Road-, Air- and Water-based Future Internet Experimentation

Project Acronym:	RAWFIE		
Contract Number:	645220		
Starting date:	Jan 1st 2015	Ending date:	Dec 31st, 2018

Deliverable Number and Title	D4.3 (a) - Pilot Experimentation Scenarios for Validation and Testing		
Confidentiality	PU	Deliverable type[1]	R
Deliverable File	RAWFIE - D4.3 (a) - Pilot Experimentation Scenarios for Validation and Testing	Date	31.05.2016
Approval Status[2]	WP leader, 1st Reviewer, 2nd Reviewer	Version	1.1
Contact Person	Ph. Dallemagne	Organization	CSEM
Phone		E-Mail	Philippe.Dallemagne@csem.ch

[1] Deliverable type: P(Prototype), R (Report), O (Other)
[2] Approval Status: WP leader, 1st Reviewer, 2nd Reviewer, Advisory Board

AUTHORS TABLE

Name	Company	E-Mail
Ph. Dallemagne	CSEM	Philippe.Dallemagne@csem.ch
Giovanni Tusa	IES	g.tusa@iessolutions.eu
Ezequiel Primo	IES	e.primo@i4es.it
Kakia Panagidi	UoA	kakiap@di.uoa.gr
Kostantinos Kolomvatsos	UoA	kostasks@di.uoa.gr
Miquel Cantero	ROBOTNIK	mcantero@robotnik.es
Marcel Heckel	FRAUNHOFER	Marcel.Heckel@ivi.fraunhofer.de
Vasil Kumanov	EPSILON	vasil.kumanov@epsilon-bulgaria.com
Kiriakos Georgouleas	HAI	GEORGOULEAS.Kiriakos@haicorp.com
Elias Kosmatopoulos	CERTH	kosmatop@iti.gr
Savvas Chatzchristofis	CERTH	savvash@gmail.com

REVIEWERS TABLE

Name	Company	E-Mail
Kakia Panagidi	UoA	kakiap@di.uoa.gr
Savvas Chatzchristofis	CERTH	savvash@gmail.com

DISTRIBUTION

Name / Role	Company	Level of confidentiality[3]	Type of deliverable
All		PU	R

[3] Deliverable Distribution: PU (Public, can be distributed to everyone), CO (Confidential, for use by consortium members only), RE (Restricted, available to a group specified by the Project Advisory Board).

CHANGE HISTORY

Version	Date	Reason for Change	Pages/Sections Affected
0.7	30.06.2015	Version for review	N/A
0.8	05.07.2015	Version for final contributions	all
0.9	27.07.2015	Version for WP leader approval	all
1.0	31.07.2015	First document iteration	all
1.1	31.05.2016	Slightly amended version submitted to the commission, after the introduction, on page 9, of the *"Plans for addressing the reviewers' recommendations after the first review"*, in subsequent versions of the deliverable	N/A

Abstract:

This deliverable is based on the results of T4.1 for what concerns the definition of the testing and validation scenarios of the RAWFIE platform. It contains the definition of a set of test and validation scenarios that will be performed in WP6 as well as the definition of the metrics and the success criteria.

Keywords:

Tested platform component system verification validation tests scenarios end users metrics

Part II: Table of Contents

List of Figures

List of Tables

Foreword

The first version of D4.3 aims at presenting the plan and the approach that will be followed to perform and document the tests for verification and validation. The details of the test scenarios themselves will be refined during the development of the RAWFIE system and will be provided in the second release of this document.

Plans for addressing the reviewers' recommendations after the first review

Taking into account the iterative process adopted in the project, and therefore the fact that each deliverable type, and so the one reporting on *"Pilot Experimentation Scenarios for Validation and Testing"*, is submitted at regular intervals corresponding to the different cycles of requirements, design, verification and validation planning and implementation, in the next iterations of this deliverable the consortium will take the actions needed to follow the recommendations received after the 1st review.

In the following, we explain how RAWFIE partners have addressed, or intend to address the above-mentioned recommendations in the subsequent versions of this deliverable, D4.6 (M21) and D4.9 (M30).

The D4.3 document included the complete list of verification tests that were identified as relevant during the first cycle, at a very early stage of the project, to ensure an extensive component and system test campaign. After the first and second implementation rounds, some tests may prove unnecessary and should be deleted from subsequent versions of the document. As a consequence, in D4.6 and D4.9 the verification and validation tests described in Section 5 and Section 6, will be kept only if they relate to any specific requirement appearing respectively in D3.2 and then D3.3.

Updated or new requirements coming from WP3, will be in turn reflected in the functionalities described in the architecture and design deliverables (D4.4, D4.5, D4.7, D4.8). Tests related to functionalities that are not explicitly mentioned in those deliverables, will not be considered as well, or existing tests will be updated accordingly.

It should also be noted that, with the preparation of deliverable D6.1, the consortium took the opportunity to proceed with the update or the removal of all tests that were not applicable anymore, after the first implementation cycle was completed.

As recommended and already stated in the first release of D4.3 on page 21, the consortium will later define the success criteria for the evaluation of the platform, and present them in deliverables D4.6 and D4.9.

Future deliverables related to the *"Pilot Experimentation Scenarios for Validation and Testing"* will also contain a specific section, where the way the recommendations are addressed will be explained.

Part III: Executive Summary

This deliverable is based on the results of T4.1 for what concerns the definition of the testing and validation scenarios of the RAWFIE platform. It describes the test and validation methodology and it defines a set of test scenarios that will be performed in WP6 as well as the definition of the success criteria.

In D3.1, the end users have specified the RAWFIE requirements at all possible levels (component, system, etc.) and many categories (functional, non-functional, etc.). These requirements shall be met by the RAWFIE testbeds, with respect to their achievement or success criteria. Deliverable D4.3 also describes those criteria (metrics) used for the assessment of the achievement of RAWFIE with respect to the requirements. It defines the minimum set of requirements to be met by the testbed and specifies the scenarios that are sufficient to validate the testbed, with respect to requirement subsets.

The test and validation scenarios deal with the global features of the RAWFIE system. They cover the test and validation of the Open interface framework, the interoperability of different sets of entities (testbeds, UxV, etc.) and the management of the RAWFIE federation.

The test scenarios will be used during the system integration and testing, in particular of the different font-ends, middle-tier, non-functional services (e.g. storage) and the operational entities (e.g. UxV, testbeds, and environment). The validation covers the entire RAWFIE Federation life-cycle, but it focuses on the deployment and operation phases.

Verification takes place during the development (e.g. in the way of unit tests) and on completion of development (integration tests), before the system is delivered to the pilot users. The purpose of verification is to ensure that each component works as expected and RAWFIE prototype components are related correctly through all expected scenarios. The verification process also offers an opportunity to test RAWFIE under extreme conditions such as realistic volumes of data, to give an indication of theoretical performance and ensure that the system is scalable to a sufficient degree when it is deployed for the users.

In order to verify components, the Consortium has identified all components of the system and verification scenarios for each of them has been prepared. Verification needs to be carried out on each component by way of unit tests to be sure that the required functionality is achieved in the way that is expected, and on the whole system to ensure that it achieves the required functionality, performance and reliability.

Evaluation takes place once RAWFIE prototype has been deployed for the pilot users to assess how the system performs under live scenarios. Evaluation covers areas such as the usability of the user interfaces, the type, quantity and quality of the data provided and overall use and usability of the system. The system will be evaluated following the metrics defined in this document.

Part IV: Main Section

1 Introduction

1.1 Scope of D4.3

This deliverable specifies the verification and validation scenarios to be exercised on a RAWFIE test-bed and the success criteria used for the evaluation of its implementation. Validation Scenarios aim at checking if the system works as expected from the End Users point of view (System Validation). They can be refined and enhanced at a later stage in cooperation with WP6, and have to be strictly linked to the Use Cases defined within WP3. This document also prepares the approach for Components and Integrated Prototype Testing (System Verification) for Task 6.1 (e.g. functional and performance tests, and so on). Finally, it describes the Verification vs. Validation activities and approaches.

D4.3 should be used as input for the work on WP6, in particular:

- Task 6.1 Prototype Integration, Testing and customization
- Task 6.2 Evaluation and Platform Validation

The document covers:

- What needs to be tested (complete testbeds, subsystems, etc.);
- Who will test (users, stakeholders, RAWFIE partners, EAB, etc.);
- When and where the tests will be performed;
- How tests are performed (tools, means, metrics, criteria, etc.).

1.2 Abbreviations

Table 1: Abbreviations

Abbreviation	Meaning
AT	Aerial Testbed
AUV	Autonomous Underwater Vehicle
DoW	Description of Work
EDL	Experiment Description Language
MT	Maritime Testbed
MM	Monitoring Manager
RC	Resource Controller
TM	Testbed Manager
UAV	Unmanned Aerial Vehicle
UGV	Unmanned Ground Vehicle
USV	Unmanned Surface Vehicle
UxV	Unmanned aerial/ground/surface Vehicle
VT	Vehicular Testbed

2 Object of the validation and testing

The RAWFIE system is made of a set of sub-systems, components, processes, etc. and, thus, it should be thoroughly validated and tested. Only through an efficient verification and validation process, possible problems and malfunctions will be revealed and corrected in order to secure the efficient execution of the RAWFIE platform. A set of scenarios have been defined to verify the properties of the RAWFIE system during the development, to verify that the RAWFIE system and components comply with the specifications and to evaluate the degree of achievement with respect to the expected performance. The RAWFIE consortium aims to secure the efficient execution of the system in two axes: (a) the *verification* of the available components and the integrated system, (b) the *validation / evaluation* of the whole system.

The verification process aims at revealing potential problems. A set of template for describing components and system integration tests that must be passed (functional tests, performance tests, etc.) will be defined. Verification scenarios are adopted to verify that the platform and the single components (as implemented within WP5) properly meet the requirements from the technical perspective (system verification). The system validation and evaluation process aims to reveal if the system also meets the defined requirements and performs as expected from the end users perspective. Similarly to the verification process, the validation will be built on top of a set of templates for describing the validation scenarios. The establishment of the scenario descriptions and specifications is based on the analysis of the user requirements defined in D3.1 and the related metrics and expected performance (success criteria).

2.1 Verification

Verification takes place during the development (e.g., in the way of unit tests) and on completion of development (integration tests) before the system is delivered to the pilot users. The purpose of verification is to ensure that each component works as expected and RAWFIE prototype components are interacting correctly through all expected scenarios. The verification process also offers an opportunity to test RAWFIE under extreme conditions such as realistic volumes of data to give an indication of the theoretical performance and ensure that the system is scalable to a sufficient degree when deployed for the users. The aim is to answer questions related to if the developed components meet the initial requirements and if they are built in the right way. In order to verify the available components, the consortium has identified all components of the system and verification scenarios for each of them has been prepared. Verification needs to be carried out on each component by way of unit tests to be sure that the required functionality is achieved in the way that is expected, and on the whole system to ensure that it achieves the required functionality, performance and reliability. Verification will help to lower the number of defects in early as well as in late stages of development and lead to better understanding of the components. Finally, it will reduce the chances of failures in the software implementation.

2.2 Validation and evaluation

Validation and evaluation takes place once the RAWFIE prototype has been deployed for the pilots to assess how the system performs under live scenarios. Evaluation covers areas such as the usability of the user interfaces, type, quantity and quality of the data provided and the overall use and the usability of the system. The system will be evaluated adopting the metrics defined in this document. The discussed process will execute extensive evaluations in order to assess the overall effectiveness and efficiency of the RAWFIE solution and to prove its added-value in a real environment. The validation campaign will include formal tests of the RAWFIE platform against the requirements set, as well as against the use cases' objectives. Validation sessions and templates, based on requirements will take place, expecting to bring valuable information about general user acceptance and usability of the provided infrastructure. Performance or other technical issues will be thoroughly evaluated. The activity will conclude with the preparation of a report summarizing the system evaluation and providing an assessment of its readiness for operational use.

2.3 RAWFIE federation lifecycle

The RAWFIE federation lifecycle will be tested through specific scenarios that 'see' the framework as a black box. The aim is to identify if the system works appropriately through a high level evaluation. At first, the tests will identify if a set of different testbeds are smoothly attached to the RAWFIE architecture. The test scenarios will define the type, the number and the location of the testbeds. Accordingly, a specific EDL script will be defined that covers the entire set of the available components and testbeds. For instance, the script will define requirements for the parallel execution of different types of testbeds in the same experiment. In combination with the stress tests, the specific approach is judged very efficient as it will identify possible problems in the RAWFIE architecture. In general, the federation lifecycle will be evaluated through a number of major phases that include: user and testbed registration, authoring, booking, launching and evaluation of an experiment. In the upcoming sections, a set of validation scenarios are provided that cover all the discussed phases accompanied by a set of metrics that will reveal the performance of the framework.

2.4 Verification and validation infrastructure and procedures

Verification will ensure that RAWFIE components meet the defined requirements while the validation phase will check if the system meets the high level requirements as defined by the consortium. Requirements are verified and the implemented components and the system are evaluated against the defined requirements. In addition, the validation process will ensure that all requirements are adequately tested or demonstrated, and that test results are as expected and can be repeated to verify correct implementation of the RAWFIE components. The consortium will follow a specific plan that follows these guidelines and it will help to ensure that the provided components can consistently meet a high level of quality and performance requirements. In short, the verification and the validation plans are as follows:

- Verification plan. For each component and sub-components the tests will manage to reveal their performance. Specific objectives will be defined for each (sub-) component and a detailed description of the verification scenario will be provided. Moreover, pre-requisites and the expected results will undertake the role of identifying if the component meets the defined requirements. Finally, specific testing scenarios could be devoted to identify the appropriate communication between components in order to secure the efficient data transfer throughout the RAWFIE architecture. The discussed plan will be realized during the implementation process in order to identify possible problems early in the development process.

- Validation plan. A set of validation scenarios will be adopted to reveal the performance of the platform. These scenarios mainly focus on testing from the stakeholder's point of view. Hence, in each scenario the main stakeholders will be defined and a detailed description will elaborate on the adopted steps. In addition, the involved (sub-) components will be referred in order to have a view on the part of the RAWFIE architecture that is evaluated. It should be noted that these scenarios will be evaluated against the already defined requirements.

