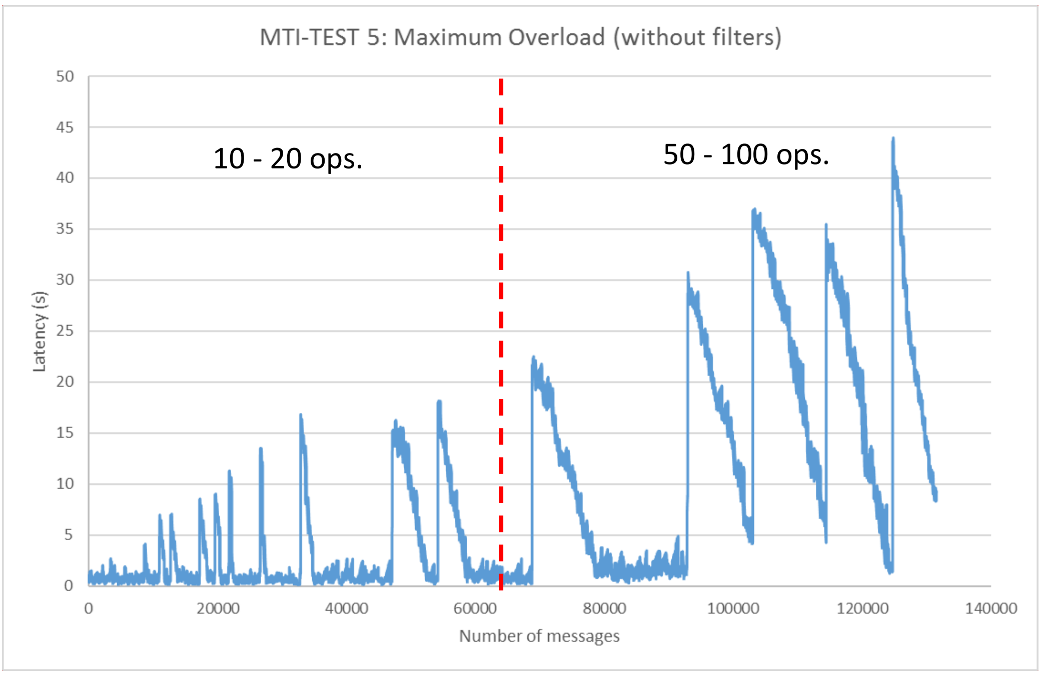


Figure 40: Latency for a maximum overload scenario in experiment MTI-5

This case is of special interest. The platform has been forced to process all the existing messages, no matter their topic, and large amounts of information have been injected in several runs causing an overload in the microservices that has forced them to increase to the maximum possible number of instances or, in other words, using all the 5 virtual machines available. Even in the case in which this is not enough to absorb the high demand (some of the microservices have registered a demand of 120% of its capacity), the platform is not collapsing (if a microservice fails it will be replaced by another one in a matter of seconds). However, the delays are too high that it should be considered to expand the number of computers available for this process. Another solution could be the integration or duplication of different message listeners, distributing the topics between them or sharing the load in these cases.

It is important to be flexible enough not only in the architecture implemented, but also in the physical container in which it is being executed to ensure that the platform will have enough resources to ensure its proper execution.

On the other hand, even in this case, all of the alert messages have been processed, although they have been provided to users with a significant delay.

As a conclusion, it is possible to say that the validation objective has not been met, if mechanisms and improvements to avoid this bottleneck are implemented.

3.2.3.2 MTI-2

This validation objective is defined as: “Demonstrate that a micro-service based architecture has enough capacity to process the huge amount of inputs expected in congested environments”. Three success criteria are tested.

Success Criteria 1: Considering increases in processing times due to overloads, the position message shall be acquired and integrated in the information flow of the platform in a time frame of less than 0.1s.

Success Criteria 2: Considering increases in processing times due to overloads, the position message shall be acquired and shown to the operator in a time frame of less than 1s. (AM)

Success Criteria 3: In congested airspaces[2], the alerts due to separation incidents shall be provided to users in less than 1s. (AM)

Before describing the different tests that have been performed to check these criteria, some common results can be extracted. In particular, and especially for SC1 and SC2: once the position has been collected by the system, it takes a very small amount of time to process this data and prepare it for the user interface. Indeed, after registering the entering and exiting time stamps, the greatest value obtained is 0.001s, which means that this process is almost instantaneous. The atomicity of processes in this service and the use of simple messages based on a predefined standard (JSON) are the cause of a lightweight execution that is not significantly affected by overloads.

The important topic in this validation objective is the processing of position information and how it is affected by increases in demand. MTI Tests 2 and 5 will be used to assess the validity of this premise as MTI-Test 4 has analogous behaviour as MTI Test 2.

MTI-Test 2: Scalability

METRIC	VALUE	SC3
Delay of a position message since it is acquired by the system, processed with the other flights in the same area and provided to users (seconds).	1.92 s	

Table 33: Metrics of MTI-Test 2: Scalability

[2] Congested airspaces are defined as volumes in which a high number of drone operations is performed simultaneously, causing conflicts between them and raising the necessity of warning the users about potentially catastrophic situation and provide mechanisms to avoid issues in the operations.

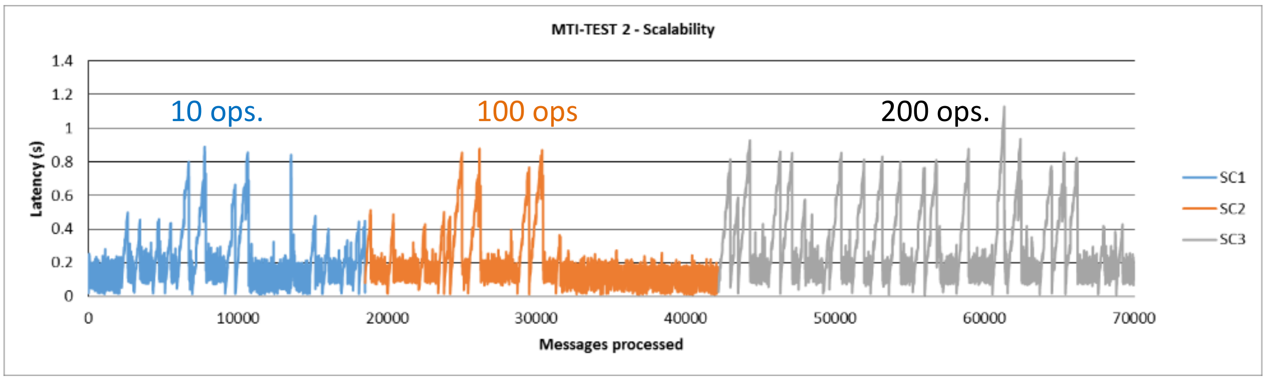


Figure 41: Latency of messages processed in experiment MTI-2 for 10, 100 and 200 drone operations

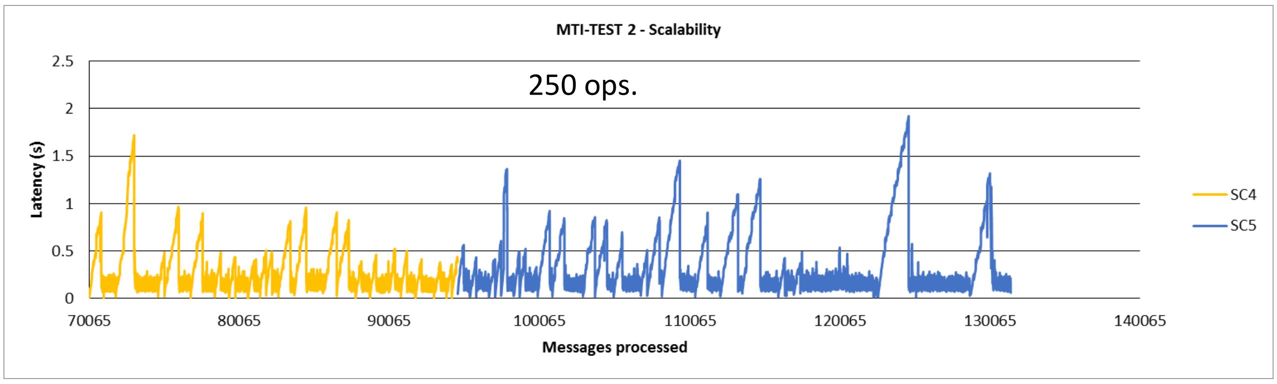


Figure 42: Latency of messages processed in experiment MTI-1 250 drone operations

The steps in the latency can be due to sudden increases of the number of drones in the scenario that is being analysed.

As it can be seen, the success criterion is not met. After reviewing the behaviour of the architecture, the problem has been detected. As microservices are configured to allow the 80% of its capacity and this limit has not ever been reached, the first conclusion that can be extracted is that it is not an architecture's fault. The problem lies in the web interface that is not capable of processing and show all these positions. This problem was also considered when defining the requirements of this architecture but it was finally decided to show all the messages (every few milliseconds) to test also the capacity of the web browser. Moreover, this raises another question: which would be the proper update rate in the user interface to access to this information?

Maybe it is not necessary to show all the information processed in the HMI but, depending on the circumstances and the criticality of the operations, this rate needs to be reduced to ensure on-time reactions to potential issues.

MTI-Test 5: Maximum workload (without filter)

METRIC	VALUE	SC2
Delay of a position message since it is acquired by the system, processed with the other flights in the same area and provided to users (seconds).	196.35 s	

Table 34: Metrics of MTI-Test 2: Maximum workload (without filter)

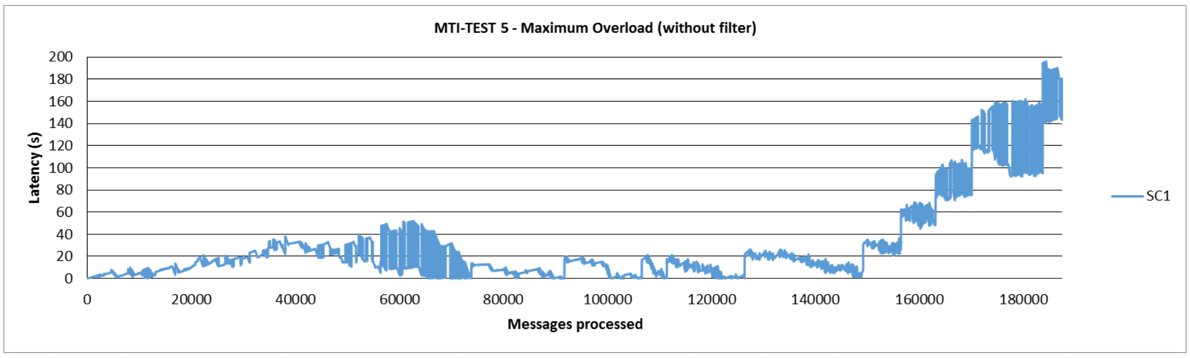


Figure 43: Latency for a maximum overload scenario in experiment MTI-5

The consequence that has been described in the previous validation objective becomes even more important in this case. Checking changes in the microservices load, it can be seen that it is due to the overload in MAVLink services due to the necessity of filtering all the variety of data streams of different nature and, after that, showing the positions in a web interface. The filtering processes are always high consuming and need to be parallelized and decoupled when possible (e.g. establishing different layers for different messages). The limited capacity of the browser makes the messages to be stored in the queue, waiting for the previous streams to be represented and delaying the delivery of this information.

MICROSERVICE	NUMBER OF INSTANCES	MÁX. WORKLOAD
Alarm	1	21%
MAVLink	3	120%
Listener	1	1%

Table 35: Number of instances and maximum workload for the considered microservices

As it has already been stated, not all the streams are of interest of each microservice running, so a mechanism to pre-filter all the inputs must be implemented. On the other hand, position datasets shall be shown to users and the update rate must be adapted to the risk and congestion of the U-space in a certain area.

3.2.4 Traffic management services dealing with dynamic airspace management (EXE 4)

The aim of these experiments was to explore the best approaches to deal with dynamic changes, addressing the balance between minimizing the impact to the existing drone operations and maximising the capacity.

3.2.4.1 TMS-1

The validation objective is defined as: "Verify the applicability of dynamic separation criteria based on drone performances which adapt the size and shapes of the safety buffers around drones".

Success Criteria 1: The Traffic Management services managed an area of airspace with fixed dimensions using the dynamic separation criteria without having any loss of separation incident.

In this scenario, the traffic pattern previously established was loaded into the RMSTE.

Over 500 seconds, the load was increased to 1,000 concurrent flights of drones attempting to travel from origin to destination. Each path was guaranteed to intersect at least once with another vehicle.

The results are below.

In this test scenario, the RMSTE reported zero separation losses during the test period:

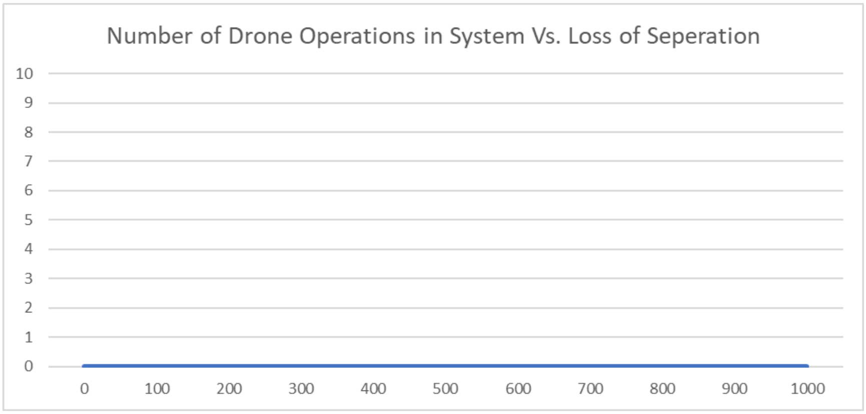


Figure 44: Graph shows no loss of separation as the number of drone operations increase

CONCLUSION: We have successfully demonstrated criteria "TMS-1 SC1 - The Traffic Management services managed an area of airspace with fixed dimensions using the dynamic separation criteria without having any loss of separation incident."

Success Criteria 2: The Traffic Management services managed an area of airspace with fixed dimensions using the dynamic separation criteria without causing any hindrance to the completion of any mission objective.

As the total vehicle traffic increased, the ability of each drone to continue unhindered to its destination was gradually reduced. In this simple evaluation in RMSTE, a reduction in efficiency below 50% would be considered the point in which the system would be considered "full" and it would be counterproductive (add too much of a hindrance) to allow more traffic even with the ability to ensure separation.

In repeated tests, mission efficiency *did not drop below 72% (averaging 78%),* yielding a maximum route deviation average of 22%*:*

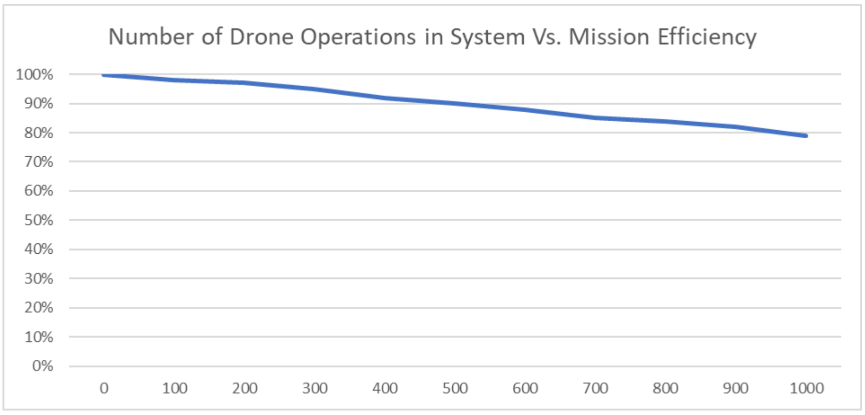


Figure 45: Graph shows average drone operations efficiency as number of drones managed increases

While these preliminary results are encouraging, and have served to successfully prove the concept, there is significant work that can be undertaken through further research focusing on optimisation.

CONCLUSION: We have successfully demonstrated "TMS-1 SC2 - The Traffic Management services managed an area of airspace with fixed dimensions using the dynamic separation criteria without causing any hindrance to the completion of any mission objective. "

Success Criteria 3: All flight plans approved by the Traffic management services coincide with airspace that the drone, Drone Operator and drone pilot are authorised to operate in.

For the purposes of the experiments, all drones were assumed to have been previously authorised to operate within the area under management by each node.

In practice, the node would not be responsible for determining whether to issue an authorisation: instead, this would be handled by another service, which was trusted by the node, and where the node can determine whether the trusted service has issued such an authorisation.

With no authorisation to fly, the drone was actually considered "uncooperative", and was not subject to management by the node. A future version of this simulation would entail the node dealing with the presence of a non-cooperative drone by notifying other nearby cooperative traffic of the incursion and how to avoid the conflict with the uncooperative drone.

CONCLUSION: We have successfully demonstrated "TMS-2 SC3 - All flight plans approved by the Traffic management services coincide with airspace that the drone, Drone Operator and drone pilot were authorised to operate in.

Success Criteria 4: The Traffic Management Services did not instruct any manoeuvre which the drone cannot safely perform.

Formally, as part of this set of experiments, we assumed a split of 75% rotor-based and 25% fixed-wing drones, accelerating gradually over a period of 15 seconds up to a maximum speed of 25m/s for fixed wings, and 20m/s for rotor-based. This would remain constant until the drone decelerated in the final 50m of its approach to its waypoint, where it would decrease linearly to 0m/s, signifying the end of its flight.

With this setup, we were able to successfully demonstrate completion of "TMS-1 SC4 - The Traffic Management Services did not instruct any manoeuvre which the drone cannot safely perform."

3.2.4.2 TMS-2

The validation objective is defined as: "Demonstrate performance-based rerouting of drones impacted by new airspace restrictions and other airspace users".

Success Criteria 1: The Traffic Management Service was able to correctly identify every drone flight to which new airspace restrictions apply.

As part of the experiment, airspace restrictions were added to the managed region during the simulated flights, by a temporary flight restriction.

Airspace restrictions of different sizes were added progressively, from 1km^2 to 10km^2 in the test region.

It was clear that drone operation that had not started and has a start location in the new restriction area were not able to start and therefore significantly hampered, as was the case for drone operations with a destination waypoint in the restricted area. For the latter, in reality the algorithm would be updated to restrict the start of the mission if it has not already begun, or return to start if already in progress if no other mission waypoints were included and the business need allowed.

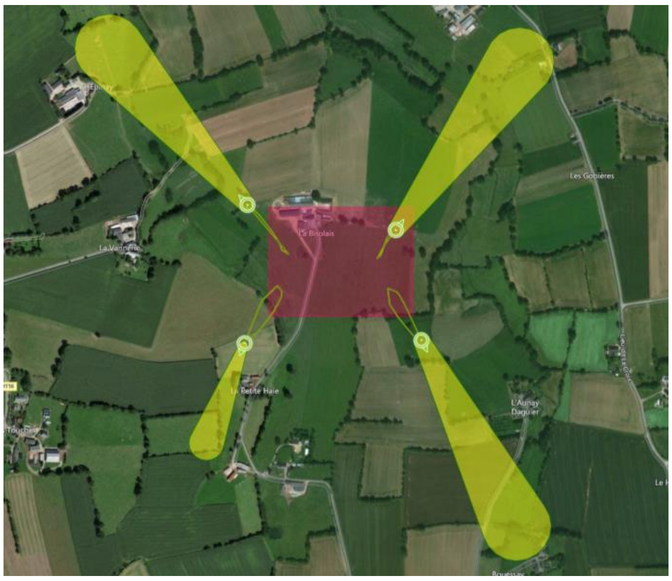


Figure 46: A new flight restriction and all airborne drones within this region exiting the restricted zone

The variation of the efficacy was broader than drone/drone deconfliction, obviously affecting those operations overlapping the restriction zone much more significantly than those without a conflict.

The RMSTE operated in a consistent way to other drones and environmental deconfliction constraints, and correctly assessed the separation requirements and any required re-routing. The deconflictor service was able to establish which drones were affected by the new temporary flight restriction and recalculated initially/tactically but ultimately in some cases strategically with the goal of re-planning their paths and the endpoint of the mission.

