Deliverable Reference	:	D5.5
Title	:	Unitary & integrated test plans, and release schedule for the Planetary RI
Confidentiality Level	:	PU
Lead Partner	:	SPACEAPPS
Abstract	:	This deliverable describes detailed unitary and integrated test plans for the Planetary RI, following the detailed specification done in T5.2. Additionally, this task will produce a detailed release goals statement with associated schedule for Planetary RI specific contributions
EC Grant N°	:	730014
Project Officer EC	:	Christos Ampatzis (REA)

InFuse is co-funded by the Horizon 2020
Framework Programme of the European Union

DOCUMENT APPROVAL SHEET			
	Name	Organization	Date
Prepared and cross-reviewed by:	Jeremi Gancet Shashank Govindaraj	SPACEAPPS	30/03/2018
	Fabrice Souvannavong Vincent Bissonnette	Magellium	
	Fabrice Souvannavong Vincent Bissonnette Arnaud Moura	Magellium	23/01/2018
	Simon Lacroix Quentin Labourey Andrea De Maio	CNRS/LAAS	
	Mark Post Alessandro Bianco Romain Michalec	Strathclyde University	
	Shashank Govindaraj Jeremi Gancet Xavier Martinez	Space Applications	
	Nassir Oumer Michal Smisek	DLR	
	Raul Dominguez	DFKI	

DOCUMENT CHANGE RECORD				
Version	Date	Author	Changed Sections / Pages	Reason for Change / RID No
0.1.0	07/12/2017	Shashank Govindaraj	All	Document structure
0.2.0	15/12/2017	Shashank Govindaraj	All	Assignment of sections to partners
0.3.0	19/12/2017	Vincent Bissonnette Fabrice Souvannavong Simon Lacroix Mark Post	All	Inserting preliminary inputs to the sub-sections
0.4.0	19/01/2018	Vincent Bissonnette Fabrice Souvannavong Mark Post Simon Lacroix	Chapter 3 & 4	Release schedule, internal unitary and integrated test plans for DFNs and DFPCs
0.5.0	23/01/2018	All Partners	Chapter 4 & 5	Testing and validating procedures for DFNs/DFPCs in OG6 facilities
1.0.0	25/01/2018	Shashank Govindaraj Joseph Salini Vincent Bissonnette Jeremi Gancet	All Chapters	Final check of content and formatting the document
2.0.0	30/03/2018	Shashank Govindaraj Xavier Martinez Vincent Bissonnette Jeremi Gancet	All Chapters	Addressing RIDs emitted by the PSA

Executive Summary

This deliverable addresses the approach and methods for testing and validating DFNs and DFPCs internally, and within OG6 facilities for the Planetary Track. CDFF core methods are designed and developed to be middleware independent. In order to test the DFNs functionality and fault tolerance, they would expose their interface via a Python interface or be wrapped in a target RCOS to allow testing them internally. The release sequence of the CDFF software components for this reference implementation are highlighted for each of the 3 internal milestones at M18, M20 and M22. The procedure for testing and evaluating each DFN category and DFPCs for Orbital-RI with the OG6 DLR PEL Mars analog with ExoMars BB2 and outdoor analog in Morocco with the SherpaTT rover and Mana-Minne multi-robot setup are elaborated.

Table of Contents

List of of Figures

1 Introduction

1.1 Purpose

This document describes the detailed approach for testing and validation of CDFF software Reference Implementations (RI) of the planetary scenario. During the development phase, internal testing of the CDFF components will be conducted in phases (pre-TRR) followed by the final validation tests within OG6 facilities (post-TRR). It includes the preparation and verification of the EGSEs and their interfaces in terms of data and software, the latter being based on the design of the CDFF as described in D5.2.

The objective of planetary RI of InFuse is to demonstrate and evaluate the full capabilities of the CDFF: from space compliance to state-of-the-art algorithms, from traditional to innovative sensors, to demonstrate that the CDFF is ready to be integrated with OG1, OG2, OG4 and OG6.

Note: This document does not focus on the validation of the supporting EGSEs (i.e. deployment of CDFF on RTEMS-Leon architecture) in the planetary reference scenarios.

1.2 Structure

This document is structured as follows:

Section 1: Introductory material to the deliverable.

Section 2: Overview of unitary and integrated testing approaches

Section 3: Software deliverables release schedule

Section 4: Pre-TRR Internal (to OG3) unitary and integrated testing plans

Section 5: Post-TRR testing within OG6 facilities

1.3 Applicable documents

AD1 InFuse Grant Agreement

AD2 InFuse Consortium Agreement

AD3 InFuse internal management manual for project partners

1.4 Reference documents

RD1 Description of Action document

RD2 D3.1 Technological Review

RD3 D3.2 System requirements

RD4 D3.3 Early CDFF architecture and ICD

RD5 D4.1 Technical Trade-off Analysis

RD6 D4.2 Advanced CDFF architecture and ICD

RD7 D4.3 CDFF Unitary and integrated test plans

RD8 D4.4 Preliminary design document

RD9 D5.2 Definition and specification of the planetary RI and EGSE

1.5 Acronyms

DF: Data Fusion

CDFF: Common Data Fusion Framework

API: Application Program Interface

OOS: On-Orbit Servicing

RCOS: Robot Control Operating System

DFN: Data Fusion Node

DFPC: Data Fusion Process Chain/Compound

DFNCI: Data Fusion Node Common Interface

DPM: Data Product Manager

MW: Middleware

LOS: Line of Sight

Fps: Frames per second

OOS-sim: On Orbit Servicing simulator

OG: Operational Grant

IMU: Inertial Measurement Unit

OT: Orbital Track

PT: Planetary track

OBC: On Board Computer

DEM: Digital Elevation Model

FPGA: Field Programmable Gate Array

2 Approach for Unitary and Integrated testing

This section describes the general approach for testing independent and integrated software component which are a part of CDFF-Core and support that are complemented by the CDFF-development tools. During the life cycle of the project, testing and verification has been distributed into 2 phases - internal to InFuse and final with OG6 facilities. This has been described in D4.3 and in the project DoA documents. The facilities for internal and final, testing and validation for the planetary RI include the PELI facility at DLR and outdoor campaigns on the SEROM site at CNES, and in Morocco.

For reference, the system requirements covered by the test plan described in this document are detailed in Appendix 8.1.

2.1 OG3 Internal testing and validation of CDFF

The following section covers the strategies for testing DFNs and DFPCs. A DFPC would be assembled as a single software component consisting of a DFPC controller (Refer to deliverable D5.1) and underlying DFNs. The DFPC controller manages the sequence of triggering data fusion functions and manages the flow of data between DFNs. Hence the DFPCs being monolithic components can be tested with the same approach as DFNs.

2.1.1 Internal testing of CDFF for the RI

The testing strategy depends upon the nature of the DFN, i.e. whether (1) the DFN developed from scratch, or is it based on legacy code (i.e. re-using previously existing code), and (2) does the DFN depend on third party libraries.

In case legacy code or third-party libraries are involved, unit tests will be developed in the DFN where these legacy or third party software are exploited (e.g. calls to functions), and shall allow verifying that the interactions with that existing software are robust.

All new code will be requested to come with a comprehensive unit test coverage. Evidences of coverage rate will be requested from contributors, accordingly.

Memory usage profiling : Valgrind profiler will be used to measure the memory usage of DFNs and DFPCs in the developers and target environment (Linux based) along with detecting memory leaks in DFNs and DFPCs. The Valgrind Massif tool is a heap profiler. It measures how much heap memory your program uses and measure the size of the program's stack(s).

Computation profiling: The gperftools CPU profiler has a very little runtime overhead, provides features like selectively profiling certain areas of interest and has no problem with multi-threaded applications. This can be complemented by a free or licensed version of the Intel VTune Amplifier (https://software.intel.com/en-us/intel-vtune-amplifier-xe) for accurate profiling of the computational loads of DFNs and DFPCs.

There are 2 methods for performing functional testing and validation.

2.1.2 Python bindings with data flow control

The DFNs/DFPCs are designed and developed to be middleware independent. In order to test the DFNs functionality and fault tolerance, DFNs would expose its interface via a Python interface (via C/C++ Python bindings) to allow testing DFNs. The DFN Python interface would connect to the data flow control within InFuse which is based on Python Pandas for reading sample sensor data from csv like files and logging the output data.

Figure 1 shows the steps required to expose the C/C++ interfaces of a DFN via an equivalent Python interface. This approach facilitates integration with the data flow control relatively fast to validate functions and behavior of the DFN logic. The steps are described as follows:

- A C/C++ library (or source code) of the DFN is considered to be developed and available for testing
- The interface for the DFN is available in the header (*.h or *.hpp) file.
- Components of the DFN interface that needs to be exposed is encoded as a Python SWIG interface (within a *.i file)
- The Swig tool chain uses the swig interface file to generate an equivalent Python interface definition file and C++ wrapper file for Python bindings.
- The generated C++ code needs to be compiled and linked to the Python interface definition file.
- In the final steps, a Python test suite program needs to be developed that can include the Python interface definition file and use it to access the DFN interface for testing them with the data flow control.

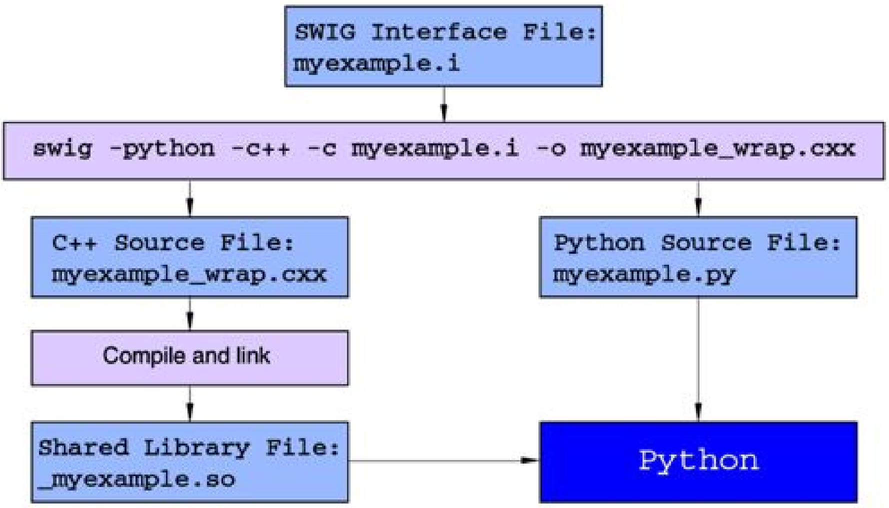

Figure 1: Workflow for exposing DFN interfaces through Python for testing off-board

2.1.3 As a robotics middleware component

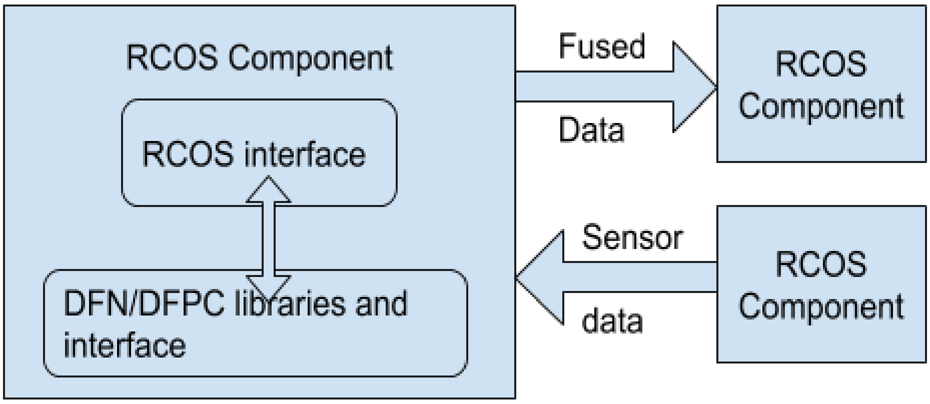

Figure 2: DFNs/DFPCs tested as RCOS components

A secondary approach would be to integrate the CDFF-core DFN into a specific robotics control software (RCOS) component and use the log replay mechanisms to access sensor data or data samples that are intermediate to the DFPC. The proposed robotics middleware within InFuse include ROS, RoCK, YARP and GenoM3. This approach would also demonstrate the ability of incorporating DFNs within multiple middleware components

2.2 Testing and validation of CDFF in OG6 facilities

The approach for validating CDFF software components are driven by OG6 testing and validation facilities, associated sensors, software and data interfaces within the representative analog. In the case of InFuse, the DLR PELI Mars analog with ExoMars BB2 will be used for PT validation respectively. A final outdoor validation scenario is foreseen for PT in Ibn Battuta, Morocco, with the deployment of CDFF in a single robot setup on the SherpaTT rover and a multi-robot setup with the Mana & Minnie mobile robots. DFPCs developed within the OT-RI (reference to D5.1) will be evaluated between M23-24 of the project in DLR OOS-Sim facility. The scenarios and use-cases identified in D5.1 will be elaborated in the following section. The testing procedure would be carried out in an open loop manner i.e commanding manipulators are not influenced by CDFF data fusion outputs. There are 2 approaches for validating CDFF software with open loop operations (i) on-line data acquisition and fusion (ii) on-line recording of data and offline data fusion.