2.4.1 Non regression and stress tests

The aim of non-regression and stress tests is to identify possible errors in the RAWFIE architecture. These errors could be caused by a number of issues like wrong interfaces design and / or implementation, insufficient data passed to / from each component and so on.

Non regression tests will be realized on the RAWFIE prototype. As it is very difficult to have a large set of UxVs during validation, specific routines will undertake the role of producing data related to UxVs behaviour (e.g., location, measurements, status of resources). Hence, the consortium will be capable of performing large scale validation producing large amounts of data in high rates. The discussed routines will be launched / combined with the prototype and will represent the behaviour of RAWFIE nodes / testbeds. A post-processing tool will undertake the responsibility of analysing the derived behaviour of the system based on a set of metrics. For instance, the number of errors, the data transferred, the time required to complete an 'action' and so on are some useful metrics that could be adopted to measure the performance of the system. In addition, the consortium will adopt an approach that will take into consideration the 'footprint' of each test. This means that every validation scenario will be combined with a specific 'view' of the system. For instance, specific tests will be realized either from the experimenter point of view or from the testbed perspective. In other words, the 'footprint' will combine each test with what is tested (i.e., RAWFIE architecture). Finally, specific reports will be realized to describe the outcome of the process.

Based on the aforementioned routines, the consortium will have the opportunity to provide extensive tests in order to reveal the performance of the platform. The aim is to bring the framework close to its limits. Fails and means for fast recovering will be realized leading to a high quality system. Stress tests will be realized in the following axes: (a) high number of users (b) high number of bookings, (c) high number of concurrent connections to the system, (d) high number of testbeds / nodes, (e) high load, (f) unpredictable events like taking a testbed / node or the DBMS offline and restarting it, etc. These tests focus on unpredictable events randomly generated during the framework execution and put emphasis on robustness, availability, and error handling under a heavy load, rather than on what would be considered correct behaviour under normal circumstances.

2.4.2 EDL Testing

The EDL testing is a special process in the verification – validation process. The reason is that EDL tests should reveal the efficiency of the system when communicating with experimenters not only through the provided functionality perspective but also through the easiness that an experimenter can create, compile and run an experiment. The aim of the EDL testing is to reveal if the scenario defined by the experimenter is smoothly processed and produces the appropriate outcomes to be adopted by the remaining RAWFIE components. Specific tests will be realized concerning important characteristics of the EDL as well as the functionalities provided by the editors. For instance, the testing process will involve two aspects: (a) the experimenter side and

(b) the components side. From the experimenter point of view, the provided editors and their functionalities should be easily initiated and commands (i.e., EDL scripts) should be efficiently translated based on the underlying EDL model. In RAWFIE, experimenters that create an experiment will need to provide a short high level description of the experiment and its purpose. The second aspect involves the definition of specific commands in the test script that will reveal if the RAWFIE components are smoothly combined. This will also test the connection between components in order to have an efficient execution of the experiment.

The test scenarios will be realized based on the defined use cases and reveal if an experimenter is capable of easily define an experiment in the EDL terms. For instance, with test scenarios, critical questions will be answered like: Can the experiment easily define the application logic of his/her experiment? How easily the experimenter can define an experiment that realizes a complex algorithm? Moreover, the test scenarios will check if the EDL script is efficiently translated based on the underlying model and, accordingly, be compiled and validated. Syntactic and semantic errors will be incorporated in the test scenarios in order to reveal if the system is capable of identifying the errors and return specific messages to the experimenter. Successful fulfilment of the compilation and the script validation process will be realized through a number of files / models assigned to specific RAWFIE components. These files / models are necessary to, finally, execute the experiment.

3 Stakeholders and actors

The end users are not the only stakeholder type to be considered in the validation plan. Other stakeholders have been identified in D3.1. Those who are the main candidate for evaluating the appropriateness of the RAWFIE testbeds to support their requirements are:

- Experimenters:
    - Users who belong to the federation. They must be acknowledged by the federation partners
- RAWFIE Platform Administrator:
    - Administrator of RAWFIE middleware framework. Middleware is own by the RAWFIE consortium
- Testbed Operators:
    - Owners and managers of testbed facilities
- UxV Manufacturers:
    - Suppliers of UxVs resources

4 Metrics

4.1 Introduction

Once the system has been verified and deployed at the testbed sites, a period of evaluation will take place during which a number of metrics will be assessed either by quantifiable measurements or by way of questionnaires/interviews. Metrics can be classified to: required or beneficial, hard or soft.

Metrics are split in different categories. Some of the validation metrics are derived from the list of Requirements described in D3.1, as these metrics are strictly related to system and users' requirements:

- PLATFORM (e.g. PT-GEN, PT-P, PT-B, PT-A, PT-L, PT-NF from D3.1)
- TESTBED (contains metrics from TB-G,TB-NF-G and TB-NF-R from D3.1)
- INTERCONNECTIVITY (e.g. TB-I from D3.1)
- RESOURCES (e.g. TB-R from D3.1)
- STORAGE (e.g. TB-D from D3.1)

Other metrics are dedicated to VISUALISATION or USABILITY.

Examples of typical metrics used for the verification and evaluation of the RAWFIE system and components are listed below.

Testbed deployment

- Metric: Percentage of working nodes; Expected percentage.

Experiments

- Total number/percentage of experiments successfully deployed
- Total number/percentage of experiments stopped
- Total number or percentage of category of UxVs participating in the experiments
- Time needed per experiment
- Time needed per category of UxVs
- Total/ Average Energy consumption per network connection
- Network connection changes from one technology to another
- Percentage of "outliers" nodes.

Control and reporting

- Percentage of retested experiments
- Percentage of incidents per testbed/ per UxV category
- Issues not solved
- Percentage of message bus errors
- Time to solve incidents
- Time to identify an incident

Note that the Fed4FIRE D6.1 deliverable defines four groups of metrics (pp. 17--18):

- component-level metrics, which applies to both virtual and physical devices (computing, network or storage);
- network metrics, which qualify the static aspects of a network;
- traffic metrics, which capture the more dynamic aspects of what is happening.

4.2 Required/Beneficial Metrics

A **Required** metric is considered to be a metric which should be applied to the project and the results will have a direct effect on the success of RAWFIE, whereas a **Beneficial** metric is when any measured benefit may assist with the commercial exploitation of RAWFIE but not have a direct impact on its success.

For instance, it is required that the system is reliable but it can be beneficial if the system is more reliable than other existing systems on the market.

4.3 Hard/Soft Metrics

A metric is considered **hard** (quantitative) if the metric can be measured purely by the use of data and give a pure statistical result. **Soft** (qualitative) metrics are more subjective in that they rely on people’s opinions such as results from user questionnaires.

An example of a hard metric is the accuracy of measurements. If, i.e., the target is 5% of accuracy and the system is able to have more accuracy then the metric is perfectly met. On the other hand, a general evaluation about system usability is a soft metric if the system itself does not implement methods for quantitatively measure it.

4.4 User defined metrics

In the RAWFIE project, metrics are used in several phases; the first phase is the design and development phase, in which the metrics are used to establish its level of achievement with respect to the specification and the requirements. The second phase is the exercise of the RAWFIE system by several stakeholders (owners, experimenters, etc.) who will also use their own metrics for the evaluation of the RAWFIE system.

From the end user perspective, the evaluation of the exercised elements varies depending on many factors, such as the application, the environment, the participating entities, etc. The end user will therefore rely on specific metrics and criteria to evaluate the performance of the exercised elements.

Should this be required, its implementation could be based on data analytics framework that will be used in RAWFIE systems (e.g. Apache storm).

4.5 Metric template definition

We propose the adoption of a descriptive table in which a set of predefined information is reproduced. Moreover, a brief textual description of the metrics is given, together with the description of "how to validate the metrics and actions to be done" for refining the specification and validation process.

Table 2: Metrics template

Metric type/ ID/ Tag	**Description**	**Required/ Beneficial**	**Hard/ Soft**	**Mean for measurement**	**Validator stakeholder**

4.6 Success criteria

Success criteria are quantitative or qualitative values (or set or ranges of values) for relevant metrics, against which the actual characteristics or performance indicators of the system and components are compared. A typical criterion is a threshold against which the performance indicator of the tested element is compared (e.g. "the temperature of the motor shall not exceed 90°C during the experiment").

The success criteria are usually combined to perform the evaluation of a given element. For example, an element will be successfully evaluated if it meets the criteria A and B and C. Another element may be successfully evaluated if it meets the criteria B and C or F.

For any given metrics, the success criteria may vary depending on the components under evaluation, or on the experiment under execution. To this intent, a template is provided to specify criteria for any component or system to be evaluated.

Table 3: Success criteria template

Criteria#	Metrics#	Component# or system#	Success criteria	Comment
Example 1	2 (e.g. average communication throughput)	UxV Control communication system	More than 10 Mbit/s	The throughput of the link between the UxV and the Ground control must be 10 Mbit/s in average over the experiment duration

Note: the success criteria will be elaborated later in the project execution.

4.7 Platform metrics

Table 4: Platform metrics.

Metric type/ ID/ Tag	Description	Required/ Beneficial	Hard/ Soft	Mean for measurement	Validator stakeholder
PLATFORM / 1 / STABLE SYSTEM	Counts the system crashes, measures the uptime.	Required	Hard	System statistics	RAWFIE system manager, Testbed owner, Experimenter
PLATFORM / 2 / MEMORY CONSUMPTION	Records the memory consumption, during the execution of experiments	Required	Hard	System statistics: maximum amount of memory required during operation.	RAWFIE system manager
PLATFORM / 3 / ERRORS	Counts the system errors	Required	Hard	System statistics	RAWFIE system manager
PLATFORM / 4/ WARNINGS	Counts the system warnings	Required	Hard	System statistics	RAWFIE system manager, Testbed owner, Experimenter
PLATFORM / 5 / SCALABILITY	Counts the resources required per node, UxV, experiment, etc.	Required	Hard	System statistics	RAWFIE system manager, testbed owner
PLATFORM / 6/ USABILITY, USER-FRIENDLINESS	End-user evaluates the system in a scale 1-10 at questionnaires if the system is understandable and easy to be used.	Required	Soft		All
PLATFORM / 7 / RECOVERY TIME	Records the time needed to recover the system operations after shutdown	Required	Hard	System statistics	RAWFIE system manager, testbed owner

PLATFORM / 1 / STABILITY: System's stability and accuracy is ensured and tested. System is considered stable if it does not crashes frequently. System is perceived as unstable if it crashes once a day. The time needed to recover the system should be not more than within minutes.

PLATFORM / 2 / MEMORY CONSUMPTION: System memory consumption is supposed to be maintained at sufficiently low levels. During the execution of experiments, the memory consumption should be low in order to guarantee the efficiency and scalability of the system.

PLATFORM / 3 / ERRORS: System errors should be kept at low levels. In the ideal case, the errors should be minimum (zero).

PLATFORM / 4 / WARNINGS: System warnings should be kept at low levels. In the ideal case, the warnings should be minimum (zero).

PLATFORM / 5 / SCALABLITY: Capacity of the system to scale according to the number of the experiments to be executed simultaneously. The system is scalable if it is able to handle any number of concurrent experiments. The possibility to execute or not some experiments should only be related to the availability or not of the resources required to execute the experiments themselves, rather than to the system capacity to handle increasing loads. Some quantitative results will be provided as soon as stress tests of the system will be performed.

PLATFORM / 6 / USABILITY, USER-FRIENDLINESS: Usability is the interaction with the system is easy and intuitive. The system must be easy to use and understandable for the user. The user should easily interact with the system through the Web Portal. A more specific list of usability elements will be specified later in the project, once the first mock-up of the Web Portal and the different Frontend tools will be designed, and a first list of elements to be shown in the GUI will be identified.

PLATFORM / 7 / RECOVERY TIME: Time elapsed between the shutdown and resuming the operations. The time needed to recover the system should be not more than within minutes.

4.8 Testbed metrics

Table 5: Testbed metrics

METRIC TYPE/ID/TAG	Description	Required/ Beneficial	Hard /Soft	Mean for measurement	Validator stakeholder
TESTBED / 1 / PERCENTAGE OF WORKING NODES	Measures the fraction of working resources to total resources of the testbed	Required	hard	System statistics	RAWFIE system manager, testbed owner
TESTBED / 2 / TOTAL EXPERIMENTS	Counts the experiments applied to specific testbed	Required	hard	System monitoring	RAWFIE system manager
TESTBED / 3 / RUNNING EXPERIMENTS	Counts the running experiments	Required	hard	System monitoring	RAWFIE system manager
TESTBED / 4 / CANCELLED EXPERIMENTS	Counts the cancelled experiments	Required	hard	System monitoring	RAWFIE system manager
TESTBED / 5/ ERRORS	Counts the errorsoccured during the execution of the experimets	Required	hard	System and testbed monitoring	RAWFIE system manager, testbed owner
TESTBED / 6 / WEATHER CONDITIONS	Gets the weather conditions during day in a testbed	Required	hard	System and testbed monitoring	RAWFIE system manager, testbed owner

4.9 UxV metrics

Table 6: UxV metrics

Metric type/ ID/ Tag	Description	Required/ Beneficial	Hard/ Soft	Mean for measurement	Validator stakeholder
UXV / 1 / CONTROLLABILITY	Actual route vs. plan	Required	Hard	Statistics of the UxV collected during the experiment	UxV operator
UXV / 2 / MISSION ACHIEVEMENT		Required	Hard	Experiment statistics: rate of achieved vs. assigned objectives	Experimenter

4.10 Interconnectivity (aka. communication) metrics

The typical data communication metrics are: Packet loss (the ratio of packet that are definitively lost over the total number of packets sent), End-to-end delay (the total time it takes for a packet to reach its destination), Meeting of time constraints (e.g. response time, jitter, synchronization, deadline) and Throughput (effective bit per second that can be sent from one or several ends to one or several other ends). Other metrics are used to evaluate the performance of specific networks or specialized equipment, such as fairness in terms of transmission opportunity, resource usage, power consumption, uptime and reliability, scalability, etc.