CONCLUSION: We have successfully demonstrated "TMS-2 SC1 - The Traffic Management Service is able to correctly identify every drone flight to which new airspace restrictions apply."

Success Criteria 2: The Traffic Management Service was able to correctly interpret the mission criteria of every drone flight to which new airspace restrictions apply.

The RMSTE was able to communicate changes in temporary airspace to participating vehicles, and, upon acceptance of a new flight into the system, understands the vehicles origin and destination. Due to the ingestion of this data, it was able to successfully interpret the mission criteria of every drone flight to determine whether the new airspace restriction affected the operation of the flight.

CONCLUSION: We have successfully demonstrated "TMS-2 SC2 -The Traffic Management Service was able to correctly interpret the mission criteria of every drone flight to which new airspace restrictions apply."

Success Criteria 3: The Traffic Management Service was able to reroute every drone flight to which new airspace restrictions apply without causing any hindrance to the completion of any mission objective, which can still be completed under the new airspace restrictions.

According to requirement TMS-2 SC2, the RMSTE is aware of a drone flight's origin and intended destination. It successfully used these data points as part of its conflict resolution algorithm when selecting a new path.

The basic success criteria of this experiment was to validate that, given a newly introduced airspace restriction, nothing in the RMSTE conflict management algorithm would interfere with the ability of a flight to reach its intended destination.

We acknowledge that in this limited experiment, no attention was paid to remaining flight capabilities/characteristics and that the placement and volume of the new temporary flight restriction would, for example, have a potentially significant impact on the ability of participating drones to be able to conduct their flight 'without hindrance'.

Further research would be required here.

CONCLUSION: We have successfully demonstrated "TMS-2 SC3 - The Traffic Management Service was able to reroute every drone flight to which new airspace restrictions apply without causing any hindrance to the completion of any mission objective which can still be completed under the new airspace restrictions."

Success Criteria 4: The Traffic Management Service was able to reroute every drone flight to which new airspace restrictions apply without adversely affecting their operational task/failure to carry out the task.

This experiment built on the output of TMS-2 SC3, but focused more on the ability of each vehicle to reach its destination following the introduction of a new airspace restriction.

For the purposes of these tests, we introduced various cylindrical airspace restrictions from 2km^2 to 10km^2, and studied how the simulation reacted. In 100% of the cases introduced, drones were able to arrive at their destination with a 0% rate of conflict.

Further study is required to optimise guidance given to conflicting drones such that additional attention is paid to their flight capabilities.

CONCLUSION: We have successfully demonstrated "TMS-2 SC4 - The Traffic Management Service is able to reroute every drone flight to which new airspace restrictions apply without adversely affecting their operational task/failure to carry out the task."

3.2.4.3 TMS-3

The validation objective is defined as: "Demonstrate that all data applicable to post-operational analysis was recorded".

Success Criteria 1: Demonstrate that all data applicable to post-operational analysis was recorded.

RMSTE has a full replay capability, which allows "time" to be sped up and slowed down for visual inspection if required via the eGUI. A full analysis of every object, timestamped, is available, as well as every instruction generated by RMSTE.

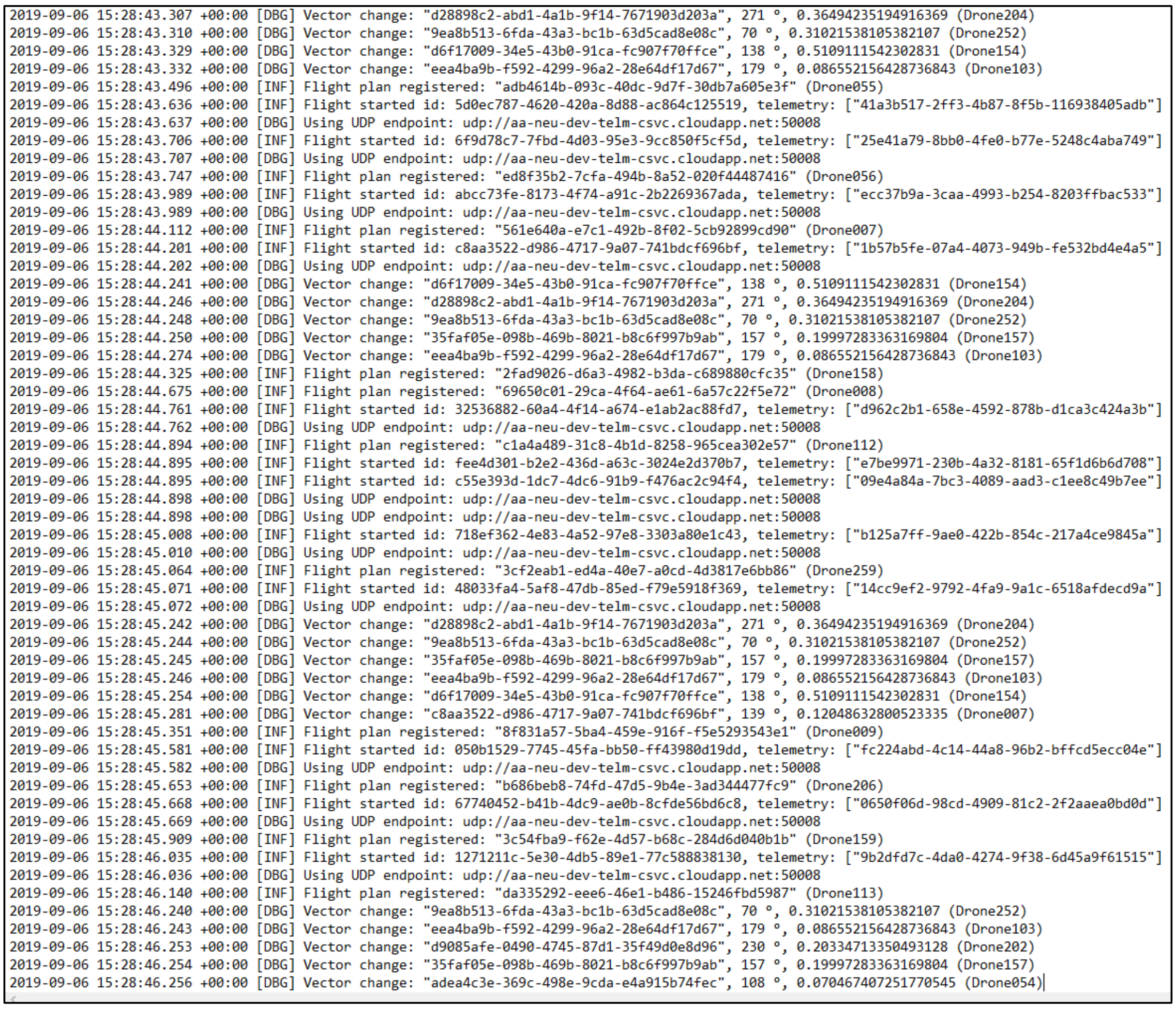
AUDIT

The Node, Node Manager, C2 Service and Traffic Management Service maintain full audit history for all state and instructions, keyed off the unique flight plan reference number.

This allows all data collected for a logical flight, regardless of the originating service, to contribute to a singular audit repository with a known flight reference for querying. We categorise audit data as either 'instrumentation' or 'action'.

Instrumentation events were emitted periodically from services to log the state of their health, performance characteristics, etc. 'Action' events were more specifically instructions generated by the service, or commands performed, that will have a specific effect on some other component. For example, a flight plan was stored as an 'action' when it was received.

Figure 47 is a sample from the trace log, which tracks every interaction with the deconflictor service:



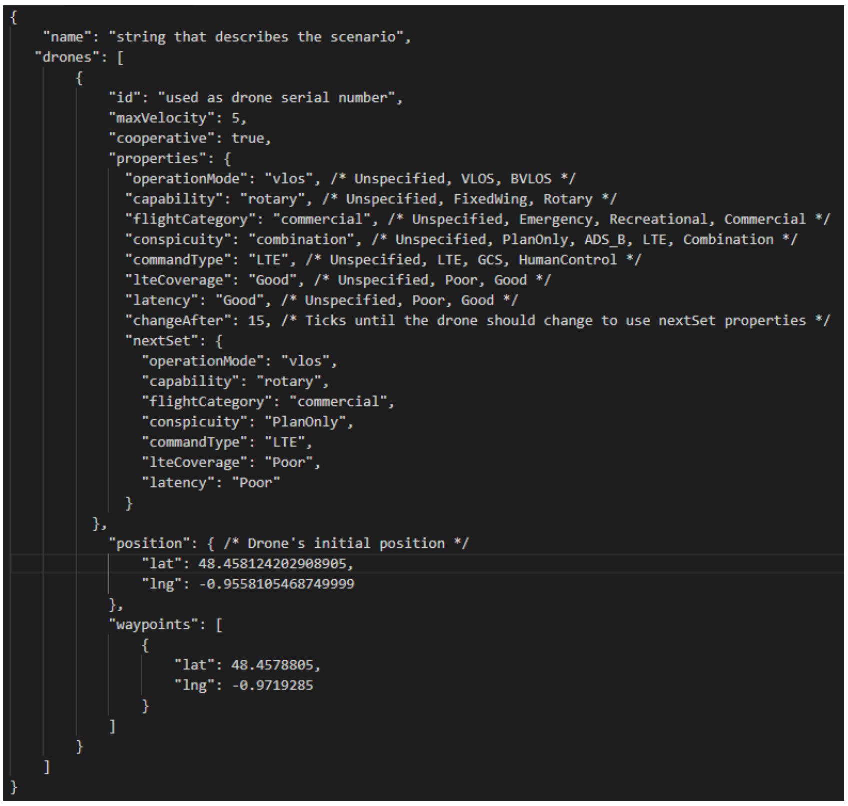
```
2019-09-06 15:28:43.307 +00:00 [DBG] Vector change: "d28898c2-abd1-4a1b-9f14-7671903d203a", 271 °, 0.36494235194916369 (Drone204)
2019-09-06 15:28:43.310 +00:00 [DBG] Vector change: "9ea8b513-6fda-43a3-bc1b-63d5cad8e08c", 70 °, 0.31021538105382107 (Drone252)
2019-09-06 15:28:43.329 +00:00 [DBG] Vector change: "d6f17009-34e5-43b0-91ca-fc907f70ffce", 138 °, 0.5109111542302831 (Drone154)
2019-09-06 15:28:43.332 +00:00 [DBG] Vector change: "eea4ba9b-f592-4299-96a2-28e64df17d67", 179 °, 0.086552156428736843 (Drone103)
2019-09-06 15:28:43.496 +00:00 [INF] Flight plan registered: "adb4614b-093c-40dc-9d7f-30db7a605e3f" (Drone055)
2019-09-06 15:28:43.636 +00:00 [INF] Flight started id: 5d0ec787-4620-420a-8d88-ac864c125519, telemetry: ["41a3b517-2ff3-4b87-8f5b-116938405adb"]
2019-09-06 15:28:43.637 +00:00 [DBG] Using UDP endpoint: udp://aa-neu-dev-telm-csvc.cloudapp.net:50008
2019-09-06 15:28:43.706 +00:00 [INF] Flight started id: 6f9d78c7-7fbd-4d03-95e3-9cc850f5cf5d, telemetry: ["25e41a79-8bb0-4fe0-b77e-5248c4aba749"]
2019-09-06 15:28:43.707 +00:00 [DBG] Using UDP endpoint: udp://aa-neu-dev-telm-csvc.cloudapp.net:50008
2019-09-06 15:28:43.747 +00:00 [INF] Flight plan registered: "ed8f35b2-7cfa-494b-8a52-020f44487416" (Drone056)
2019-09-06 15:28:43.989 +00:00 [INF] Flight started id: abcc73fe-8173-4f74-a91c-2b2269367ada, telemetry: ["ecc37b9a-3caa-4993-b254-8203ffbac533"]
2019-09-06 15:28:43.989 +00:00 [DBG] Using UDP endpoint: udp://aa-neu-dev-telm-csvc.cloudapp.net:50008
2019-09-06 15:28:44.112 +00:00 [INF] Flight plan registered: "561e640a-e7c1-492b-8f02-5cb92899cd90" (Drone007)
2019-09-06 15:28:44.201 +00:00 [INF] Flight started id: c8aa3522-d986-4717-9a07-741bdcf696bf, telemetry: ["1b57b5fe-07a4-4073-949b-fe532bd4e4a5"]
2019-09-06 15:28:44.202 +00:00 [DBG] Using UDP endpoint: udp://aa-neu-dev-telm-csvc.cloudapp.net:50008
2019-09-06 15:28:44.241 +00:00 [DBG] Vector change: "d6f17009-34e5-43b0-91ca-fc907f70ffce", 138 °, 0.5109111542302831 (Drone154)
2019-09-06 15:28:44.246 +00:00 [DBG] Vector change: "d28898c2-abd1-4a1b-9f14-7671903d203a", 271 °, 0.36494235194916369 (Drone204)
2019-09-06 15:28:44.248 +00:00 [DBG] Vector change: "9ea8b513-6fda-43a3-bc1b-63d5cad8e08c", 70 °, 0.31021538105382107 (Drone252)
2019-09-06 15:28:44.250 +00:00 [DBG] Vector change: "35faf05e-098b-469b-8021-b8c6f997b9ab", 157 °, 0.19997283363169804 (Drone157)
2019-09-06 15:28:44.274 +00:00 [DBG] Vector change: "eea4ba9b-f592-4299-96a2-28e64df17d67", 179 °, 0.086552156428736843 (Drone103)
2019-09-06 15:28:44.325 +00:00 [INF] Flight plan registered: "2fad9026-d6a3-4982-b3da-c689880cfc35" (Drone158)
2019-09-06 15:28:44.675 +00:00 [INF] Flight plan registered: "69650c01-29ca-4f64-ae61-6a57c22f5e72" (Drone008)
2019-09-06 15:28:44.761 +00:00 [INF] Flight started id: 32536882-60a4-4f14-a674-e1ab2ac88fd7, telemetry: ["d962c2b1-658e-4592-878b-d1ca3c424a3b"]
2019-09-06 15:28:44.762 +00:00 [DBG] Using UDP endpoint: udp://aa-neu-dev-telm-csvc.cloudapp.net:50008
2019-09-06 15:28:44.894 +00:00 [INF] Flight plan registered: "c1a4a489-31c8-4b1d-8258-965cea302e57" (Drone112)
2019-09-06 15:28:44.895 +00:00 [INF] Flight started id: fee4d301-b2e2-436d-a63c-3024e2d370b7, telemetry: ["e7be9971-230b-4a32-8181-65f1d6b6d708"]
2019-09-06 15:28:44.895 +00:00 [INF] Flight started id: c55e393d-1dc7-4dc6-91b9-f476ac2c94f4, telemetry: ["09e4a84a-7bc3-4089-aad3-c1ee8c49b7ee"]
2019-09-06 15:28:44.898 +00:00 [DBG] Using UDP endpoint: udp://aa-neu-dev-telm-csvc.cloudapp.net:50008
2019-09-06 15:28:44.898 +00:00 [DBG] Using UDP endpoint: udp://aa-neu-dev-telm-csvc.cloudapp.net:50008
2019-09-06 15:28:45.008 +00:00 [INF] Flight started id: 718ef362-4e83-4a52-97e8-3303a80e1c43, telemetry: ["b125a7ff-9ae0-422b-854c-217a4ce9845a"]
2019-09-06 15:28:45.010 +00:00 [DBG] Using UDP endpoint: udp://aa-neu-dev-telm-csvc.cloudapp.net:50008
2019-09-06 15:28:45.064 +00:00 [INF] Flight plan registered: "3cf2eab1-ed4a-40e7-a0cd-4d3817e6bb86" (Drone259)
2019-09-06 15:28:45.071 +00:00 [INF] Flight started id: 48033fa4-5af8-47db-85ed-f79e5918f369, telemetry: ["14cc9ef2-9792-4fa9-9a1c-6518afdecd9a"]
2019-09-06 15:28:45.072 +00:00 [DBG] Using UDP endpoint: udp://aa-neu-dev-telm-csvc.cloudapp.net:50008
2019-09-06 15:28:45.242 +00:00 [DBG] Vector change: "d28898c2-abd1-4a1b-9f14-7671903d203a", 271 °, 0.36494235194916369 (Drone204)
2019-09-06 15:28:45.244 +00:00 [DBG] Vector change: "9ea8b513-6fda-43a3-bc1b-63d5cad8e08c", 70 °, 0.31021538105382107 (Drone252)
2019-09-06 15:28:45.245 +00:00 [DBG] Vector change: "35faf05e-098b-469b-8021-b8c6f997b9ab", 157 °, 0.19997283363169804 (Drone157)
2019-09-06 15:28:45.246 +00:00 [DBG] Vector change: "eea4ba9b-f592-4299-96a2-28e64df17d67", 179 °, 0.086552156428736843 (Drone103)
2019-09-06 15:28:45.254 +00:00 [DBG] Vector change: "d6f17009-34e5-43b0-91ca-fc907f70ffce", 138 °, 0.5109111542302831 (Drone154)
2019-09-06 15:28:45.281 +00:00 [DBG] Vector change: "c8aa3522-d986-4717-9a07-741bdcf696bf", 139 °, 0.12048632800523335 (Drone007)
2019-09-06 15:28:45.351 +00:00 [INF] Flight plan registered: "8f831a57-5ba4-459e-916f-f5e5293543e1" (Drone009)
2019-09-06 15:28:45.581 +00:00 [INF] Flight started id: 050b1529-7745-45fa-bb50-ff43980d19dd, telemetry: ["fc224abd-4c14-44a8-96b2-bffcd5ecc04e"]
2019-09-06 15:28:45.582 +00:00 [DBG] Using UDP endpoint: udp://aa-neu-dev-telm-csvc.cloudapp.net:50008
2019-09-06 15:28:45.653 +00:00 [INF] Flight plan registered: "b686beb8-74fd-47d5-9b4e-3ad344477fc9" (Drone206)
2019-09-06 15:28:45.668 +00:00 [INF] Flight started id: 67740452-b41b-4dc9-ae0b-8cfde56bd6c8, telemetry: ["0650f06d-98cd-4909-81c2-2f2aaea0bd0d"]
2019-09-06 15:28:45.669 +00:00 [DBG] Using UDP endpoint: udp://aa-neu-dev-telm-csvc.cloudapp.net:50008
2019-09-06 15:28:45.909 +00:00 [INF] Flight plan registered: "3c54fba9-f62e-4d57-b68c-284d6d040b1b" (Drone159)
2019-09-06 15:28:46.035 +00:00 [INF] Flight started id: 1271211c-5e30-4db5-89e1-77c588838130, telemetry: ["9b2dfd7c-4da0-4274-9f38-6d45a9f61515"]
2019-09-06 15:28:46.036 +00:00 [DBG] Using UDP endpoint: udp://aa-neu-dev-telm-csvc.cloudapp.net:50008
2019-09-06 15:28:46.140 +00:00 [INF] Flight plan registered: "da335292-eee6-46e1-b486-15246fbd5987" (Drone113)
2019-09-06 15:28:46.240 +00:00 [DBG] Vector change: "9ea8b513-6fda-43a3-bc1b-63d5cad8e08c", 70 °, 0.31021538105382107 (Drone252)
2019-09-06 15:28:46.243 +00:00 [DBG] Vector change: "eea4ba9b-f592-4299-96a2-28e64df17d67", 179 °, 0.086552156428736843 (Drone103)
2019-09-06 15:28:46.253 +00:00 [DBG] Vector change: "d9085afe-0490-4745-87d1-35f49d0e8d96", 230 °, 0.20334713350493128 (Drone202)
2019-09-06 15:28:46.254 +00:00 [DBG] Vector change: "35faf05e-098b-469b-8021-b8c6f997b9ab", 157 °, 0.19997283363169804 (Drone157)
2019-09-06 15:28:46.256 +00:00 [DBG] Vector change: "adea4c3e-369c-498e-9cda-e4a915b74fec", 108 °, 0.070467407251770545 (Drone054)
```

Figure 47: Sample of Audit Log

CONCLUSION: We have successfully demonstrated TMS-3 SC1: All traffic management events were re-playable in post-ops.