The OOS-Sim facility is equipped with a set of baseline sensors and calibrated in the current configuration (IMU, Stereo-camera). Additional sensors for validating OG3 would need to be mounted and calibrated for InFuse such as 3D Lidar, ToF camera and/or Force/Torque sensor. The OBC for this setup will consist of the existing computer infrastructure at the facility for commanding and acquiring data from the OOS-Sim manipulators that would be interfaced with an additional computer running DFPCs via an ethernet interface.

The DFPCs would be wrapped in the target middleware (typically ROS) and deployed on the OBC to acquire data sensors to perform on-line or off-line data fusion. Parametric and model specifications are required for specific DFPCs that will be addressed in the following sections of this document. Existing sensor data acquisition interfaces will need to be customized for transferring data to the computer hosting CDFF for online data fusion.

2.3 Recap of Planetary RI scenarios

The planetary RI is designed to address the following scenarios :

- Long Traverse Localisation,
- Long Traverse DEM,
- Rendezvous,
- Return to Base.

The long traverse scenario has been split into two scenarios, addressing respectively localisation and DEM data products.

The objective of the Long Traverse mission is to autonomously reach a target located about 1 km away, defined by its absolute coordinates. The localisation function is a key element for the success of the mission. Indeed, localisation data is used by three components of the system: trajectory control, fusion of navigation maps (or DEM), and localisation of the target with respect to the rover.

The Long Traverse DEM shares the same objectives, but also intends to produce 3 types of DEMs:

- The rover map: this is the map of the surroundings of the rover at each observation. It is attached to the reference frame of the rover.
- The fused rover map: this is the fusion of rover maps over a given period of time. It is attached to a local reference frame (site frame).
- The total fused map: this is the fusion of all rover maps to create a complete map of the robot surrounding. It is attached to the planet’s reference frame, and can be used in future journeys of the rover.

The Rendezvous mission objectives consist in guiding the rover towards a precise position and orientation with respect to a man-made asset, e.g. the sample analysis module, for instance, to perform the transfer of a soil sample. The target is considered non-cooperative, as there is no direct communication between it and the rover.

Finally, the Return to Base mission consists in autonomously executing a trajectory that has been executed beforehand (*e.g.* after a long traverse, or after having fetched a sample, or performed a scientific analysis), but in the opposite direction. The localisation system can thus benefit from the use of maps created during the first pass.

For a detailed mapping of which DFPC is applied to each use case, as well as which DFN is included in which DFPC, the reader is invited to refer to D5.2 - Section 4.1.

2.4 Preparation of datasets

Two sources of simulation data, with different uses, are foreseen to be used. Firstly, an existing CNES planetary rover simulator may be used to test and validate a subset of the DFPCs to be developed, specifically the ones using stereo camera data as input (e.g Visual Odometry, Visual SLAM, Visual Map-based Localisation, 3D Model Detection and Tracking). Secondly, the MORSE simulator will be used, on the one hand, to test and validate the DFPCs by generating all other types of sensor data (cameras, LiDARs, IMUs, ToF) and, on the other hand, to validate the functional interfaces between the DFPCs, the sensors, and the robot actuators. Indeed, the MORSE simulator is designed to emulate the exact hardware configurations and interfaces as the actual Mana and Minnie rovers to be used during the 3M validation track.

It is foreseen that several datasets captured by DLR LRU (Lightweight Rover Unit) might be available for InFuse internal testing in the form of ROS bag files. These datasets contain visual odometry, wheel odometry, IMU measurements as sensor input data. ROS topic specification should be provided. Datasets were captured on Etna volcano on Sicily. Some of the datasets were captured during the preparations for ROBEX field test, in September 2016, and some of them were captured during the final ROBEX field test demonstration, in June/July 2017. Ground-truth localization information was captured by tachymeter in the former case and by DGPS in the latter case, and will ideally be provided alongside the datasets.

3 Release Schedule

3.1 DFPCs (with associated DFNs)

Priority	**LAAS-CNRS**
M18	DFPC Wheel Odometry for the robots Mana and Minnie DFPC DEM, without the associated DPM services, i.e. no possibility to satisfy the “produce Total Rover Map” request DFPC Position Manager, with fusion of Wheel Odometry and Visual Odometry
M20	DFPC Visual Odometry (LAAS flavor) DFPC Pose graph SLAM DFPC Lidar Map-based Localization Update of DFPC Position Manager, with fusion of SLAM estimates
M22	DFPC Absolute Localisation Update of DFPC DEM, with all the associated DPM services Update of DFPC Position Manager, with all the associated Localisation DPM services

Priority	**USTRATH**
M18 (High)	Feature Detection and Matching
M20 (Medium)	Point Cloud Triangulation and Construction Point Cloud Model-Based Localization
M22 (Low)	Bundle Adjustment and Optimization

Priority	**MAG**
M18 (High)	Navigation Map Building Visual Odometry - MAG/CNES
M20 (Medium)	Visual SLAM Visual Map-based Localisation Mid-range 3D Model Detection Mid-range 3D Model Tracking

M22 (Low)	Long-range Tracking Point Cloud Model-Based Localisation

Priority	**DLR**
M18 (High)	Camera calibration, Stereo camera calibration
M20 (Medium)	Body-to-camera calibration
M22 (Low)	N/A

Priority	**SPACEAPPS**
M18 (High)	DFPC Haptic scanning
M20 (Medium)	DFPC 3D Model Detection
M22 (Low)	DFPC 3D Model Tracking

3.2 CDFF-Support and Dev utilities (iterative updates)

This section describes the release schedule for CDFF-Support and CDFF-Dev sub-system components. A list of features or capabilities (based on requirements) that will be developed and released is provided.

Schedule	**Orchestrator features or capabilities**
M18	● Identify the queries from OG2 and associated parameters ● Handle essential queries from OG2 to map into corresponding DFPC ● Trigger start and stop of DFPCs based on OG2 query ● Monitor and log basic run-time status of DFPCs
M20	● Interface Orchestrator with the DPM at the DFPC level to acquire fused data and forward it to OG2 ● Command OG4 sensor operating modes, implement handshaking sequence and monitor error states

	● Acquire sensor data from OG4 sensors via the mock data acquisition interface
M22	● Interface Orchestrator with the central DPM ● Notification of run-time states to OG2 for intervention of unknown states. ● Monitor complete DFPC run time and error states and correction (FDIR) ● Commanding of all OG4 operation modes and associated FDIR ● Running multiple DFPCs simultaneously exchanging data via the central DPM.

Schedule	**DPM capabilities**
M18	● Position Manager DFPC: ability to fuse Wheel Odometry and Visual Odometry
M20	● Update of Position Manager DFPC: ability to fuse SLAM pose estimates
M22	● Final version of Position Manager DFPC: ability to fuse and manage the outputs of all the localisation DFPCs ● DFPC DEM, with all the associated DPM services

Schedule	**Visualiser features or capabilities**
M18	● Offline visualization example of application with small subset of ESROCOS types (e.g. pointclouds and transformation)
M20	● Offline visualization for the sensor data types (ESROCOS types) that OG4 provides.
M22	● Offline visualization of logged data for all InFuse data products.

Schedule	**Data flow control features or capabilities**
M18	● Offline replay of the execution of an example DFCP based on logged data from this same DFPC.
M20	● Automatic reconstruction of any implemented DFPC for offline replay from its description file.

M22	• Offline replay of the execution of any DFCP based on logged data. • Replacement of DFNs from the original DFPC to alternative ones for evaluation.

Schedule	Filtering and outlier removal features or capabilities
M18	• Identify use cases for outlier removal • Implement prototypes of outlier removal methods
M20	• Implement outlier removal methods in C++ • Integration of C++ components in pySPACE
M22	• Evaluate outlier removal methods with data from robot

Schedule	MW Facilitator features or capabilities
M18	• Automatic generation of the DFN Common Interface and of the DFPC Common Interface
M20	• Generation of the required code for running a DFPC in a Rock RCOS • Experimental code generation of ROS wrappers for DFNs and experimental assembly DFN ROS nodes to create a distributed DFPC (build and launch files to be evaluated)
M22	• Rock RCOS integration of the Orchestrator • Rock RCOS integration of the Central Data Products Manager • Generation of the required code for running a complete Data Fusion Solution for the Planetary Track in a Rock RCOS • Building the generated DFN ROS Nodes and run time testing of interfaces • Testing the ROS node DFNs (and possible configuration of a distributed DFPC) with sensor data from ROS bags and evaluating outputs

3.3 Development plan

Organisation

The development period until the TRR is decomposed in 3 weeks sprints with two main objectives :

- 18/06 - experiments @CNES :

    - online validation of key algorithms,
    - acquisition campaign for algorithms evaluation.
- 20/08 - algorithms evaluation and test report filled.

In order to ensure a proper collaboration of partners and facilitate the cross integration of DFN and DFPC, intermediate synchronisation points are foreseen :

- 16/04 : All partners to deliver key DFPCs for online validation with the simulator,
- 07/05 : All partners to deliver calibration tools to be used on Rovers.
- 28/05 : Integration days @LAAS.
- 30/07 : All DFPS / DFNs provided.

Partners detailed road map

The detailed roadmap is presented in the document :

"INFUSE-PDV-020-MAG-v01.10-PlanetaryTrack-TRR-planning".

4 Internal unitary and integrated testing (pre-TRR)

4.1 Unitary test plan with data flow control

4.1.1 DFNs

Describes the procedure for testing and evaluating each DFN category.

4.1.1.1 DFN: Image Pre-Processing (USTRATH)

Inputs: - Image - Camera Calibration Matrix - Distortion Coefficients - Downsampling Factor	**Outputs:** - Downsampled, Calibrated and Undistorted image
Test procedure: - Run the node on desired image with calibration matrix and coefficients as parameters - Evaluate resulting image	**Evaluation criteria or metrics:** - Image should be the exact resolution requested - Image should exhibit distortion of no more than 10% difference in relative distortion compared high-quality

	undistorted and calibrated reference of the same scene. - Note that no reference is perfect

4.1.1.2 DFN: Feature Detection (USTRATH)

Inputs:	**Outputs:**
- Image	- List of features from image (Harris or ORB)
Test procedure: - Run the node on desired undistorted and downsampled image - Check that features have been detected	**Evaluation criteria or metrics:** - The number of features detected should be within 10% of the number of features in a reference case for feature detection obtained by a standard detection algorithm (open source and in OpenCV) implemented for a given descriptor. These open-source descriptor implementations are considered to be industry standard, only parameters are expected to cause differences between implementations of the same descriptor on the same data. - 90% of manually detected features should lay within a 5 pixel distance of the a detected feature. Manually detected features are considered to be features identified by a human inspecting a close-up of an image to establish whether a feature is centered correctly on a part of the image. A human will inspect features to identify any that are misplaced on a part of an image.

4.1.1.3 DFN: Feature Matching (USTRATH)

Inputs:	**Outputs:**
- First Image - Features in First Image - Second Image - Features in Second Image	- List of matches between features (using Brute Force or FLANN)
Test procedure:	**Evaluation criteria or metrics:**
- Run the node on desired pair of images with features - Check that matches have been detected	- The number of features detected should be within 10% of the number of features in a reference case for feature detection obtained by a standard detection algorithm (open source and in OpenCV) implemented for a given descriptor. These open-source descriptor implementations are considered to be industry standard, only parameters are expected to cause differences between implementations of the same descriptor on the same data. - 90% of manually defined matches on manually defined features should be detected correctly. Manually defined matches are considered to be matches between features made by a human on close inspection of a set of features on a pair of images. A human will inspect each set of matches and identify those that are incorrect.

4.1.1.4 DFN: Fundamental Matrix Calculation (USTRATH)

Inputs:	**Outputs:**
- Set of matches between two images	

	- Calculated fundamental matrix describing homography between images
Test procedure: - Run the node on set of matches - Check that fundamental matrix has been calculated	**Evaluation criteria or metrics:** - A valid Fundamental Matrix should be found between two sets of features that are correctly located (validated by the eyes of a human) and correctly matched (validated by the eyes of a human). - There is no measure applicable for accuracy between two images, this is just mathematical validation and a fundamental matrix is not guaranteed to be found if all matches are not correct.

4.1.1.5 DFN: Triangulation (USTRATH)

Inputs: - Set of matches between two images - Fundamental matrix	**Outputs:** - Local point cloud describing triangulated point locations in space
Test procedure: - Run the node on set of matches and valid fundamental matrix - Check that point cloud is valid	**Evaluation criteria or metrics:** - Triangulated points are guaranteed by the algorithm to be placed within the field of view of the camera. - No measure is applicable for validation, only the production of a triangulated point is considered to be a success.