Table 7: tests that will be performed on a given component

Metric type/ ID/ Tag	Description	Required/ Beneficial	Hard/ Soft	Mean for measurement	Validator stakeholder
INTERCONNECTIVTY / 1 / NETWORK THROUGHPUT	Data throughput available to the RAWFIE functions/ effective bit per second that can be sent from one or several ends to one or several other ends	Required	Hard	Network statistics	RAWFIE system manager, Testbed owner, UxV operator
INTERCONNECTIVTY / 2 / ERROR RATE	Raw and residual rates in terms of packets in error vs. the total number of packet sent or received, depending on the perspective.	Required	Hard	Statistics about raw and residual transmission error rate	UxV operator, testbed owner
INTERCONNECTIVTY / 3 / SIGNAL STRENGTH – LINK QUALITY	Assess the link quality	Required	Hard	Radio front end, typically the Received Signal Strength indicator. Ultimately more refined methods can estimate the link quality	UxV operator, testbed owner
INTERCONNECTIVTY / 4 / INTERFERERS	Assess the level of interferences in the wireless communication environment	Required	Soft	Number of interferers with the wireless communication within a given range.	Testbed owner
INTERCONNECTIVTY / 5 / NUMBER OF NODES	Number of participants in the network	Required	Hard	Network statistics	RAWFIE system manager, Testbed owner. UxV operator
INTERCONNECTIVTY / 6 / REACHABILITY	Distance at which the communication quality of service cannot be offered anymore. To be elaborated	Required	Hard	Application and network statisitics.	RAWFIE system manager, Testbed owner, UxV operator
INTERCONNECTIVTY / 7 / DISCONNECTIONS	Number of disconnections (per time unit, per experiment…)	Required	Hard	Application and network statisitics.	RAWFIE system manager, Testbed owner, UxV operator
INTERCONNECTIVTY / 8 / AUTHENTICATION RATES	Authentication metrics, such as the	Required	Hard	RAWFIE system and UxV communication	RAWFIE system

	number of successful and failed authentication attempts.			statistics	manager, Testbed owner. UxV operator
INTERCONNECTIVTY / 9 / RESIDUAL PACKET LOSS	the ratio of packet that are definitively lost over the total number of packets sent (see also Interconnectivity / 2)	Required	Hard	Network statistics	RAWFIE system manager, Testbed owner, UxV operator
INTERCONNECTIVTY / 10 / END-TO-END DELAY	the total time it takes for a packet to reach its destination	Required	Hard	Network statistics	RAWFIE system manager, Testbed owner, UxV operator
INTERCONNECTIVTY / 11 / MEETING OF TIME CONSTRAINTS	e.g. jitter, synchronization, deadlines, etc.	Required	Hard	Network statistics and application temporal statistics	RAWFIE system manager, Testbed owner

INTERCONNECTIVITY / 3 / SIGNAL STRENGTH – LINK QUALITY: Assess the link quality, based on a number of indicators, given by the radio front end, such as the Received Signal Strength indicator, or statistical information, consolidated on the wireless node or the RAWFIE system/testbed.

INTERCONNECTIVITY / 4 / INTERFERERS: Interferers are sources of perturbation. Their presence should be controlled and mitigated in testbeds, e.g. by limiting the presence of other electromagnetic sources or their transmission power. In addition, the experiments may require the simulation of specific interferences generated by the real application environment. In a basic approach, we may only account for interferers such as WIFI hotspots or other wireless communication sources within a range depending on the technologies, regulations, etc. For a finer assessment, specific equipment should be used to analyse the impact over the link quality in the wireless communication (see Link Quality, above). Testbed owners should consider this as a property of their testbed: it may be “interferer-clean”.

Note: the simulation of wireless communication interference should be considered as a function provided by the Testbed operator or injected by the experimenter or the UxV provider.

4.11 Resources metrics

Table 8: Metrics for resources

Metric type/ ID/ Tag	Description	Required/ Beneficial	Hard/ Soft	Mean for measurement	Validator stakeholder
RESOURCES / 1 / USED MEMORY	Counts used memory of UxV	Required	Hard	UxV node	RAWFIE system manager, UxV operator
RESOURCES / 2 / FREE MEMORY	Counts free memory	Required	Hard	UxV node	RAWFIE system manager, UxV operator
RESOURCES / 3 / USED STORAGE	Counts used storage	Required	Hard	UxV node	RAWFIE system manager, UxV operator
RESOURCES / 4 / FREE STORAGE	Counts remaining free storage of the device	Required	Hard	UxV node	RAWFIE system manager, UxV operator
RESOURCES / 5 / CPU UTILISATION	Measures CPU utilization	Required	Hard	UxV node	RAWFIE system manager, UxV operator
RESOURCES / 6 / CUMULATIVE CPU TIME	Counts CPU time per experiment	Required	Hard	UxV node	RAWFIE system manager, UxV operator
RESOURCES / 7 / CUMULATIVE CPU USAGE	Counts CPU time per experiment	Required	Hard	UxV node	RAWFIE system manager, UxV operator
RESOURCES / 8 / NUMBER OF PROCESSES	Counts the number of processes during UxV lifetime	Required	Hard	UxV node	RAWFIE system manager, UxV operator
RESOURCES / 9 / NUMBER OF RUNNING PROCESSES	Counts the number of processes per experiment	Required	Hard	UxV node	RAWFIE system manager, UxV operator
RESOURCES / 10 / BATTERY LIFETIME	Counts battery lifetime per experiment	Required	Hard	UxV node	RAWFIE system manager, UxV operator
RESOURCES / 11 / PACKET ARRIVAL RATE	Counts packet arrival rate per experiment	Required	Hard	UxV node	RAWFIE system manager, UxV operator
RESOURCES / 12 / DEVICE STATUS	Identifies if UxV is in normal condition or in an emergency situation	Required	Soft	UxV node	RAWFIE system manager, UxV operator

4.12 Storage metrics

Table 9: Metrics for storage

Metric type/ ID/ Tag	Description	Required/ Beneficial	Hard/ Soft	Mean for measurement	Validator stakeholder
STORAGE / 1 / EXPERIMENT DATA	The system shall be able to store experiment data in case of communication link failure between the testbed and the upstream components deployed in the cloud.	Required	Hard	Completeness and consistency of the stored data.	RAWFIE system manager, experimenter

STORAGE / 1 / EXPERIMENT DATA: Experiment data: the system shall be able to store experiment data in case of communication link failure between the testbed and the upstream components deployed in the cloud. If for some reason the communication between the testbed and upper layers components is interrupted, experiment data will be stored internally to the testbed, and be sent as soon as the communication is recovered.

4.13 Visualisation metrics

Table 10: Metrics for visualisation

Metric type/ ID/ Tag	Description	Required/ Beneficial	Hard/ Soft	Mean for measurement	Validator stakeholder
VISUALIZATION / 1 / BALANCE	End user measures the distribution of the optical weight (number of objects) in a picture via questionnaires	Required	Soft	Questionnaire	Experimenter
VISUALIZATION / 2/ SYMMETRY	Evaluation of symmetry at GUI	Required	Soft	Questionnaire	Experimenter
VISUALIZATION / 3 / SIMPLICITY	Experimenter evaluates if the objects appearing to the screen are the minimum needed	Required	Soft	Questionnaire	Experimenter
VISUALIZATION / 4 / DENSITY	Counts the percentage of screen is covered with objects.	Required	Soft	RAWFIE system	RAWFIE system
VISUALIZATION / 5 / CONSISTENCY	Experimenter evaluates if the similar actions lead to similar results and the elements in the GUI (fonts, patterns, tables) are similar to all pages	Required	Soft	Questionnaire	Experimenter
VISUALIZATION / 6/ UTILITY	Experimenter evaluates the utility of the tools in order to define, manage and execute an experiment.	Required	Soft	Questionnaire	Experimenter

VISUALIZATION / 1 / BALANCE: Balance can be defined as the distribution of the optical weight in a picture. The optical weight refers to the perception that some objects appear heavier than others. Larger objects are heavier, whereas small objects are lighter. Balance in screen design is achieved by providing an equal weight of screen elements, left and right, top and bottom.

VISUALIZATION / 2 / SYMMETRY: Symmetry is the extent to which the screen is symmetrical in three directions: vertical, horizontal, and diagonal.

VISUALIZATION / 3 / SIMPLICITY: Simplicity is directness and singleness of form, a combination of elements that results in ease in comprehending the meaning of a pattern. Simplicity in screen design is achieved by optimising the number of elements on a screen and minimising the alignment points. Tullis has derived a measure of screen complexity for text-based screens. It involves counting the number of different rows or columns on the screen that are used as starting positions of alphanumeric data items. Information theory is then used to calculate the complexity of this arrangement of starting positions.

VISUALIZATION / 4 / DENSITY: Density is the extent to which the screen is covered with objects. Density is achieved by minimizing screen density levels.

VISUALIZATION / 5 / CONSISTENCY: Wolf (1989) suggests that consistency means that similar user actions lead to similar results. Another definition is that a consistent user interface is one that maximizes the number of shared rules across tasks (Polson et al., 1986). Consistency within an application should facilitate human perception and cognitive processes such as visual scanning, learning, and remembering. This applies to spatial properties which include the organization of menus, placement of frequently used widgets, symmetry, and alignment of widgets. This also applies to fonts, colours, common actions, sequences, terms, units, layouts, typography and more within an application program.

VISUALIZATION / 6 / UTILITY: The GUI effectively and efficiently accomplishes the tasks for which it was designed. It is correct with respect to the functional objectives of the application.

4.14 Usability metrics

Table 11: Metrics for usability

Metric type/ ID/ Tag	Description	Required/ Beneficial	Hard/ Soft	Mean for measurement	Validator stakeholder
USABILITY / 1 / CONSISTENCY	Experimenter tests if same tasks lead to same output	Required	Soft	Pre-defined test-cases	Technical partners
USABILITY / 2/ GUIDANCE	Experimenter tests if help guidance or error messages appear in order to guide him/her to the right option	Required	Soft	Pre-defined test-cases	Technical partners
USABILITY /3 /EXPLICIT CONTROL	Experimenter tests if help guidance is personalised to the user based on the group that belongs or his reputation	Required	Soft	Pre-defined test-cases	Technical partners
USABILITY / 4 / ERROR MANAGEMENT	Experimenter can report an error to the technical team	Required	Soft	Pre-defined test-cases	Technical partners

USABILITY / 1 / CONSISTENCY: Certain aspects of an interface should behave in consistent ways at all times for all screens; terminology, icons, color etc. should be consistent between screens or within a screen.

USABILITY / 2 / GUIDANCE: Guidance refers to the means available to advise, orient, inform, instruct, and guide the users throughout their interactions with a computer (messages, alarms, labels, etc.), including from a lexical point of view.

USABILITY / 3 / EXPLICIT CONTROL: Explicit Control concerns both the system processing of explicit user actions, and the control users have on the processing of their actions by the system.

USABILITY / 4 / ERROR MANAGEMENT: Error Management refers to the means available to prevent or reduce errors and to recover from them when they occur. Errors are defined in this context as invalid data entry, invalid format for data entry, incorrect command syntax, etc.

5 Verification

The verification of components is included in this chapter in an attempt to capture, from the earliest stage of the project, as most input as possible discussing the scenarios and tests about the verification and validation.

5.1 Verification scenarios

5.1.1 Web Portal (Graphical User Interface)

5.1.1.1 System Monitoring Tool

Table 12: System Monitoring Tool verification scenario

Component	***System Monitoring Tool* (Web Portal)**
Component Behaviour	System Monitoring Tool offers a graphical interface for system monitoring related functionalities.
Test	Visualisation of system and UxV health status
Test ID	SMT01
Pre-requisites	• connection to the System Monitoring Service (may not be necessary if System Monitoring Service collects all necessary data anyway) • administrative knowledge about the system state needed on user side (to check results)
Test description	• user opens System Monitoring Tool in the Web Portal • since the System Monitoring Service has a global view on the middle tier components, the System Monitoring Tool displays views with status of ○ middleware components, ○ testbeds components ○ UxVs components Note: with reference to the term “Service”, Data tier elements may be included here, in order to retrieve data for resources and the testbeds. In addition, System Monitoring will have the overview of the components at the middle tier (possibly as a service in the server). This requires some elaboration.
Expected results	• the displayed result should reflect the system state

5.1.1.2 *Resource Explorer Tool*

Table 13: Resource Explorer Tool verification tests

Component	***Resource Explorer Tool* (Web Portal)**
Component Behaviour	Resource Explorer Tool offers a graphical interface for browsing resources (testbeds and UxVs) inside the RAWFIE system.
Test	Browse testbeds and UxVs and start booking
Test ID	RET01
Pre-requisites	• connection to the Testbeds Directory Service ok • data about testbeds and UxVs available
Test description	• user opens Resource Explorer Tool in the Web Portal • Resource Explorer Tool displays a view with all available testbeds • user selects a testbed • Resource Explorer Tool displays all testbed details and a list of available UxVs • user selects a UxV • Resource Explorer Tool displays all UxVs details
Expected results	• Correct display of the data

5.1.1.3 *Experiment Monitoring Tool*

Table 14: Experiment Monitoring Tool verification tests

Component	***Experiment Monitoring Tool* (Web Portal)**
Component Behaviour	Experiment Monitoring Tool offers a graphical interface for experiment monitoring related functionalities.
Test	Visualisation of experiment status
Test ID	EMT01
Pre-requisites	• connection to the Launching Service ok • knowledge about the experiments state needed on user side (to check results)
Test description	• user opens Experiment Monitoring Tool in the Web Portal • Experiment Monitoring Tool displays a view with all experiments

	of the current user (ordered by date descending). The list also contains a sort summary of the experiments state • user selects a experiment • Experiment Monitoring Tool displays all experiment details ○ date / timespan ○ related testbed ○ list of used UxVs ○ execution state (planned, running, successful, error, canceled) ○ link to the used EDL
Expected results	• the displayed result should reflect the experiment state

5.1.1.4 Booking Tool

Table 15: Booking Tool verification tests

Component	Booking Tool (Web Portal)
Component Behaviour	Management of bookings of resources (UxVs)
Tests	BT01, BT02, BT03

Test	Visualisation of booking status
Test ID	BT01
Component (parent component if applicable)	***Booking Tool (Web Portal)***
Pre-requisites	• connection to the Booking Service ok • user opened Booking Tool though the Resource Explorer Tool (selected UxVs as parameter)
Test description	• user opens Booking Tool though the Resource Explorer Tool (selected UxVs as parameter) • Booking Tool displays a calendar view with the dates where the UxVs are already reserved

Expected results	The reserved dates should completely reflect all reservations.