Success Criteria 2: Demonstrate that all filed flight plan information was accessible.

All flight plans alongside with information relating to the decision making was accessible.



```
{
    "name": "string that describes the scenario",
   "drones": [
        {
            "id": "used as drone serial number",
            "maxVelocity": 5,
            "cooperative": true,
            "properties": {
              "operationMode": "vlos", /* Unspecified, VLOS, BVLOS */
              "capability": "rotary", /* Unspecified, FixedWing, Rotary */
              "flightCategory": "commercial", /* Unspecified, Emergency, Recreational, Commercial */
              "conspicuity": "combination", /* Unspecified, PlanOnly, ADS_B, LTE, Combination */
              "commandType": "LTE", /* Unspecified, LTE, GCS, HumanControl */
              "lteCoverage": "Good", /* Unspecified, Poor, Good */
              "latency": "Good", /* Unspecified, Poor, Good */
              "changeAfter": 15, /* Ticks until the drone should change to use nextSet properties */
              "nextSet": {
                "operationMode": "vlos",
                "capability": "rotary",
                "flightCategory": "commercial",
                "conspicuity": "PlanOnly",
                "commandType": "LTE",
                "lteCoverage": "Poor",
                "latency": "Poor"
              }
          },
            "position": { /* Drone's initial position */
                "lat": 48.458124202908905,
                "lng": -0.9558105468749999
            },
            "waypoints": [
                {
                    "lat": 48.4578805,
                    "lng": -0.9719285
                }
            ]
        }
    ]
}
```

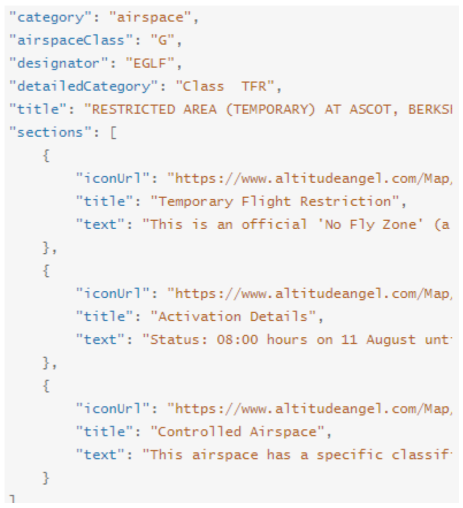
Figure 48: Example Flight Plan

CONCLUSION: We have successfully demonstrated TMS-3 SC2 - All filed flight plan information was accessible.

Success Criteria 3: All third-party information related to the filing of a flight plan is accessible.

When the conflict resolutions service builds a situation representation and undertakes a transformation, data from external services used to make the decision is stored in the audit log.

Below is an extract of input from an external data service that was considered during the filing of a flight plan.



```
"category": "airspace",
"airspaceClass": "G",
"designator": "EGLF",
"detailedCategory": "Class  TFR",
"title": "RESTRICTED AREA (TEMPORARY) AT ASCOT, BERKSI
"sections": [
    {
        "iconUrl": "https://www.altitudeangel.com/Map,
        "title": "Temporary Flight Restriction",
        "text": "This is an official 'No Fly Zone' (a
    },
    {
        "iconUrl": "https://www.altitudeangel.com/Map,
        "title": "Activation Details",
        "text": "Status: 08:00 hours on 11 August unt
    },
    {
        "iconUrl": "https://www.altitudeangel.com/Map,
        "title": "Controlled Airspace",
        "text": "This airspace has a specific classif
    }
]
```

Figure 49: Temporary Flight Restriction Data Example

CONCLUSION: We have successfully demonstrated TMS-3 SC3 - All third-party information related to the filing of a flight plan is accessible.

3.3 Detailed analysis of results per research area of the microservice-based architecture

3.3.1 Safety and Failure Modes

Regardless of the utilized implementation technology, it can be stated that, under certain conditions, all the applications will fail. When looking into a microservice implementation, it is paramount to ensure an architecture that is capable to absorb failures and that incorporates countermeasures able to react in real-time. By means of resilience tests and the application of failure mode mechanisms in specific critical services, it is demonstrated in this section how an environment of microservices can be robustly implemented.

Resilience tests for the Weather Service (EXE 1)

In order to tests the benefits of a micro-service approach and data flow the functionality of the micro-weather service has been packaged with different approaches. These approaches pack the functionality differently, from a single service instance offering all interfaces to clients to a fully compartmentalized one, running a dynamic amount of Docker containers:

1. Fully monolithic: Single service running all clients' interactions plus a single broker for clients;
2. Separation in two services: Gateway for service configuration and Single client agent for managing all clients (threaded);
3. Fully micro: Gateway for service configuration and an independent container with a single agent for one client;
4. Fully micro plus exclusive broker per client: Same as before but adding a separate light AMQP broker to every agent container instead of an external broker.

For testing the resilience of the different architectures, a synthetic test scenario has been designed.

In the test, 100 processes will simulate clients asking for weather data within a certain area concurrently.

They follow the essential steps described above (request to gateway, request to agent, and consume from their queue). A client that is able to read for 5 minutes and requests to its agent at the end of the service is considered a success.

Every thread is continuously executing petitions so that there are always 100 concurrent clients in total. That means, right after finishing one simulation of a client (or failing) it is starting a new one.

The computational core of the weather service is simulated by a series of dummy containers that fill their corresponding queues with synthetic data that simulates the prediction data at a rate of 10 packages per minute of 500KB each.

Different runs of this setup varying the percentage of faulty clients. Every run lasts 20 minutes and takes place in a server running Ubuntu18.04 with 28 Intel Xeon cores (HT activated) and 256 GB of RAM. All docker images are also build over the basic Ubuntu 18.04 image.

The four different approaches where tested at 0%, 10%, 25%, 50% and 90% of faulty clients.

As there is the infinite possible error of clients' implementation, we have implemented three simple kinds that will happen evenly: 1st execution will be halted at a random point during its execution by killing the process, 2nd one sends a malformed REST requests, 3rd makes malformed petitions to consume data from queue.

The next graph summarizes all experimental results of this test with the four different approaches:

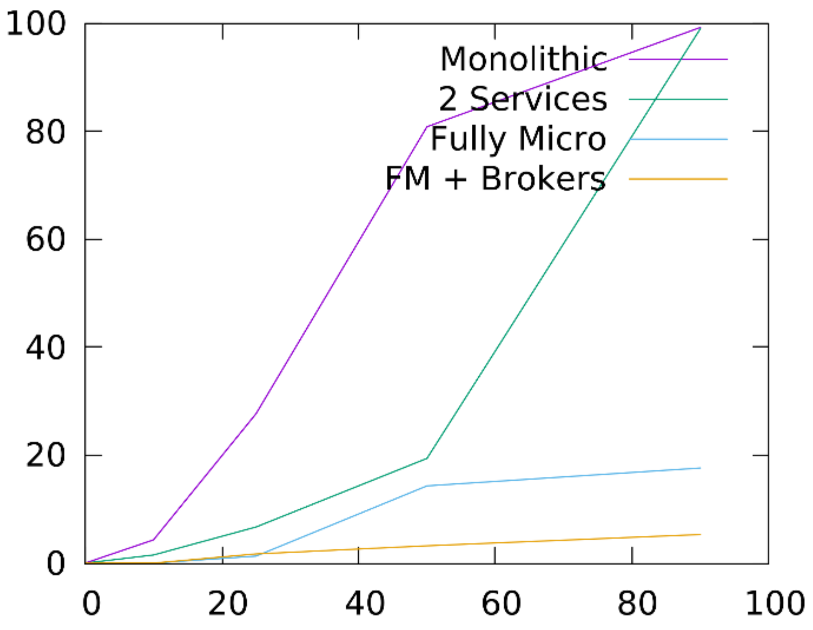


Figure 50: Load handling of implemented architecture

The X-axis represents the percentage of simulated clients with faulty behaviour and the Y-axis is the percentage of unsuccessful clients from the ones that were canon.

It can be observed that the four competing architectures were fully capable to handle the load in the absence of errant clients.

In the fully monolithic case the number of errors rises very quickly with the percentage of errant clients as once a non-managed exception rises the service is aborted and has to restart, terminating all present clients. The restart also causes a 30 to 35 seconds downtime during which petitions are lost.

The second case shows a bit more resilience as the two independent blocks are reset independently and present clients and incoming petitions are not always lost. In any case, one errors are frequent enough almost no client is successful as it can be seen at the 90% errant clients' data point.

The third architecture in this synthetic test is the one that best resembles the local weather service architecture, having the gateway instance creating a process for running an independent docker container for every incoming client. It can be seen that the number of errors stays low until 50 percent of errant clients at which point nearly 16% of good client failed.

After some investigation, we discovered that most of the errors were due to the huge number of client queues that were not being properly clean in the client's broker. That is why; during the execution of these tests we proposed the 4th case: adding a lightweight data broker per client instead of a shared one. This increases the use of RAM by less than 60MB per agent due to the memory allocations for the reserves for the queues of the particular broker and requires a dynamic assignment of sockets to the different brokers but eliminates a major cause of inter-client error propagation as it can be seen in the orange line showing the data for this case.

The remaining errors in the 4th scenario (only 5,3% at 90% of errant clients) are in their majority caused by errors in the gateway. In our opinion, in order to further improve results replication of the gateway should be implemented. This will be taken into account for future iterations of the service.

Of the success criteria defined, for the proposed architecture (case 3) here we reproduce a corresponding table to the one in section § 2.3.2.1:

- **Success Criteria 1**: Ideal scenario should have over 99% of clients correctly served in proposed architecture;
- **Success Criteria 2**: Normal scenario should have over 95% of clients correctly served in proposed architecture;
- **Success Criteria 3**: High scenario should have over 95% of clients correctly served in proposed architecture;
- **Success Criteria 4**: Very High scenario should have over 90% of clients correctly served in proposed architecture;
- **Success Criteria 5**: Extreme scenario should have over 90% of clients served in proposed architecture;
- **Success Criteria 6**: In all scenarios, the proposed architecture should be more resilient then the monolithic approach.

RESULT	SC1	SC2	SC3	SC4	SC5	SC6
Success. 100% of `well behaved´ clients were served correctly	x					
Success. 100% of `well behaved´ clients were served correctly		x				
Success. 98.7% of `well behaved´ clients were served correctly			x			
Failed. 85.7% of `well behaved´ clients were served correctly				x		
Failed. 82.4% of `well behaved´ clients were served correctly					x	
Success. The micro services approach improved all scenarios tested						x

Table 36: Summarized results of experiment related to resilience

While most criteria were surpassed, it can be seen that the last two criteria for success were not met with our proposed solution. This was our initial motivation for testing the fourth architecture, which also includes a specific message broker per client agent container. This forth case met all success criteria having a 96.8% success in the very high case and 94.7% in the extreme one.

We understand that the value of this test is limited as it does not capture the full range of possible errors, but we consider the results as a contributing evidence that- a micro-service architecture approach enables the creation of more robust architectures.

It is also true that the code of the monolithic approach could be hardened to have a better management of exceptions. Doing this to cover all cases would be difficult, complicating a lot the internal code flow and it would be prone to memory leaks or even security bugs due to the increased complexity.

We determined the fact that a higher degree of robustness achieved by design is a clear advantage of the micro-service approach.

Service availability in congested scenarios for the Traffic Information Services (EXE 3)

The validation objectives that will be assessed in this section have already been described with more detail in section 2.3.2.1. In general terms, the final objective of these exercises is to demonstrate the feasibility of a microservice architecture in terms of robustness, addressing the possible and unavoidable failures inherent in every architecture and showcasing the different mechanisms to mitigate harmful effects in the normal operation of U-space.

We must assume that, eventually, the system will fail but maybe we do not even realize. In D3.1 it was stated that a microservice architecture is defined as a failure-resistant structure. That means that the environment (comprising the configuration of the architecture and the different mechanisms provided by frameworks to manage the execution of the whole system) around these microservices has a suite of Core functions that are continuously monitoring and reporting the status of the architecture (D3.1, § 6.2). In addition to this, as an active/reactive system, it is also capable of recovering from these issues, clearing and closing the microservices in error mode and creating new instances with the same functionality with quick response times. These premises will be the main objectives of these validations: obtain conclusive evidence about the reaction of the platform to its own failure.

This is a very relevant topic in an environment in which huge and sudden increases/decreases in demand may emerge (so the platform should be capable of growing) and critical services may be continuously available (e.g. Traffic Information Services in congested scenarios).

Along the proposed exercises, it is possible to identify two common objectives:

<u>Additional delays/Recovery time due to a failure</u>

The chain reaction in the failure of a microservice comprises the loss of communication until another instance (or instances) takes charge of its work after balancing the load or, if needed, another microservice is instanced and created. Whatever the case, this implies that there will be a time frame in which this information will not be provided to the system and users, so one of the key objectives of the architecture is to minimize this number.

In this exercise, the validation objective is described as "Service availability due to failure mode implementation" and a list of success criteria have been defined to measure these times throughout the tests defined.

Success Criteria 1: In critical services, the delay of information due to failures in the execution of a service shall be less than 2s.

Success Criteria 2: A platform focused on information management for U-space shall ensure that, in case of a service failure, a backup microservice for critical services shall be deployed in less than 1 second.

Success Criteria 3: A platform focused on information management for U-space shall ensure that the failures in services must be detected within 0.1s since the issue happened.

MTI-Test 6 of this exercise is focused on this specific research area and can provide valuable outputs to assess these criteria.

METRIC	VALUE	SC1	SC2	SC3
Delay of information due to service failures (seconds).	26.49s (3.15)			
Recovery time to execute another instance of the service (seconds).	193s			
Time to detect the failure of the service (seconds).	0.001s			X

Table 37: Summarized results of experiment related to delays and recovery time

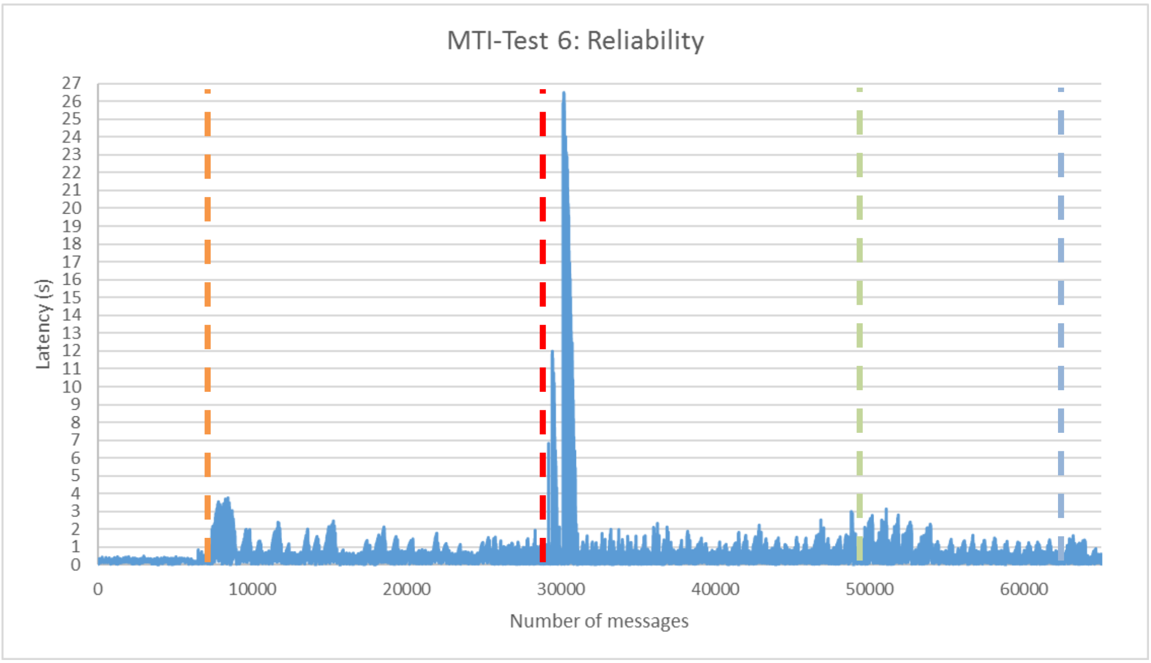


Figure 51: Latency of messages processed in experiment MTI-6

From these results, the first impression is that the microservice architecture does not fit the requirements for U-space environments but the previous chart shall be analysed in detail. In the end, it will rely on its main configuration and the availability of computing resources.

The microservice architecture has initially been executed with one microservice per instance. That means that, if one of them enters in failure mode, the delays and losses of information will be inevitable until another instance is created and stabilised. That is what happens when the MAVLink microservice is destroyed (orange line) but the issue becomes even more important when the Alarm MS is removed (red line), that provides the greater delay. In case of duplicated services (red and blue lines), the delay is not so important (just a few seconds, in brackets) because the excess workload can be absorbed by the other instance and, in this case, it is not necessary to deploy another microservice, so the conclusion is that there will be a duplicated microservice to absorb the effects of failures in microservices. Alternatively, even a standby microservice (initialized) that can take control of the demand.

On the other hand, microservices are launched almost instantaneously, the problem relies on their initialization, that may take a few minutes, so the solution will also be to implement duplicated microservices by default for crucial services in U-space. On the other hand, and as it can be seen, it takes more than 3 minutes to fully initialize one of these microservices but the maximum delay is 26 s. The reason of this behaviour is the implemented countermeasures: initialization is understood as the capacity of a microservice to fully perform its objective, but once a microservice is removed from the system, a set of microservices, not fully initialized, appear which are trying to absorb the demand and are waiting for the substitute to be fully operative. For this reason, the delays can be further reduced.

Mechanisms against faults or error.

The common reaction to service failure will be the creation of new instances of the microservices, trying to absorb the workload. If the processes are atomic enough, only one microservice will be created (due to fast initialization and stabilization). If not, a set of microservices will be developed, trying to distribute the excess of workload in less demanding modules and, after stabilization, they will be “destroyed” to the minimum set of resources necessary to conduct the normal operation of the platform.

In the same validation objective of exercise 3, several success criteria have been defined to measure this behaviour.

Success Criteria 4: A platform focused on information management for U-space shall be capable of dealing with increases in demand creating, at least, an additional instance of the same functionality to load the balance and prevent its overload.

Success Criteria 5: The maximum workload of a service shall be less than 80% of its capacity, providing that a new instance will be created when overloads above this percentage are detected.

METRIC	VALUE	SC4	SC5
A platform focused on information management for U-space shall be capable of dealing with increases in demand creating, at least, an additional instance of the same functionality to load the balance and prevent its overload (IC).	3-5	X	
The maximum workload of a service shall be less than 80% of its capacity, providing that a new instance will be created when overloads above this percentage are detected.	165% (42%)		X

Table 38: Summarized results of experiment related to mechanisms against faults

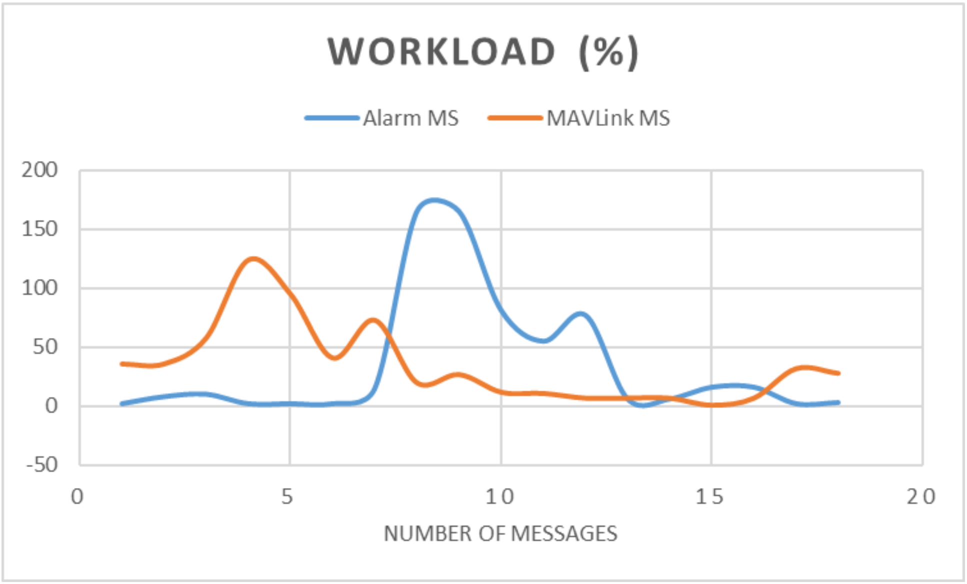


Figure 52: Workload of different microservices in relation the number of processed messages

As expected, the microservice architecture has created microservice under demand, as a response of the failure, trying to absorb not only the workload of the faulty MS, but also the initialization times of these additional microservices.