4.1.1.6 DFN: Point Cloud Assembly (USTRATH)

Inputs: - Local point cloud - Global point cloud	**Outputs:** - Global point cloud incorporating new points
Test procedure: - Run the node on local and global point cloud - Check that common points have been merged and new points have been added	**Evaluation criteria or metrics:** - Resulting point cloud should be within the expected bounds of error described in D5.2. Expected performance is no more than 10% outliers as estimated by a human inspecting the point cloud, position estimation less than 1% of R, where R is the maximum operational distance of the camera/sensor, and 90% similarity in shape to the object viewed with less than 10% error in dimensional analysis (only for components larger than 10% of the total size of the object)

4.1.1.7 DFN: 3D Keypoint Selection (USTRATH)

Inputs: - Point cloud - Keypoint descriptor radius - Keypoint spacing	**Outputs:** - Keypoints and 3D descriptors for point cloud (SHOT or FPFH)
Test procedure: - Run the node on point cloud - Check that keypoints have been selected within point cloud	**Evaluation criteria or metrics:** - Keypoints should be selected at regular locations throughout the point cloud as viewed by a human observer. “Regular” refers to keypoints not exhibiting gaps or clustering exceeding 20% of the

	average separation between keypoints.

4.1.1.8 DFN: 3D Descriptor Matching (USTRATH)

Inputs:	**Outputs:**
- 3D keypoint descriptors for scene - 3D keypoint descriptors for model	- Matches between descriptors (Brute Force or FLANN)
Test procedure: - Run the node on scene and model - Check that appropriate matches have been made	**Evaluation criteria or metrics:** - 90% of manually defined matches on manually defined features should be detected correctly. Manually defined matches are considered to be matches between features made by a human on close inspection of a set of features on a pair of images. A human will inspect each set of matches and identify those that are incorrect.

4.1.1.9 DFN: 3D Correspondences (USTRATH)

Inputs:	**Outputs:**
- Scene point cloud - Model point cloud - List of matches between model and scene	- List of potential poses in order of likelihood (number of matches)
Test procedure: - Run the node on scene and model with matches - Check that poses are reasonable	**Evaluation criteria or metrics:** - The matched model within the scene should be located and oriented within 10% of the size and relative orientation of the model.

	- Distance between ground truth pose and estimate pose should not be higher than 5% of the operating distance.

4.1.1.10 DFN: Bundle Adjustment (USTRATH)

Inputs:	**Outputs:**
- Existing Target Point Cloud - Matches for features in original input images - Original input 2D images	- Point cloud with improved geometric consistency
Test procedure: - Run the node on point cloud - Check that resulting point cloud has improved geometry	**Evaluation criteria or metrics:** - Relative geometric size and orientation of the point cloud should be quantitatively closer to the size and orientation of the actual scene with respect to the un-adjusted point cloud. - Point cloud should be located and oriented within 10% of the size and relative orientation of the scene. - Outlier percentage of points in point cloud should be no more than 10% with outliers estimated by a human viewing the resulting point cloud.

4.1.1.11 DFN: 3D Point Computation (MAG)

Inputs:	**Outputs:**
- Detected Harris features - Previously created disparity map	- 3D position of the detected Harris features

Test procedure:	**Evaluation criteria or metrics:**
- Run the node on a given set of Harris features, obtained by imaging a mapped terrain, or by simulation, - Check that reconstructed 3D points fit with ground truth terrain data.	- Standard deviation of euclidean distance between estimated 3D points and ground truth terrain model shall be within the bounds set by the used stereobench parameters (baseline, camera resolution, focal length, point depth).

4.1.1.12 DFN: 3D-3D Motion Estimation (MAG)

Inputs:	**Outputs:**
- 3D points - Previous 3D points	- Relative pose between point clouds
Test procedure: - Run node with 2 3D point clouds between which the relative motion is known, - Evaluate std of reprojection error in sensor frame.	**Evaluation criteria or metrics:** - RMS reprojection error after motion estimation should be lower than 1 pixel in sensor frame.

4.1.1.13 DFN: DEM building (MAG)

Inputs:	**Outputs:**
- Disparity map	- DEM
Test procedure: - Run node with disparity map - Check that output matches with the disparity map,	**Evaluation criteria or metrics:** - Similarity of computed DEM to actual scene

- In simulation or real test ground, check that the computed DEM matches with the ground truth DEM.	- The standard deviation of the distance between DEM and ground truth terrain is within 5 cm.

4.1.1.14 DFN: DEM Fusion (MAG)

Inputs:	**Outputs:**
- Current DEM - Previous DEM	- Updated DEM
Test procedure: - Run node on DEMs - Check that the output DEM is updated with the current DEM, - Evaluate fusion error wrt to DEM ground truth.	**Evaluation criteria or metrics:** - The DEM updated part is at the proper position, - The standard deviation of the distance between fused DEM and ground truth terrain is within 5 cm.

4.1.1.15 DFN: FeatAndSigExtractor (MAG)

Inputs:	**Outputs:**
- Depth image - RGB Image	- Keypoints with descriptors
Test procedure: - Run the DFN on a set of images exhibiting slight differences in translation, rotation and scale, representative of rover motion. - Use detection time to estimate the cost of any feature extractor.	**Evaluation criteria or metrics:** - A sufficient number of features is extracted in an individual image, - Feature extraction is sufficiently robust to homography transformations representative of rover movement (10 cm displacement, 10° rotation), - We expect to recover 50% of detected features with 1-pixel accuracy.

	- Descriptor extraction time is compatible with the 1Hz target update rate of the full SLAM chain.

4.1.1.16 DFN: Feature Matching (MAG)

Inputs:	**Outputs:**
- 3D points - Previous 3D points - Current pose from odometry	- Matched 3D points between current and previous 3D points
Test procedure: - Run the DFN on a set of images exhibiting slight differences in translation, rotation and scale representative of rover motion, with known detected keypoints. - Check that 3D point matches are valid. - Evaluate robustness to image variations.	**Evaluation criteria or metrics:** - Feature matching is sufficiently robust to homography transformations representative of rover movement (10 cm displacement, 10° rotation), - We expect to match 60% of detected features correctly. - Matching time is compatible with the 1Hz target update rate of the full SLAM chain.

4.1.1.17 DFN: Image Geometric Correction (MAG)

Inputs:	**Outputs:**
- Image - Camera Calibration Matrix - Distortion Coefficients - Downsampling Factor	- Downsampled, Calibrated and Undistorted image
Test procedure: - Run the node on a set of images containing a standard camera calibration target with calibration	**Evaluation criteria or metrics:** - Image should be the correct resolution and qualitatively undistorted,

matrix and coefficients as parameters, - Evaluate resulting image.	- Corners on the standard calibration target (chessboard or other) are reprojected with a RMS error lower than 0.5 pixels.

4.1.1.18 DFN: Loop Closure (MAG)

Inputs: - Previously created SLAM map of given environment, - 1 chosen image frame where a loop should be detected.	**Outputs:** - Loop detection success, - Corrected SLAM map.
Test procedure: - Run the DFN on a previously built SLAM map and a chosen frame in which a loop closure is likely to be detected. - If a closure is detected, the DFN computes a correction and propagates it to the entire SLAM map. - Evaluate the effect of the applied correction on rover poses.	**Evaluation criteria or metrics:** - Loop detection success - Relative rover position error on the whole SLAM map wrt to ground truth, computed after loop closure correction propagation is below 2%.

4.1.1.19 DFN: Mapper (MAG)

Inputs: - Shared SLAM map - Current Frame	**Outputs:** - Map points - Key frames
Test procedure: - Run the DFN on a previously built SLAM map (M) and a chosen frame (F).	**Evaluation criteria or metrics:** - Successful integration of the current frame in the map.

- Check if the computed keyframes are coherent with the frame F. - Check if current frame and corresponding points have been added to the map.	- Successful culling of redundant keyframes in the map. - Quantitative effects are evaluated at the DFPC level.

4.1.1.20 DFN: Navigation Map Building (MAG)

Inputs: - DEM	**Outputs:** - Navigation Map
Test procedure: - Run node on a reference DEM with known obstacles (slopes, steps) - Check that features are correctly detected given the input rover parameters(max slope, rover size and margins).	**Evaluation criteria or metrics:** - Manual verification that obstacles are detected or not wrt their size, - Manual verification that slopes are detected or not wrt their inclination, - Check that a traversable path is correctly detected between obstacles separated by a known distance.

4.1.1.21 DFN: Navigation Map Fusion (MAG)

Inputs: - Current Navigation Map - Previous Navigation Map	**Outputs:** - Updated Navigation Map
Test procedure: - Run node on several standalone navigation maps, - Then, perform the same evaluation on the fused maps as with a single map: - Run node on a reference DEM with known obstacles (slopes, steps)	**Evaluation criteria or metrics:** - Manual verification that obstacles are detected or not wrt their size, - Manual verification that slopes are detected or not wrt their inclination, - Check that a traversable path is correctly detected between

- Check that features are correctly detected given the input rover parameters(max slope, rover size and margins).	obstacles separated by a known distance.

4.1.1.22 DFN: Point Cloud Registration (MAG)

Inputs:	**Outputs:**
- Initial pose estimation - 2 input point clouds	- Transformation minimizing the distance between the two input clouds
Test procedure: - Provide two point clouds of varying densities with known transformation and the initial pose estimation, - Check if the computed transformation corresponds to the ground truth.	**Evaluation criteria or metrics:** - Euclidean distance is around 5% of R, where R is the maximum operational distance of the camera, and angular distance is around 10°.

4.1.1.23 DFN: Point Tracking (MAG)

Inputs:	**Outputs:**
- Current set of detected image features - Previous set of detected image features - Coarse relative pose estimation between previous and current frames.	- Matched set of 3D points in previous and current image frames
Test procedure: - Run the DFN on a chosen set of detected features and related pose estimation, - Evaluate matching performance.	**Evaluation criteria or metrics:** - Ratio of matched features over detected features is 60% over 2 consecutive images, - Tracking accuracy (reprojection error) is lower than 1 pixel.

4.1.1.24 DFN: Pose Estimator - Simple Predictor (MAG)

Inputs: - Rover pose estimation from wheel odometry,	**Outputs:** - Rover pose prediction
Test procedure: - Run the DFN on a current rover pose, with detected 3D landmarks. - Evaluate estimated pose prediction, through reprojection error, assuming geometries are preserved within the 3D reconstruction accuracy of the stereo process.	**Evaluation criteria or metrics:** - RMS reprojection error of known features in image is lower than 10 pixels.

4.1.1.25 DFN: Relocaliser (MAG)

Inputs: - Current Keypoints - Previous frame Keypoints - Image features (bag-of-words model)	**Outputs:** - Estimated initial pose
Test procedure: - Run the node on keypoints - Check the two scenarios: tracking was successful in the previous frame, tracking was not successful, - Validate that the pose estimation is correct.	**Evaluation criteria or metrics:** - Compute the standard deviation between the estimated pose and ground truth. - The target localisation accuracy after relocalisation step is within 0.5m of the closest and detected taught image.

4.1.1.26 DFN: Stereo Rectification (MAG)

Inputs: - Stereo Image Pair	**Outputs:** - Rectified Stereo Image Pair
Test procedure: - Run the node on a stereo image pair - Check that the image planes are aligned (rotation, focal length).	**Evaluation criteria or metrics:** - Compute the standard deviation of the distance between corresponding epipolar lines., identified using a standard camera calibration target (chessboard), is below 0.5 pixels.

4.1.1.27 DFN: Stereo Correlation (MAG)

Inputs: - Rectified Stereo Images Pair	**Outputs:** - Image of Filtered Disparities
Test procedure: - Run the node on a rectified stereo image pair - Check that the Disparity Image corresponds to the observed environment, either in simulation or real images.	**Evaluation criteria or metrics:** - In simulation, compute the standard deviation of the distance between the the ground truth depth and the depth computed by the disparity image, - In real images, compute the standard deviation of the distance between a localised plane (standard camera calibration target) and the computed stereo depth. - The distance should be within the 3D reconstruction accuracy bounds dictated by parameters of the stereobench used (baseline, camera resolution, focal length, point depth).

4.1.1.28 DFN: Template Detection (MAG)

Inputs:	**Outputs:**
- Target template - RGB image - Depthmap	- Target template pose
Test procedure: - Provide an RGB image in which the template position is known - Run the DFN to perform detection, - Verify success of detection and check pose accuracy.	**Evaluation criteria or metrics:** - Detection success rate is higher than 80%. - Coarse pose estimation is within angular and position bounds of the spatial sampling of the template.

4.1.1.29 DFN: Template Tracking (MAG)

Inputs:	**Outputs:**
- RGB Image - Initialization pose	- Target pose
Test procedure: - Run DFN on RGB image sequence, - Evaluate tracking success rate and pose estimation accuracy.	**Evaluation criteria or metrics:** - Successful tracking from 50m to 1m range, to allow for further rendezvous operations. - Final relative pose error should be lower than 0.1m in position, and 10° in angle.

4.1.1.30 DFN: Tracker (MAG)

Inputs:	**Outputs:**
- Camera pose - Detected keypoints - SLAM map	- Optimized estimated camera pose - Subset of detected keypoints

Test procedure:	**Evaluation criteria or metrics:**
- Run node with known camera pose, - Evaluate output camera pose with ground truth,	- Optimized tracking accuracy (reprojection error of known keypoints wrt to ground truth pose) is lower than 1 pixel.