Test	Booking on free date
Test ID	BT02
Component (parent component if applicable)	***Booking Tool (Web Portal)***
Pre-requisites	• connection to the Booking Service ok • user opened Booking Tool though the Resource Explorer Tool (selected UxVs as parameter)
Test description	• User selects “New booking“ from the UI • Booking Tool shows booking form • User enters data (name, time, comments) and a date where no reservation exist and submits the form • Booking Tool saves the booking
Expected results	The new booking should be persistently saved in the DB

Test	Booking on reserved date
Test ID	BT03
Component (parent component if applicable)	***Booking Tool (Web Portal)***
Pre-requisites	• connection to the Booking Service ok • user opened Booking Tool though the Resource Explorer Tool (selected UxVs as parameter)
Test description	• User selects “New booking“ from the UI • Booking Tool shows booking form • User enters data (name, time, comments) and a date where already

	reservations exist and submits the form Note: Either we book on a free date or on a reserved one. If on a free date, the service returns success; on a reserved date, an error message appears. The test may be ultimately merged with BT02.
Expected results	The Booking Tool should inform the user that there are already reservations and so the resources cannot be booked. No new booking should be saved in the DB

5.1.1.5 *Data Analysis Tool, engine and results DB*

Table 16: Data Analysis Tool, engine and results DB verification tests

Component	**Data Analysis Tool**
Component (parent component if applicable)	***Compute Engine, Data Analysis Engine, Result DB***
Component Behaviour	The Data Analytics Tool resides on the portal and connects to the CLI to pass commands to and from the Compute Engine & the Results Database.
Tests	DAT01, DAT02, DAT03, DAT04, DAT05, DAT06, DAT07, DAT08

Test	Verify correct command relay from GUI to the Data Analysis Engine.
Test ID	DAT01
Component (parent component if applicable)	***Data Analysis Engine***
Pre-requisites	• Requires web portal to be functioning and accessible. • Requires Data Analysis engine to be functioning & accessible.
Test description	1. Hard code a specific payload for a sequence of operations 2. Create a message payload for this sequence

	3. Verify that the payload the engine received
Expected results	If the predefined template matches that of the one sent then the transmission was a success.

Test	Pull predefined results from results database
Test ID	DAT02
Component (parent component if applicable)	***Results DB***
Pre-requisites	• Requires web portal to be functioning and accessible. • Requires Data Analysis engine to be functioning & accessible. • Require Results DB to be functioning and accessible.
Test description	1. Create predefined dataset 2. Store dataset 3. Pull dataset into Data Analysis Engine 4. Send dataset to web portal
Expected results	• The dataset pulled into the engine should be the same as the original • The dataset received by the portal should be the same as the original.

Test	Write test schema and validate received schema in analysis engine
Test ID	DAT03
Component (parent component if applicable)	***Data Analysis Engine***
Pre-requisites	• Requires Web Portal to be functioning and accessible. • Requires Data Analysis engine to be functioning & accessible. • Require Message Bus to be functioning and accessible.

Test description	1. Create test schemas in the Data Analysis Engine 2. Receive test schemas in the Data Analysis Engine 3. Verify that the one sent is consistent with the one received
Expected results	• The schema received should be consistent with the one created and sent from the Data Analysis engine.

Test	Read data from measurements data
Test ID	DAT04
Component (parent component if applicable)	***Data Analysis Engine***
Pre-requisites	• Requires Web Portal to be functioning and accessible. • Requires Data Analysis engine to be functioning & accessible. • Require Message Bus to be functioning and accessible.
Test description	1. Verify that the data that has to be processed can be accessed through the Web Portal. 2. Verify that the data that has to be processed can be read through the Web Portal.
Expected results	• Data should be accessible and readable through the Web Portal.

Test	Send packaged compute operations to compute engine and verify payload
Test ID	DAT05
Component (parent component if applicable)	***Data Analysis Engine, Compute Engine***
Pre-requisites	• Requires Web Portal to be functioning and accessible. • Requires Data Analysis Engine to be functioning & accessible. • Require Compute Engine to be functioning and accessible.
Test description	1. Create jobs on the Dara Analysis Engine through the Web Portal 2. Send this job to the Compute Engine to be executed

	3. Collect the results of this execution that are stored in the result DB 4. Verify that those results exist in the result DB and are consistent with the job sent to the Compute Engine 5. Collect output feedback of the execution from the compute engine 6. Verify payload
Expected results	• Results associated with the execution of the job sent to the Compute Engine should be stored in the expected format in the Result DB • These results should be consistent with the job executed • Payload should be consistent

Test	Validate Data Analysis experiment package created vs predefined one
Test ID	DAT06
Component (parent component if applicable)	***Data Analysis Engine***
Pre-requisites	• Requires Web Portal to be functioning and accessible. • Requires Data Analysis engine to be functioning & accessible.
Test description	1. Access the Data Analysis Engine through the Web Portal 2. Create an experiment package associated with a specific experiment the user wants to perform 3. Verify the consistency of this package structure and content by comparing it to a predefined package associated with the same experiment
Expected results	• The created experiment package should be similar to the predefined package for the same experiment

Test	Implement Identity Operator and verify results of some test operations on Result DB
Test ID	DAT07
Component (parent component if	***Data Analysis Engine, Compute Engine, Result DB***

applicable)	
Pre-requisites	• Requires Web Portal to be functioning and accessible. • Requires Data Analysis engine to be functioning & accessible. • Require Result DB to be functioning and accessible.
Test description	1. Access the Data Analysis Engine through the Web Portal 2. Create an experimental package containing a job implementing the Identity Operator 3. Send the package to the Compute Engine which will store the results in the Result DB 4. Access the Result DB through the Web Portal and verify the results associated with the previous job execution
Expected results	• The results associated with the job execution stored in the Result DB should be identical to the input data the job is executed on

Test	Retrieve predefined results from Result DB
Test ID	DAT08
Component (parent component if applicable)	***Data Analysis Engine, Result DB***
Pre-requisites	• Requires Web Portal to be functioning and accessible. • Requires Data Analysis engine to be functioning & accessible. • Require Result DB to be functioning and accessible.
Test description	1. Access the Data Analysis Engine through the Web Portal 2. Read result DB from the Data Analysis Engine and retrieve data from this DB as input data 3. Create packaged computed operation on this retrieved data 4. Along with the identity test above, verify if the results retrieved coming from an identity operation are identical to the initial input data on which the identity operator had been executed.
Expected results	• Results should be retrievable from the Data Analysis Engine and executions should be able to be performed on them

5.1.1.6 *Experiment authoring tool*

Table 17: Experiment authoring Tool verification tests

Component	**Experiment Authoring Tool**
Component (parent component if applicable)	***Textual Editor, Visual Editor, Launching Tool***
Component Behaviour	The Experiment Authoring Tool aims at providing functionalities related to the EDL editors and to help experiments start their experiments.
Tests	EAT01, EAT02, EAT03, EAT04, EAT05, EAT06

Test	Define experiments in the textual editor
Test ID	EAT01
Component (parent component if applicable)	***Textual Editor***
Pre-requisites	• Requires the EDL model to be accessible. • Requires the Web portal / EDL plugin to be accessible.
Test description	1. Access the textual editor through the Web portal or through an IDE with the EDL plugin installed. 2. Write an experiment. 3. Utilize code completion, content assist and compilation functionalities. 4. Define some erroneous commands in the experiment workflow. 5. Save the experiment.
Expected results	• Errors should be identified; code completion and content assist should efficiently work together with the compilation.

Test	Update experiments in the textual editor
Test ID	EAT02
Component	***Textual Editor***

(parent component if applicable)	
Pre-requisites	• Requires the EDL model to be accessible. • Requires the Web portal / EDL plugin to be accessible.
Test description	1. Access the textual editor through the portal or through an IDE with the EDL plugin installed. 2. Open an already defined experiment stored in the repository. 3. Make changes in the experiment workflow. 4. Save the experiment.
Expected results	• The already defined experiment should be smoothly accessed, errors should be identified; code completion and content assist should efficiently work together with the compilation.

Test	Define experiments in the visual editor
Test ID	EAT03
Component (parent component if applicable)	***Visual Editor***
Pre-requisites	• Requires the EDL model to be accessible. • Requires the Web portal / EDL plugin to be accessible.
Test description	1. Access the visual editor through the portal or through an IDE with the EDL plugin installed. 2. Access the available toolbar. 3. Create an experiment by utilizing the available tools. 4. Define some erroneous connections between tools in the experiment workflow (to be detailed later). 5. Save the experiment.
Expected results	• Visual editor works as specified. In particular, errors should be clearly identified.

Test	Update experiments in the visual editor

Test ID	EAT04
Component (parent component if applicable)	***Visual Editor***
Pre-requisites	• Requires the EDL model to be accessible. • Requires the Web portal / EDL plugin to be accessible.
Test description	1. Access the visual editor through the portal (or through Eclipse) with the EDL plugin installed. 2. Open an already defined experiment stored in the repositories. 3. Make changes in the experiment workflow. 4. Save the experiment.
Expected results	• The already defined experiment should be smoothly accessed, errors should be identified; should efficiently work.

Test	Switch between editors
Test ID	EAT05
Component (parent component if applicable)	***Textual Editor, Visual Editor***
Pre-requisites	• Requires the EDL model to be accessible. • Requires the Web portal / EDL plugin to be accessible.
Test description	1. Access the textual or the visual editor through the portal or through an IDE with the EDL plugin installed. 2. Create an experiment. 3. Switch to the alternative editor and make changes. 4. Save the experiment.
Expected results	• The editor works as specified. In particular, errors should be clearly identified. Code completion and content assist should efficiently work.

Test	Launch Experiments

Test ID	EAT06
Component (parent component if applicable)	***Launching Tool***
Pre-requisites	• The repositories should be up and running.
Test description	1. Initiate the launching tool. 2. Select the experiment. 3. Start the experiment execution.
Expected results	• The experiment should smoothly start and the appropriate RAWFIE components should be initiated.

5.1.1.7 EDL Compiler and Validator

Table 18: EDL Compiler and Validator Tool verification tests

Component	**EDL Compiler and Validator**
Component (parent component if applicable)	***Experiment Validation Service***
Component Behaviour	The EDL compiler and validator should receive an EDL script for the editors, identify possible syntactic and semantic errors and provide a set of files to be adopted by the remaining components in the RAWFIE architecture.
Tests	ECV01, ECV02

Test	Compile experiments
Test ID	ECV01
Pre-requisites	• Requires the EDL editors' model to be accessible. • Requires the EDL model to be accessible.
Test description	1. Access the textual editor or the visual editor. 2. Write a simple experiment.

	3. Compile the experiment.
Expected results	• Errors should be transferred in the available editors otherwise a set of files should be placed at the appropriate locations in the RAWFIE architecture.

Test	Validate experiments
Test ID	ECV02
Component (parent component if applicable)	***Experiment Validation Service***
Pre-requisites	• Requires the EDL editors' model to be accessible. • Requires the EDL model to be accessible.
Test description	1. Access the textual editor or the visual editor. 2. Write a simple experiment. 3. Validate the experiment.
Expected results	• Errors or successful results should be transferred in the available editors. Compilation may create an output of EDL script.

5.1.1.8 UxV Navigation Tool

Table 19: UxV Navigation Tool verification tests

Component	***UxV Navigation Tool***
Component Behaviour	This component will provide to the user the ability to remotely navigate a squad of UxVs. Through a user friendly interface, the experimenter will specify the required details of the experiment, providing information regarding the number of the vehicles, the number of the units etc.
Test	Access the UxV navigation tool and validate the produced instructions
Test ID	UxVNT01
Pre-requisites	• Requires Web Portal to be functioning and accessible.