Related to the workload, in case of 1 microservice per service, the demand has forced it to work over 100% (it is was expected that, to absorb these increases in demands, the microservice shall work at 165% of its capacity (due to the coincidence within the failure and a rise in the number of messages), which is not possible and could even be catastrophic for it) which could cause a catastrophic failure (the system is not capable of recovering) but, in case of duplicated services, the greater value has only reached 42%, which reinforces the idea of the necessity of duplicated services.

Failure mode mechanisms in the Traffic Management Service dealing with dynamic airspace management (EXE 4)

In case of this exercise the validation objective is defined as "To test a failure in any services that we were dependent on, as well as how failures were handled in components of the deconfliction service."

As an important U-Space component, the Tactical service has been designed with automatic monitoring by an independent set of software components (on independent cloud infrastructure). The separation from production systems and monitoring systems on to separate physical infrastructure provides for a more robust monitoring and alerting solution.

In this experimental system, we were primarily focused on testing for the following failure conditions:

- Failure of a node (or, the failure of an individual subsystem upon which a node is reliant);
- Failure of a cluster;
- Failure of a datacentre;
- Failure of a dependent data source (such as weather data);
- Failure of the C2 service.

An element of 'self-healing' is built into the RMSTE (Region Management Simulation Test Environment) via the Node Manager, which monitors the physical health of each of the dependent nodes in the cluster and automatically de-provisions and provisions new node members into a cluster.

In these experiments, we successfully demonstrated the removal of 25% of the nodes in an RMSTE cluster while the cluster was under 70% total load, without any loss of service or capability to any connected simulated vehicles.

To simulate this, we manually took a node offline while the system was running normal simulations. The Node Manager detected this failure of one of the nodes, and provisioned a new node on average in 47 seconds, over three successive scenarios.

Because Node Manager recovered and the health of the cluster was not impacted, no forward or onward alerting to consumers was made.

<u>Cluster Failure</u>

In a total cluster failure, Node Manager first checked whether any other clusters were present in the same data centre. If none were present, the Node Manager calls out to a Node Manager instance running in a different data centre and requests instantiation of a new cluster equivalent to the cluster that has just gone offline.

Cluster failures are harder to recover from if only one cluster exists in a single datacentre**.** In this scenario, under repeated testing, if no other local-datacentre clusters were available, time-to-failover was between 1-2 minutes. However, the impact to connected simulated vehicles was high – with excessive latency and timeouts – throughout the failover duration. This is because, for these experiments, we did not provision a global elastic load balancer, which is a key take-away as a requirement for more seamless failover.

Failure of a dependent data source

To solve for this failure mode, we have a categorisation system of data providers. For example, weather data is regarded as a lower priority to connections to ADS-B data feeds, or the NOTAM system. Each provider in the system is listed with an alternative data source and it is possible for RMSTE to "load balance" between data providers (*round robin*), or to operate a primary/secondary failover (straight swap). Through the Discovery Service, it is even possible for RMSTE to 'call-out' to see if any alternatives exist at regular intervals during a degraded performance scenario, and switch to them automatically.

CONCLUSION: We have successfully demonstrated "We were able to switch provider if a service we were dependent upon fails or increase separation to the maximum level while it recovers. Both ensuing that separation is maintained.

Failure of all data providers in a priority 1 group (the group of data providers attributed as critical for any operations with the system) would cause RMSTE to flag a cluster as "unavailable to new traffic" due to the fact it does not have the situation awareness it needs in order to be able to provide the tactical service. In this mode, RMSTE stops accepting any *new* vehicles into the cluster by rejecting their take-off request, sending a "temporarily unavailable" error code.

Vehicles that were currently connected were, for this experiment, asked to "Return to Base".

While vehicles were executing the Return to Base, RMSTE operates in 'reduced capability mode' by sending the appropriate data message to connected vehicles and then performing de-confliction only against cooperative vehicles within the cluster.

Failure of the service

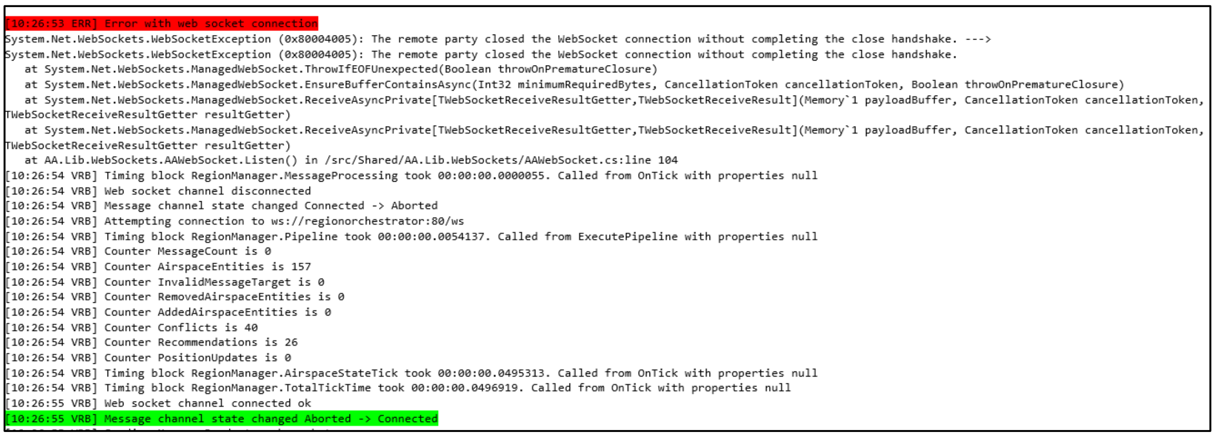
RMSTE manages conflicts by issuing 'instructions' via the C2 interface.

Should the C2 interface service suffer a failure, or degradation, the overall performance of the RMSTE cluster can be considered 'degraded' or 'offline'.

Note: The C2 service is capable of being geographically disperse, and can actually span multiple data centres more easily than the nodes themselves. This lends itself to a more robust deployment paradigm than other services, so the likelihood of a total failure to be able to deliver any C2 message to connected vehicles is greatly reduced by a factor roughly equivalent to the SLA available within a single data centre, multiplied by the total number of data centres the C2 service is deployed within.

Depending on the scenario, the Node Manager will execute the appropriate action based on the following criteria:

- If the C2 service is completely offline, the Node Manager will trigger the “cluster offline” and try to auto-heal the C2 service. During this time, incoming connections to the service were met with a 503 – Service Unavailable, and connected vehicles should execute a return to base;
- If the C2 service is suffering performance degradation, Node Manager can dynamically vary the size and shape of the management zones around each tracked vehicle to cater for an additional round-trip latency.



```
[10:26:53 ERR] Error with web socket connection
System.Net.WebSockets.WebSocketException (0x80004005): The remote party closed the WebSocket connection without completing the close handshake. --->
System.Net.WebSockets.WebSocketException (0x80004005): The remote party closed the WebSocket connection without completing the close handshake.
   at System.Net.WebSockets.ManagedWebSocket.ThrowIfEOFUnexpected(Boolean throwOnPrematureClosure)
   at System.Net.WebSockets.ManagedWebSocket.EnsureBufferContainsAsync(Int32 minimumRequiredBytes, CancellationToken cancellationToken, Boolean throwOnPrematureClosure)
   at System.Net.WebSockets.ManagedWebSocket.ReceiveAsyncPrivate[TWebSocketReceiveResultGetter,TWebSocketReceiveResult](Memory`1 payloadBuffer, CancellationToken cancellationToken,
TWebSocketReceiveResultGetter resultGetter)
   at System.Net.WebSockets.ManagedWebSocket.ReceiveAsyncPrivate[TWebSocketReceiveResultGetter,TWebSocketReceiveResult](Memory`1 payloadBuffer, CancellationToken cancellationToken,
TWebSocketReceiveResultGetter resultGetter)
   at AA.Lib.WebSockets.AAWebSocket.Listen() in /src/Shared/AA.Lib.WebSockets/AAWebSocket.cs:line 104
[10:26:54 VRB] Timing block RegionManager.MessageProcessing took 00:00:00.0000055. Called from OnTick with properties null
[10:26:54 VRB] Web socket channel disconnected
[10:26:54 VRB] Message channel state changed Connected -> Aborted
[10:26:54 VRB] Attempting connection to ws://regionorchestrator:80/ws
[10:26:54 VRB] Timing block RegionManager.Pipeline took 00:00:00.0054137. Called from ExecutePipeline with properties null
[10:26:54 VRB] Counter MessageCount is 0
[10:26:54 VRB] Counter AirspaceEntities is 157
[10:26:54 VRB] Counter InvalidMessageTarget is 0
[10:26:54 VRB] Counter RemovedAirspaceEntities is 0
[10:26:54 VRB] Counter AddedAirspaceEntities is 0
[10:26:54 VRB] Counter Conflicts is 40
[10:26:54 VRB] Counter Recommendations is 26
[10:26:54 VRB] Counter PositionUpdates is 0
[10:26:54 VRB] Timing block RegionManager.AirspaceStateTick took 00:00:00.0495313. Called from OnTick with properties null
[10:26:54 VRB] Timing block RegionManager.TotalTickTime took 00:00:00.0496919. Called from OnTick with properties null
[10:26:55 VRB] Web socket channel connected ok
[10:26:55 VRB] Message channel state changed Aborted -> Connected
```

Figure 53: Console output demonstrating recovery time

CONCLUSION: We have successfully demonstrated “We were able to recover from an internal service failure in line in 10 seconds.” In fact, it only took 2 seconds

3.3.2 Data Management

The validation objectives have been described in § 2.3.2.2 in detail. In a general form all the defined validation objectives aim at assessing the consistency and reliability of the different data exchange mechanisms implemented as part of the developed services and capabilities.

As every set of microservices processes data in closed environments and utilizes multiple mechanisms for external communication, a frictionless data flow is not automatically ensured. Monitoring the implemented mechanisms through the initial simulations has indicated critical locations where the data consistency and integrity could be compromised. Applying reactive measurements and adjustments to the initially implemented microservices (e.g. utilization of backup mechanisms, implementation of synchronous/ asynchronous communication simultaneously) has made it possible to address the initial objectives adequately.

Adequate processing of user requests and data consistency in the Flight Planning Management Service (EXE 2)

<u>DM-1: Robust flight plan data exchange</u>

In the case of the Flight Planning Management Service, the main challenge of the microservice implementation is to cope with the requirements for the service in regard to the communication mechanisms necessary for processing the flight plan requests and providing a timely notification to the MPM service.

Besides the formal check-up of submitted flight plan requests and general allocation of these in a centralized database, the FPM service prototype includes a conflict detection capability for validating flight trajectories against external trajectories in the system. If the microservice implementation is able to process all the initial requests, manage updates and provide notifications, this implementation could be contemplated as an initial reference implementation framework for integrating further capabilities that are related to the flight planning management, such as more enhanced conflict detection capabilities or a procedural interface with ATC.

For the assessment of data exchange, the flight plan request processes from the experiment simulations in § 3.2.2.1 have been analysed. These are initial flight plan submissions, so as reiterating update request derived from the conflict detection of several flight trajectories with other trajectories. The success criteria metric (FPM) represents how successful is in general the FPM service in processing the data (management, allocation, notification), which is highly depending on the outputs of the conflict manager microservice. Furthermore, this success rate metric applies only to the internal data exchange within the FPM service prototype and its interaction with the MPM service.

Success Criteria 1: 99% success rate in the data exchange between the MPM and FPM services.

Success Criteria 2: The message queue delivers over 99% of transmitted requests for all the scenarios considered.

METRIC	VALUE	SC1	SC2
Total request processes simulated (RP) – (number)	302	x	x
Average response time of FPM Service to flight plan submissions (RT) – (sec)	0.08	x	
Average update time of FPM Service to flight plan modifications	0.05	x	
Maximum response time of FPM Service to flight plan requests (MT) – (sec)	0.13	x	
Success rate in the requests to the FPM service (FPM) - (% requests)	100	x	
Success rate in the message queue requests (MQ) - (% requests)	100		x

Table 39: Summarized results of experiment related to the processing of user requests

From all the 302 requests simulated in this exercise, there has been a 100 % success rate in processing user requests efficiently. This efficient processing can also be appreciated in the average and maximum response times of the service. It was expected that the message broker microservice could have represented a bottleneck in the system, as it queues flight plan data for the validation microservice. However, here we could also see a successful process rate in delivering flight plans as messages. The number of simulated requests derives from simulating a point-to-point mission over a 2-month timeframe.

With this results it has been demonstrated that all flight plan data requests have been correctly processed by the FPM service prototype and that the service data routing has efficiently processed flight plan data for the validation microservice.

DM-2: Consistent and available flight plan data

To demonstrate that the flight plan data samples in the FPM service remain consistent and available, the uniqueness in the data sets and the availability of the service have been measured.

Success Criteria 1: No duplicate flight plan records exist in the database.

Success Criteria 2: Uptime of the service is at least 99%.

METRIC	VALUE	SC1	SC2
Uniqueness in the flight plan data sets (UQ) – (number of records in the database for each flight plan)	1	x	
Availability of the service in uptime (AVS) - (%)	99.5		x

Table 40: Summarized results of experiment related to data consistency

In the first aspect, the centralized management has been able to maintain unique data sets in the database, even after modifications requested by the conflict manager microservice or the MPM service itself. For the second aspect, the cloud-based instance of the FPM service was not fully available during the scope of the simulation (downtime of around 7 minutes per day), which means that at particular times the flight plan was not able to be submitted for validation at all.

Although the availability uptime is considerably high (99.5%), this means that still the FPM service was not able to be reached from every geographical region at every time. If it is considered that for this experiment only one instance of the service was implemented in the cloud environment, a situation where instances of the service are located among various servers and need to be accessed from various regions possibly makes it more challenging to achieve high availability values. On the other side, service availability is a characteristic that can be improved through the product selection in the cloud computing platform (e.g. utilization of multiple instances or virtual machines). Naturally, this will lead to higher implementation costs.

DM-3: Equitable distribution of data management

The objective is the assessment of the processing time in the Flight Planning Management Service against the respective processing time in Mission Planning Management Service

Contemplating the mission and flight planning of drone operations as an integrated capability, it is assessed in this section how the data management required for the planning of drone's operations is distributed between the MPM and FPM service. Initially, the success criteria are focused on distributing at least 50% of the complete planning time (characterised in processing time) into the MPM service.

Success Criteria 1: The Mission Planning Management service has a ratio of at least 50% in the complete processing time.

METRIC	VALUE	SC1
Average processing time of FPM Service to flight plan requests (FPT) – (sec)	0.07	x
Average processing time of MPM Service to generate flight plan requests (modelling and generation) (MPT) – (sec)	0.34	x
Ratio in the request duration between the MPM and the FPM service (RRD) - (% of processing time distribution)	83-17	x

Table 41: Summarized results of experiment related to the assessment of processing time

From the measurements it turns out however that the processing time of the MPM service is considerably much larger than the time from the FPM (83% of the complete processing time). The reason for this resulted from the consuming tasks performed by the trajectory optimization and modelling performed during the mission planning.

Efficient data processing in the Traffic and Monitoring Services (EXE 3)

In case of exercise 3, the remaining success criteria that were not considered in the previous section will be assessed, although the information can be extracted from the described conclusions.

Success Criteria 1: The platform shall be able to process, at least, 99% of the messages collected.

Success Criteria 2: The maximum delay that can be admitted by a U-space information management system shall be 5s when considering traffic information service (processing the data, integrate it with the rest of the operations and provide the view to the users).

METRIC	VALUE	SC1	SC2
(MP) Percentage of messages processed (%messages)	100	x	
(MD) Maximum delay of the information considering the number of flights processed (seconds).	1.921		x

Table 42: Summarized results of experiment related to data processing

As it has already been stated, it is possible to extract some additional requirements for the modules that comprise this architecture from DM-1. On the one hand, if the resources that will be used to execute it are also flexible enough to deal with every increase in demand, the delays will not be critical (less than 1s). On the other hand, and this relies on the “user” segment, the amount of information expected is such that it is possible that the current GIS systems, web browsers and all the features used to show information will not be able to handle it, so a filter based on an update rate shall be defined in terms on the criticality of the environment. Finally, as it expected that a variety of types of messages shall be processed (considering the heterogeneity of drone configurations, the growing list of U-space services, the interoperability between authorities, etc.) it is crucial to define several information listeners that splits information flow into more simple processes to be capable of understanding a certain set of topics, decoupling the workload into, also, small modules.

Up-to-date situational representation of the data in the Traffic Management Service dealing with dynamic airspace management (EXE 4)

Validation Objective: Ensure data set is current when internal service decisions were made from it.

There were different classes of data:

- Situation data, which is comprised primarily of cooperative telemetry sources;
- Environment data, which is comprised primarily of weather;
- Aeronautical data, which is comprised primarily of volumetric information describing regulated or restricted airspace.

RMSTE is able to operate on the basis that data providers in each of the categories, except for situation data, have been assured to be of a minimum quality bar and so it is not the duty for RMSTE or the Node Manager to ensure consistency within data suppliers' systems.

What is important, however, is the consistency of the 'state' managed within each node, and, operationally, across a cluster.

In considering operational data, there is the minimum set of data consistency required to maintain an adequate air situation picture and to provide the service, and a secondary set of conditions regarding to consistency of analytical/audit data. The latter is handled by sending analytics/audit events into a cluster-wide message queue, which is 'eventually' written to permanent storage by a separate service.

For live, operational data, it is first important to recall the failover characteristics of the cluster are, for node-level failures, sub second. In this mode, since the primary state for each node is saved in the node's volatile memory, if the node goes offline, the state of traffic handled by that node effectively is lost. However, vehicles operating in the node report their location to RMSTE at least every 2 seconds, thus the maximum exposure window is 2 seconds.

It is therefore not considered necessary in these experiments to share node state across nodes since the entire cluster state is rebuilt every 2 seconds anyway.

CONCLUSION: We have successfully demonstrated "An up to date situational representation at all times for the area being monitored."

3.3.3 Scalability

Scalability of the Weather Service in dense utilization areas (EXE 1)

This section presents the results of the load tests to see how the micro weather service solution front end behaves under an increasing load of requests.

As the core part of the service in independent of the actual number of clients (as explained it in § 2.4.1 only depends on the total area and frequency of prediction requested) this test focusses on the front end for data delivery using synthetic data. This setup is set in a similar fashion to the resilient test presented in 3.3.1.

In order to simplify the test, synthetic data is used and no format translation is performed at the client's agent level. The computational core of the weather service is simulated by a series of dummy containers that fill their corresponding queues with synthetic data that simulates the prediction data at a rate of 10 packages per minute of 500KB each.

All the microservices modules and all data brokers are running on the same server with 28 Xeon cores (HT enabled) and 256GB of DDR4-2400 of RAM. The physical connection with the clients is a single link 10 Gbits Ethernet run through a 10 Gbit capable switch.

The server, the gateway docker container and the internal data broker container are running initially and then clients start to be added performing requests at a fixed rate of 50 every minute. All clients behave well and consume the AMQP messages in their respective queues at an adequate rate. As the objective is to simulate the scalability of the system alone, no failure in clients were simulated in this test.

As the number of concurrent clients increases so does the number of docker container instances of the client agent running.

Using the monitoring tool "*docker stat*" it was sampled the RAM and CPU usage of the server containers machine at different work load levels. The measures were taken at increments of 50 clients in the load from zero to 1000.