4.1.1.31 DFN: Background learning (SPACEAPPS)

Inputs:	**Outputs:**
- 2D depth Image	- 2D background Image
Test procedure:	**Evaluation criteria or metrics:**
- Run the node with sample data containing a background and a foreground object	- The resulting image does not contain the foreground depth information

4.1.1.32 DFN: Depth Filtering (SPACEAPPS)

Inputs:	**Outputs:**
- 2D depth Image	- 2D filtered depth Image
Test procedure:	**Evaluation criteria or metrics:**
- Run the node, Send a depth image containing outliers	- The resulting image has a higher PSNR than the input image - The outliers have been removed from the input image

4.1.1.33 DFN: Depth Normalisation (SPACEAPPS)

Inputs:	**Outputs:**
- 2D depth Image	- 2D Normal map
Test procedure: - Generate a normal map from a height map - Run the node with sample data	**Evaluation criteria or metrics:** - The resulting image contains normals extracted from the depth Image, the resulting image and ground truth normal are equivalent.

4.1.1.34 DFN: World Transformation (SPACEAPPS)

Inputs:	**Outputs:**
- 2D depth Image - Relative Reference Frame - Absolute Reference Frame	- 2D depth Image Transformed
Test procedure: - Run the node with sample data	**Evaluation criteria or metrics:** - The resulting image contains absolute positions instead of relative positions equivalent to the inverse transformation from relative to absolute reference frame.

4.1.1.35 DFN: Voxelization (SPACEAPPS)

Inputs:	**Outputs:**
- 2D depth Image	- Voxel sparse datastructure - Octomap

Test procedure:	**Evaluation criteria or metrics:**
- Run the node with sample data	- Every input point has been binned into the Voxel sparse data structure. - An octomap has been produced.

4.1.1.36 DFN: K-Means (SPACEAPPS)

Inputs:	**Outputs:**
- Octomap	- Array of K regions
Test procedure: - Run the node with sample data - Change the number of possible regions (K)	**Evaluation criteria or metrics:** - For every node present in the octomap, the Node is present in one of the K regions. - K regions are present with data for each region

4.1.1.37 DFN: Feature Extraction (SPACEAPPS)

Inputs:	**Outputs:**
- Octomap	- Vector of features
Test procedure: - Run the node with sample data twice with rotated and translated data.	**Evaluation criteria or metrics:** - Feature extraction is deterministic and identical for the two datasets.

4.1.1.38 DFN: Edge Extraction (SPACEAPPS)

Inputs:	**Outputs:**
- 2D depth Image	- 2D Edges Image
Test procedure: - Run the node with sample data first. - Re-run the node with the output of the first round.	**Evaluation criteria or metrics:** - On first round an edge Image has been produced. - On second round, an image identical to the input image has been produced.

4.1.1.39 DFN: Primitive Matching (SPACEAPPS)

Inputs:	**Outputs:**
- 2D depth Image - Primitives to Match	- Array of primitives ordered by matching probabilities
Test procedure: - Run the node with sample data containing a planar surface and an elliptical object - Require a plane and an ellipse to be found. - Run the node with outliers	**Evaluation criteria or metrics:** - The array returns positions for primitives near the locations of the planar surface and the elliptical object. - The output array does not change with outliers.

4.1.1.40 DFN: Weighting expert (SPACEAPPS)

Inputs:	**Outputs:**
- Array of primitives ordered by matching probabilities - 2D Edges Image - 2D Depth Image - Vector of features	- Pose estimation of the target object - Probability of pose estimation

- Array of K regions - Octomap	
Test procedure: - Run the node with sample data	**Evaluation criteria or metrics:** - Compare the obtained position with ground truth position

4.1.1.41 DFN: Levenberg-Marquardt fitter (SPACEAPPS)

Inputs: - Pose estimation of the target objects - Probability of pose estimation - Joint constraints between target objects	**Outputs:** - Kinematic Pose
Test procedure: - Run the node with sample data - Input URDF file describing geometric relation between objects (eg gripper should be connected to arm)	**Evaluation criteria or metrics:** - Ground truth with known kinematic pose of a robotic arm.

4.1.1.42 DFN: Contour Matching and Pose Estimation (SPACEAPPS)

Inputs: - Image edge and normals - Model contours	**Outputs:** - Position and orientation
Test procedure: - Input model file and parse it - Input tracker and camera parameter - Run the node with sample image data	**Evaluation criteria or metrics:** - Compare ground truth with known pose of a robotic arm. With estimated pose

4.1.1.43 DFN: Force Mesh Generator (SPACEAPPS)

Inputs:	**Outputs:**
- Desired end-effector position - End-effector force measurements - Estimated rover pose relative to target.	- 3D point cloud representing touched shapes
Test procedure: - Run the node with sample data - A ground truth pointcloud will be used.	**Evaluation criteria or metrics:** - The 3D point cloud generated is a subpart of the ground truth pointcloud.

4.1.2 DFPCs

This section describes procedures for testing DFPCs applicable to PT- RI.

A DFPC test report will be produced after each test DFPC testing session, reporting on: the protocol followed, DFNs involved, expected results vs. obtained results, ref and location (on server) of the log of the tests, and any remarks and recommendations in relation to the results.

4.1.2.1 DFPC: LAAS Visual odometry

Inputs:	**Outputs:**
- Left and Right images at two consecutive positions + metadata - Estimated rover motion (*e.g.* as produced by wheel odometry)	- Estimated rover motion between the two consecutive positions
Test procedure: - Teleoperate the robot while the visual odometry module is running and compare the estimated poses with a ground truth provided by a GPS RTK	**Evaluation criteria or metrics:** - Error between the cumulated estimated motions and GPS RTK ground truth - Relative position error shall be below 2.5% of the travelled distance,

	- Drifting over distance depending on terrain - Computation performance of algorithm (pose update rate)

4.1.2.2 DFPC: MAG/CNES Visual odometry

Inputs: - Left and right images + metadata - Estimated rover pose from wheel odometry.	**Outputs:** - Estimated rover pose
Test procedure: - Teleoperate the robot while the visual odometry module is running and compare the estimated pose with a ground truth provided by an RTK-GPS receiver.	**Evaluation criteria or metrics:** - Relative error between estimated pose and GPS RTK ground truth, - Relative position error shall be below 2% of the travelled distance, - Drifting over time depending on terrain, - Computation performance of algorithm (pose update rate).

4.1.2.3 DFPC: Absolute localization

Inputs: - Fused rover map, enriched with a luminance layer if the used point clouds are produced by stereovision - Initial pose estimate at previous time	**Outputs:** - Estimated rover pose in the orbiter map frame of reference

Test procedure:
- The orbiter map is:
    - In CNES SEROM: the ground truth provided by CNES
    - In Morocco: the orbiter map will be built by the drone MAYA (TBC)
	Evaluation criteria or metrics:
	- Error between estimated pose and GPS RTK ground truth,
	- Localization error shall be on the order of the DEM resolution spatial

- Drive the robot around a given place to build a Fused Rover Map (with the DEM DFPC) -- alternatively, build a DEM from a point cloud panorama acquired at a single position - Activate the DFPC with the built DEM	- Influence of the terrain type on the required surface of rover mapped area - Computation performance of algorithm

4.1.2.4 DFPC: DEM Building

Inputs: - LIDAR point cloud OR Stereo point cloud - Rover pose ground truth	**Outputs:** - Rover Map - Fused Rover Map - Fused Total Rover Map
Test procedure: - Teleoperate the robot while running the DEM building DFPC, using the position ground truth, and compare the resulting maps with a ground truth - The terrain ground truth will either be provided by CNES in the case of testing on the SEROM site or built with the drone MAYA in Morocco (TBC)	**Evaluation criteria or metrics:** - Errors between rover map and ground truth (RMSE, Min/Max errors…) - Evolution of the errors with the number of fused point clouds - Computation performance of algorithm (map update rate)

4.1.2.5 DFPC: Lidar SLAM

Inputs: - Sequences of LIDAR point clouds - Estimated rover pose at the time of the point cloud acquisition (e.g. as produced by Wheel or Visual odometry)	**Outputs:** - Estimated rover poses at the time of point cloud acquisitions - Pose-graph (internal): environment model made of a selection of keyframe point clouds and associated poses

Test procedure:	**Evaluation criteria or metrics:**
- Teleoperate the robot while running the Lidar SLAM DFPC, and compare the resulting poses with a ground truth provided by GPS RTK.	- Error between estimated poses and ground truth - Translation error shall be <2% of travelled distance - Rotation error shall be <0.04 deg/m - Computation performance of the algorithm (time taken for a pose-graph update, and for a pose graph optimization)

4.1.2.6 DFPC: Lidar-based localisation

Inputs:	**Output:**
- LIDAR point cloud - Estimated rover pose - Environment model produced by the Lidar SLAM DFPC (pose graph: selection of keyframe point clouds and associated poses)	- Estimated rover pose in the pose-graph frame of reference
Test procedure:	**Evaluation criteria or metrics:**
- Teleoperate the robot on an area previously mapped by the Lidar SLAM DFPC, while running the Lidar-based localization DFPC and compare the resulting poses with a ground truth provided by GPS RTK.	- Error of the estimated pose with respect to the pose-graph frame of reference should be below 0.5cm in translation and 0.5 degree in rotation - Computation performance of the algorithm (time taken to estimate the rover pose in the pose-graph reference frame)

4.1.2.7 DFPC: Visual SLAM

Inputs:	**Outputs:**
- Left and right images + metadata - Depth images + metadata - RGB images + metadata - Estimated rover pose	- Estimated rover pose in map
Test procedure: - Teleoperate the robot while the visual slam module is running and compare the estimated pose with a ground truth provided by an RTK-GPS ground truth.	**Evaluation criteria or metrics:** - Error between estimated pose and RTK-GPS ground truth, - Relative position error shall be within 2% of the distance travelled, - Drifting over time depending on terrain, - Computation performance of algorithm (pose update rate) is compatible with the 1Hz target.

4.1.2.8 DFPC: Visual Map-based Localisation

Inputs:	**Outputs:**
- Left and Right images + metadata - Depth images + metadata - RGB images + metadata - Estimated rover pose - Previously created SLAM landmark map	- Estimated rover pose in map
Test procedure: - Teleoperate the robot while the visual map-based localisation module is running and compare the estimated pose with a ground truth provided by an RTK-GPS.	**Evaluation criteria or metrics:** - Number of lost tracking occurrences, - Error between estimated pose and GPS RTK ground truth of the closest map keyframe, - Position error shall be within 0.5m of the closest map keyframe,

	- Drifting over time depending on terrain, - Computation performance of algorithm (pose update rate) is compatible with the 1Hz target update rate.

4.1.2.9 DFPC: Long-range Tracking

Inputs: - RGB images + metadata - Rover AHRS data + metadata	**Outputs:** - Estimated rover pose wrt to target.
Test procedure: - Precisely measure the fixed target position and orientation wrt to test field global frame. - Teleoperate the robot while the target is kept within the sensors field of view and the tracking DFPC is activated.	**Evaluation criteria or metrics:** - Bearing error between estimated pose and true rover relative pose, computed with an RTK-GPS ground truth (of target and rover). - Tracking accuracy of the center of the target is in the order of magnitude of 1 pixel on the sensor, which would be sufficient to allow for further rendezvous operations. - Update rate is expected to be sufficiently fast to perform autonomous navigation at 1Hz.

4.1.2.10 DFPC: Mid-range 3D Model Detection

Inputs: - Left and Right images + metadata - Depth images + metadata - RGB images + metadata	**Outputs:** - Target detection success/failure - Coarse estimation of rover pose wrt to target.

Test procedure:	**Evaluation criteria or metrics:**
- Precisely measure the fixed target position and orientation wrt to test field global frame. - Teleoperate the robot while the target is kept within the sensors field of view and the detection DFPC is activated. Follow a predefined trajectory.	- Detection success rate over trajectory higher than 80%, - Error between coarse pose estimation and true rover relative pose, computed with an RTK-GPS ground truth lower than 15° (angular) and 0.5m (linear) - Computational performance of algorithm (pose update rate) compatible with an autonomous operation rate of 1Hz.

4.1.2.11 DFPC: Mid-range 3D Model Tracking

Inputs:	**Outputs:**
- Left and Right images + metadata - Depth images + metadata - RGB images + metadata - Rover AHRS data	- Estimated rover pose relative to target
Test procedure:	**Evaluation criteria or metrics:**
- Precisely measure the fixed target position and orientation wrt to test field global frame. - Teleoperate the robot while the target is kept within the sensors field of view and the tracking DFPC is activated. Follow a predefined trajectory.	- Error between pose estimation and true rover relative pose, computed with an RTK-GPS ground truth, - Position error is lower than 10cm at 10m range, angular accuracy should be lower than 10° in heading, - Computational performance of algorithm (pose update rate) is compatible with the 1Hz update target.

4.1.2.12 DFPC: Point Cloud Model-Based Localisation with ICP Point Cloud Matching

Inputs:	**Outputs:**
- Rover attitude from AHRS + metadata - Point cloud + metadata	- Estimated rover pose with regard to target.
Test procedure: - Precisely measure the fixed target position and orientation wrt to test field global frame. - Load an existing 3D point cloud model of the target. - Operate the rover in the vicinity of the target while the DFPC is executing, keeping the target in the FOV of the sensors. - Check resulting poses for accuracy.	**Evaluation criteria or metrics:** - Error between pose estimation and true rover relative pose, computed with an RTK-GPS ground truth. - Final relative position accuracy should be lower than 0.1m, and angular accuracy should be lower than 10° in heading. - Computational performance of algorithm (pose update rate) should be compatible with 1Hz update rate target.