Test description	• Access the UxV Navigation Tool through the portal • Validate the produced instructions ○ Validate the schema of the JSON output file ○ Validate the data format of the JSON output file ○ Validate the size of the JSON output file
Expected results	• The output data should be accessible and compatible with the required format

5.1.1.9 *Visualization Tool*

Table 20: Visualization Tool verification tests

Component	**Visualization Tool**
Component (parent component if applicable)	***Visualization Tool, Visualization Engine***
Component Behaviour	The Visualization Tool provides visualization of the geospatial data. Additionally it enables the user to show and track all UxV resources and to apply additional modifications (layers, filters, etc.) to the geospatial data and to show different sensor data.
Tests	VIS01, VIS02, VIS03, VIS04, VIS05, VIS06

Test	User request handling
Test ID	VIS01
Component (parent component if applicable)	***Visualization Tool , Visualization Engine***
Pre-requisites	• Requires visualization tool to be functioning & accessible. • Requires visualization engine to be functioning & accessible.
Test description	1. User sends a predefined http request via the visualization tool 2. The visualization tool forwards it to the visualization engine 3. Visualization engine handles it and sends back the response

	4. The response is visualized on the user screen
Expected results	The request from the user is properly handled and plotted on the screen

Test	Geospatial data handling
Test ID	VIS02
Component (parent component if applicable)	***Visualization Tool , Visualization Engine***
Pre-requisites	• Requires message bus to be functioning & accessible. • Requires visualization tool to be functioning & accessible. • Requires visualization engine to be functioning & accessible.
Test description	1. Acquire predefined geospatial data (WMS, WFS) via the message bus 2. Modify for proper visualization using geoserver 3. Send to visualization tool 4. Render and plot on user screen
Expected results	The predefined geospatial data is plotted properly on the user screen

Test	Geospatial data modification
Test ID	VIS03
Component (parent component if applicable)	***Visualization Tool , Visualization Engine***
Pre-requisites	• Requires message bus to be functioning & accessible. • Requires visualization tool to be functioning & accessible. • Requires visualization engine to be functioning & accessible.
Test description	1. Acquire predefined geospatial data (WMS, WFS) via the message bus 2. Add a predefined layer on top of the data

	3. Send to visualization tool 4. Render and plot on user screen
Expected results	The layer appears properly on top of the geospatial data on the user screen

Test	Experiment Controller communication
Test ID	VIS04
Component (parent component if applicable)	***Visualization Engine***
Pre-requisites	Requires Experiment Controller to be functioning & accessible. Requires visualization engine to be functioning & accessible.
Test description	Receive a message that the experiment has started from the Experiment Controller Handle the message appropriately in the visualization engine by starting the experiment in the visualization engine (e.g. forward the necessary information to the visualization tool) Receive an experiment status message from the Experiment Controller Handle it properly in the visualization engine (e.g. forward the necessary information to the visualization tool) Receive a “stop experiment” message from the Experiment Controller Stop the experiment on the visualization engine
Expected results	The start/stop and status messages about the experiment that come from the experiment controller are handled properly in the visualization engine

Test	Visualization Tool Interaction
Test ID	VIS05
Component (parent component if	***Visualization Tool***

applicable)	
Pre-requisites	• Requires visualization tool to be functioning & accessible.
Test description	1. Enable/Disable different features of the visualization tool (e.g. show/hide speed web widget) 2. Handle the user requests in the visualization engine 3. Update the plotted data (e.g. show/hide speed web widget)
Expected results	The complete user interaction with keyboard/mouse is handled properly and the screen is updated accordingly

Test	Camera interaction
Test ID	VIS06
Component (parent component if applicable)	***Visualization Tool, Visualization Engine***
Pre-requisites	• Requires visualization tool to be functioning & accessible. • Requires visualization engine to be functioning & accessible. • Requires Experiment controller to be functioning & accessible.
Test description	1. Retrieve with the visualization engine quasi real time data from one UxV, processes it and send it to the visualization tool 2. Plot the data with the visualization tool on the screen and start following the UxV with the camera 3. Add additional UxV to the experiment 4. Adjust the camera view for the new scenario 5. Change the camera view through the visualization tool 6. Camera should update the perspective 7. Change the camera view to follow one of the UxVs again 8. Camera perspective is updated and it follows the UxV
Expected results	The camera perspective is updated properly according to the current scenario and is adjusted in real time dependent on changes of the scenario or because of user interaction

5.1.2 Communication and storage components

5.1.2.1 Testbeds directory service

Table 21: Testbeds directory service Tool verification tests

Component (and parent component if applicable)	***Testbeds directory service***
Component Behaviour	***Manages the RAWFIE platform federation resources and testbeds***
Tests	**TD01, TD02, TD03, TD04**

Test	Retrieve resources from testbed facility
Test ID	**TD01**
Component (parent component if applicable)	
Pre-requisites	• The resource must exist in the facility • The user must be granted to access to the testbed • The user must know the resource characteristics
Test description	1. User performs the login through the web portal 2. Search for the resource detailing the device information 3. The queries are executed against the repository 4. Verify that resource data retrieved is consistent and its actual status is valid
Expected results	Resource information shall be displayed, if the resource/s cannot be found, a notification message is shown.

Test	Add new testbed facility to the RAWFIE federation

Test ID	**TD02**
Component (parent component if applicable)	***Testbeds and Resources Repository***
Pre-requisites	• The platform administrator must count with the appropriate access rights • The new testbed facility must complain with certain standards and regulations
Test description	1. Admin performs the login to manage the infrastructure 2. Controls that the information is correct, just exactly as it was defined beforehand 3. Check the data to persist in order to avoid future security issues and information disclosure 4. Register the testbed data into the repository 5. Verify that the recent added facility is fully operative and available for the experimental applications
Expected results	The new test-bed works as expected offering UxV devices for diverse researches in different contexts.

Test	Register UxV node into a testbed facility
Test ID	**TD03**
Component (parent component if applicable)	
Pre-requisites	• The resource information must be available and accurately provided by the manufacturer •
Test description	1. The testbed operator logins with the sufficient access rights 2. Registers the resource information into the repository 3. Verify that the registered resource was successfully added, and is fully operative and available for the experimental applications
Expected results	The new resource at the given testbed facility is fully operative and ready

	to use for experiment executions.

Test	Retrieve testbeds information and belonging resources
Test ID	**TD04**
Component (parent component if applicable)	***Resource Explorer Tool***
Pre-requisites	• The testbed facility must exist in the federation • The user must have granted access to the testbed • The user must have a minimal knowledge regarding the testbed searched
Test description	1. The experimenter sends the testbed parameters information in order to list the desired testbeds, as well as a determined search criteria 2. Consider the filtering specification for the testbed search and execute the query against the respository 3. Retrieve a result set with different testbeds and its existing available resources 4. Forward the fetched data to the Resource Explorer Tool for its consequent visualisation
Expected results	Testbed information is displayed, then the experimenter can visualize, analyse and evaluate the best choice to carry out the experiment.

5.1.2.2 Message Bus

Table 22: Message Bus Tool verification tests

Component	***Message Bus***
Component Behaviour	***A message broker that allows an asynchronous communication among the heterogeneous components across all application tiers of the RAWFIE architecture***
Tests	**MB01, MB02, MB03, MB04**

Test	Monitor resource
Test ID	**MB01**
Component (parent component if applicable)	***Monitoring Tool, Visualization Tool, Visualization Engine***
Prerequisites	• The resource must be active performing the experimentation • The UxV must be equipped with GPS system and corresponding resource controllers
Test description	1. The experimenter wants to locate the used resource for an experiment 2. Executes the adequate device query 3. Retrieves the current geospatial information of the device
Expected results	The UxV coordinates are fetched in real time and are currently available.

Test	Generate resource booking notification
Test ID	**MB02**
Component (parent component if applicable)	***Launching Service***
Prerequisites	• The experimenter must be registered in the platform with the necessary credentials • The experimenter must also count with a registered email account pertaining to the federation • A long-term selection must be scheduled as launching selection with the previous resource booking
Test description	1. Book any resource in order to carry on a certain experiment in the near future 2. Wait till the established date and time to be launched 3. Verify that user has received the corresponding notification regarding the booking information and experiment prepared

Expected results	The user has received an email with the experiment information and relevant launching data.

Test	Generate platform event notification
Test ID	**MB03**
Component (parent component if applicable)	***System Monitoring Service***
Prerequisites	• The platform administrator must be registered at the federation and be able to receive these type of critical notification messages • The system must be correctly configured and the monitoring service active and fully operating
Test description	1. A particular event triggers the notification process, for example a threshold was reached 2. Immediately the notification template is generated and forward to the responsible user administrator to take control of the situation 3. The platform administrator receives the message with the detailed description of the hardware or software issue.
Expected results	The platform administrator gets the proper information right on time and with enough level of detail and exactness.

Test	Execute experiment workflow
Test ID	**MB04**
Component (parent component if applicable)	***Experiment Controller***

Prerequisites	• The experimenter have already created the script for the experiment of interest • The chosen resource must be completely available and ready to use
Test description	1. The experimenter forwards the script to the Experiment Controller in order to start or barely execute the next action of the resource mission 2. The instructions are forwarded to the corresponding testbed facility 3. The resource receives the new set of instructions as generated from the script for overriding the experiment workflow
Expected results	The execution of the experiment happens just as the experimenter defined it in the EDL script and the action was successfully performed.

5.1.2.3 *Users and Rights Service*

Table 23: Users and Rights Service Tool verification tests

Component	Users & Rights Service
Component Behaviour	Management of users and rights. (Components may also be users)
Tests	URS01, URS02

Test	Check user login
Test ID	URS01
Component (parent component if applicable)	Users & Rights Service
Pre-requisites	• user name and password known thought login form
Test description	• user name and password sent to the Users & Rights Service • Users & Rights Service load the data of the given user • Users & Rights Service hashes the password and compares it with the stored password hash • Users & Rights Service returns the login check result

Expected results	Result should be negative, if user does not exist or the password hash is different Result should be positive, if user password hash equals the stored password hash

Test	Check user rights
Test ID	URS02
Component (parent component if applicable)	Users & Rights Service
Pre-requisites	• user or component ID known thought certificate or standard login procedure
Test description	• user ID and required rights sent to the Users & Rights Service • Users & Rights Service load the data of the given user • Users & Rights Service returns if the all rights are available
Expected results	If user does not exist, the result should be an error. Otherwise, the result should be a true of all requested right are available or false if not.

5.1.2.4 *Launching Service*

Table 24: Launching Service Tool verification tests

Component	**Launching Service**
Component (parent component if applicable)	-
Component Behaviour	The launching service gives the opportunity to experimenters to select and start an already defined, validated and uploaded experiment.
Tests	LS01, LS02

Test	Short term launching
Test ID	LS01
Component (parent component if applicable)	-
Pre-requisites	• Requires the Web portal to be accessible. • Requires the launching tool to be accessible. • Requires the testbed proxy and the experiment controller to be accessible. • The repositories should be up and running.
Test description	1. Access the launching tool. 2. Select an already defined experiment. 3. Start the experiment. 4. Initiate the Testbed Proxy. 5. Initiate the Experiment Controller.
Expected results	• The experiment should smoothly start and the appropriate RAWFIE components should be initiated.

Test	Long term launching
Test ID	LS02
Pre-requisites	• The Web portal should be up and running.
Test description	1. Initiate the checking process of the booking repository. 2. Identify the experiments that should be immediately started. 3. Start the appropriate experiments. 4. Initiate the Testbed Proxy. 5. Initiate the Experiment Controller.
Expected results	• The experiments should smoothly start and the appropriate RAWFIE components should be initiated.

5.1.3 Testbed control, monitoring and analysis components

5.1.3.1 Experiment Controller

Table 25: Experiment Controller Tool verification test

Component	***Experiment Controller***
Component Behaviour	The Experiment Controller (EC) is a service placed in the Middle tier and is responsible to monitor the smooth execution of each experiment. The task of the EC is not the control of the UxVs directly as this will be done through the Testbed Proxy. The main task is the monitoring of the experiment execution while acting as 'broker' between the experimenter and the resources in (near) real time.
Tests	EC01

Test	Connection Test
Test ID	EC01
Pre-requisites	• Requires web portal to be functioning and accessible. • Requires testbed proxy to be functioning & accessible.
Test description	• Register an experiment (Testbed manager) • Send Network Requirements (Testbed manager) • Send basic instructions to the Resource Controller • Transmit simulated or real results back to the Experiment Monitoring Tool
Expected results	• The experiment controller should be able to receive instructions from the web-portal and transmit these instructions to the resource controller. Additionally, the EC01 test should guarantee the transmission of the results back to the Experiment Monitoring Tool.

5.1.3.2 Monitoring Manager

Table 26: Monitoring Manager Tool verification tests

Component	**Monitoring Manager**
Component	Monitoring manager is responsible to monitor the status of the testbed and the devices belonging to it, at functional level, i.e., the 'health of the

Behaviour	devices' and current activity.
Tests	MM01

Test	Check Monitoring Activity
Test ID	MM01
Pre-requisites	• Requires the resource controller to be accessible. • Requires the network controller to be accessible. • Requires the testbed proxy to be accessible. • Requires the data tier to be accessible.
Test description	1. Requires the resource controller to be accessible. 2. Requires the network controller to be accessible. 3. Requires the testbed proxy to be accessible. 4. Requires the data tier to be accessible
Expected results	• The experiment should smoothly start and the appropriate RAWFIE components should be initiated.

5.1.3.3 Network Controller

Table 27: Network Controller Tool verification tests

Component	**Network Controller**
Component Behaviour	The Network Controller manages the network connections and changes between different technologies in the testbed and their resources.
Tests	NC01

Test	Switch network interface due to connectivity problems.
Test ID	NC01
Pre-requisites	• Requires the Testbed Manager to be accessible
Test description	• The Network Controller 'checks' the connectivity of the resources through the Resource Controller. • The Resource Controller informs the Network Controller for malfunctions in the network connectivity of the resources.

	• The Network Controller receives the incoming messages from the Resource Controller. • The Network Controller identifies problems in the connectivity and triggers the Resource Controller to force the change of the network interface.
Expected results	• The appropriate network interface is selected.

5.1.3.4 *Resource Controller and Navigation Service*

Table 28: Resource Controller Tool verification test

Component	***Resource Controller (and Navigation Tool)***
Component Behaviour	The core component of the navigation system is the "Resource Controller" which ensures the safe and accurate guidance of the UxVs. RC commands each device to switch on board sensors on and off. At the same time, Resource Controller informs the "Monitoring Tool" and Data Tier for the gathered measurements of the sensors of each device.
Tests	RC01

Test	Connection Test and Validation of the Accuracy of the Given Instructions
Test ID	RC01
Pre-requisites	• The proxy should be connected to the testbed • Requires the UxV to be ready to operating (e.g. en route). • Requires the UxV to be reachable by any communication mean.
Test description	• Receive instructions from the Experiment Controller • Validate the Obstacle Avoidance Mechanism using known simulated scenarios • Validation of the Collision Avoidance Mechanism using known simulated scenarios • Send basic instructions to the UxVs through the Testbed Manager so as to perform UxV01- UxV05 tests • Transmit the results buck to the Experiment Controller
Expected results	• The UxV follows the instruction correctly, in order and timely,

	according to the specified parameters.