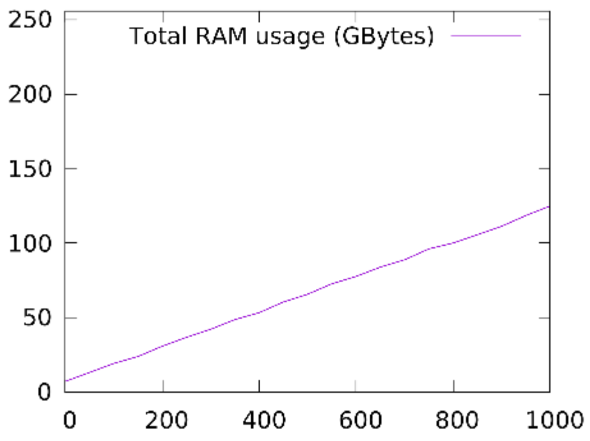


Figure 54: RAM usage as a function of the load

Initially, when zero clients are being served, the running of the gateway and the data brokers simulating the weather core interface already make use of 6.78 GB of RAM. The first instance of a client produces an increase in the footprint of 274MB in RAM with all subsequent ones being 110 and 140 MB range. This is translated in a fairly linear increase from the second client onwards reaching the amount of 124.858 GBytes of total RAM consumption when 1000 clients are consuming data. This amount is well below the 256 GB mark set as a maximum RAM usage. Extrapolating the obtained data, it can be expected a single server for this frontend to be able to sustain 2000 clients before hitting its RAM limit.

As a note, it can be observed that deploying the first client agent had a bigger RAM impact than all subsequent ones. Deploying the different modules of the service such as the client agents as Docker containers instead of fully independent virtual machines has the added advantage that instruction parts of the different copies are shared between all the instances, making the RAM consumption grow much less for any instance after the first one on the same physical machine.

As a result, only one of every binary library linked to the agents per physical machine exists. This is very significant as all JSON and XML parsing libraries to parse and generate all messages can be otherwise quite big in size. These blocks of RAM are marked as execution code and not as user data by the OS if the hardware supports it (such as any modern Xeon processor) preventing it to be a factor for fault propagation.

CPU and Bandwidth

The CPU and Bandwidth usage on the server was measured on the same points as the RAM (every 50 clients, up to 1000).

The server without any client load was using an average of 4.75% CPU. This usage can be attributed to the gateway and, mainly, to the processes of the data brokers for interaction with the core weather modules. They have to be managing the data produced by the core regardless of client activity, keeping the topics of every cell updated just in case any client agent start reading from them.

The rest of the results were linear as expected from the first client, increasing the measured load on the server until reaching 48.24% at 1000 clients.

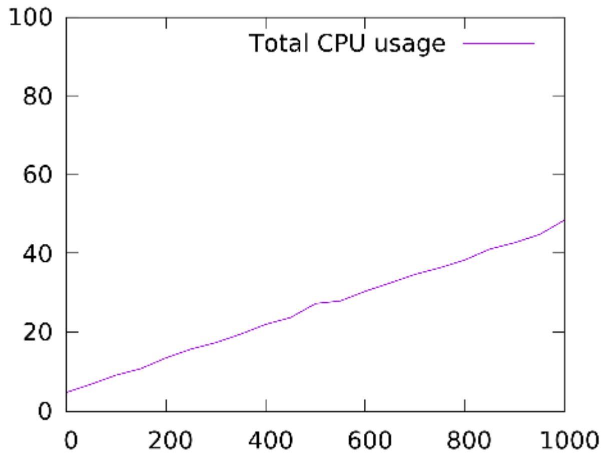


Figure 55: CPU usage as a function of the load

The combined CPU usage of the gateway, agents and data brokers stays well within the limits for this server during the whole ramp up to 1000 clients. In fact, the CPU usage never reached 50% load during the test.

While it would bring the system close to the limit, the server could theoretically also scale up to 2000 clients per server regarding CPU requirements.

The host server monitoring tools were used to measure the total bandwidth usage of the server. As predicted, the total average network traffic also grows linearly with the number of clients reaching a value of 678Mbit/s with 1000 clients. The burst nature of AMQP made the instantaneous maximum bandwidth usage seems during the experiments more than 3 times higher than this value but it always stayed below the 25% capacity of the network adapter.

The fact that in this scenario CPU and RAM usage both are around the same mark does not completely capture the behaviour of the service. In more complex tests, where clients required different data formats from the service (for example a JSON encapsulation or a defined WXXM extension) it is expected that the CPU requirements would grow faster than the RAM ones. On the other hand, bandwidth requirements are not expected to be bigger in more complex scenarios.

It is clear that further testing of scenarios that are more complex should be performed before any production deployment. In any case, the results of this test show that the proposed architecture for the front end of the service scales really well with the number of clients and shows to be a valid path for the development of a commercially viable future version of this service.

Success Criteria 1: At least 500 users served concurrently with 256 GB installed memory.

Success Criteria 2: At least 500 users served concurrently with two Xeon Gold processors (56 concurrent threads total).

Success Criteria 3: At least 500 users served concurrently with 10 gigabit/s of bandwidth.

RESULT	SC1	SC2	SC3
Success. 1000 concurrent clients used 124.86 GB of RAM on the server machine.	x		
Success. 1000 concurrent clients caused and average load of 48.24% of CPU on the server machine.		x	
Success: 1000 concurrent clients caused an average bandwidth consumption of an average of 678Mbits/s on the server machine.			x

Table 43: Summarized results of experiment related to the scalability of the Weather Service

Self-management in the Traffic Information Service (EXE 3)

In this section, the different mechanisms to response to increases in demand and peak workloads shall be analysed. It is expected that the microservice architecture shall detect these increments (while monitoring the status of the services) and will react creating more microservices to cushion the expected rise in the demand of resources. As a consequence, the exercises that have been presented shall provide an answer not only to the behaviour of the system, but also the management of resources, the activation mechanisms and even the expected needs in resources, always ensuring that the loads do not affect the proper operation of the platforms.

In case of exercise 3, the number of inputs and so the outputs will be increased by the integration of more and more drones (up to 256) in the platform, sending their position and creating alarms when necessary and other messages related to other topics. The variation in the delay of the information (since it is processed and shown to the users) and the creation of microservices will be measured, highlighting the validity of this architecture.

<u>Deployment of additional microservices</u>

For exercise 3, two analogous validation objectives have been defined: self-management of the system to increases in response to in demand and instantiation of a backup microservice instead of one that has failed due to overload. In the end, these validations can be summarised as “the behaviour of the system when there is an overload of information". For this reason, there will be analysed jointly. The associated success criteria are:

S1-Success Criteria 1 (S1-SC1): The platform shall create at least 1 microservice when workloads close to the maximum limit are detected.

S1-Success Criteria 2 (S1-SC2): The delays of information must be reduced to half of its value when another microservice is deployed.

S1-Success Criteria 3 (S1-SC3): Additional microservices shall be developed within 5s after the workload is detected.

S2-Success Criteria 1 (S2-SC1): The failure must be detected within 0.1s since the issue happened.

S2-Success Criteria 2 (S2-SC2): There will be at least 1 extra microservice instead of the one that has failed

S2-Success Criteria 3 (S2-SC3): Backup microservices shall be deployed within 3s since the failure was detected.

METRIC	VALUE	SC	
Instances created per service (number of services).	3-5	S1-SC1	S2-SC2
		X	X
Reduction of delays due to workload balance (%)	75%	S1-SC2	
		X	
Setup time to execute another instance of the service (seconds).	193s	S1-SC3	S2-SC3
		-	-
Time to detect the failure of the service (seconds).	0.001s	S2-SC1	
		X	

Table 44: Summarized results of experiment related to the deployment of additional services

The scalability tests (MTI-Test 2) represent the way of working of a microservice architecture: once these increases in workloads are detected, the platform reacts to it and provides the necessary mechanisms to absorb these peaks, creating more microservices if needed.

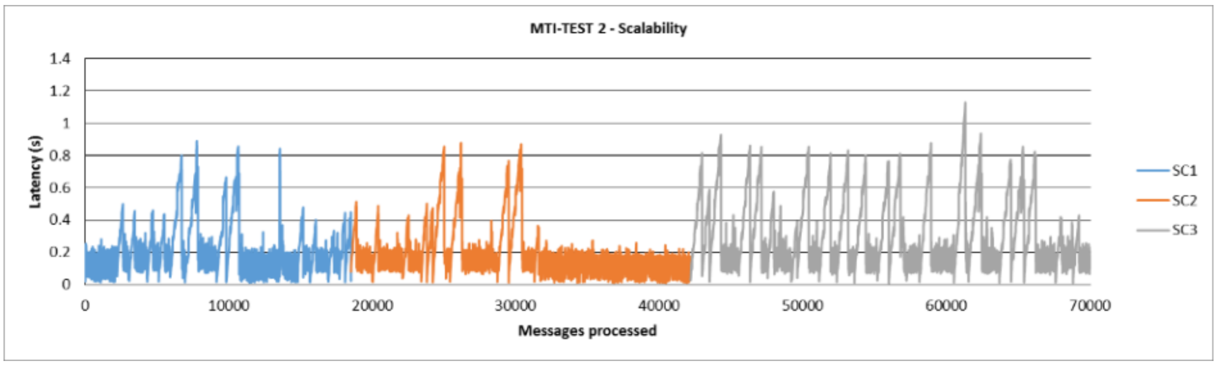


Figure 56: Latency of messages processed in experiment MTI-2

Although scalability tests have been performed to analyse this fact, MTI-Test 5 (Maximum workload without filters) is a more illustrative example. As it can be seen from the fluctuations in the delay:

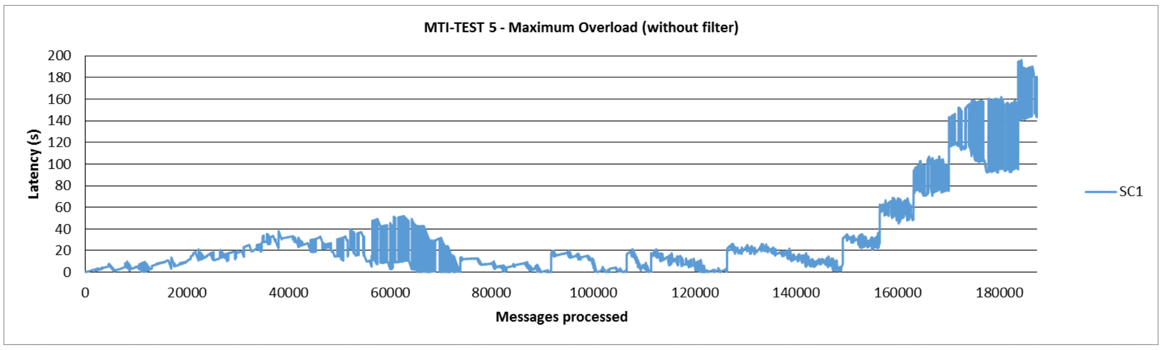


Figure 57: Latency of messages processed in experiment MTI-5

These variations are the response of the system to huge increments in the workload of services. The most important metric here is the reduction of the delays as a consequence of balancing the load between different but identical microservices (the other success criteria have already been explained in § 2.3.2.3). For instance, the rise in the band of 60000 messages is mitigated and the delays are reduced from more than 50 seconds to 15 seconds. In terms of the workload of the microservices, it can be stated that a microservice with 120% workload is reduced to 28% and 26 %, improving the capacity of the system and allowing it to absorb these increases. In the following steps, the behaviour of the system is similar (except for the final one that is expected to crash the system).

As it has been aforementioned, 15 seconds is not an acceptable delay for critical services that will operate in U-space, so other countermeasures shall be implemented to provide agile and efficient mechanisms to absorb these increases in demand (e.g. microservices in standby, maximum target levels that will deploy microservices even when they are not near to overload…).

Automatic scaling of the Traffic Management Service dealing with dynamic airspace management (EXE 4)

Validation Objective: To ensure the system automatically scales to manage the UAV load within the maximum possible capacity of the test area, ensuring the system is never constrained by processing load of UAVs being managed but instead only by the environmental conditions being faced.

RMSTE (Region Management Simulation Test Environment) was designed to run on modern, low-cost public cloud infrastructure. The infrastructure is therefore virtualised and runs on clusters consisting of nodes with configurable RAM, CPU, Disk and Network performance.

It is good practice in cloud-scale system design to understand what vertical (size of disk, amount of RAM) scale-up options exist, and what scale-out (number of nodes in a cluster) options exist, for each service.

RMSTE was designed around these principles. Factors such as the volume of non-drone activity within the test area, as well as the frequency with which cooperative drones report in telemetry, affect the capacity of each node. Performance targets, such as response time and latency, are metrics, which enable the Node Manager to determine if an additional node is required.

The Node Manager acts like a governor for the system and for the health of the clustered deployment. By evaluating key metrics every few milliseconds, the Node Manager can quickly instigate a change in deployment characteristics without human intervention, ensuring Quality of Service criteria (which can be configured) were maintained.

During these experiments, Node Manager watched for:

- CPU Capacity crossing 60% utilisation for 3 minutes
- RAM utilisation crossing 85% utilisation for 3 minutes
- Bandwidth utilisation crossing 90% utilisation for 2 minutes
- Total drones-under-management within a node exceeding 250

When these situations were found, Node Manager correctly (< 15 seconds) instantiated another node, and joined it into the load balancer, thereby increasing the aggregated capacity. When the conditions that caused the increase initially subsided after a 30-minute period, the node manager correctly de-provisioned the last node added, back to the minimum level set for fault-tolerance.

For the experimental exercise, the vertical scale was set as a manual task: rather than to create a cluster with truly elastic node capabilities, the physical allocation of resources to virtual nodes was preconfigured, and the node manager was simply allowed to auto-scale the number of nodes between a minimum and maximum.

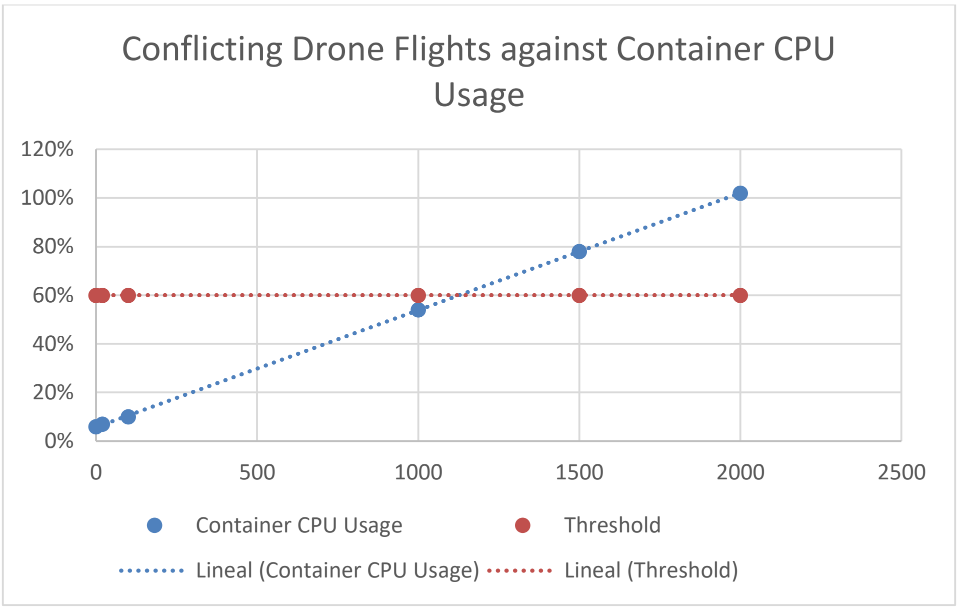


Figure 58: Shows Number of Drones in Conflict against CPU Usage

Figure 58 shows the breaking down of the simulated environment and the continuous increasing of the number of drones in the system to the point of saturation.

Memory was not exhausted in the experiments. However, the CPU was the primary reason for scaling requirements. The CPU increased based primary, not as the number of drone flights increased but rather as the number of the drone flights requiring conflict resolution in the region increased.

As the CPU threshold was reached, a new container was created to balance the CPU load requirements, and once load was decreased alongside the CPU usage, after 30 minutes the additional containers were decommissioned.

CONCLUSION: It has been successfully demonstrated that it is possible to "Ensure the system was never constrained by processing load of UAVs being managed but instead only by the environmental conditions being faced."

CONCLUSION: It has been successfully demonstrated that it is possible to "Ensure the system was not over provisioned (to ensure commercial efficiency)."

3.3.4 Flexibility

Self-monitoring in the Traffic Information Service (EXE 3)

An effective use of resources is mandatory in an architecture that must deal with wide variations in demand and, many events of different type and a variety of messages of different nature. The capacity requirements cannot be clearly defined and so the different modules to connect all the services, so the platform must be "open" to changes and expandable in an autonomous way.

Adaptability of the architecture to the existing demand

The validation objective that will be assessed is “Address the processes of self-management in a microservice architecture to respond to fluctuations in demand”, which means that the monitoring of microservices in this architecture shall answer not only to increases, but to off-peak times in which the use of duplicated resources is not needed.

Success Criteria 1: The maximum workload of a microservice shall be less than 80% of its capacity, to avoid peak increases that will lead it to a failure.

Success Criteria 2: Once the workload of a microservice reaches its maximum rate, another instance of the service shall be created.

Success Criteria 3: In off-peak periods, the number of microservices shall decrease in, at least, one instance.

METRIC	VALUE	SC1	SC2	SC3
Instances created per service (number of services).	3-8	X		
Instances reduced per service (number of services).	5-8		X	
Maximum workload supported per service (%).	165%			-

Table 45: Summarized results of experiment related to the adaptability of the architecture

MTI-Test 3 for flexibility checking have not been enough to assess this topic. Furthermore, the conclusions that can be extracted from stress tests (especially MTI-Test 5) and resilience (MTI-Test 6) will lead us to significant outputs.

The stress test reflects the growth of the architecture due to increases in demand, being capable of using all the resources available trying to provide an answer to these sharp rises. However, on the other hand, if the platform has not been properly sized and without enough robustness it will lead it to overloaded microservices (e.g. using more virtual machines, the regular workload percentage is estimated to be 40% of the total).