4.1.2.13 DFPC: Reconstruction 3D, Part 1: Image Feature Detection and Matching

Inputs:	**Outputs:**
- Images to be matched - Camera calibration matrix - Ground truth pre-computed matched features.	- List of features per image input (optional) - List of matches between locations in undistorted input images
Test procedure: - Run the DFPC with input images available - Store output feature matches - Compare list of feature locations with pre-computed feature list	**Evaluation criteria or metrics:** - Features are accurately detected and matched in the same way as pre-computed list (>90% accuracy)

4.1.2.14 DFPC: Reconstruction 3D, Part 2: Image Feature Detection, Matching and Pose Estimation

Inputs:	**Outputs:**
- Images to be matched - Camera calibration matrix - Ground truth camera poses	- List of estimated camera poses.
Test procedure: - Run the DFPC with input images available - Store estimated poses - Compare list of poses with ground-truth poses	**Evaluation criteria or metrics:** - Average euclidean and angular distance deviation should not be larger than 10% of the DFPC use range.

4.1.2.15 DFPC: Reconstruction 3D, Part 3: Point Cloud Triangulation and Construction

Inputs:	**Outputs:**
- Images to be matched - Camera calibration matrix - Ground truth Point Cloud	- Updated point cloud with new features inserted
Test procedure: - Run the DFPC - Check resulting point cloud for geometric accuracy	**Evaluation criteria or metrics:** - No more than 10% outliers as estimated by a human inspecting the point cloud, position estimation less than 1% of R, where R is the maximum operational distance of the camera/sensor, and 90% similarity in shape to the object viewed with less than 10% error in dimensional analysis (only for

	components larger than 10% of the total size of the object)

4.1.2.16 DFPC: Point Cloud Model-Based Localization with SHOT-based matching

Inputs:	**Outputs:**
- Existing Target Scene Point Cloud - Model Point Cloud to be found	- Poses of matches (in order of quality) for model point cloud within scene
Test procedure: - Run the DFPC on existing point cloud and model - Check resulting poses for accuracy	**Evaluation criteria or metrics:** - The matched model within the scene should be located and oriented within 10% of the size and relative orientation of the model. - Distance between ground truth pose and estimate pose should not be higher than 5% of the operating distance.

4.1.2.17 DFPC: Reconstruction 3D, Part 4 (optional): Bundle Adjustment and Optimization

Inputs:	**Outputs:**
- Existing Target Point Cloud - Matches for features in original input images - Original input 2D images	- Point cloud with improved geometric consistency
Test procedure: - Run the DFPC on existing point cloud with matches and original images available	**Evaluation criteria or metrics:** - Relative geometric size and orientation of the point cloud should be quantitatively closer to the size and orientation of the actual scene

- Check resulting point cloud for geometric accuracy	with respect to the un-adjusted point cloud. - Point cloud should be located and oriented within 10% of the size and relative orientation of the scene.

4.1.2.18 Navigation Map Building

Inputs: - Left and right images + metadata	**Outputs:** - Navigation Map
Test procedure: - Run the Navigation Map Building DFPC over various terrains, for a given rover geometry. - Validate that rover constraints (tipping angle, contacts) are satisfied throughout whole trajectory, using IMU measurements. - Validate that obstacles (slopes, objects) are correctly identified are traversable/non-traversable given a rover geometry.	**Evaluation criteria or metrics:** - Non-traversable slopes are identified in the correct category, - Obstacles of given height are correctly identified, - Rover tipping angle and contact constraints (defined by rover used in the test) shall be satisfied on the whole trajectory. - Planned trajectory distance ratio (reference distance/executed distance) is between 80 to 110 %, with a standard deviation of below 10% between runs, - Computational performance: CPU and memory usage, update rate is compatible with a autonomous navigation at 1Hz.

4.1.2.19 DFPC: Position Manager

Inputs:	**Outputs:**
- Pose estimates produced by any of the localisation DFPC (Wheel Odometry, Visual Odometry, Lidar SLAM, Lidar-based Localisation, Visual SLAM, Visual Map-based Localization, Absolute Localisation)	- Current estimated rover pose, in any reference frame - Estimates of past rover poses, in any reference frame
Test procedure:	**Evaluation criteria or metrics:**

- Teleoperate the robot while a selection of localisation DFPCs is running, comparison of the resulting estimated poses with a ground truth provided by an RTK-GPS ground truth. The tests will be run in a incremental manner:
  - Integration of Wheel Odometry and Visual Odometry
  - Integration of the Lidar SLAM
  - Integration of the Visual SLAM
  - Integration of the Lidar-based Localisation
  - Integration of the Visual Map-based Localisation
  - Integration of the Absolute Localisation
  - Integrating of a pair or triplet of the localisation DFPCs
	- Error between the estimated poses and true rover poses, computed with an RTK-GPS ground truth.
	- Time performance of the integration of the estimated pose produced by each of the localisation DFPCs.
	- The performance of the localisation produced by PoM depends on which localisation results have been fused.

4.1.2.20 DFPC: Haptic Scanning

Inputs:	**Outputs:**
- Desired end-effector position - End-effector force measurements - Estimated rover pose relative to target.	- 3D point cloud representing touched shapes
Test procedure:	**Evaluation criteria or metrics:**
- Run the node with sample data - A ground truth pointcloud will be used.	- The 3D point cloud generated is a subpart of the ground truth pointcloud.

4.2 Integrated test plan with data flow control

Using the CDFF-Dev features, the data fusion solutions will be tested offline, once log data correspondent to the mission is collected.

4.2.1 DFPCs + Orchestrator

Validate the correct functioning of the orchestrator using real data and DFPCs offline. The data to be used for these test should be similar to one that will be obtained on the final scenarios. When possible acquired from the same sensors.

It will be possible to analyze the capabilities of the orchestrator using different configurations of the orchestrator with a variety of DFPCs.

Test approach:

The user will provide the path to the logged data in a Python script. In this python script the orchestrator will be accessible (thanks to Python bindings) as well as the DFPCs. The script will provided the data to the different DFPCs and its correspondent DFNs. The Orchestrator based on the results of the processed data and on the incoming data itself, can decide to stop or activate different DFPCs. The activation and deactivation of DFPCs will be implemented in the python side by querying the orchestrator as to which DPFCs should be activated and its runtime frequency that will be defined by the developer of the Data Fusion Solution (i.e. user of the CDFF). The orchestrator will provide a configuration file where the activation or runtime frequency of each DFPC can be configured before deploying it on the target platform.

Pass criteria: The test will be considered as successful if the result of the DFPC orchestration is sound and allows the proper generation of DFPC products, that typically would meet OG2 / ERGO needs.

4.2.2 DFPCs + DPM (local & central DPM)

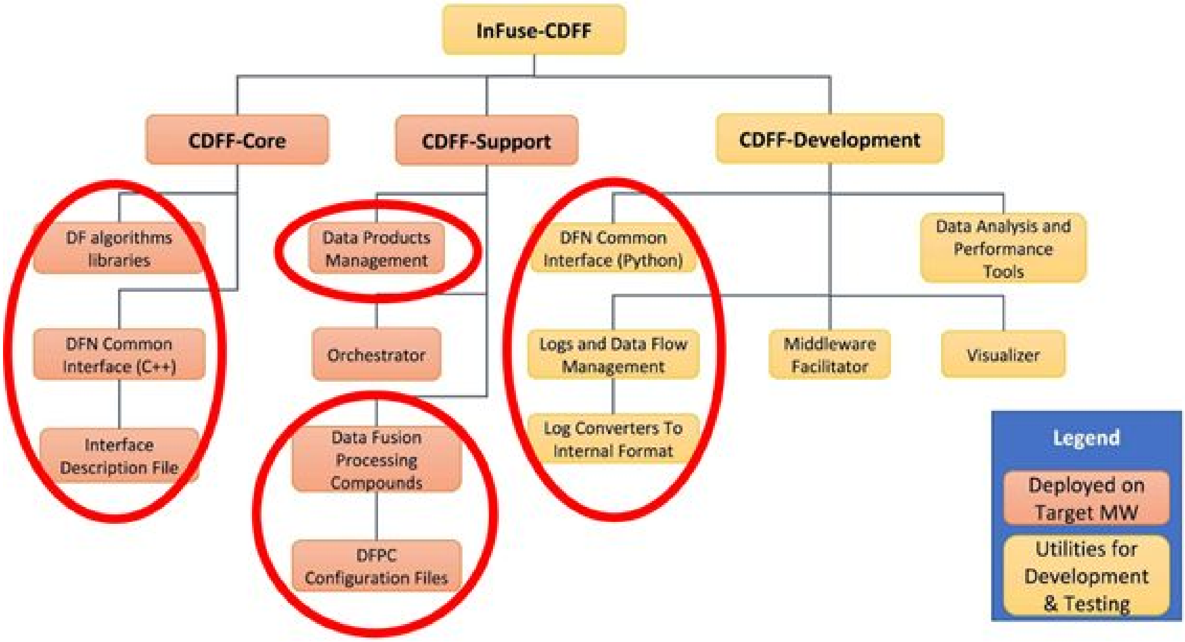

Figure 3: CDFF product tree: focus on DFPC + DPM + data flow

In this test, the focus is on the interactions between the DPM and the DFPCs. Such testing activities require the Data Flow component of the CDFF, so that to conveniently and effectively transfer various types of data between the DFNs involved in DFPCs to be tested, and the DPM components.

Also, the Data Analysis and Performance Tools (aka. DAP Tools) will optionally be used as part of the tests, so that to (1) verify the proper functioning of these tools and (2) obtain relevant information on the performances of the data fusion algorithms involved in the tests.

The test will consist of:

(1) Running a single DFPC, and verifying that relevant data products are properly received and handled by the DPM.
(2) Running a single DFPC, and verifying that it can effectively retrieve and access data products it needs, from where it is stored in the DPM.
(3) Running several (at least 2) DFPCs in parallel, and ensuring that data products transfer from a DFPC to another, through the DPM, works properly.

Pass criteria: The test will be considered as successful if it is verified that the DPM can properly handle (i.e. receive and store) the data products that need to be shared among the DFPC, and can suitably make data products available to DFPCs, on request.

The global DPM can be evaluated through its common interface in the data flow control environment. Local DPMs will as well be supported, for this the python bindings to the local DPMs will have to be provided by the developers.

An example of a test that will be performed is the sharing of data through the central DPM between two DFPCs: One of the DFPCs will store a map in the DPM and a second DFPC will query and use internally this map.

A python script where this test is performed will be provided.

4.2.3 DFPCs + DPM (centralized) + Orchestrator

This test deals with a fully integrated CDFF, with the purpose of ensuring overall consistency of data exchanges and process triggered within each component. No mock components are required in this setup though.

A baseline test scenario would consist in the following steps:

(1) The Orchestrator selects and enables a DPFC, as a reaction to an assumed request from OG2 (will in the test be triggered directly at the Orchestrator level).

(2) The selected DPFC is enabled, waiting for inputs from sensors.

(3) In the absence of OG4 sensors, pre-recorded data sets are fed (“manually”) in the CDFF, through the Data Flow.

(4) The DFPC should be able to receive and process the input data, and shall produce and dispatch (through the Data Flow) a sound data fusion product.

(5) The data product is sent to the DPM, and the Orchestrator is notified of its availability.

(6) The Orchestrator requests the resulting data product from the DPM.

(7) The DPM passes the data product material to the Orchestrator.

During all the process, the Data Visualizer is able to render relevant data samples and data products, on request. Similarly the DAP tools may be used in support to performances shaping.