5.1.4 Testbed resources

5.1.4.1 Testbed Proxy

Table 29: Testbed Proxy Tool verification tests

Component	***Testbed Proxy***
Component Behaviour	***Handles the communication between the testbed facility and the rest tiers of RAWFIE Architecture***
Tests	TP01

Test	Detect a proxy server
Test ID	**TP01**
Prerequisites	• The proxy should be connected to the testbed • Testbed proxy is connected to internet • User knows testbed proxy configurations
Test description	1. User pings proxy server 2. Verify the port number
Expected results	Proxy server details like IP address is demonstrated to the user

5.1.4.2 Testbed Manager

Table 30: Testbed Manager Tool verification tests

Component	***Testbed Manager***
Component Behaviour	Contains accumulated information from the experiments and the devices in the testbed
Tests	TM01, TM02, TM03

Test	Registers an experiment
Test ID	TM01

Pre-requisites	• Requires middle tier to be accessible • Requires the testbed proxy and the experiment controller to be accessible. • Requires Data Tier to be accessible
Test description	1. Experiment Controller sends a request for a new experimentAccess the launching tool. 2. Testbed manager registers this experiment locally 3. Testbed manager registers this experiment to the data tier 4. Testbed manager sends the experiment commands to the Resource Controller 5. Sends response to the Experiment Controller
Expected results	• No error occurs • Experiment Controller receives a “success” response

Test	Checks the status of the experiments
Test ID	TM02
Pre-requisites	• Requires middle tier to be accessible • Requires the testbed proxy and the experiment controller to be accessible. • Requires Data Tier to be accessible
Test description	1. Experiment Controller sends a request to Testbed manager 2. Testbed Manager checks locally the status of the experiments 3. Sends a list with the experiments and their status to Experiment Controller
Expected results	• No error occurs • Experiment Controller receives a response with the list of the experiments • The data from the Testbed Manager should be consistent with the relevant of the Data Tier

Test	Manage the experiments without middle-tier connection
Test ID	TM03
Pre-requisites	• Testbed loses the connection with the middle tier

Test description	1. Testbed Manger checks the status of the experiments 2. Testbed Manager informs for Resource Controller for “emergency” situation and pause experiments 3. Resource Controller sends a response
Expected results	• No error occurs • Testbed Manager receives a response from Resource Controller

5.1.4.3 UxV Node

Table 31: UxV Node Tool verification tests

Component	**UxV Node**
Component Behaviour	The UxV node is an unmanned vehicle (ground, underground, aerial, water surface or sub-surface)
Tests	UxV01 to UxV15

Test	Return to base
Test ID	UxV01
Pre-requisites	• Requires the RAWFIE system to be operational • Requires the mission to be defined and running. • Requires the UxV to be ready to operating (e.g. en route). • Requires the UxV to be reachable by any communication mean.
Test description	1. Establish the communication with the UxV. 2. Establish a secure control session, 3. Send the return to base command, 4. If the UxV is not autonomous, instruct it with the necessary waypoint or guidance information, possibly until the end of the test, 5. Close the secure control session.
Expected results	• The UxV is home after a safe return.

Test	Follow a route

Test ID	UxV02
Pre-requisites	• Requires the RAWFIE system to be operational • Requires the mission to be defined and running. • Requires the UxV to be ready to operating (e.g. en route). • Requires the UxV to be reachable by any communication mean.
Test description	1. Establish the communication with the UxV. 2. Establish a secure control session, 3. Send the instructions for the new route, 4. If the UxV is not autonomous, continue to instruct it in real-time with the necessary waypoint or guidance information, possibly until the end of the test, 5. Close the secure control session after having reached a safe state for the UxV.
Expected results	• The UxV follows the route as specified.

Test	Acquire sensor samples
Test ID	UxV03
Pre-requisites	• Requires the RAWFIE system to be operational • Requires the mission to be defined and running. • Requires the UxV to be ready to operating (e.g. en route). • Requires the UxV to be reachable by any communication mean.
Test description	1. Establish the communication with the UxV. 2. Establish a secure control session (if not done already), 3. Send the acquisition commands 4. Store sensor samples and, if possible, transmit them via the data communication system, 5. If opened specifically for the matter of the test, close the secure control session.
Expected results	• Sensor samples have acquired correctly and are stored in the UxV memory or in the experiment database.

Test	Fidelity to commands
Test ID	UxV04

Pre-requisites	• Requires the RAWFIE system to be operational • Requires the mission to be defined and running. • Requires the UxV to be ready to operating (e.g. en route). • Requires the UxV to be reachable by any communication mean.
Test description	1. Establish the communication with the UxV. 2. Establish a secure control session (if not done already), 3. Send repeatedly pre-defined sets of commands, covering the full range of possible UxV actions, 4. Check the conformance of the undertaken actions and corrections (if necessary) to the commands, 5. Record all fine grained status of the UxV over the duration of the test, to be able to reconstruct the behaviour of the UxV, 6. If opened specifically for the matter of the test, close the secure control session.
Expected results	• The UxV follows the instruction correctly, in order and timely, according to the specified parameters.

Test	Verification of the UxV health status monitoring
Test ID	UxV05
Pre-requisites	• Requires the RAWFIE system to be operational • Requires the mission to be defined and running. • Requires the UxV to be ready to operating (e.g. en route). • Requires the UxV to be reachable by any communication mean.
Test description	1. Establish the communication with the UxV. 2. Establish a secure control session (if not done already), 3. Repeatedly request for the UxV status, 4. Record locally and transmit the requested status to the experiment database, 5. If opened specifically for the matter of the test, close the secure control session.
Expected results	• The UxV acquires, stores the requested status and correctly transmits them.

5.1.4.4 UxV Network Communication

Test	Continuous communication
Test ID	UxV06
Pre-requisites	• Requires the RAWFIE system to be operational • Requires the mission to be defined and running. • Requires the UxV to be ready to operating. • Requires the UxV to be reachable by any communication mean.
Test description	1. Establish the communication with the UxV. 2. Exchange a predefined set of commands and data.
Expected results	• The UxV is “home” (to be defined, since it may depend on the type of UxV, the running experiment, the host testbed) after a safe return. “Home” may be an attribute of the UxV.

Test	Secure communication
Test ID	UxV07
Pre-requisites	• Requires the RAWFIE system to be operational • Requires the UxV to be ready to operating. • Requires the UxV to be reachable by any communication mean.
Test description	1. Establish the communication with the UxV. 2. Establish a secure control session, 3. Check communication parameters, 4. Exchange a pre-defined set of commands and data, 5. Close the secure control session.
Expected results	• The end to end communication between the UxV and the ground control is established, secured and maintained.

Test	Real-time communication
Test ID	UxV08
Pre-requisites	• Requires the RAWFIE system to be operational • Requires the mission to be defined and running. • Requires the UxV to be ready to operating (e.g. en route). • Requires the UxV to be reachable by any communication mean.
Test description	1. Establish the communication with the UxV. 2. Establish a secure control session, 3. Send safe commands and measure the temporal characteristics of the communication (e.g. response time, synchronisation of reception across a swarm of UxV (coordinated group of UxV), etc.).
Expected results	• The temporal characteristics are within the acceptable boundaries.

Test	Resume communication and data transfer
Test ID	UxV09
Pre-requisites	• Requires the RAWFIE system to be operational • Requires the mission to be defined and running. • Requires the UxV to be ready to operating. • Requires the UxV to be reachable (at least sporadically) by any communication mean.
Test description	1. Establish the communication with the UxV. 2. Start a transaction. 3. Interrupt the communication at the low-level (e.g. disconnect the antenna) 4. Re-establish the communication low level means 5.
Expected results	• The UxV detects the communication interruption and the re-establishment of the communication link and resume the interrupted transaction (may be by restarting it).

Test	UxV Device Management
Test ID	UxV10

Pre-requisites	• Requires the RAWFIE system to be operational • Requires the mission to be defined and running. • Requires the UxV to be ready to operating (e.g. en route). • Requires the UxV to be reachable by any communication mean.
Test description	6. Establish the communication with the UxV. 7. Establish a secure control session, 8. Send the return to base command, 9. If the UxV is not autonomous, instruct it with the necessary waypoint or guidance information, possibly until the end of the test, 10. Close the secure control session.
Expected results	• The UxV is home after a safe return.

Component	UxV Node
Component (parent component if applicable)	Network Communication
Component Behaviour	Provide communication services to the UxV
Pre-requisites	UxV is in operation state and the parent UxV node has been launched
Test	Connection Test
Test ID	UxV11
Test description	Connectivity between resource controller and UxV node is ensured. The connection is bi-directional. UxV Data is being streamed and Commands are being received by the UxV. In order to test the functionality, remote-controlled movement commands can be sent to the robot and information streamed such as robot state or Image data can be checked. Loss of connection due to poor signal needs special attention
Expected results	The data streamed should be received with quality enough; Commands have to result in smooth movement of the robot.

Component	UxV Node
Component (parent component if applicable)	Sensors and localization
Component Behaviour	Provides data gathered by each sensor placed on the robot
Pre-requisites	UxV is in operation state and the parent UxV node has been launched. Network Communication is also fully functional
Test	Sensor Data Acquisition 1
Test ID	UxV12
Test description	Individual sensor data is tested Data streamed of every sensor is tested individually.
Expected results	Every sensor works as expected

Component	UxV Node
Component (parent component if applicable)	Sensors and localization
Component Behaviour	Provides data gathered by each sensor placed on the robot
Pre-requisites	UxV is in operation state and the parent UxV node has been launched. Network Communication is also fully functional
Test	Sensor Data Acquisition 2 (localization)
Test ID	UxV13
Test description	Localization of the robot is tested.

	Robot is moved to a precisely located point and a comparison is made with robot localization data.
Expected results	Every sensor works as expected

Component	UxV Node
Component (parent component if applicable)	Onboard Storage
Component Behaviour	Requested data are stored in the robot system
Pre-requisites	UxV is in operation state and the parent UxV node has been launched. Sensor node is functional
Test	Data Storage
Test ID	UxV14
Test description	A request for storing certain data is done After a mission given, data storage in the system is checked.
Expected results	The data is stored and identified in the robot system

Component	UxV Node
Component (parent component if applicable)	Onboard Processing
Component Behaviour	The robot process the data and executes its tasks
Pre-requisites	UxV is in operation state and the parent UxV node has been launched. Sensor node is functional, network communication is functional

Test	Waypoints Processed
Test ID	UxV15
Test description	Semi-autonomous mission is tested. The robot has to process a set of waypoints and move to each waypoint. Collision avoidance is also processed correctly
Expected results	The trajectory matches the one calculated by the navigation node. Robot stops, informs and recalculate its route to next waypoint if an unexpected obstacle is found.

5.2 Integrated system testing

As well as testing each individual component, the system will also be tested as a whole unit to validate its overall behaviour. Testing will be covered in the following areas:

The integrated testing procedure will be detailed during the first development iteration. The testing procedure will be based on the successful chain of verification scenarios described in Section 2 of this document.

Such scenarios will correspond to sequences and combinations of the components tests.

6 Validation scenarios

This chapter describes the validation scenarios. Some have been defined by the selected users of the RAWFIE system. Other simpler and more dedicated scenarios can focus on the evaluation of specific characteristics or behaviours of the RAWFIE components, testbeds, federation, etc. They are defined on the basis of requirements described in D3.1. Other scenarios may be defined on the basis of user defined use cases.

6.1 User defined scenarios

The set of user defined scenarios, as defined in D3.1 in the mentioned sections, are large scale, full-fledged validation scenarios:

- Scenario 1 – Environmental Monitoring of Water Canals
- Scenario 2 – Border Surveillance or Perimeter protection of large areas
- Scenario 3 – On demand deployable Internet facilities
- Scenario 4 – Exploration & Assessment of Network Technologies Robustness

- Scenario 5 – Efficient Coordination for phenomena or mission
- Scenario 6 – Over the Air (OTA) UxV Re-programming

Some of the above-mentioned scenarios are described in the next paragraphs to form a first basis of scenarios that could be later refined, completed and finally experienced.

Validation scenario	**Efficient coordination of multiple UxVs**
Main stakeholder	Providers, Experimenters
Secondary stakeholder	UxV manufacturers
Description	This scenario deals with the efficient coordination of multiple UxVs for the purpose of covering certain phenomena (e.g. fire spreading in an area) or executing a certain sensing mission (e.g. mapping or scanning of an unknown area). • The experimenter wants to test an algorithm for spatial coverage of an area of interest with the minimum energy consumption • The experimenter looks for a UxV-specific testbed (e.g. Surface, Aerial, Ground, etc.) • The experimenter books a group of UxV resources • The experimenter write the experiment steps with EDL, e.g. • Defines a specific area of interest • Defines the algorithm for the coordination of the UxVs (like Particle Swarm Optimization algorithm) • The experiment will be started a the given date • The UxVs execute the given script correctly • Measurements are sent to the database • The experiment finishes • The experimenter evaluates the results • View experiment log • Examine measurements • Percentage of the covered area • Nodes lifetime • Nodes energy consumption • Final positions • The experimenter runs the Data Analysis to find potential problems with the coordination of the resources
Metrics and success criteria	• all steps are completed without error

Involved Sub-systems	Resource Explorer Tool Testbeds Directory Service Booking Tool Booking Service Experiment Authoring Tool EDL Compiler & Validator Experiment Validation Service Launching Service Experiment Monitoring Tool Testbed Proxy Resource Controller Testbed Manager
Validated requirement	PT-GEN-001, PT-P-001, PT-P-003, PT-A-001, PT-A-003, PT-A-004, PT-A-005, PT-A-006, PT-A-007, ,PT-A-009, ,PT-A-014, PT-A-016, PT-B-001, PT-L-002, PT-E-002, PT-E-003, TB-G-004, TB-G-006, TB-I-001, TB-G-013, TB-D-001

6.1.1 Over the Air (OTA) UxV Re-programming

Validation scenario	**Over the Air (OTA) Re-programming**
Main stakeholder	Providers, Experimenters
Secondary stakeholder	UxV manufacturers
Description	The aim of this scenario is to use the OTA re-programming procedure that will be triggered by nodes, after a failure occurs or when extensions on board applications are required. 1. An experiment is selected involving a number of nodes. 2. Each node transfers loads from one place to another without the supervision of the experiment control. 3. A node requires extension on the on board processing capabilities (simulation). 4. The node sends a message to the swarm. 5. The nodes that are capable of fulfilling the request change the processing scheme. 6. The requested results are sent back.
Metrics and success criteria	• all steps are completed without error
Involved Sub-systems	Resource Controller Testbed manager UxV nodes
Validated requirement	PT-GEN-001, PT-GEN-002, TB-G-003, TB-G-005, TB-I-001, TB-I-002, TB-I-004, TB-R-001, TB-R-002, TB-R-004, TB-R-005, TB-R-010, TB-R-011
Validation scenario	Over the Air (OTA) Re-programming
Main stakeholder	Providers, Experimenters
Secondary stakeholder	UxV manufacturers
Description	The aim of this scenario is to use the OTA re-programming procedure that will be triggered by nodes, after a failure occurs or when extensions on board applications are required. 1. An experiment is selected involving a number of nodes. 2. Each node transfers loads from one place to another without the supervision of the experiment control. 3. A node requires extension on the on board processing capabilities (simulation). 4. The node sends a message to the swarm. 5. The nodes that are capable of fulfilling the request change the processing scheme. 6. The requested results are sent back.