On the other hand, from the resilience test, the following log highlights the normal behaviour of this architecture:

	NAME-ID	READY	STATUS
INITIAL STATUS	activemq-0	1/1	Running
	activemq-1	1/1	Running
	activemq-2	1/1	Running
	apache-gzpch	1/1	Running
	alarm-52nlz	1/1	Running
	listener-72vls	1/1	Running
	mavlink-8jrx9	1/1	Running
	mavlink-qwcvr	1/1	Running
	zonas-vqn4j	1/1	Running

	postgres-8n5h2	1/1	Running
	unomongo-0	1/1	Running
	mavlink-8jrx9	1/1	Terminating
INCREASES AND DROPS IN DEMAND	mavlink-8q9dm	0/1	Pending
	mavlink-8q9dm	0/1	Creating
	mavlink-8q9dm	1/1	Running
	mavlink-c6t8l	0/1	Pending
	mavlink-c6t8l	0/1	Creating
	mavlink-c6t8l	1/1	Running
	mavlink-x9bv9	0/1	Pending
	mavlink-x9bv9	0/1	Creating
	mavlink-x9bv9	1/1	Running
	mavlink-8jrx9	0/1	Terminating
	mavlink-8jrx9	0/1	Terminating
	mavlink-5ddmk	0/1	Pending
	mavlink-gpdlc	0/1	Pending
	mavlink-r2r44	0/1	Pending
	mavlink-gpdlc	0/1	Pending
	mavlink-r2r44	0/1	Pending
	mavlink-5ddmk	0/1	Creating
	mavlink-gpdlc	0/1	Creating
	mavlink-r2r44	0/1	Creating
	mavlink-gpdlc	1/1	Running
	mavlink-5ddmk	1/1	Running
	mavlink-r2r44	1/1	Running
	mavlink-qrqd5	0/1	Pending
	mavlink-jbn6j	0/1	Pending
	mavlink-qrqd5	0/1	Creating
	mavlink-jbn6j	0/1	Creating
	mavlink-qrqd5	1/1	Running
	mavlink-jbn6j	1/1	Running
	alarm-52nlz	1/1	Terminating
	alarm-c9mrn	0/1	Pending
	alarm-c9mrn	0/1	Container
	alarm-c9mrn	1/1	Running
	alarm-q92hd	0/1	Pending
	alarm-q92hd	0/1	Container
	alarm-q92hd	1/1	Running
	alarm-j6wsz	0/1	Pending
	alarm-j6wsz	0/1	Container
	alarm-j6wsz	1/1	Running

	alarm-52nlz	0/1	Terminating
	alarm-n24zn	0/1	Pending
	alarm-szc6w	0/1	Pending
	alarm-n24zn	0/1	Creating
	alarm-szc6w	0/1	Creating
	alarm-szc6w	1/1	Running
	alarm-n24zn	1/1	Running
	mavlink-5ddmk	1/1	Terminating
	mavlink-9bshv	0/1	Pending
	mavlink-9bshv	0/1	Pending
	mavlink-9bshv	0/1	Creating
	mavlink-9bshv	1/1	Running
STABILIZATION	mavlink-jbn6j	1/1	Terminating
	mavlink-qrqd5	1/1	Terminating
	mavlink-9bshv	1/1	Terminating
	mavlink-r2r44	1/1	Terminating
	mavlink-gpdlc	1/1	Terminating
	mavlink-5ddmk	0/1	Terminating
	mavlink-gpdlc	0/1	Terminating
	mavlink-r2r44	0/1	Terminating
	mavlink-jbn6j	0/1	Terminating
	mavlink-9bshv	0/1	Terminating
	mavlink-r2r44	0/1	Terminating
	mavlink-qrqd5	0/1	Terminating
	mavlink-jbn6j	0/1	Terminating
	mavlink-qwcvr	1/1	Terminating

The services (identified by the Name-ID pair) are created and destroyed under demand. It has already been mentioned that, in order to absorb the increase in the use of resources during the initialization of a microservice and the peak information loads several microservices are created at the same time and that is the reason of the creation and termination of a lot of microservices in this experiment, when the desired microservice is working properly the rest of "support microservices" are not needed anymore, so they are removed from the system. Another additional output is that, after removing one of the additional microservices, it was not necessary to instance a new one due to the demand at that time was not high enough to produce workload in the existing one.

3.3.5 Commercial Implications

The challenge addressed in this section is to accurately calculate implementation and computational costs for satisfying microservice demands of a highly variable number of users. Considering this, the consortium performed 3 different cost analysis in order to have a better understanding about the cost implications of implementing and maintaining an eco-system of microservices running in a cloud-based platform and more important, what is the impact for users that shall consume resources from U-space services. Differently as the previous sections, this section is structured in three parts showing the results from the different approaches on how to calculate costs.

3.3.5.1 Cost estimation for requests to the Flight Planning Management Service (EXE 2)

Service costs distribution as function of the flight plan management requests

The FPM service involves the interaction of different microservices and is subjected to various internal work flow processes based on the type of request submitted by the clients. For the examinations considered in the scope of the project, 5 general processes (or so-called *transactions*) are considered:

- Flight plan validation/update request;
- Flight plan cancelation request;
- Flight plan information update request (no modification of the flight trajectory);
- Flight plan status request;
- Flight plan database status request.

These cover the centralized flight plan planning management capability of the implemented FPM service. The next sequence of figures describes the distribution of tasks in all the transactions considered. It is expected that each task and ultimately each transaction requires different computational resources to satisfy the requests. Moreover, it is important to remark that these transactions are not necessarily invoked in every case during the pre-flight phase. For instance, if the flight plan validation proves to be successful, and no modifications in the flight plan are requested, only one transaction would be sufficient for the client to request (in this case: flight plan validation).

During the simulations performed in the exercise 2, the computing resources needed to satisfy the requests have been obtained through the measurement of the processing time that the single microservices require. A summary of processing time in relation to the single transactions is given in Table 46. The values have been normalized to the largest value measured.

TRANSACTION TYPE	Number of microservice invocations per 100 transactions	Average processing time (%)
Flight plan validation/update (Transaction 1)	48	100
Flight plan cancelation (Transaction 2)	23	12.5
Flight plan information update (Transaction 3)	21	9.1
Flight plan status (Transaction 4)	5	9.1
Flight plan database status (Transaction 5)	3	12.5

Table 46: Processing time in relation to certain transactions

With this information, it is possible to make a first estimation about how the various service functions can be billed based on the resources that they consumed, regardless of the pricing model in which the use of the flight plan management functionality is included. From this distribution it can be extracted that most resource demanding process is the flight plan submission processing and associated flight trajectory validation or update. This transaction is up to ten times more resource demanding than the other typical transactions.

Estimation of the service transaction costs under the consideration of a pre-defined pricing model

Based on the forecast of future drone operations [9], a number of drone operations taking place during a defined timeframe can be projected. From this projection a number of requests per month can be further derived. This can be used as a groundwork for estimating service transaction costs. Following assumptions are relevant to mention:

- For estimating the number of drones in activity, a segmentation by the main industry sectors (agriculture, energy…) has been performed;
- A static projection for the year 2025 in Europe is considered [9].

Lastly, besides an expected projection scenario, two more scenarios (conservative and optimistic) can be elaborated in order to scale up the estimation. The table below summarizes the results of the projection for 3 different scenarios.

	Conservative	Expected	Optimistic
Number of drones in activity	54,167	195,000	433,333
Agriculture (transactions/month)	429	5,714	44,571
Energy (transactions/month)	95,333	202,500	758,333
Public Safety and Security (transactions/month)	83,333	2,083,333	7,500,000
Delivery (transactions/month)	225,000	1,061,667	4,666,667
Others (transactions/month)	54,167	195,000	433,333
Flight plan requests in a month	458,262	3,548,214	13,402,905

Table 47: Summarized results for the estimation of flight plan requests

Although this study is limited, it provides an initial view on how large the demand for flight plan requests can be. Subsequently, the implementation costs of deploying and maintaining the microservices - ideally implemented as self-managed container instances - in a cloud-computing platform can be estimated based on this potential demand. The following factors will affect the implementation cost estimation for the most part:

- The pricing scheme offered by the cloud-computing platform (e.g. list prices from Microsoft Azure [10].
- Optimization of microservice implementation based on performance characteristics that the different microservices require.

- Expected maximum load in the number of requests per time unit that the microservices need to process.
- Supporting services needed for the adequate functionality of the microservices, such as registry, service bus and database storage.
- Infrastructure costs that apply for the entire microservice ecosystem in general (API gateway, service discovery, etc.)

For the different projection scenarios presented above, the required computing resources can be obtained and finally the single service transaction costs can be derived. The following figure shows an exemplary distribution of the price per transaction as a function of the projection scenario.

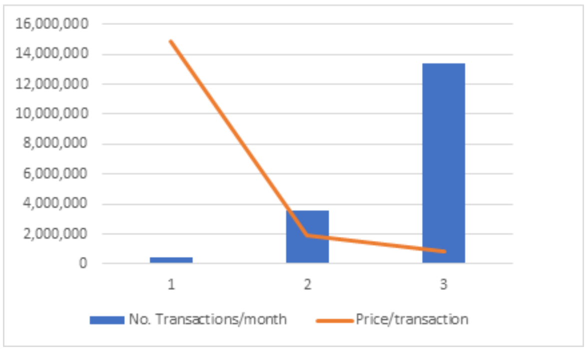


Figure 59: Comparison of the number of transaction against the transaction price

From this distribution it can be seen that the price decreases rapidly as soon as the transaction demand increases. The main reason for this decrease is the low implementation costs that microservices have when they are scaled up in the cloud-computing platform.

3.3.5.2 Implementation cost estimation for the provision of traffic information services (EXE 3)

The microservice architecture that has been used in exercise 2 has been mainly implemented using open source technologies (MongoDB, ActiveMQ, PostregsXL/PostGIS, IGNITE and Kubernetes). The cost of this platform, despite the efforts to develop it, relies in the initial Listener to process the information and the physical support that contains this architecture and, even in this case, the NGINX module (listener) is already integrated in AWS, so the cost is also reduced.

As part of AWS environment, there are simple mechanisms to extract the exact costs of the modules that have been used as part of the platform. The following table shows the cost of exercise 3, considering only the physical support of the platform and the different communications mechanisms that allows the connection of the different parts of the architecture and considering that the total number of outputs may have reached 1 million of messages (more than 1 message per second) and the number of input messages is more significant as they have been previously filtered. As this exercise was performed in two days, the costs are distributed in two columns.

Cost (€)	Service Items CONTAINER	DAY 1	DAY 2
Direct Charges (14)		20.11 €	20.03 €
	EC2 - Compute	12.84 €	12.84 €

Cost (€)	Service Items CONTAINER	DAY 1	DAY 2
	EC2 - Transfer	0.08 €	0.03 €
	EBS - Storage (remote servers)	2.22 €	2.22 €
	EC2 - EBS Surcharge	- €	- €
	EC2 - Load Balancer Transfer	0.03 €	0.00 €
	EC2 - Load Balancer Usage	0.54 €	0.54 €
	S3 - API	0.04 €	0.04 €
	S3 - Transfer	0.00 €	0.00 €
	S3 - Storage	0.00 €	0.00 €
	Amazon Elastic File System	0.03 €	0.03 €
	Route 53 - Hosted Zone		
	EKS - Cluster Hours	4.32 €	4.32 €
	ECR - Storage	0.01 €	0.01 €
	ECR - Data Transfer		
Indirect Charges (10)		0.84 €	0.83 €
	EC2 - EBS Snapshot	0.18 €	0.18 €
	EC2 - Elastic IP		
	AWS CloudTrail	- €	- €
	DynamoDB - Data Transfer In	- €	- €
	DynamoDB - Data Transfer Out	0.00 €	0.00 €
	CloudWatch - Other	0.19 €	0.19 €
	Amazon Simple Notification Service	- €	- €
	AWS Key Management Service	0.00 €	0.00 €
	AWS Config		
	EC2 Usage	0.48 €	0.47 €
	Total	**20.96 €**	**20.87 €**

Table 48: Cost of exercise 3 in support systems

The following diagram provides an overview of the cost per module:

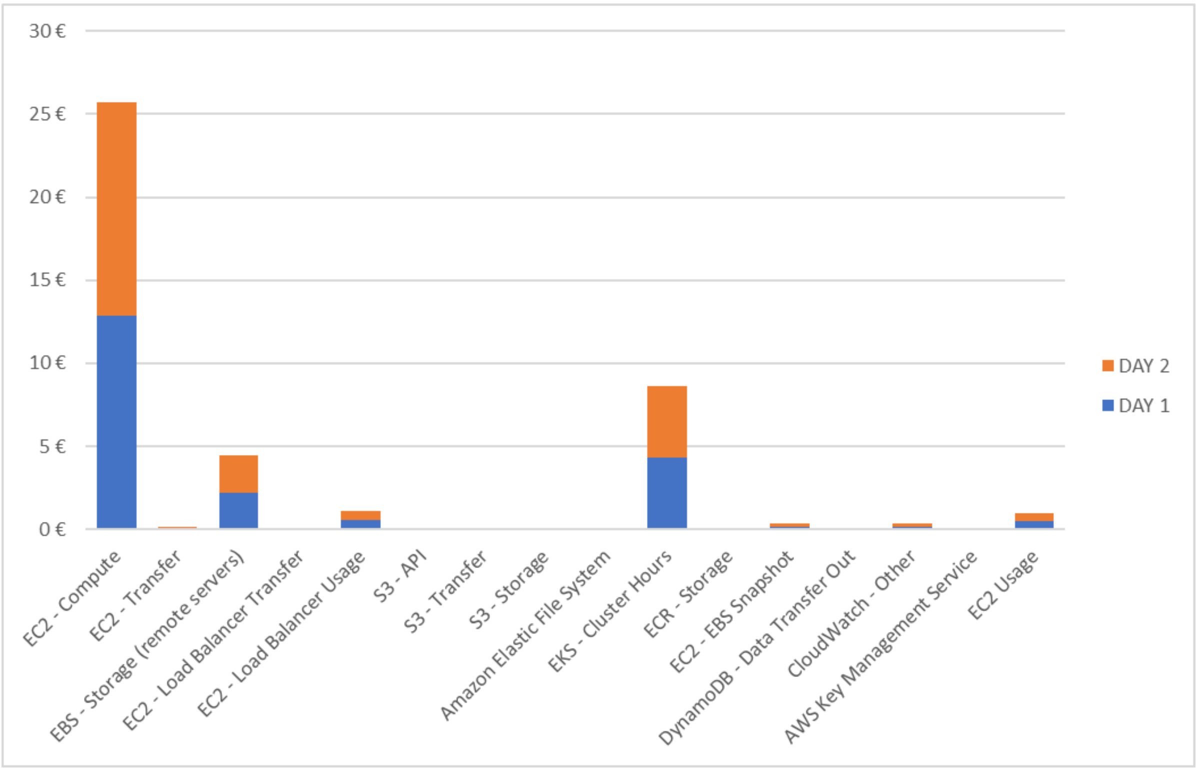


Figure 60: Cost distribution per AWS module

As a rule, the higher costs will be distributed in the computing resources (the virtual machines running the architecture), the use of remote servers (cluster hours, storage and elastic cloud -EC- services) and the load balancer. The increase in number of microservices implemented (and so U-space services), computing resources and mechanisms to keep a high performance will increase them. A more complete analysis, considering all the U-space services defined and implemented (or maybe not all of them, taking into account that maybe some services shall rely in the USSP side, becoming data sources) shall be performed to provide a global vision of the costs of this U-space information management platform.

Considering a mean cost per day (20.92 €) and dividing that number by 250 (close to the maximum number of drone operations measured) it is possible to obtain the cost per operation of the platform (0.08366 €/day & operation). This result has been extrapolated using the forecast in the growth of drone operations detailed in 3.3.5, from the European Drone Outlook Study [9].

	Conservative	Expected	Optimistic
Number of drones in activity	54,167	195,000	433,333
Total cost per day (€)	4531	16311	36247

Table 49: Cost of the microservice architecture from exercise 3 per day

The predictions involve a yearly cost of 6M€-13M€ which, considering the number of drone operations, it is not significant. It is true that other modules shall be attached to this architecture and the number of computing resources and transactions will be increased but in this case, in which real-time information is crucial, this budget is affordable. In addition, even, if these costs are increased dramatically, a simple pay-per-use methodology can be implemented in which each user will have to pay a small fee (maybe less than 10 € per operation, saving millions of money yearly) to be connected to this platform.

Finally, as this architecture has been designed to support a number of 250 simultaneous drone operations in the same area and considering that pre-flight validation has not been taken into account, the number of alert messages is expected to decrease and so the number of drone messages to be processed (this platform is capable of ingesting all the data streams, consisting of 1 per 0.05 s) and shown to the users, so the total cost of resources and data exchange mechanisms will be reduced.

3.3.5.3 Computational costs for the Weather Core Service (EXE 1)

The cost of the Weather service can be decomposed in two blocks:

- Cost of the core service;
- Cost of the client agents and data brokers.

The scalability test has shown that the cost of the client agents and data brokers grows mostly linearly, being able to use a single middle size server (see tests for hardware details) for 1000 clients, easily scaling to more with multiple servers.

The server acquired for these tests has an initial cost of 19000 Euros and, supposing 4 years of use the approximate TCO (total cost of ownership) would be 11500 Euros, including maintenance, bandwidth, power, IT labor costs and space (software maintenance not included).

To provide rational resilience at least 2 independent deployments should be set up which give us the approximate cost of 23.000 Euros per 1000 users per year for having this capacity hosted in-house. That is, to be ready to serve 10000 users a cost of 230.000 Euros would be fixed every year, regardless of the number of actual users.

Opting for cloud deployment, a similar server to the one used, for example in the m5 series of the AWS cloud would cost between 3000 and 3500 Euros a month for exclusive use, that give as a cost of between 36.000 Euros and 42.000 Euros (using an AWS provided redundancy solution, that would only be deployed and charged for during emergencies).

While it looks like the Cloud provided solution would be more expensive, it has to be taken into account that there would be no need to use an instance of this size in the number of users is smaller and the service could be seamlessly migrated from one instance type to another when the number of users significantly changes. That is, the cost would be more expensive per user but only actual use would be charged so depending on the capability to predict demand this could not be only a simpler solution but even a cheaper one.

Core Weather Service computational costs:

The cost of the Core Weather service would be the biggest part of the server costs, especially in cases where the densities of users is low.

For example, for a test area like the one used for the tests performed:

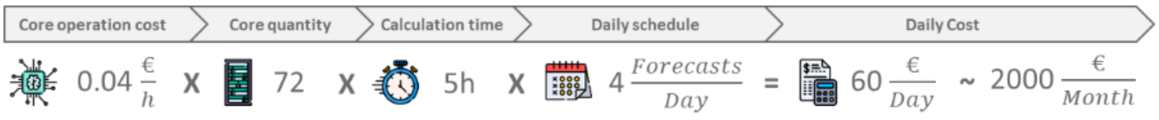


Figure 61: Summary of the cost analysis for the Weather Core Service

This gives us a cost of 24000 Euros a year for a single domain a year.

With all the previous data points, we can roughly estimate the cost computing cost of providing the service for a certain scenario. For example, 300 users using a combined area of 5 domains in a purely cloud solution would give a total cost of approximately 135000 Euros, being the core service cost 89% of it. This is a computing cost of 450 Euros per user per year. In denser areas, the cost would be dramatically less as the core service cost would be further diluted.

Potential cost reduction measures:

- Parallelization of forecast computation (more cores, less time): New instances of cloud instances with more cores could be used to reduce the computation time of each forecast. These instances are usually more expensive but when taking into account the total computing power they provide, allowing for shorter use, the final cost tends to be smaller. An added advantage is that the latency of the predictions would be lower;
- Model computation decoupled from client's requests and sensor inputs: As explained in or architecture, the frontend is designed so that the core computation is independent from the number of clients and only depends on the extension and frequency of the requested combined forecasts resulting in multiple clients/providers of sensor data for a given domain. This means that the cost per user of the service is potentially very low for high-density areas;
- Low and mid-resolution models reused for several domains: All pending jobs for the core service are composed and share all layers of calculated data where possible.

Other economic considerations:

- Potential need for human intervention to set up models in response to non-standard client requests: That is, for scenarios that are not covered by the predictions performed for the average user, for example other variables requested or special data format, an extra cost could be charged. Better service, higher cost;
- Computational cost correlated with number of domains, not consumers: It could be convenient for the provider to deploy the service in areas of high user density first and expand coverage once the service is economically viable;
- Potential to define tiered pricing schemes:
  - Less accuracy, lower price;
  - Discounts to sensor data providers;
  - Light Agents, e.g. consumer accepts standard data (GRIB, WXXM), so need for filtering or format customization;

Storage requirements for analytics, validation and resilience:

A typical 24h forecast for a domain like the one tested has a total size of 6 GB. That makes the data requirements for four forecasts a day, covering 72 hours of prediction window in advance would be 72 GB a day. The data created per year would be 26,2 TB.

4 Conclusions and recommendations

4.1 Conclusions

Conclusions and recommendations were obtained from the individual experiment results as well as from expert feedback in a dedicated workshop.

4.1.1 Technological Feasibility

Drone-specific Weather Service

During the development of this prototype, we have come to many conclusions that can be summarized into four main blocks:

1. A probabilistic approach to weather information provides higher quality information: After comparing results of the single forecast with our probabilistic approach, we see that it results in better-informed decisions. We proved that this approach allows for the propagation of the uncertainty to all applications that use it, giving, not a single plan or result of their actions, but a range of what to expect or do with the different expected weather scenarios.

   While the cases executed in our demonstrations clearly show the specific advantages when using probabilistic weather information for the different planning or executing tasks, they also show that each use of this information is going to require adaptation of the client's process in order to take full advantage of the statistical nature of the information. That is, the probabilistic weather information offers many opportunities to improve services that depend on weather information but adaptation is required to fully exploit it, a profound one in many cases. Regardless of this difficulty, we think that in a drone-dense environment, especially one where drones are fully automated, this way of providing weather information is going to be a game changer for many applications.