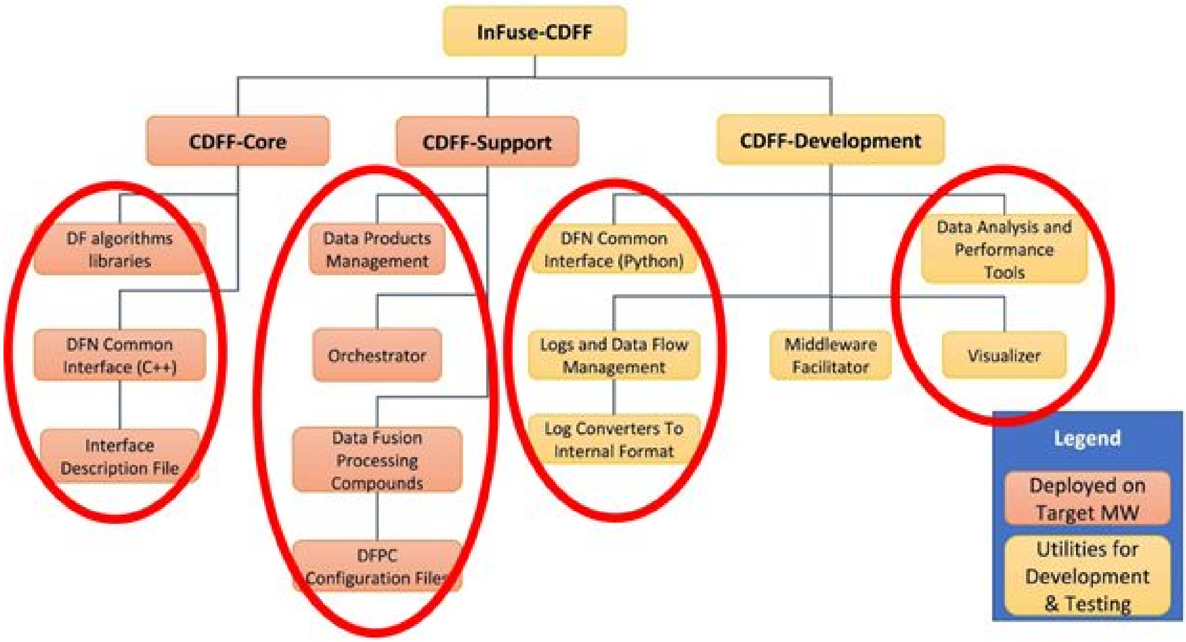

Figure 4: CDFF product tree: All integrated

Pass criteria: The test will be considered as successful if it can be verified that the integrated CDFF components behave nominally, and that multiple (and representative) DFPCs can be successfully solicited during the execution of scenarios comparable to the baseline test scenario above. The central DPM would play a role in allowing DFPCs to exchange historic and current pose estimates generated from other DFPCs.

Tests of the usage of the data flow control combining the orchestrator and the central data product manager will be provided.

These tests will consist on scripts in which more than one DFPC is used offline, providing to it logged sensor data. It will be proved that it is possible to test the functionalities of the orchestrator with Python language by querying it which DFPC should be activated. In this test, the central Data Product Manager will as well be incorporated.

4.2.4 Orchestrator with OG2 (data+command) and OG4 (command) mock interfaces

For what concerns the interface between the Orchestrator and OG2, the interfaces concerned are essentially the ones allowing OG2 to request data products to OG3, and allowing OG3 to return requested data products.

For what concerns the interface between the Orchestrator and OG4, the interfaces concerned are those allowing to set the operational mode of OG4, i.e. selecting one of the working modes for the sensors.

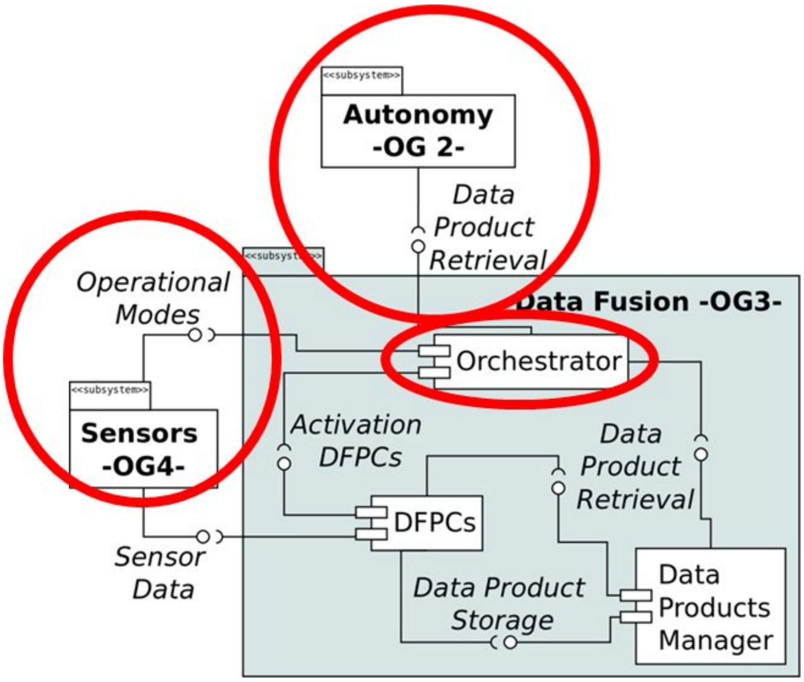

Figure 5: Orchestrator interactions with OG2 and OG4.

Test approach:

In order to test the identified interfaces, a “mock OG2” software (aka. M-OG2) and “OG4 emulation software” (aka. E-OG4) will be purposely developed respectively as a mean to issue requests towards the Orchestrator and to ensure that received data products match expectations (structure, content) in the case of OG2 interfaces, and as a mean to receive and verify the structure and content of mode setting requests, for concerns the OG4 interfaces. The set of interfaces that will be tested between OG3, OG2 and OG4 can be found in D5.2, Appendix 7.5.

In order to provide the Orchestrator with relevant data products, either a mock DPM or real DPM will be used to provide the Orchestrator with the required data products (as pre-recorded samples) – through the CDFF Data Flow, for the sake of convenience.

These interfaces tests will be carried out by wrapping the in Orchestrator’s OG2 and OG4 related APIs in TASTE environment, a common RCOS between OG4 and OG3 or directly linking interfaces between them, and the data products will be expressed in (extended) ASN.1 formalism as defined by OG1.

Pass criteria: The test will be considered as successful if it can be verified that the Orchestrator:

- Can handle requests issued by M-OG2 for all relevant Data Products that OG2 may possibly request, and accordingly each time returns to the M-OG2 software the expected Data Products in the desired data format (Ex. ASN1.0) and quality (resolution, respecting error margins..).
- Can properly send operation mode setting requests to E-OG4, i.e. as a data structure and content that E-OG4 will validate as correct.

4.2.5 DFPC with OG4 emulator sensor data interfaces

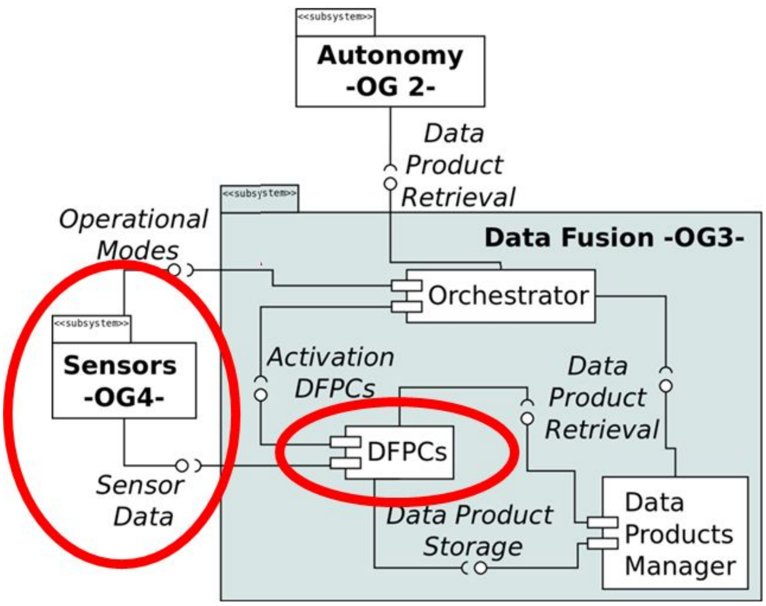

Figure 6: DFPCs interactions with OG4

The interface concerned is the one allowing DFPCs to access sensor data produced and provided by the OG4 emulator.

Test approach:

In order to test that, OG4 emulator software (aka. E-OG4) will be purposely developed as a mean to provide sensors data (whose source may be either pre-recorded samples, simulated, or produced online by real sensors) to the data flow of the CDFF, so that to make these data available to the DFPC

These interfaces tests will be carried out in TASTE environment, a common RCOS between OG4 and OG3 or directly linking interfaces between OG3-OG4. The data issued by E-OG4 will be expressed in (extended) ASN.1 formalism as defined by OG1. OG4 cooperation will be required for defining and developing the emulated sensor data interfaces. The set of interfaces that will be tested between OG3 and OG4 can be found in D5.2, Appendix 7.5.

To test the different types of sensor data interfaces (Ex. IMU, Cameras, Lidar, stereo cameras etc.), will require a few sample DFPCs (one or more) to be wrapped in TASTE, a common RCOS between OG4 and OG3 or directly linking interfaces to acquire data from OG4 emulated sensor data interfaces.

Pass criteria: The test will be considered as successful if it can be verified that the DFPCs are able to access and use efficiently the sensors data incoming from E-OG4. The test coverage should make provision for all sensors and data types foreseen as part of this RI.

4.3 Test schedule with EGSE M18, M20, M22

4.3.1 LAAS Mobile robots

Schedule	Test approach

M18	- The EGSE Mana and Minnie are finalized: all sensors and on-board, all functional module that gather their data are operational
	- The development of the following DFPCs is finalized:
	    - Visual odometry (MAG/CNES version)
	    - DEM (without associated DPM services)
	    - Position Manager (able to integrate outputs from Wheel Odometry and Visual Odometry)
	- The developed DFPCs are integrated and tested in simulation (with the Morse simulator)
	- First tests of data acquisition and processing are initiated at LAAS premises

M20	- First field tests on the CNES Serom test site.
	    - All the DFPCs developed at M18 are integrated on-board the Mana and Minnie robots.
	    - Data acquisitions are performed while manually driving the robots for further processing
	    - Additional functionalities that allow autonomous navigation are integrated, the integrated DFPCs run in real time - while logging data for further analysis

M22	- The whole suite of the PT DFPCs are developed - Their testing in simulation begins

M23	- Second field tests on the CNES Serom test site
	    - All the PT DFPCs are integrated on-board the Mana and Minnie robots
	    - Tests are made for each DFPC, both under manual control of

	the robots and autonomous navigation mode - First tests of the multi-robot features of the DFPCs are made

4.3.2 HCRU

Schedule	Test approach
M18	- All components of HCRU should be fully mechanically and electrically integrated - Computational hardware should have a specified OS (OSL 42.1) installed - Computer should be bootable - Cameras should be software-integrated, triggerable, visual data should be arriving within expected time delay correctly timestamped - IMU should be integrated, triggerable, data should be arriving within expected time delay correctly timestamped - SSD should be integrated and able to write and read data - Set of functional tests will be performed to verify above-mentioned capabilities
M20	- HCRU is foreseen to be married with an early-iteration of InFuse - It is planned to physically take HCRU to a collocation in Toulouse, to have early version of InFuse integrated with HCRU, so that it receives its sensor data and performs basic functionality of CDFF using them. No integration with LAAS rovers is expected. - HCRU should be able to capture sensor data and bridge them over to InFuse, which should perform an early-iteration - HCRU is supposed to be early-integrated with DLR BB2 - HCRU is supposed to be early-integrated with DFKI Sherpa
M22	- Full demonstration of CDFF running on HCRU/BB2 - Full demonstration of CDFF running on HCRU/Sherpa - Online demonstration (qualitative testing) - Offline validation (quantitative testing)

5 Testing and validation with OG6 facilities (post-TRR)

For OG3, three test tracks stand ready: DLR ExoMars BB2 demonstrator integrated with HCRU and scheduled for a test in Planetary Exploration Lab, DFKI SherpaTT demonstrator integrated with HCRU and scheduled for a field test in Morocco, and CNRS-LAAS MANA and MINI, integrated with their proprietary sensor suite and scheduled for a field test in Morocco. Similarities and differences of these three test tracks were analyzed. The test plan is designed to leverage those, so that the most crucial DFPCs are extensively tested on the highest amount of different platforms possible, while more experimental DFPCs will be tested on a rover they are most suitable for.

In the chart below, one can find the overview of the current plan of post-TRR testing and validation of deliverable DFPCs:

	BB2@PEL	**Sherpa@Morocco**	**3M@Morocco**
	Post-TRR	Field test	Field test
	online	offline	online
Wheel Odometry	OK - DLR version	OK - DFKI version	OK - LAAS version
Visual Odometry - MAG/CNES	OK	OK	OK
Visual Odometry - LAAS	TBC	TBC	OK
Visual SLAM	no	OK	OK
Visual Map-based Localisation	no	OK	OK
Point Cloud Model-Based Localization	OK	no	OK
Haptic scanning	no	no	no
Absolute Localization	no	OK	OK
DEM Building	OK	OK	OK
LIDAR Pose Graph SLAM	no	TBC (Lidar on HCRU or on Sherpa?)	OK
LIDAR Map-based Localisation	no	TBC (Lidar on HCRU or on Sherpa?)	OK
Pose Fusion	OK - fusion of wheel and visual odometries	OK - fusion of wheel and visual odometries	OK - full fledged version
Multi-robot requests	no	no	OK

A few remarks:

- Wheel odometry is very dependent on robot chassis. Therefore, we envisage use of three flavors: DLR, LAAS and DFKI, optimized for the respective demonstrator. Note that the DFKI flavor is part of OG6 and not a solution developed by OG3. Thus, a final release of the OG6 odometry is not foreseen along with the OG3 code.
- Visual Odometry note: two solutions will be provided:
    - A baseline one by MAG - robust version
    - An experimental one from LAAS - to be deployed as time and level of maturity allows.
- Testability of LIDAR-based DFPCs is considered to be a high-risk option for HCRU, as it is still unclear if LIDARs will manage to be integrated within time limit.