Metrics and success criteria	• all steps are completed without error
Involved Sub-systems	Resource Controller Testbed manager UxV nodes
Validated requirement	PT-GEN-001, PT-GEN-002, TB-G-003, TB-G-005, TB-I-001, TB-I-002, TB-I-004, TB-R-001, TB-R-002, TB-R-004, TB-R-005, TB-R-010, TB-R-011

6.1.2 Experimenter scenarios

Validation scenario	**Experimenter registration**
Main stakeholder	Administrator
Secondary stakeholder	Experimenters
Description	The purpose of these scripts is to check if experimenters can make a successful registration through the Web portal. 1. The experimenter visits the Web Portal. 2. The experimenter should register on the Web Portal by filling the form with the desired username, email address, etc. 3. The administrator accepts the registration and emails the password with an activation link. The administrator doesn't accept the registration of the specific experimenter due to mistyped/error information applied and sends a notification. 4. The experimenter activates his account by visiting the respective link or in the case of unsuccessful registration he/she performs the process again. 5. The experimenter enters the Web portal after successful login.
Metrics and success criteria	• all steps are completed without error
Involved Sub-systems	Web Portal Users & Rights Service
Validated requirement	PT-GEN-001, PT-GEN-002, PT-GEN-003

Validation scenario	**Experimenter Booking**
Main stakeholder	Experimenters
Secondary stakeholder	Testbed Managers
Description	The purpose of this scenario is to allow the experimenter to book successfully a testbed with a number of (required) resources for the execution of her/his experiment. 1. The experimenter successfully logins into the RAWFIE system through the Web Portal. 2. The experimenter gains access to the booking tool. 3. The experimenter can have a view of all the available testbeds and, thus, the experimenter can access the testbeds list through the Resource Explorer. 4. The experimenter books a testbed or multiple testbeds for a specific period of time accompanied by the appropriate UxV resources. 5. The platform books the resources in the specific testbed for the pre-defined period of time. The platform doesn't allow the booking in case the resources are already booked to another experiment (at the same date/time). In both cases, a notification is delivered to the experimenter.
Metrics and success criteria	• all steps are completed without error
Involved Sub-systems	Web Portal Booking Tool Users & Rights Service Testbed and Resources Directory Bookings repository
Validated requirement	PT-GEN-001, PT-GEN-003, PT-P-003, PT-B-001, PT-B-002, PT-B-003, PT-B-005, PT-B-006

Validation scenario	**Experiment Definition**
Main stakeholder	Experimenters
Secondary stakeholder	-
Description	The aim of this scenario is to show the validation steps during the process of an experiment definition. The experiment definition process will be validated through the following steps: 1. The experimenter successfully logins into the RAWFIE system through the Web Portal. 2. The experimenter initiates the authoring tool. 3. He/she uses the functionalities provided by the EDL and the authoring tool to describe an experiment in the EDL terms. 4. He/she successfully interacts with the authoring tool (the textual and the visual editors) and uses the facilities like auto-completion, the graphical interface, etc. 5. He/she successfully identifies and corrects possible errors in the experiment workflow. 6. He/she successfully saves the experiment in the provided repository. 7. He/she successfully compiles the experiment and produces the final files to be adopted by the remaining components of the RAWFIE architecture. 8. He/she successfully validates the experiment. 9. The experimenter successfully leaves the authoring tool.
Metrics and success criteria	• all steps are completed without error
Involved Sub-systems	Web Portal Experiment Authoring Tool Users & Rights Service EDL Compiler and Validator Experiment Validator Experiments / EDL Repository
Validated requirement	PT-GEN-001, PT-GEN-003, PT-A-001, PT-A-002, PT-A-003, PT-A-004, PT-A-005, PT-A-006, PT-A-007, PT-A-008, PT-A-009, PT-A-0010, PT-A-0011, PT-A-0012, PT-A-0013, PT-A-0014, PT-A-0015, PT-A-0016

Validation scenario	**Manual Launching of Experiments**
Main stakeholder	Experimenters
Secondary stakeholder	-
Description	The scenario aims to validate the manual launching of pre-defined experiments. 1. The experimenter successfully logins into the RAWFIE system through the Web Portal. 2. The experimenter initiates the launching tool. 3. He/she selects the already defined / configured experiment and launches it. 4. The experiment successfully leaves the RAWFIE system.
Metrics and success criteria	• all steps are completed without error
Involved Sub-systems	Web Portal Experiment Authoring Tool Launching Service Users & Rights Service Testbed Proxy Resource Controller Testbed Manager
Validated requirement	PT-GEN-001, PT-GEN-002, PT-GEN-003, PT-P-001, PT-L-001, PT-L-002, PT-L-003

Validation scenario	**Interacting with Nodes (see also Monitoring Experiments)**
Main stakeholder	Experimenters
Secondary stakeholder	-
Description	The aim of this script is to show the interaction between the experimenter and his/her experiment during the execution. 1. The experimenter successfully logins into the RAWFIE system through the Web Portal. 2. The system starts the experiment as defined by the booking service and the experimenter starts to observe the experiment execution. 3. The status of the experiment and the corresponding resources are presented in the provided interface. 4. The experimenter interacts with the experiment by manually changing the navigation instructions of nodes and other experiment parameters. 5. The RAWFIE system validates the incoming directions and acts accordingly. 6. The experiment successfully leaves the RAWFIE system.
Metrics and success criteria	• all steps are completed without error
Involved Sub-systems	Web Portal Experiment Authoring Tool Launching Service Users & Rights Service Visualization Tool Navigation Tool Visualization Engine Experiment Control Experiment Monitoring Tool Testbed Proxy Resource Controller Testbed Manager
Validated requirement	PT-GEN-001, PT-GEN-002, PT-GEN-003, PT-P-001, PT-L-001, PT-L-002, PT-L-003, PT-L-004, PT-L-005, PT-L-006, PT-L-007, PT-L-008, PT-L-009, PT-L-0010, PT-E-001, PT-E-002

Validation scenario	**Monitoring Experiments**
Main stakeholder	Experimenters
Secondary stakeholder	-
Description	The aim of this scenario is to validate the experiment monitoring tool. 1. The experimenter successfully logins into the RAWFIE system through the Web Portal. 2. The system starts an experiment as defined by the booking service and the experimenter starts to observe the experiment execution. 3. The status of the experiment and the corresponding resources are presented in the provided interface. 4. The experiment successfully leaves the RAWFIE system.
Metrics and success criteria	• all steps are completed without error
Involved Sub-systems	Web Portal Monitoring Tool Visualization Tool Launching Service Experiment Controller Experiment Monitoring Tool Testbed Proxy Resource Controller Testbed Manager
Validated requirement	PT-GEN-001, PT-GEN-003, PT-L-004

6.1.3 Monitoring of Water Canals

Validation scenario	**Monitoring of Water Canals**
Main stakeholder	Experimenter
Secondary stakeholder	Testbed Operators
Description	1. Experimenter has a program/algorithm for USV for monitoring water channels) 2. Experimenter looks for some testbed where some water channels and USVs are available (Resource Explorer Tool) 3. Experimenter writes the program for the USV for monitor the water channels (in EDL or platform code?) 4. Experimenter write the experiment steps with EDL, e.g. o Load monitoring program into the USV system o USV go to the beginning of the water channel o Move into the water channel o Monitor the water channel o Back to base station when done 5. Experimenter books the testbed and needed USVs 6. Experiment will be started a the given date 7. USVs execute the given script correctly 8. Measurements are sent to the database 9. Experiment finishes 10. Experimenter evaluates the results o View experiment log o examine measurements
Metrics and success criteria	• PLATFORM / 1 / STABLE SYSTEM = 100% • PLATFORM / 3 / ERRORS = 0 • PLATFORM / 4 / WARNINGS = 0 • PLATFORM / 6 / USABILITY, USER-FRIENDLINESS • UXV / 1 / CONTROLLABILITY • UXV / 2 / MISSION ACHIEVEMENT
Involved Sub-systems	Resource Explorer Tool Testbeds Directory Service Booking Tool Booking Service Experiment Authoring Tool EDL Compiler & Validator Experiment Validation Service Launching Service Experiment Monitoring Tool
Validated requirement	PT-GEN-001, PT-P-001, PT-P-003, PT-A-001, PT-A-003, PT-A-004, PT-A-008, PT-A-009, PT-A-016, PT-B-001, PT-L-002, PT-L-009, PT-E-002, PT-E-003

6.1.4 Border Surveillance or Perimeter protection of large areas

Validation scenario	**Border Surveillance or Perimeter protection of large areas**
Main stakeholder	Experimenter
Secondary stakeholder	
Description	1. Experimenter has a program/algorithm for UGVs that can detect intruders 2. Experimenter looks for a UAV and UGV testbed with an large testbed area (Resource Explorer Tool) 3. Experimenter also needs many UxVs: a. UGVs that will play the intruders b. UAVs that observe the area 4. UAVs need appropriate sensors (cameras), the mounting of these sensors must also be booked 5. The Booking Tool supports the Experimenter to find a date where enough UxVs are available 6. Experimenter books the resources 7. Experimenter write the experiment steps with EDL, e.g. a. UGVs should go to their starting position b. UAVs start the detection of intruders at the perimeter c. UGVs start “intruding” d. UAVs execute the detection program 8. Experiment will be started a the given date 9. UxVs execute the given script correctly 10. Measurements are sent to the database 11. Experiment finishes 12. Experimenter evaluates the results a. View experiment log b. Examine measurements
Metrics and success criteria	• PLATFORM / 1 / STABLE SYSTEM = 100% • PLATFORM / 3 / ERRORS = 0 • PLATFORM / 4 / WARNINGS = 0 • PLATFORM / 6 / USABILITY, USER-FRIENDLINESS • UXV / 1 / CONTROLLABILITY • UXV / 2 / MISSION ACHIEVEMENT
Involved Sub-systems	Resource Explorer Tool Testbeds Directory Service Booking Tool Booking Service Experiment Authoring Tool EDL Compiler & Validator Experiment Validation Service Launching Service Experiment Monitoring Tool

Validated requirement	PT-GEN-001, PT-P-001, PT-P-003, PT-A-001, PT-A-003, PT-A-004, PT-A-009, PT-A-016, PT-B-001, PT-L-002, PT-L-009, PT-E-002, PT-E-003, TB-G-004

6.1.5 On demand deployable Internet facilities

Validation scenario	**On Demand Internet Facilities**
Main stakeholder	Experimenter
Secondary stakeholder	-
Description	This validation scenario tackles the rapidly expanding domain of on-demand deployable Internet facilities through UxVs. The experimenter has a program/algorithm for UAVs that provide internet access over the air. The steps of the scenario is as follows: 1. The experimenter looks for a UAV testbed where the UAVs could be equipped with the radio technology to be used. 2. The experimenter books the testbed and the resources. 3. The experimenter writes the experiment steps with EDL. 4. The experiment will be started at the given date. 5. UxVs execute the given script correctly. 6. The RAWFIE platform prioritizes needs, maximizes data volumes and conserves UAVs battery by excluding devices not needed by the task. 7. Measurements are sent to the database. 8. The experiment finishes. 9. The experimenter evaluates the results through the experiment log and the provided measurements. 10. Experimenter runs the Data Analysis to find potential problems with the network coverage.
Metrics and success criteria	• all steps are completed without error
Involved Sub-systems	Web Portal Experiment Authoring Tool Monitoring Tool Resource Explorer Tool Booking Tool Data Analysis Tool Visualization Tool Users & Rights Service Visualization Engine Experiment Controller Experiment Monitoring Tool Testbed Directory Service

	EDL Compiler & Validator Data Analysis Engine Launching Service Testbed Proxy Resource Controller Testbed Manager
Validated requirement	PT-GEN-001, PT-GEN-002, PT-GEN-003, PT-P-001, PT-P-003, PT-A-001, PT-A-003, PT-A-005, PT-A-014, PT-B-001, PT-B-002, PT-B-006, PT-L-001, PT-L-004, PT-L-006, PT-E-002, PT-E-003, PT-E-005

6.1.6 Exploration & Assessment of Network Technologies Robustness

Validation scenario	**Exploration of Network Technologies Robustness**
Main stakeholder	Experimenter
Secondary stakeholder	-
Description	This validation scenario aims to assess the networking performance and robustness. The steps of the scenario is as follows: 1. The experimenter looks for a UxV testbed. 2. The experimenter books the testbed and the resources. 3. The experimenter writes the experiment steps with EDL. 4. The experiment will be started at the given date. 5. UxVs execute the given script correctly. 6. UxVs diverge from a common location. 7. UxVs switch network interfaces to secure communications. 8. Data are sent to the central node and the RAWFIE platform and stored to the database. 9. The experiment finishes. 10. The experimenter evaluates the results through the experiment log and the provided measurements. 11. Experimenter runs the Data Analysis to find potential problems with the network coverage.
Metrics and success criteria	• all steps are completed without error
Involved Sub-systems	Web Portal Experiment Authoring Tool Monitoring Tool Resource Explorer Tool Booking Tool Data Analysis Tool Visualization Tool Users & Rights Service Visualization Engine Experiment Controller Experiment Monitoring Tool Testbed Directory Service EDL Compiler & Validator Data Analysis Engine