2. A micro-service oriented architecture and data flow provides a robust, scale solution to offer this service to a highly variable number of clients. This provides the required flexibility- in the form of individualized agents per client that can accommodate the information provided by the core weather service to every user needs -, scalability - allowing the feeding of information to a large amount of clients at a sensible cost - and robustness - isolating client errors from affecting each other. The combination of REST interfaces with AMQP messages has provided a middleware set that has proven very flexible and scalable, being able to transport all required information and thus providing a guaranteed QoS.

3. Many actors in the industry are actively looking for more suitable weather information that is directed towards drone operators. Drone-specific weather information will be an enabler for this industry and we have shown here that a probabilistic approach can improve the quality of many services built on top of it, from mission planning and path planning to traffic management.

4. The presented architecture would allow for a very reasonable infrastructure cost for this service, regardless of a cloud based deployment or an "on premise" one. Of course, a cloud deployment cost would adapt better to a dynamic number of users.

Mission and Flight Planning Management Services

Besides considering operational safety-critical aspects, a great emphasis was placed in the mission planning capability for fulfilling the overall mission requirements. The employed modelling functionality considered multiple factors (power consumption, clearance from ground surface) for the modelling of a feasible trajectory. However, one relevant aspect that has not been considered is the operational uncertainty when modelling trajectories in the spatial and temporal domain. Furthermore, according to the type of the mission and location, additional aspects should be considered for the trajectory optimization, such as population density and further geospatial information.

A functionality for conflict detection has been developed and tested in this exercise. In this initial development, the scope of the conflict detection function is limited to 4D trajectories which consist of a sequence of waypoints in the spatial and temporal domain and only covered a specific type of drone system. However, it has been demonstrated that the conflict detection can serve as a central controlling instance in U-space and interact with a service provider mission planning capability for processing and managing flight plan requests.

In a secondary exercise, the impact of applying conflict resolution solutions to modelled trajectories based on micro-scaled weather models and airspace restrictions was tested. For weather-based deconfliction, it was useful to complement weather models consisting of measurements from available local sensor platforms with official reports. However, it is relevant to consider existing deviations derived from this resulting process and special attention has to be placed in the employed interpolation methods. For the deconfliction solution based on airspace restrictions it still remains a challenge to extract all the necessary information from temporal airspace sectorisation publications in order to accurately model a restricted volume and therefore perform a strategic deconfliction in the spatial domain. Additionally, given the occupying volume of current airspace restrictions, it becomes challenging to work with deconflicted trajectories that can also still meet the overall mission goals, such as a maximum flight duration.

It should be stressed that the conflict resolution capability has been placed on the side of the service provider (MPM service) in this framework. If this shall not be the case, there has to be further research on how to exchange flight plan data between the FPM and MPM services during the conflict management phase. In case that the centralized FPM service shall cover the conflict detection and resolution functions, it is expected that the drone operator will still be part of the renegotiation process when formulating a deconflicted feasible flight trajectory. When looking back to the proposed framework, a further aspect yet to be evaluated in conflict management is the potential design of a set of proposals or recommendations that can help the operator in his/her re-planning of the mission.

As the main contribution of these exercise remains the architecture proposal for designing a centralized flight plan management service that has the components for processing and managing flight plan requests from external users. The flexibility of the microservice implementation approach has facilitated the modular and self-managed design of the main service components. For the microservice implementation of the different service capabilities (message broker, conflict manager, etc.) a dedicated implementation technology could be chosen (e.g. RabbitMQ, Java Spring Boot) that is best suitable for the operational requirements of the service. However, the key aspect to prioritize during the conceptualization and implementation process is the assurance of a common API contract in order to facilitate a seamless communication between the microservices.

During the simulation campaign of the mission and planning process the data flow management has been measured and analysed in terms of processing time. Compared with the processing time required by MPM service for the flight plan generation, the processing time required by the FPM was significantly lower, which shows that the microservices implemented in the FPM service are lightweight functionalities that can respond efficiently to flight plan requests. Although the FPM service is not the most critical service among the set of U-space services when it comes to workload and response time demands, it becomes relevant on the other hand to supervise and monitor to ensure a consistent flight plan database within the centralized system.

Traffic Information and Monitoring Services

The use of microservice-based architectures in the previous exercises have provided meaningful outputs to foster its use in the information exchange processes in U-space.

On the one hand, the demand capacity of this architecture has been addressed in a variety of tests. Considering simple messages (based in simple JSON-formatted streams integrated by the development teams), the data processing stages have proven to be agile and efficient, as it is easy to read and parse this information without significant issues (delays, increase in workload, etc.). In addition to this, the simplicity in the microservices that have been integrated and the parsing methods[3] plays a key role in the management of this architecture, ensuring that the workload of the system is not increased significantly when the number of data sources (and as a consequence, the results to be processed and shown to the users) is increased.

[3] In some cases, it consists of extracting the position of the drone and transform it into a renderable object that can be interpreted by a GIS system; in other cases, it involves a distance-to-all calculation and, if a condition is met, an alert based also on simple streams is shown; and finally the requests to/from distributed databases in which, due to the indexation mechanisms and agile searches, the information can be provided more quickly.

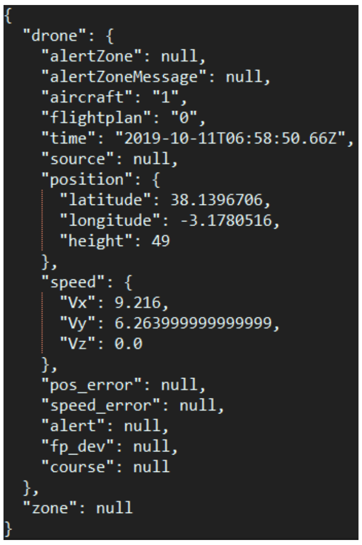

```
{
  "drone": {
    "alertZone": null,
    "alertZoneMessage": null,
    "aircraft": "1",
    "flightplan": "0",
    "time": "2019-10-11T06:58:50.66Z",
    "source": null,
    "position": {
      "latitude": 38.1396706,
      "longitude": -3.1780516,
      "height": 49
    },
    "speed": {
      "Vx": 9.216,
      "Vy": 6.263999999999999,
      "Vz": 0.0
    },
    "pos_error": null,
    "speed_error": null,
    "alert": null,
    "fp_dev": null,
    "course": null
  },
  "zone": null
}
```

Figure 62: Example of messages used in Exercise 3

On the other hand, the capacity of self-management of a microservice architecture shall be highlighted. As it has been stated, the microservice architecture is capable of absorbing slight increases in demand without conditioning the performance of the whole architecture. In case of peak growths in the number of messages or more permanent rises in demand this architecture counts on a set of mechanisms to detect[4], mitigate[5] and stabilize[6] its performance. These mechanisms will ensure that, providing enough resources to support them, the system can process and absorb all workload increases with no substantial impact on performance and delays in the whole information process environment. As main countermeasures to avoid issues in the data management, duplicity in critical services shall be provided whose operation can be defined as balanced (both services with similar load balance) or as backup system (one of them is working and the other one is in standby until a certain load limit is reached). It must be ensured that at least one microservice is in execution waiting for a failure to provide an initial back up to cover failure modes. Another consideration is that some modules in the architecture should not be completely redundant, as some services may operate in a degraded manner.

Moreover, not only microservices can be duplicated, but also the information ingestion methods (filters, message layers). These can be parallelized and decoupled, allowing certain parts of the architecture to be in charge of processing specific types of information. This mechanism can also be integrated into the architecture, ensuring that the acquisition of new datasets is not affected by bottlenecks caused by a variety of streams of different formats and promoting the agile management

[4] Monitoring of the status of each microservice (workload, failure modes, working thresholds, etc.).

[5] Backup services, workload balancing, creation of a set of microservices to absorb peak demands, message queues in which the datasets are stored, waiting to be processed and preventing information losses.

[6] Deletion of supporting microservices, management of the duplicated ones with the existing level of demand and target objectives to avoid bottlenecks.

of information, focusing on datasets of interest and even redirecting these inputs to a certain microservice to process it. In this way, the use of more than one queues is a suitable option, taking into account that when all of the U-space services are implemented, the first increase in demand will be located in the entry point of the platform.

Something not considered in exercise 3 are situations where the system is not capable of recovering, meaning that the failure is caused in the management mechanisms of the architecture. Therefore, contingency scenarios must be defined and rules for actuation developed. These could include the use of on-board capabilities of the drones for direct drone-to-drone coordination in case the U-space system fails entirely.

In addition to this, as previously stated, the platform will not only rely on its configuration, but also on the physical support through which it is implemented. In this document, the use of contracted cloud services has also been addressed that consist of a set of computers that are switched on and configured on demand, in seconds (or even milliseconds). The main advantage of this configuration is the flexibility in demand that brings to U-space environment, as the manager of the architecture must not be worried about overloads at any given time. Concerning the evolution of U-space operations, the configuration provided by the architecture in exercise 3 tested a set of 250 drones and continuous alerts and found that, in certain cases, some problems arose related to its performance. Therefore, it is crucial that resource allocation is managed incrementally to allow the system to expand and adapt to the expected growth of U-space environments. This resource allocation can be linked to profitability of U-space system.

However, the opposite point of view should also be considered. The number of drones that can be managed simultaneously for a certain area and during a certain time frame should in some way consider system limitations.

The implementation of U-space services in a microservice architecture is also a key role of this architecture. If the proper mechanisms for joining and a compatible information layer are in place, these services will be able to communicate and improve the outputs of the whole system. This leads to the commercial feasibility of this architecture, as it provides a methodology and standards to allow connection of new microservices and, as a consequence, incentivise solutions proposed by different companies that participate in this environment. Moreover, the decoupling of all the processes of a certain service into a set of simple functions (microservices) supported by a self-management platform that links them together is a valuable contribution to the information exchange processes.

Finally, it should be stressed that the value of every powerful architecture is meaningless if the information cannot be shown to the user in a simple way and in the expected time. The user segment (comprising drone operators and data consumer) must be considered, as the outputs of these processes are data streams that shall be transformed into simple, eye-catching messages and rendered for easy interpretation. Moreover, another issue to be considered is which information is necessary for users and the update rates in these outputs, allowing the interfaces to show all the required information and avoiding bottlenecks in the final part of the chain. It is clear that this approach must be based on an operational point of view, analysing the criticality of each operation, the congestion in specific areas, the capabilities of each aircraft and operator to react to sudden issues and the use of critical U-space services. Exercise 3 defined a one second latency as the baseline, which may be a viable solution in low-congested environment, but this number should be modified dynamically after analysing the whole set of components in U-space.

Traffic management services dealing with dynamic airspace management

We have observed that as the number of drone flights increase within a defined region (Node) that there is a threshold in which the average efficiency for each drone operation starts to decrease due to required conflict resolution actions.

There is a point where the operational manager of the airspace node will need decide on the capacity limit, and whilst it has always been possible to maintain safety in the simulations, as the efficiency drops there would be real world scenarios where a drone operation would no longer be feasible based on this drop in efficiency. It is anticipated that this limit will be dynamic based on the factors highlighted throughout the experiments, and that the performance requirements for drones entering the system may also be dynamic for a given period.

Noticeably, pre-flight strategic deconfliction proactively reduces the amount of tactical deconfliction actions, and therefore decreases the number of in-flight conflict resolutions. This increases the overall efficiency of the system's drone operations.

The original modelled algorithms required some adaptation due to edge cases where drones were taking unexpected deviations or even getting stuck as part of their tactical conflict resolution actions. The algorithmic changes primarily focused on adding additional strategic deconfliction recalculations where tactical resolution was required. It is noted that while the project has used a foundation technique for tactical conflict resolution there is still room for improvements to these algorithms to increase efficiency.

The weights used in the experiments have been tweaked, however are still on the more cautious side of things. It is possible that these could be decreased further over time as higher degrees of certainty and confidence is built in the data feeding the system, as well as those to services used to communicate with drone operators or the drones themselves.

Being able to replay and trace tactical conflict resolution activities has helped review and improve efficiency of the service.

We have observed CPU usage increased, as the number of drones being managed increased, more specifically the increase in drones, which were marked as "in conflict" and required conflict resolution were the main consumer of computational resources.

The resilience and reliability of the service itself is configurable, in these experiments we have specified this at a very high level which we believe would be required for when such a safety system running in production. However, provision of such a highly available solution has a cost implication. This reliability of the overall U-Space system, is also closely tied to each drone's separation requirements and therefore the overall airspace capability.

We have found that by using the discovery service model that it is possible to query other available U-Space services. However, for a run-time failure scenario, the integration/adapter must have already been configured. For example, while supplementary data service providers (e.g. for weather), have a consistent standard, we needed to have had integrated and tested it prior to being able to failover to it in the event of a failure of a primary provider. For more complex external services (e.g. strategic deconfliction) which another U-Space service may wish to utilise (e.g. tactical conflict resolution), due to the maturity of the standards and the variety of capabilities offered under such a title, heterogeneous integration has been found to be somewhat more complex.

4.1.2 Economic Feasibility

Through the mission and flight planning simulation the management processes, which derived from user requests to the FPM service, have been treated as single transactions (e.g. flight plan validation, update, cancellation, etc.). The benefit of considering flight planning management processes as particular transactions, and having them implemented as a set of modular microservices, is that it is possible to characterize typical user service requests.

Assessing the average processing time of these service request makes possible then to assign the consumption of resources to computational costs. In the analysis performed in this exercise, it has been demonstrated that the most resource demanding service requests can be identified and quantified. This is quite beneficial for economic aspects, as it could be assessed and estimated how the various flight planning service requests could be precisely (and potentially individually) billed based on the computational resources that they consumed in a cloud computing environment.

As initial extrapolation exercise, the costs of flight planning service requests have been estimated for a future projection scenario (2025) under the consideration of a punctual pay-per-use model. Although some assumptions were made in terms of the future number of drones in activity and their associated expected service requests, the implementation costs per microservice could have been accurately estimated, given that the pricing models of most of the commercial cloud computing platforms are well defined and available. Adding the remaining infrastructure costs (API gateway, service discovery) to the base flight planning microservices costs would result in an overall cost estimation.

Besides the resulting numbers from the exercise, the main conclusion from the performed analysis is that implementing the micro-service approach is both technically and economically feasible. Vendors can readily develop business models to support this type of product offering.

4.2 Recommendations

The outcomes from IMPETUS experiments focusing on the use of a microservice-based architecture have proven that this software architectural framework is suitable to be integrated in the information exchange processes in a complex and congested environment (considering the number of operations, data streams of a variety of natures and flexibility in demand). However, to integrate this framework into the U-space system the key axis in the success of this architecture is closely linked to its default configuration (promoting its scalability and flexibility) and the available resources to be capable of "growing". For this reason, the following recommendations shall be considered:

- **Standardization**: IMPETUS project has initially analysed the number and nature of the different inputs to be considered by the tested services in a U-space environment. It is important to clearly address, from these data types, the specific attributes that must be part of this information processes and provide a standard format that can be spitted into simple pieces. This will allow the use of different queues and the targeting into the messages of interest, avoiding bottlenecks in the data entry and, on the other hand, will allow the interconnection to other services, expanding their functionalities and completing the U-space service schema.

- **Atomicity**: split the functions involved in the information exchange processes presents clear advantages. The proposed microservices approach allows splitting the information services in simple functions, focused on a specific calculation that will improve the performance of the system, as it is easier to process data and create outputs using a target approach.
- **Maximum allowed workload**: configuring the architecture so the maximum workload admitted by each microservice is the 80% of its capacity allows that, when this limit is reached, another microservice is deployed. In practice, it has been proven that peak demands will increase this workload to more than 100%, increasing the likelihood of a failure and causing delays in the information process. The mechanisms to prevent this shall include a more conservative limitation in the workload (50% is an acceptable threshold as it is assumed that peak workloads will make it work beyond this limit) but also including different barriers to balance it (backup and duplicated services).
- **Smart filtering**: prior to include the datasets in the information layer and as part of the standard, mechanisms to detect the nature of the message shall be included in the platform in a parallelized way, avoiding the sequential processing of messages. This will ensure that small pieces of information are available to the proper services, facilitating the exchange of information and improving the performance of the microservice environment.
- **Update rate**: the update rate of identified drones, when possible, should be shown in the HMI of the information service client, allowing the user to have a clear idea of the environment in which they are operating. This rate shall also be increased when detecting a certain proximity between operations (e.g. defining a set of emergency levels and updating more often when they are closer).
- **Duplicity in modules**: the default configuration shall consist on duplicated services for those considered critical and, when the demand is increased and it is necessary to count on more resources, it should always be ensured that there will be a backup microservice apart from the ones that are being executed. This will allow to enter easily in failure mode and always have a countermeasure against issues.
- **Use of resources**: It is hardly recommended, once more U-space services are included in a final information management platform, to perform different stress tests using different sets of resources, to assess the scalability and flexibility of the final solutions, always considering that it is even expected to grow in the near future.
- **Data exchange**: a more comprehensive information exchange is recommended for the Flight Planning Management Service than the one used in the IMPETUS exercise dealing with detailed assessment of the flight plan data exchange between the mission and flight planning management services. The provision of all approved traffic information could reduce considerably the number of conflicts detected by a centralized flight planning management instance. This would imply however that an agreement has to be found at first about the extent of flight planning information that the centralized system could provide to the various MPM service providers.
- **Service development testing**: not only for the testing of critical services that have a high workload, but also for services that shall interact with different types of users it is relevant to test the provision of reliable service request responses. Through the development of the

service prototype in IMPETUS exercises, dedicated tools (e.g. *Postman*) have proven to be very useful for testing and monitoring the response of API requests. Especially when it comes to test instances in a cloud environment and to monitor resources across multiple regions, this type of tools can facilitate the assessment of the performance of every implemented microservice. It is recommended also contemplate this type of tests along the development process, as it is usually the case when developing microservices in an agile environment.

5 References

[1] IMPETUS, „D2.1 Drone Information User's Requirements," 00.01.00, 2018.

[2] IMPETUS, „D2.2 Drone Information Services," 00.01.00, 2018.

[3] IMPETUS, „D5.1 Experimental Plan," 00.01.00, 2018.

[4] IMPETUS, „D3.1 IMPETUS Architecture and Technical Requirements," 00.01.00, 2019.

[5] SESAR Joint Undertaking, „European ATM Master Plan: Roadmap for the safe integration of drones into all classes of airspce," Publications Office of the European Union, Luxembourg, 2017.

[6] SESAR Joint Undertaking, „Project Execution Guidelines for SESAR 2020 Exploratory Research," 1.00.00, 2016.

[7] National Oceanic and Atmospheric Administration, „Global Forecast System," [Online]. Available: https://www.ncdc.noaa.gov/data-access/model-data/model-datasets/global-forcast-system-gfs. [Zugriff am 17 12 2019].

[8] National Center for Atmospheric Research, „Weather Research and Forecasting Model," [Online]. Available: https://www.mmm.ucar.edu/weather-research-and-forecasting-model. [Zugriff am 17 12 2019].

[9] SESAR Joint Undertaking, „European Drones Outlook Study," 2016.

[10] Microsoft Azure, „Pricing calculator," [Online]. Available: https://azure.microsoft.com/en-us/pricing/calculator/. [Zugriff am 17 12 2019].