5.1 Preparing and verifying PEL & ExoMars BB2 EGSE and interfaces

5.1.1 Sensors and Calibration

HCRU should provide sensor data from stereo cameras and IMU as a baseline. Additionally, integration of LIDAR is being pursued. BB2 should provide wheel odometry data to HCRU via DLR-internal means.

DLR is not envisaging additional sensors to be included apart from those specified.

Mono-camera calibration, if needed, will be performed using DLR CalDe/CalLab. For more details, see D5.2, chapter 4.3.1. Stereo camera calibration will also be performed using DLR CalDe/CalLab. For more details, see D5.2, chapter 4.3.2. Body-to-camera calibration is described in D5.2, chapter 4.3.3. BB2’s wheel odometry is expected to be calibrated using DLR standard wheel calibration routine.

Preparing BB2 for testing:

1. Connect rover over cable to AC/DC power supply
2. Enable rover and HCRU
3. Start relevant processes on rover
4. Start relevant processes on HCRU
5. Disconnect power supply
6. See if steering works and datastreams arrive

5.1.2 Software and Data interfaces with OBC

HCRU is expected to carry a full computational stack consisting of Intel i7 processor, SSD drive and sensor connections via standard interfaces such as Ethernet. No additional hardware is expected/necessary to be provided from OG3’s side.

DLR is planning to deploy OpenSUSE Leap 42.3 on HCRU and utilize DLR-internal systems (RobotKernel, SensorNet) and respective ROS bridges to provide sensor data wrapped as ROS topics.

5.1.3 Ground truth

A 6DOF localization ground-truth will be provided from external pose tracking system - we are using a COTS solution infrared-camera based system utilizing reflective markers, called Vicon. Precision of localization ground-truth is supposed to be in order of millimetres.

Mapping ground-truth will be provided by an external digital elevation map capture system based on stereo cameras placed over the testbed. Ground-truth precision for mapping is expected to be of a few-centimetre accuracy, of possibly different value in horizontal plane and in elevation direction.

Localization ground-truth data are expected to be provided in a time-synchronized and pose-registered manner. Robot self-localization will be compared against this ground-truth, using standard measures, like L_2-norm.

Mapping ground-truth data data are expected to be provided in a time-synchronized and pose-registered manner. Robot mapping will be compared against this ground-truth, using standard map comparison metrics.

All robot-internal data can be stored to an exchangeable SSD drive in a format specified by InFuse. Robot-external data, i.e. localization and mapping ground-truths will be stored separately.

5.1.4 Validation schedule

Approximately 1 week time slot will be scheduled around M23, where HCRU mounted on BB2 will perform a dry-run of InFuse framework. This workshop will likely take place in Oberpfaffenhofen, Germany, at DLR-RMC premises. Presence of representatives of all PT-relevant partners is advisable.

Final validation will also take part in Oberpfaffenhofen, Germany, at DLR-RMC premises. It will be in Post-TRR phase, likely M24. We assume 1 week time slot for this testing scenario. Presence of all PT-relevant partners is advisable.

We can expect to identify issues and problems during the testing on DLR premises, and need to adapt InFuse or HCRU accordingly, to perform optimally in a follow-up test track in Morocco. We expect to be granted a time window of approximately one month for these adaptations - after the post-TRR testing in Oberpfafenhofen and before the shipping of HCRU to Morocco.

Day #	Validation Steps
Preparation	Setup of sensors. Connecting OBC and network interfaces. Calibrating

	sensors. Basic test coverage, ensuring the entire setup properly works
1-5	List of DPCSs that will be tested - in an open loop, sequentially Wheel Odometry - OK - DLR version Visual Odometry(MAG/CNES) - OK Visual Odometry(LAAS) - TBC Point Cloud Model-Based Localization - OK Position Manager - OK (early version: fusion of wheel and visual odometries)
Postprocessing	Evaluation of localization and mapping results in comparison to ground-truth

5.2 Preparing and verifying Sherpa in an Outdoor Analogue

5.2.1 Sensors and Calibration

HCRU should provide sensor data from stereo cameras and IMU as a baseline. Additionally, integration of LIDAR is being pursued. Sherpa should supply it with wheel odometry data.

SherpaTT incorporates already a Velodyne HDL-32E LIDAR, a Laser range finder: Hokuyo UST-20LX, a Camera: Basler Ace (2048 x 2048 px, 25 fps), and an IMU Xsens MTi-28A AHRS.

We are not envisaging additional sensors to be included apart from those specified.

Mono-camera calibration, if needed, will be performed using DLR CalDe/CalLab. For more details, see D5.2, chapter 4.3.1. Stereo camera calibration will also be performed using DLR CalDe/CalLab. For more details, see D5.2, chapter 4.3.2. Body-to-camera calibration is described in D5.2, chapter 4.3.3. Sherpa's wheel odometry is expected to be calibrated by OG6.

Before performing any experiment it must be checked:

1. on the transport-cart and powered by AC/DC adaptor (probably using power generator in-field)
2. All (three) batteries fully charged, connected and recognized by the CPMB (Central Power Management Board)
3. WiFi connection to control PC
4. Remote Emergency-switch is "manned"
5. Start hardware drivers

6. Start Motion Control System (MCS), Ground Adaptation Process (GAP) at this point being disabled
7. Drive to Zero-Pose (all joints in defined zero), check if actually zero. If yes, proceed, if not: manually correct/re-calibrate joints, goto 4
8. Move SherpaTT down from cart by
    a. move to PreferredPoseC (High) -> Wheels into ground contact
    b. remove cart
    c. move to PreferredPoseC
    d. move to PreferredPoseA
9. Enable GAP of MCS
10. Start OG3 software on the HRCU
11. Check that clocks in both HRCU and SherpaTT are synchronized
12. The sensor are activated and producing correct data
13. GPS incorporates the differential correction
14. The data produced by all the sensors as well as from the odometry component is been logged
15. The remote control is operational and SherpaTT responds as expected to the commands
16. Disconnect AC/DC adaptor, SherpaTT switches over to first battery automatically

5.2.2 Software and Data interfaces with OBC

HCRU is expected to carry a full computational stack consisting of Intel i7 processor, SSD drive and sensor connections via standard interfaces, so no additional hardware is expected/necessary to be provided from OG3's side.

Interfacing software between the HCRU and Sherpa OBC is not required for this demonstration as the data processing of the sensor data will be done off-line. The only requirement is that both clocks are synchronized so that the timestamps match.

DLR is planning to deploy OpenSUSE Leap 42.3 on HCRU and utilize DLR-internal systems (RobotKernel, SensorNet) and respective ROS bridges to provide sensor data wrapped as ROS topics.

An interface has been provided by OG6 to develop the communication with the HRCU, the main goal of the test is though to demonstrate the offline features of InFuse based on data taken from the sensors on both SherpaTT and the HCRU. The most important requirement is thus to make sure that the data is been logged in both systems and that the initial timestamps are synchronized. Interface between OG6-Sherpa and HRCU :

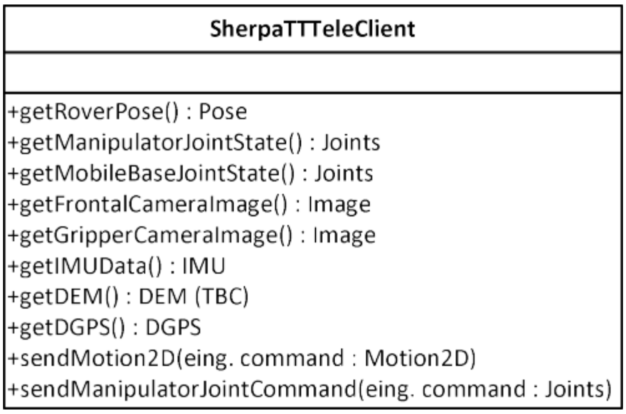

Figure 7: API Provided by OG6 to exchange data with SherpaTT

5.2.3 Ground truth

OG6 is proving Differential GPS which will be used as ground truth to compare the results on the pose estimation DFPC offline. As far as it is known to us there is no ground truth map available of the region of the test, but LAAS plans to deploy a SenseFly UAV to produce a high resolution DEM and orthoimage (0.05 cm resolution). Thus, comparison of the generated DEM against ground truth DEMs may be possible.

The resulting logged data will be then compared using the tools provided by CDFF-Dev. Here, the tools for visualization and offline replay will be used to compare the Ground Truth with the resulting poses and if available with existing maps of which the error is know.

The data from SherpaTT will be logged using the rock log tool and stored in the robot's hard drive. The data from the HRCU will be logged in the correspondent SSD using the logging mechanisms that the HRCU provides (Ex. ROS bags).

5.2.4 Validation schedule

M16: Identification of the DFPCs that will be evaluated offline

M18: OG6 provides sensor data in the same format to be tested in the final demonstration. From both the HRCU and the SherpaTT.

Interfaces for DFPCs to be evaluated as well as for the Orchestrator and DPM should be finalized.

M19: In case that the data format provided by the HRCU and or the SherpaTT differs from the ESROCOS types, converters to these types will be implemented in this month.

M20: Offline evaluation of the provided data using the Data Fusion Solutions to be tested should be working.

M21: The HRCU physical integration on the SherpaTT robot should take place.

Collection of data sets using SherpaTT and the HRCU will take place in Bremen

M22: Integration of Data Fusion analysis tools to compare the results from different DFPCs.

The final tests on Morocco should take an estimated time of 3 weeks.

For this tests developers knowing the DFPCs to be evaluated should be present. At least one of the developers of the CDFF-Dev and CDFF-Support tools. Experts on the calibration of the sensors for the HRCU are also required. Experts on the functioning of SherpaTT will also be required.

The exact days of work with SherpaTT have to be coordinated with OG6 because the system is also used in the demonstrations of OG2 therefor the following schedule might change.

Day #	**Validation Steps**
1	Arrival to the site. Deployment of the robot and initial tests. Setup of sensors Connecting OBC and network interfaces
2	Calibrating sensors Basic test coverage, ensuring that the entire setup properly works
3	Open loop control of the set up by OG6 - remote control maneuvers will be tested in not analog location Offline Evaluation of the Visual Odometry produced by MAG/CNES.
4	Open loop control of the set up by OG6 - remote control maneuvers will be tested at an analog location Offline Evaluation of the Visual Odometry produced by LAAS.
5	Open loop control of the set up by OG6 - remote control maneuvers will be tested at an analog location Offline Evaluation of the Visual SLAM produced by LAAS.
6	Open loop control of the set up by OG6 - remote control maneuvers will be tested at an analog location Offline Evaluation of the DEM Building
7	Offline Evaluation of the Visual Map-based Localisation

8	Offline Evaluation of the LIDAR Pose Graph SLAM
9	Offline Evaluation of the LIDAR Map-based Localization
10	Offline Evaluation of the Position Manager
11-14	Days for tackling addressing issues detected on site and planning the details of the final demonstration (selection of the areas for demonstration and DFPCs to show)
15	Final Demonstration at selected places showing the functioning of some DFPCs offline with data recorded on the moment of the demonstration.

5.3 Preparing and verifying Multi-robot setup (3M) in an Outdoor Analogue

5.3.1 Sensors and Calibration

The sensors that equip the robots Mana and Minnie are listed in D5.2, section 5.3. Among those, the ones to calibrate are:

- The cameras, which will be calibrated using either the usual camera calibration tools available or the DLR CalDe/CalLab
- The automotive Lidar (Sick LD-MRS 400001) mounted on a 2-axis orientable turret will be calibrated using home made tools

The inter-sensor and sensor-robot relative position estimates will be made using ad-hoc tools.

5.3.2 Software and Data interfaces with OBC

All the DFNs and DFPCs will be integrated within Genom3 functional modules and ROS nodes, able to communicate together through the ROS communication protocol. The integration granularity associates one DFPC to one Genom3 module or one ROS node. The drivers for each on-board sensors are also encapsulated within Genom3 modules and sensor nodes.

The OBC are i7 2.53GHz processors, running the Linux Ubuntu 16.04 distribution.

5.3.3 Ground truth

For the robot position estimates, the ground truth is a RTK GPS for the translation components, the IMU (used at rest) for the attitude components, and the FOG gyro for the heading component. The 3 sigma precision associated to the RTK GPS is 1cm horizontal and 2cm vertical, the precision of the heading provided by the FOG gyro is a function of elapsed time, and is of 1 degree after one hour of operation.

An orthorectified map and reference DEM of the CNES Mars Yard outdoor site will be available. For the tests in Morocco, it is foreseen to fly a SenseFly drone to obtain a high precision terrain DEM and associated orthoimage (0.05 m resolution). The resulting DEM will be subsampled down to the resolution of orbiter maps provided by the Hirise instrument on Mars: 1.0m for the DEM and 0.25 m for the orthimage.

The resulting logged data will be then compared using the tools provided by CDFF-Dev. Here, the tools for visualization and offline replay will be used to compare the Ground Truth with the resulting poses and if available with existing maps of which the error is know.

5.3.4 Validation schedule

The daily validation schedule during M24 in Morocco is yet to be detailed. The tests planned at the CNES SEROM site during M23 will in particular allow to rehearse the validation schedule.