	Launching Service Testbed Proxy Resource Controller Testbed Manager
Validated requirement	PT-GEN-001, PT-GEN-002, PT-GEN-003, PT-P-001, PT-P-003, PT-A-001, PT-A-003, PT-A-005, PT-A-014, PT-B-001, PT-B-002, PT-B-006, PT-L-001, PT-L-004, PT-L-006, PT-E-002, PT-E-003, PT-E-005

6.1.7 Efficient Coordination for phenomena or mission

Validation scenario	**Coordination for Phenomena or Mission**
Main stakeholder	Experimenter
Secondary stakeholder	-
Description	This validation scenario aims to assess the performance of the coordination a set of UxVs to observe specific phenomena or to conclude specific missions. The steps of the scenario is as follows: 1. The experimenter looks for an UxV testbed. 2. The experimenter books the testbed and the resources. 3. The experimenter writes the experiment steps with EDL. 4. The experiment will be started at the given date. 5. UxVs execute the given script correctly. 6. UxVs perform self-organization and re-organization (an incident is simulated to check the nodes behaviour). 7. Data are sent to the central node and the RAWFIE platform and stored to the database. 8. The experiment finishes. 9. The experimenter evaluates the results through the experiment log and the provided measurements. 10. Experimenter runs the Data Analysis to find potential problems with the network coverage.
Metrics and success criteria	• all steps are completed without error
Involved Sub-systems	Web Portal Experiment Authoring Tool Monitoring Tool Resource Explorer Tool Booking Tool Data Analysis Tool Visualization Tool Users & Rights Service Visualization Engine Experiment Controller Experiment Monitoring Tool Testbed Directory Service EDL Compiler & Validator Data Analysis Engine

	Launching Service Testbed Proxy Resource Controller Testbed Manager
Validated requirement	PT-GEN-001, PT-GEN-002, PT-GEN-003, PT-P-001, PT-P-003, PT-A-001, PT-A-003, PT-A-005, PT-A-014, PT-B-001, PT-B-002, PT-B-006, PT-L-001, PT-L-004, PT-L-006, PT-E-002, PT-E-003, PT-E-005

6.1.8 RAWFIE Platform Administrator scenarios

6.1.8.1 Administrators manages the user

Validation scenario	**Administrator manages the user rights**
Main stakeholder	RAWFIE Platform Administrator
Secondary stakeholder	Experimenters
Description	1. Administrator opens the user management of the Web Portal 2. Administrator searches for a given user 3. Administrator changes the rights of the given user
Metrics and success criteria	• PLATFORM / 6 / USABILITY, USER-FRIENDLINESS
Involved Sub-systems	Web Portal Users & Rights Service
Validated requirement	PT-GEN-001, PT-GEN-002

6.1.8.2 Administrator adds a new user

Validation scenario	**Administrators adds a new user**
Main stakeholder	RAWFIE Platform Administrator
Secondary stakeholder	Experimenters
Description	1. Administrator opens the user management of the Web Portal 2. Administrator clicks on “new user” 3. Administrator inserts the user data and submits the data 4. Users & Rights Service save the user 5. Information is sent to the new user via email
Metrics and success criteria	• PLATFORM / 6 / USABILITY, USER-FRIENDLINESS
Involved Sub-systems	Web Portal Users & Rights Service
Validated requirement	PT-GEN-001, PT-GEN-002

6.1.8.3 System monitoring

Validation scenario	**System monitoring and error notifications**
Main stakeholder	RAWFIE Platform Administrator
Secondary stakeholder	
Description	1. Launching Service crashes

	2. System Monitoring Service checks system state and detects that Launching Service is not running 3. System Monitoring Service sends a notification email to the administrator 4. Administrator opens the System Monitoring Tool 5. Administrator checks system state 6. Administrator restarts Launching Service via some SSH client 7. Administrator checks system state (now Launching Service is running again)
Metrics and success criteria	• PLATFORM / 6 / USABILITY, USER-FRIENDLINESS • PLATFORM / 7 / RECOVERY TIME
Involved Sub-systems	Web Portal System Monitoring Tool System Monitoring Service (Launching Service)
Validated requirement	PT-GEN-001, PT-GEN-002

6.1.9 Testbed operator scenarios

6.1.9.1 Schedule maintenance

Validation scenario	**Schedule maintenance**
Main stakeholder	Testbed Operator
Secondary stakeholder	Experimenters
Description	1. Testbed operator wants to maintain its UxVs 2. Via the Booking Tool he tries to find a period where all involved UxVs are free 3. He could not find one in the near future and decides to cancel some bookings 4. The affected experimenters are notified via email that there bookings were cancelled 5. The involved UxV where block for the period of the planned maintenance
Metrics and success criteria	• PLATFORM / 6 / USABILITY, USER-FRIENDLINESS
Involved Sub-systems	Web Portal Booking Tool
Validated requirement	

6.1.9.2 *Cancel running experiment*

Validation scenario	**Cancel running experiment**
Main stakeholder	Testbed Operator
Secondary stakeholder	Experimenters (e.g. via the Experiment Monitoring tool and Experiment Controller)
Description	1. the Testbed Operator notices that something goes wrong 2. he opens the Experiment Monitoring Tool and browsed to the experiment 3. he initiate the cancelation of the experiment via the Experiment Monitoring Tool 4. the Experiment Monitoring Tool sends the cancelation to the Experiment Controller 5. the Experiment Controller issues the appropriate commands to send the UxVs back to the port 6. the Navigation Service receives the commands and guides the UxVs back.
Metrics and success criteria	• PLATFORM / 6 / USABILITY, USER-FRIENDLINESS • UXV / 1 / CONTROLLABILITY
Involved Sub-systems	Web Portal Experiment Monitoring Tool Experiment Controller Navigation Service
Validated requirement	

6.1.10 UxV Manufacturers scenarios

Install new UxVs in a testbed

Validation scenario	**Install new UxVs in a testbed**
Main stakeholder	UxV Manufacturers
Secondary stakeholder	Testbed Operator
Description	1. UxV Manufacturer ask the Testbed Operator if new UxVs could be installed in the testbed 2. Testbed Operator agrees 3. UxV Manufacturer sends the new UxVs to the testbed site 4. UxV Manufacturer give the information about the UxVs to the Testbed Operator 5. Testbed Operator update the resource description for its testbed via the Resource Explorer 6. UxV Manufacturer and Testbed Operator configure the testbed to control the new UxVs
Metrics and success criteria	• PLATFORM / 6 / USABILITY, USER-FRIENDLINESS
Involved Sub-systems	Web Portal Resource Explorer
Validated requirement	PT-P-003, TB-G-004

6.2 Early sub-system tests and validation

Matching pilot experimentation scenarios for validation to the use cases described in D3.1 one-to-one postpones testing for validation to a very late stage of project development and requires a lot of resources. Even though RAWFIE focuses on large scale experimentation of real UxVs, it is envisaged to show some evidence that the RAWFIE platform works well in smaller scale experiments or with a reduced set of functions or components.

As a consequence of the above, at least two additional pilot experimentation scenarios have been introduced to allow for early tests and validation of sub-systems or reduced scale RAWFIE systems.

Both cases assume that all Front-end tier, middle tier and data tier components are fully functional and running. The end user can write and launch validated experiments which can be conducted using limited or no UxV resources.

In the future this section may be augmented with additional tests needed to verify the correctness of different UxVs subsystems integration to RAWFIE platform prior the phase of executing the end-user defined validation scenarios as described in 6.1.2-6.1.7.

6.2.1.1 *UxV Data Generator validation*

Validation scenario	**UxV Data Generator**
Main stakeholder	Experimenter
Secondary stakeholder	RAWFIE Platform Administrator / Testbed Operators
Description	An "UxV Data Generator" component is implemented in the lower layer of Testbed and feeds the system with messages identical the ones generated from the UxV resources. A suitable log file also verifies that commands/responses from the RAWFIE platform arrive in testbed tier in the expected format. The "UxV Data Generator" component simulates to an extent the behaviour of an UxV device implementing incrementally from basic to more complex features. The scope of this verification scenario is to give to the experimenter the ability to write and run experiments in the RAWFIE platform in the absence of UxV resources and validate that the steps of the experiment are executed in the order and time specified in the scripts. 1. Experimenter logins to the RAWFIE portal with the appropriate credentials 2. Experimenter looks for the testbeds and UxV resources (simulated resources) available 3. Experimenter uses the Experiment Authoring tool to write the experiment steps with EDL, e.g. o Ask UxV's current status and location (x1, y1) o Move to location x2, y2 o Monitor this location point o Return to the initial location 4. Experimenter books the testbed and needed UxVs 5. Experiment will be started at the given date/time 6. EDL script is executed correctly using the UxV Generator component as end device that simulates UxVs behavior 7. Measurements are sent to the database 8. Experiment finishes 9. Experimenter evaluates the results o View experiment log o examine measurements
Metrics and success criteria	• all steps are completed without error
Involved Sub-systems	Web Portal Users & Rights Service Resource Explorer Tool Testbeds Directory Service Experiment Authoring Tool EDL Compiler & Validator Experiment Validation Service Booking Tool Booking Service

	Launching Service Experiment Controller Experiment Monitoring Tool
Validated requirement	PT-GEN-001, PT-P-001, PT-P-003, PT-A-001, PT-A-003, PT-A-004, PT-A-005, PT-A-006, PT-A-008, PT-A-009, PT-A-013, PT-A-014, PT-A-016, PT-B-001, PT-L-002, PT-E-002, PT-E-003

6.2.1.2 *UGV navigation validation*

Validation scenario	**UGV navigation and monitoring**
Main stakeholder	Experimenter
Secondary stakeholder	RAWFIE Platform Administrator / Testbed Operators
Description	A UGV (a ROBOTNIK Summit XL Robot) properly navigates to the coordinates described by end-user experiments and takes some action based on its sensing capabilities (e.g. take photos when predefined coordinates where reached). The scope of this verification scenario is to provide evidence that the UxV node interacts correctly with the RAWFIE platform using the appropriate testbed components and its network communication and navigation subcomponents behave as expected. Besides the Front-end tier, middle tier and data tier this verification test assumes that the Vehicular Testbed (VT) component in the testbed tier is fully functional and running. **1.** Experimenter logins to the RAWFIE portal with the appropriate credentials **2.** Experimenter looks for the testbeds and UxV resources available **3.** Experimenter uses the Experiment Authoring tool to write the experiment steps with EDL, e.g. o Ask UGV's current status and location (x1, y1) o Move to different locations o Monitor these location points o Return to the initial location **4.** Experimenter books the testbed and needed UxVs 5. Experiment will be started at the given date/time 6. EDL script is executed correctly and UGV behaves as expected 7. Measurements are sent to the database 8. Experiment finishes 9. Experimenter evaluates the results o View experiment log o examine measurements
Metrics and success criteria	• all steps are completed without error
Involved Sub-systems	Web Portal Users & Rights Service Resource Explorer Tool Testbeds Directory Service Experiment Authoring Tool EDL Compiler & Validator Experiment Validation Service Booking Tool Booking Service Launching Service Experiment Controller Experiment Monitoring Tool

	Vehicular Testbed Resource Controller UGV node(s)
Validated requirement	PT-GEN-001, PT-P-001, PT-P-003, PT-A-001, PT-A-003, PT-A-004, PT-A-005, PT-A-006, PT-A-008, PT-A-009, PT-A-013, PT-A-014, PT-A-016, PT-B-001, PT-L-002, PT-E-002, PT-E-003

7 ANNEX 1. Validation scenario template

The templates for the validation scenarios and their evaluation metrics are described in the tables hereafter.

Table 32: Validation scenario: Scenario 1

Validation scenario	**Scenario 1**
Main stakeholder	End user 1
Secondary stakeholder	Operator, Governmental organisation for UxV regulation, …
Description	Short description of the scenario
Metrics and success criteria	Availability (d/y): 300 days per year …
Involved Sub-systems	Flight control Sensors …
Validated requirement	Ref to Req of D3.1

The validation scenario addressing a specific feature of function of the RAWFIE testbed are described in the tables hereafter.

Table 33: specific validation scenario: xxxx

Validation scenario	**Scenario 1**
Main stakeholder	End user 1
Secondary stakeholder	Operator, Governmental organisation for UxV regulation, …
Description	Short description of the scenario
Metrics and success criteria	Availability (d/y): 300 days per year …
Involved Sub-systems	Flight control Sensors …
Validated requirement	Ref to Req of D3.1

Table 34: Validation scenario: Scenario 1

Component, feature, function	**Short description**	**Metrics**	**success criteria**	**comment**

8 ANNEX 2. Component testing - how to read the verification scenarios

Even if at conceptual level in most of the case, the members of the consortium have identified all main system components, both hardware and software, and for each of them the template that will be described in the following will be adopted, in order to describe each required testing scenario.

We will assume that each component can consist of zero, one or more sub-components. If there are no sub-components the testing scenario is related to the component itself, whereas in the other cases it will be related to each sub-component.

Moreover, each component (or sub-component) can have one or more verification scenarios.

Two cases can be distinguished:

1. Only one test for the component (or sub-component). In this case the following table template is used:

Table 35: test for the component (or sub-component)

Component (parent component if applicable)	Name of the component/s involved in the verification test
Component Behaviour	If useful, please refer to the corresponding section in D4.1.
Pre-requisites	Working condition for the component in order to be able to execute the test
Test	Name of the test
Test ID	ID of the test
Test description	Condition that should be verified; Sequence of steps to perform the test (if applicable)
Expected results	The expected results from the execution of the steps described.

2. More tests for the component (or sub-component). In this case the following template is used:

   A) first template describing the component and identifying the tests that will be performed on that component, by attributing a TEST ID at each test;

Table 36: tests that will be performed on a given component

Component (parent component if applicable)	Name of the component
Component Behaviour	If useful, please refer to the corresponding section in D4.1.
Tests	List of the test IDs

	All tests in this section… As in the following the test…

B) a second template describing the specific test and the expected results.

Table 37: specific test of a given component and the expected results

Test	Name of the test
Test ID	ID of the test
Component (parent component if applicable)	Name of the component/s involved in the verification test
Pre-requisites	Working condition for the component in order to be able to execute the test
Test description	Condition that should be verified; Sequence of steps to perform the test
Expected results	The expected results from the execution of the steps described.