Appendix A Objectives and success criteria to validate the tested services

Validation Objective Title	Validation Objective description	Success Criterion title	Success Criterion description	Services/capabilities addressed
WX-1: Quantification of drone trajectory prediction uncertainty	Quantitatively measure how the uncertainty associated with the forecast of atmospheric conditions relevant to drone trajectory prediction (pressure, temperature and wind) translates into trajectory prediction uncertainty in terms of 3D position, timing and fuel/energy consumed.	WX-1 SC1	Predicted along-track position, timing and fuel/energy consumed as computed with the different weather scenarios exhibit significant differences (e.g. above 5%) so to be operationally relevant.	Weather information
		WX-1 SC2	Sensitivity of the uncertainty in predicted along-track position, timing and fuel/energy consumed to the prediction horizon is significant (e.g., differences above 5% appear for prediction horizons within 2 hours).	Weather information
WX-2: Quantification of the impact of weather forecast uncertainty in mission planning	Quantitatively measure how trajectory prediction uncertainty caused by the uncertainty associated with the weather forecast, used at mission planning time, impacts mission effectiveness and efficiency.	WX-2 SC1	Optimal orientation of the survey pattern as computed with the different weather scenarios exhibits significant differences (e.g. above 5%) so to be operationally relevant from the standpoint of mission efficiency.	Weather information
		WX-2 SC2	Optimal sequencing of UAVs over the survey pattern as computed with the different weather scenarios exhibits significant differences so to be operationally relevant in terms of mission effectiveness and/or efficiency. For instance, the optimal sequencing as computed with the deterministic weather forecast proves ineffective (e.g. does not meet the required image refresh rate) or inefficient (e.g. exceeds the required image refresh rate).	Weather information
WX-3: Quantification of the impact of weather forecast uncertainty in flight planning	Quantitatively measure how trajectory prediction uncertainty caused by the uncertainty associated with the weather forecast, used at flight planning time, affects the feasibility of the flight plan – and therefore safety, in both nominal and off-nominal conditions.	WX-3 SC1	Predicted fuel/energy consumed as computed with the different weather scenarios exhibits significant differences so to be relevant to ensure feasibility of the flight plan produced. E.g., Significant differences between forecasted and actual winds make the planned nominal trajectory unfeasible because of fuel/energy shortage.	Weather information
		WX-3 SC2	Predicted glide slope of the UAV attempting an emergency landing in engine-inoperative condition as computed with the different weather scenarios exhibits significant differences so to compromise reachability of the target emergency landing spot.	Weather information

Validation Objective Title	Validation Objective description	Success Criterion title	Success Criterion description	Services/capabilities addressed
WX-4: Quantification of the impact of weather forecast uncertainty in traffic management	Quantitatively measure how trajectory prediction uncertainty caused by the uncertainty associated with the weather forecast, used at traffic planning time (pre-tactical timeframe) impacts safe traffic separation.	WX-4 SC1	Plane trespassing window as computed with the different weather scenarios exhibits significant differences so to compromise separation between the UAVs performing the mission and the AVs entering/exiting the airfield through it.	Weather information
FPM-1: Enhanced Mission Planning	Demonstrate that an enhanced mission planning capability can improve the overall level of safety of the approved missions as opposed to standard mission planning.	FPM-1 SC1	80% success rate in planning trajectories which avoid areas marked by standard NOTAMS and drone related NOTAMS.	Mission Plan management
		FPM-1 SC2	60% success rate in planning trajectories which avoid areas of adverse weather and dangerous winds.	Mission Plan management
FPM-2: Weather-based Trajectory Deconfliction	Demonstrate that the stability of using strategic deconfliction based on trajectory models is not significantly affected by the uncertainty of predicted weather conditions and inaccuracies in local measurements.	FPM-2 SC1	Less than 10% of the results of strategic deconfliction solutions which utilize weather models deviate by more than +/- 5% from the total trajectory length and flight time from those that do not.	Flight planning management
		FPM-2 SC2	The resolution certainty of weather-based strategic deconfliction solutions is greater than 95%.	Flight planning management
FPM-3: Aeronautical Information-based Trajectory Deconfliction	Demonstrate that the strategic deconfliction based on trajectory models can provide feasible alternatives in the presence of large airspace sectorisation.	FPM-3 SC1	Less than 50% of the results of strategic deconfliction solutions which utilize specific aeronautical information deviate by more than +/- 10% from the total trajectory length and flight time from those that do not.	Flight planning management
		FPM-3 SC2	The resolution certainty of non-permanent airspace sectorisation information-based strategic deconfliction is greater than 95%.	Flight planning management
MTI-1: Situational Awareness Provision	Quantitatively measure how many alerts about dangerous traffic situations are received by users in an appropriate amount of time	MTI-1 SC1	All dangerous situation alerts are received by the end users with less than 5 second latency.	Traffic information
		MTI-1 SC2	99% of created alert messages are received by the end user (the Alarm Microservice).	Traffic information
MTI-2: Data fusion	Demonstrate that a micro-service based architecture has enough capacity to process the huge amount of inputs expected in congested environments	MTI-2 SC1	Considering increases in processing times due to overloads, the position message shall be acquired and integrated in the information flow of the platform in a time frame of less than 0.1s.	Monitoring
		MTI-2 SC2	Considering increases in processing times due to overloads, the position message shall be acquired and shown to the operator in a time frame of less than 1s.	Monitoring

Validation Objective Title	Validation Objective description	Success Criterion title	Success Criterion description	Services/capabilities addressed
		MTI-2 SC3	In congested airspaces , the alerts due to separation incidents shall be provided to users in less than 1s.	Monitoring
TMS-1: Dynamic separation criteria	Verify the applicability of dynamic separation criteria based on drone performances, which adapt the size and shapes of the safety buffers around drones.	TMS-1 SC1	The Traffic Management services can manage an area of airspace with fixed dimensions using the dynamic separation criteria without having any loss of separation incident.	Dynamic Capacity Management
		TMS-1 SC2	The Traffic Management services can manage an area of airspace with fixed dimensions using the dynamic separation criteria without causing any hindrance to the completion of any mission objective.	Dynamic Capacity Management
		TMS-1 SC3	All flight plans approved by the Traffic management services coincide with airspace that the drone, Drone Operator and drone pilot are authorised to operate in.	Dynamic Capacity Management
		TMS-1 SC4	The Traffic Management Services do not instruct any manoeuvre which the drone cannot safely perform.	Dynamic Capacity Management
TMS-2: Dynamic rerouting	Demonstrate performance-based rerouting of drones impacted by new airspace restrictions and other airspace users.	TMS-2 SC1	The Traffic Management Services are able to correctly identify every drone flight to which new airspace restrictions apply.	Dynamic Capacity Management
		TMS-2 SC2	The Traffic Management Services are able to correctly interpret the mission criteria of every drone flight to which new airspace restrictions apply.	Dynamic Capacity Management
		TMS-2 SC3	The Traffic Management Services are able to reroute every drone flight to which new airspace restrictions apply without causing any hindrance to the completion of any mission objective, which can still be completed under the new airspace restrictions.	Dynamic Capacity Management
		TMS-2 SC4	The Traffic Management Services are able to reroute every drone flight to which new airspace restrictions apply without adversely affecting their operational task/failure to carry out the task.	Dynamic Capacity Management
TMS-3: Data recording	Demonstrate that all data applicable to post-operational analysis is recorded.	TMS-3 SC1	All traffic management events are replayed in post-ops.	Dynamic Capacity Management
		TMS-3 SC2	All filed flight plan information is accessible.	Dynamic Capacity Management

Validation Objective Title	Validation description	Objective	Success Criterion title	Success Criterion description	Services/capabilities addressed
			TMS-3 SC3	All third party information related to the filing of a flight plan is accessible.	Dynamic Capacity Management

Appendix B Objectives and success criteria to validate the architecture

Validation Objective Title	Validation Objective description	Success Criterion title	Success Criterion description	Services/capabilities addressed
SFM-1: Robust Weather service data management	Demonstrate capability to correctly manage and supervise data flows even in the presence of misbehaving actors.	SFM-1 SC1	Ideal scenario should have over 99% of clients correctly served in proposed architecture.	Weather information
		SFM-1 SC2	Normal scenario should have over 95% of clients correctly served in proposed architecture.	Weather information
		SFM-1 SC3	High scenario should have over 95% of clients correctly served in proposed architecture.	Weather information
		SFM-1 SC4	Very High scenario should have over 90% of clients correctly served in proposed architecture.	Weather information
		SFM-1 SC5	Extreme scenario should have over 90% of clients correctly served in proposed architecture.	Weather information
		SFM-1 SC6	In all scenarios, the proposed architecture should be more resilient then the monolithic approach.	Weather information
SFM-2: Service availability due to failure module implementation	Assure a high service availability by a self-monitoring failure detection module to identify and replace failed services in the architecture.	SFM-2 SC1	In critical services, the delay of information due to failures in the execution of a service shall be less than 2s. (ID)	Traffic information
		SFM-2 SC2	A platform focused on information management for U-space shall ensure that, in case of a service failure, a backup microservice for critical services shall be deployed in less than 1s. (RT)	Traffic information
		SFM-2 SC3	A platform focused on information management for U-space shall ensure that the failures in services must be detected within 0.1s since the issue happened. (FD)	Traffic information
		SFM-2 SC4	A platform focused on information management for U-space shall be capable of dealing with increases in demand creating, at least, an additional instance of the same functionality to load the balance and prevent its overload (IC).	Traffic information
		SFM-2 SC5	The maximum workload of a service shall be less than 80% of its capacity, providing that a new instance will be created when overloads above this percentage are detected.	Traffic information
SFM-3: Robust micro-service architecture	To test a failure in any services that we are dependent on, as well as	SFM-3 SC1	Ability of the system to switch provider if a service Tactical Deconfliction is dependent upon fails, or increase separation to the maximum	Dynamic Capacity Management

Validation Objective Title	Validation Objective description	Success Criterion title	Success Criterion description	Services/capabilities addressed
against different failure modes	how failures are handled in components of the deconfliction service		level while it recovers. Both approaches ensuing that separation is maintained.	
		SFM-3 SC2	Ability to recover from an internal service failure in line in 10 seconds.	Dynamic Capacity Management
DM-1: Robust flight plan data exchange	Demonstrate that all flight plan data requests are processed by the FPM service and all the data is correctly transmitted	DM-1 SC1	99% success rate in the data exchange between the MPM and FPM services (RP) (FPM)	Flight Plan management
		DM-1 SC2	The message queue delivers over 99% of transmitted requests for all the scenarios considered.	Flight Plan management
DM-2: Consistent and available flight plan data	Demonstrate that the flight plan data samples remain consistent and available through the complete validation process	DM-2 SC1	No duplicate flight plan records exist in the database	Flight planning management
		DM-2 SC2	Uptime of the service is at least 99%	Flight planning management
DM-3: Equitable distribution of data management	Quantitatively measure the computing time distribution required for the mission planning and validation process	DM-4 SC1	The Mission Planning Management service has a ratio of at least 50% in the complete processing time (RRD)	Flight planning management
DM-4: Data capacity and performance in traffic information service	Demonstrate that a micro-service based architecture has enough capacity to process the huge amount of inputs expected in congested environments.	DM-4 SC1	The platform shall be able to process, at least, 99% of the messages collected. (MP)	Traffic information
		DM-4 SC2	The maximum delay that can be admitted by a U-space information management system shall be 5s when considering traffic information service (processing the data, integrate it with the rest of the operations and provide the view to the users). (MD)	Traffic information
DM-5: Data consistency in tactical deconfliction service	Ensure data set is current when internal service decisions are made from it.	DM-5 SC1	An up to date situational picture at all times for the area being monitored.	Dynamic Capacity Management
S-1: Scalability of number of users for the weather service in dense utilization areas	Sufficient RAM per server	S-1 SC1	At least 500 users served concurrently with 256 GB installed memory.	Weather information
	Sufficient CPU per server	S-1 SC2	At least 500 users served concurrently with two Xeon Gold processors (56 concurrent threads total).	Weather information
	Sufficiency of bandwidth	S-1 SC3	At least 500 users served concurrently with 1 gigabit/s of bandwidth.	Weather information

Validation Objective Title	Validation Objective description	Success Criterion title	Success Criterion description	Services/capabilities addressed
S-2: Self-management of the system in response to increases in demand	Creation of microservices related to a certain U-space services due to increases in demand	S-2 SC1	The Traffic Management Services are able to correctly identify every drone flight to which new airspace restrictions apply.	Traffic information
		S-2 SC2	The Traffic Management Services are able to correctly interpret the mission criteria of every drone flight to which new airspace restrictions apply.	Traffic information
		S-2 SC3	The Traffic Management Services are able to reroute every drone flight to which new airspace restrictions apply without causing any hindrance to the completion of any mission objective, which can still be completed under the new airspace restrictions.	Traffic information
	Instantiation of a backup microservice instead of one that has failed due to overload.	S-2 SC4	The failure must be detected within 0.1s since the issue happened. (FD)	Traffic information
		S-2 SC5	There will be at least 1 extra microservice instead of the one that has failed. (IC)	Traffic information
		S-2 SC6	Backup microservices shall be deployed within 3s since the failure was detected. (ST)	Traffic information
S-3: Automatic scaling of the cloud system microservices	To ensure the system automatically scales to manage the UAV load within the maximum possible capacity of the test area.	S-3 SC1	Ensure the system is never constrained by processing load of UAVs being managed but instead only by the environmental conditions being faced.	Dynamic Capacity Management
		S-3 SC2	Ensure the system is not over provisioned (to ensure commercial efficiency).	Dynamic Capacity Management
F-1: Self-management of microservices	Address the processes of self-management in a microservice architecture to respond to fluctuations in demand.	F-1 SC1	The maximum workload of a microservice shall be less than 80% of its capacity, to avoid peak increases that will lead it to a failure. (ML)	Traffic information
		F-1 SC2	Once the workload of a microservice reaches its maximum rate, another instance of the service shall be created. (IC).	Traffic information
		F-1 SC3	In off-peak periods, the number of microservices shall decrease in, at least, one instance. (IR)	Traffic information

Appendix C Assumptions for the exercises

A detailed list of assumptions within the scope of the experimental purposes of the exercises is given in the following table.

EXP-1-01	On-board sensing capability	CNS technologies	The navigation capability on-board the UAVs permits sensing relevant meteorological information periodically.	Utilization of different weather scenarios.	Planning	High
EXP-1-02	Communication and information management	CNS technologies	Communication and information management enables to downlink atmospheric observations	Solution needed for the local-scale weather service	Planning	High
EXP-1-03	Ground-based facility	Service pre-requisites	The Local-scale Weather service runs in a ground-based facility with enough computational processing capability	Required for processing the weather forecast model	Planning	High
EXP-1-04	Coverage of geographical domain	Service pre-requisites	Tiling scheme in place ensuring that Weather service covers the geographical domain	Coverage of an instance of weather model is limited	Planning	Medium
EXP-1-05	Internet access for Weather service	Service pre-requisites	The Weather service can access the Internet to periodically retrieve the global/mesoscale weather forecast	Global/mesoscale weather forecast required	Planning	Medium
EXP-1-06	Access to local weather observations	Service pre-requisites	The Weather service can access the local weather observations periodically	Local weather observations required	Planning	Medium
EXP-1-07	Invariance of local-scale weather service	Operational environment	The local-scale weather service is agnostic to the operational paradigm	Mission, flight and traffic planning can benefit from weather forecasts	Planning	High
EXP-1-08	Geographical context	Operational environment	The context can be either rural or suburban with moderate terrain roughness or the operation should occur at enough altitude over terrain and obstacles	Avoid terrain boundary layer effects that state-of-the-art local-scale models do not capture well	Planning	High
EXP-1-09	Link to geospatial information	Existing U-space services	The Local-scale Weather service relies on geospatial information	Terrain elevation of particular interest	Planning	Medium

EXP-2-01	DTM provider and Orchestrator contract	CNS technologies	The contract between the DTM provider and the Orchestrator is in place and working properly	Needed for a continuous information exchange	Planning	High
EXP-2-02	Local weather sensors	Service pre-requisites	At least 2 local weather sensors are available for filling the weather output model	Required for consistent local weather observations	Planning	High
EXP-2-03	Interaction with ATC	Service pre-requisites	Clearance from ATC has been granted for the operation	No examination of ATC interaction	Planning	Medium
EXP-2-04	Alternate flight plans	Service pre-requisites	Alternate flight plans are not part of the flight planning validation process	No examination of contingency management	Planning	Medium
EXP-2-05	Aeronautical Information	Operational environment	There is no expected conflict between the area of operations and permanent airspace sectorisation	Only examination of non-permanent airspace sectorisation	Planning	Medium
EXP-2-06	Separation minima	Operational environment	50ft horizontal and 50ft vertical SM is selected for the strategic deconfliction functionality	Alignment with other SESAR project	Planning	High
EXP-2-07	Link to registration and identification	Existing U-space services	Registration and identification of the Drone Operator take place successfully	No examination of these services	Planning	Low
EXP-2-08	Link to data recording services	Existing U-space services	Submitted flight plans are available for Data Recording services	Availably of flight plans	Planning	Low
EXP-2-09	Mission Approval	Regulatory framework	The purpose of the mission has been approved by the Regulating Authority	No examination of interaction with Regulating Authority	Planning	Low
EXP-2-10	Request transmission	Implementation architecture	The FPM service requests are transmitted appropriately through a centralized API Gateway	No examination of the centralized API Gateway	Planning	Medium
EXP-3-01	Standard communications protocol	CNS technologies	There is a standard communications protocol to interact with the Orchestrator	Ensure that the information is sent with enough quality and little delay	Planning/ Execution	High

EXP-3-02	Visual detection sensors	CNS technologies	The visual detection algorithms and sensors can detect an obstacle (2m min)	Dependent of utilized drone system	Planning/ Execution	Medium
EXP-3-03	GSM coverage	CNS technologies	GSM coverage is available	Needed for the planned scenario	Planning/ Execution	High
EXP-3-04	Contract with U-Space Service Provider	Service pre-requisites	Contract includes Tracking services for the implemented sensors and antennas	Needed for the planned scenario	Planning/ Execution	High
EXP-3-05	Risk environment	Operational environment	Low-risk environment with low population density in non-controlled airspace in VFR/ VLOS conditions	In alignment with planned scenario	Planning/ Execution	High
EXP-3-06	Detection algorithms	Operational environment	Detection algorithms for obstacles have been properly tested and validated	Ensure system performance	Planning/ Execution	High
EXP-3-07	Simulation of real operations	Operational environment	Simulation of manned and unmanned aviation reflects real operations	Data is consistent with standard drone trajectories	Planning/ Execution	Medium
EXP-3-08	Link to U1 and U2 services	Existing U-space services	All U1 U-Space services and Flight Planning Management, Strategic de-confliction and tracking services from U2 are implemented and working properly	Needed for the planned scenario	Planning/ Execution	Medium
EXP-3-09	Approval of flight plans	Existing U-space services	Submitted flight plans have been checked and approved by the Orchestrator	No examination of flight planning services	Planning/ Execution	Medium
EXP-3-10	Airspace	Regulatory framework	This experiment will take place in amber airspace	Alignment with CORUS project	Planning/ Execution	Medium
EXP-3-11	Certification	Regulatory framework	Drone Operator, pilots and U-Space Service Provider are legally certified	Needed for planned scenario	Planning/ Execution	High
EXP-3-12	Regulation compliance	Regulatory framework	The operations to be performed shall comply with the current regulations in Spain	Needed for planned scenario	Planning/ Execution	High

EXP-3-13	Protection areas	Regulatory framework	Regulation considers protection areas defined in terms of drone capabilities	Needed for planned scenario	Planning/ Execution	
EXP-4-01	Registration and identification	Service pre-requisites	Registration and identification of the Drone Operator has taken place	GuardianUTM has access to the information	Planning/ Execution	High
EXP-4-02	Information services	Service pre-requisites	Altitude Angel Information Services will be utilised to provide input into the Traffic Management Services	Needed for planned scenario	Planning/ Execution	High
EXP-4-03	Separation minima	Operational environment	The SM which is used for the experiments is defined	Alignment with CORUS project	Planning/ Execution	High
EXP-4-04	Link to DCM and strategic deconfliction	Existing U-space services	DCM – and Strategic Deconfliction, has already undertaken	Needed for planned scenario	Planning/ Execution	Medium
EXP-4-05	Link to tactical deconfliction	Existing U-space services	Basic Tactical Deconfliction is not enough to allow missions to continue with efficiency and minimal disruptions	Needed for planned scenario	Planning/ Execution	High

CRIDA
I·D·i - ATM

BOEING

JEPPESEN

ALTITUDE
A·N·G·E·L

C-ASTRAL
AEROSPACE Ltd.

ineco

TECHNISCHE
UNIVERSITÄT
DARMSTADT