6 Data Exploitation

Each testing step in the proposed plan will generate a wealth of result data (DFN outputs, DFPC outputs, performance measurements) which will be processed and analysed. In the case of integrated tests, some parametric studies will be carried out when deemed necessary to evaluate the effect of various DFPC parameters on resulting performances. Required DFPC or DFN evolutions to comply with test criteria will be identified and implemented along the way, mainly during pre-TRR testing. Finally, results produced by testing and validation activities with OG6 facilities will allow to highlight lessons learned and determine remaining future evolutions.

In all cases, synthetic technical results will be produced and released after each rounds of tests.

7 Conclusion

This deliverable documents the unitary and integrated test plans for the Planetary RI of the InFuse CDFF.

Besides describing the overall approach, the release schedule of the core components of InFuse is provided as a best estimate at the time of issuing this deliverable.

The release schedule is synchronized with internal integration and testing rounds, that will take place every 2 months during the implementation phase.

The produced source code will be continuously tested and verified, relying on a common git framework with proper automatic build routines. This will ensure that the overall CDFF software is all the time sound while components are progressively added to it.

Finally this deliverable also documents the testing and validation foreseen with OG6 facilities, during the post-TRR activities.

This document should serve as a reference for testing and integration activities of InFuse, but the content may likely be the object of adjustments during the implementation phase, due the possibilities for contingency (e.g. adapting the release timing or order for some of the components, or adapting the testing procedures).

The Test Readiness Review should anyway refer to the test plan proposed in this document (or its latest version, should changes be recorded).

8 Appendix

8.1 CDFF Requirements Coverage by Test Plan

The requirements were organized in the System Requirements Document (deliverable 3.2) according to a previous categorization of the features. In this appendix the requirements remain organized as previously to facilitate tracking with the previous document.

8.2 CDFF Core

8.2.1 User Requirements (UserR)

Req. ID	Compliance (yes / no / partial)	Section where it is addressed	Comments
SR_UserR_A101	Yes	4.1.1 (4.1.1.2, 4.1.1.3, 4.1.1.8, 4.1.1.16), 4.1.2.10, 4.1.2.11, 4.1.2.13	
SR_UserR_A102	Yes	4.1.1.5, 4.1.1.24, 4.1.2.1, 4.1.2.2, 4.1.2.3, 4.1.2.6, 4.1.2.8, 4.1.2.15	
SR_UserR_A103	Yes	4.1.1, 4.1.2.1, 4.1.2.5, 4.1.2.7	
SR_UserR_A104	Yes	4.1.2.4, 4.1.2.5, 4.1.2.7, 4.1.2.14, 4.1.2.12, 4.1.2.16	
SR_UserR_A105	Yes	4.1.1, 4.1.2	
SR_UserR_A106	Yes	4.1.1, 4.1.2	
SR_UserR_A107	Yes	4.1.1, 4.1.2	
SR_UserR_A108	Yes	4.1.1, 4.1.2	
SR_UserR_A109	Yes	4.1.1, 4.1.2	
SR_UserR_A110	Yes	4.1.1, 4.1.2	
SR_UserR_A111	Partial	5.1, 5.2	This will be tested in OG6 facilities with regular computing

			platforms with imposed memory and processing constraints. Covered in D5.3.
SR_UserR_A112	Yes	4.2, 4.2.4	
SR_UserR_A113	N/A		DFNs in 4.1.1 are already selected as a result of the trade-off
SR_UserR_A114	Yes	4.1.1, 4.1.2	Each DFN & DFPC has a test plan associated with it to meet RAMS
SR_UserR_A115	Yes	5.1, 5.2, 5.3	
SR_UserR_A116	Partial	5.1, 5.2, 5.3	Visualization is only provided on the Developer's Environment
SR_UserR_A117	N/A		Covered in D5.3

8.2.2 Functional Requirements (FuncR)

Req. ID	Compliance (yes / no / partial)	Section where it is addressed	Comments
SR_FuncR_A201	N/A		Orbital track
SR_FuncR_A202	N/A		Orbital track
SR_FuncR_A203	N/A		Orbital track
SR_FuncR_A204	N/A		Orbital track
SR_FuncR_A205	N/A		Orbital track
SR_FuncR_A206	N/A		Orbital track

SR_FuncR_A207	Yes	4.1.2.13, 4.1.2.16	
SR_FuncR_A208	Yes	4.1.2.5, 4.1.2.14, 4.1.2.15, 4.1.2.16	
SR_FuncR_A209	Yes	4.1.2.3, 4.1.2.10, 4.1.2.11	
SR_FuncR_A210	Yes	4.1.2.10, 4.1.2.11	
SR_FuncR_A211	Yes	4.1.2.17, 4.1.2.13	
SR_FuncR_A212	Yes	4.1.2.3, 4.1.2.6, 4.1.2.8, 4.1.2.12, 4.1.2.18	
SR_FuncR_A213	Yes	4.1.2.18	
SR_FuncR_A214	Yes	4.1.2.17, 4.1.2.4	
SR_FuncR_A215	Yes	4.1.2.1, 4.1.2.2, 4.1.2.7, 4.1.2.10, 4.1.2.11, 4.1.2.13	
SR_FuncR_A216	Yes	4.1.2.5, 4.1.2.6, 4.1.2.12, 4.1.2.14, 4.1.2.15, 4.1.2.16	
SR_FuncR_A217	Yes	4.1.2.1, 4.1.2.2, 4.1.2.4, 4.1.2.7, 4.1.2.8, 4.1.2.13	
SR_FuncR_A218	Yes	4.1.2.5, 4.1.2.6, 4.1.2.12, 4.1.2.14, 4.1.2.15, 4.1.2.16	
SR_FuncR_A219	Yes	4.1.2.5, 4.1.2.6, 4.1.2.12, 4.1.2.14, 4.1.2.15, 4.1.2.16	
SR_FuncR_A220	Yes	4.1.2.1, 4.1.2.2, 4.1.2.18	
SR_FuncR_A221	Partial	4.1.2.3	For global heading information in absolute localization

SR_FuncR_A222	Yes	4.1.2.19	
SR_FuncR_A223	Yes	4.1.1.7, 4.1.2.17 (many more in 4.1.1)	
SR_FuncR_A224	Yes	4.1.1.10, 4.1.1.16 (many more in 4.1.1)	
SR_FuncR_A225	Yes	4.1.1, 4.1.2	
SR_FuncR_A226	Yes	4.1.1.12, 4.1.1.5, 4.1.1.24, 4.1.1.30 (many more in 4.1.1)	
SR_FuncR_A227	Yes	4.2.5, 5.1.1, 5.2.1, 5.3.1	
SR_FuncR_A228	Yes	4.1.2.18, 5.3	

8.2.3 Performance Requirements (PerfR)

Req. ID	Compliance (yes / no / partial)	Section where it is addressed	Comments
SR_PerfR_A301	Yes	4.1.2, 5.1.3, 5.2.3, 5.3.3	
SR_PerfR_A302	Yes	4.1.2.18, 5.1.3, 5.2.3, 5.3.3	
SR_PerfR_A303	Yes	4.1.2.1, 4.1.2.2, 4.1.2.18	
SR_PerfR_A304	N/A		Orbital track
SR_PerfR_A305	N/A		Orbital track
SR_PerfR_A306	N/A		Orbital track
SR_PerfR_A307	N/A		Orbital track
SR_PerfR_A308	N/A		Orbital track

8.2.4 Interface Requirements (IntR)

Req. ID	Compliance (yes / no / partial)	Section where it is addressed	Comments
SR_IntR_A403	Yes	4.2.2, 4.2.3, 4.2.4, 4.2.5	
SR_IntR_A404	Yes	4.2.2, 4.2.3, 4.2.4, 4.2.5	
SR_IntR_A405	Yes	4.2.2, 4.2.4	
SR_IntR_A406	Yes	4.2.2, 4.2.4	
SR_IntR_A407	Yes	4.2.2, 4.2.4	
SR_IntR_A408	Yes	4.2.2, 4.2.4	
SR_IntR_A409	Yes	4.2.2, 4.2.3	

8.2.5 Resource Requirements (ResR)

Req. ID	Compliance (yes / no / partial)	Section where it is addressed	Comments
SR_ResR_A501	Yes	5.1, 5.2, 5.3, 4.1.2	
SR_ResR_A502	Yes	5.1.2, 5.2.2, 5.2.3	
SR_ResR_A503	Yes	4.1	
SR_ResR_A504	No	-	PROM usage is not known in this phase.
SR_ResR_A505	Yes	5.1.2, 5.2.2, 5.2.3	

8.2.6 Operational Requirements (OpR)

Req. ID	Compliance (yes / no / partial)	Section where it is addressed	Comments
SR_OpR_A601	Partial	4.2.1, 5.1, 5.2, 5.3	Visualization is only provided on the

			Developer's Environment
SR_OpR_A602	Partial	4.2.1, 5.1, 5.2, 5.3	Visualization is only provided on the Developer's Environment
SR_OpR_A603	Yes	4.2.1	Orchestrator core functionality
SR_OpR_A604	Partial	4.1.2	Possible in the developers environment

8.2.7 Product assurance and safety requirements (ProdR)

Not Applicable.

8.2.8 Configuration and implementation requirements (ConfR)

Req. ID	Compliance (yes / no / partial)	Section where it is addressed	Comments
SR_ConfR_A801	Yes	4.2.2, 4.2.3	
SR_ConfR_A802	Yes	4.2.2, 4.2.3, 4.2.5	
SR_ConfR_A803	Yes	4.1	All DFPCs and DFNs are coded as per InFuse coding guidelines making it compatible for space grade architectures.
SR_ConfR_A804	Yes	4.1	All DFPCs and DFNs are coded as per InFuse coding guidelines making it compatible for space grade architectures.
SR_ConfR_A805	Yes	5.1.2, 5.2.2, 5.3.2	RTEMS is in D5.3

SR_ConfR_A806	Yes	5.1.2, 5.2.2, 5.3.2	Architecture described in D5.3
SR_ConfR_A807	Yes	5.1.2, 5.2.2, 5.3.2	Architecture described in D5.3

8.2.9 Test and Validation (ValR)

Req. ID	Compliance (yes / no / partial)	Section where it is addressed	Comments
SR_ValR_A901	N/A		Orbital Track
SR_ValR_A902	N/A		Orbital Track
SR_ValR_A903	N/A		Orbital Track
SR_ValR_A921	Yes	5.1.3, 5.2.3, 5.3.3	
SR_ValR_A922	Yes	5.1.3, 5.2.3, 5.3.3	
SR_ValR_A923	Yes	5.1.3, 5.2.3, 5.3.3	
SR_ValR_A924	Yes	5.1, 5.2, 5.3	
SR_ValR_A951	N/A		Orbital Track
SR_ValR_A961	Yes	2.4, 5.1, 5.2, 5.3	

8.3 CDFF Dev

8.3.1 User Requirements (UserR)

Req. ID	Compliance (yes / no / partial)	Section where it is addressed	Comments
SR_UserR_B101	Yes	2.1, 4.2	
SR_UserR_B102	Yes	2.1, 4.2.4, 4.2.5	
SR_UserR_B103	Yes	4.2.2, 4.2.3	

SR_UserR_B104	Yes	4.2	Visualization is provided in CDFF-Dev tools
SR_UserR_B105	Yes	2.1	Provided in CDFF-Dev tools
SR_UserR_B106	Yes	4.2	
SR_UserR_B107	Yes	4.2	

8.3.2 Functional Requirements (FuncR)

Req. ID	Compliance (yes / no / partial)	Section where it is addressed	Comments
SR_FuncR_B201	Yes	2.4, 5.1.4, 5.2.4, 5.3.4	
SR_FuncR_B202	Yes	4.2.2, 4.2.3	
SR_FuncR_B203	Yes	4.2	Visualization is provided in CDFF-Dev tools
SR_FuncR_B204	Yes	2.1	Provided in CDFF-Dev tools
SR_FuncR_B205	Yes	4.2.1	
SR_FuncR_B206	Yes	4.2.1	
SR_FuncR_B207	Yes	4.2.1	
SR_FuncR_B208	Yes	4.2.1	
SR_FuncR_B209	Yes	4.2	Visualization is provided in CDFF-Dev tools
SR_FuncR_B210	Yes	4.2.2, 4.2.3	
SR_FuncR_B211	Yes	4.2.1	

8.3.3 Performance Requirements (PerfR)

NA.

8.3.4 Interface Requirements (IntR)

Req. ID	Compliance (yes / no / partial)	Section where it is addressed	Comments
SR_IntR_B401	Yes	4.2.1, 4.2.4	

8.3.5 Resource Requirements (ResR)

Req. ID	Compliance (yes / no / partial)	Section where it is addressed	Comments
SR_ResR_B501	Yes	2.1, 2.2	
SR_ResR_B502	N/A		Depends on DFPC and is addressed in D5.3
SR_ResR_B503	Yes	5.1.2, 5.2.2, 5.3.2	

8.3.6 Operational Requirements (OpR)

NA.

8.3.7 Product assurance and safety requirements (ProdR)

NA.

8.3.8 Configuration and implementation requirements (ConfR)

Req. ID	Compliance (yes / no / partial)	Section where it is addressed	Comments
SR_ConfR_B801	Yes	4.2	