



Deliverable Reference : D4.3

Title : CDFF Unitary and Integrated Tests

Confidentiality Level : PU

Lead Partner : SPACEAPPS

Abstract : This deliverable presents the testing strategy for the CDFF, addressing individual CDFF components testing as well as combined CDFF components testing, and finally the testing approach for external interfaces (with OG2 and OG3).

EC Grant N° : 730014

Project Officer EC : Christos Ampatzis (REA)



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

D4.3: CDFF Unitary and Integrated Test Plans

DOCUMENT APPROVAL SHEET			
	Name	Organization	Date
Prepared and cross-reviewed by:	Jeremi Gancet	SPACEAPPS	22/06/2017
	Shashank Govindaraj		23/06/2017
	Jeremi Gancet	SPACEAPPS	28/07/2017 13/09/2017

D4.3: CDFF Unitary and Integrated Test Plans

DOCUMENT CHANGE RECORD				
Version	Date	Author	Changed Sections / Pages	Reason for Change / RID No
0.5	22/06/2017	Jeremi Gancet	All	Partial release
1.0	28/07/2017	Jeremi Gancet	All	Addressing post-PDR RIDs.
1.1	13/09/2017	Jeremi Gancet	All	Finalizing document

D4.3: CDFF Unitary and Integrated Test Plans

Table of Contents

1	Introduction	4
1.1	Purpose	4
1.2	Document structure	4
1.3	Applicable Documents.....	4
1.4	Reference Documents.....	4
1.5	Acronyms.....	4
2	Approach for Testing Activities.....	6
2.1	Scope	6
2.2	Approach to testing	6
2.3	Supporting tools.....	7
3	Individual CDFF Components Testing	9
3.1	CDFF Core	9
3.2	CDFF Dev.....	9
3.2.1	Data Product Management (DPM)	9
3.2.2	Orchestrator (Orch)	11
3.2.3	Data Visualization (DV)	11
3.3	CDFF Supp.....	13
3.3.1	Data Flow Control (DFC)	13
3.3.2	MW Facilitators (MWF).....	14
4	OG3 Internal Integration Tests.....	15
4.1	Overall strategy and timeframe	15
4.2	DFPC (+ DAP Tools) + Data Flow.....	15
4.3	DPM + Data Flow	16
4.4	Orch + Data Flow	17
4.5	DV + Data Flow	18
4.6	Orch + DPM + Data Flow	19
4.7	DFPC (+ DAP Tools) + DPM + Data Flow	20
4.8	All integrated: DFPC (+ DAP Tools) + DPM + Orchestrator + DV + Data Flow	21
5	External Interfaces Testing	22
5.1	OG3-OG2 interfaces	22
5.2	OG3-OG4 interfaces	23
6	Conclusion.....	24
	Appendix 1: Mock DFPC test report template	25
	Appendix 2: Illustration of Magellium MAGVALID tools	26

D4.3: CDFF Unitary and Integrated Test Plans

List of Figures

Figure 1: CDFF tests schedule.....	7
Figure 2: CDFF product tree: focus DFPC + data flow	15
Figure 3: CDFF product tree: focus on DPM + data flow	16
Figure 4: CDFF product tree: focus on Orchestrator + data flow	17
Figure 5: CDFF product tree: focus on DV + data flow	18
Figure 6: CDFF product tree: focus on Orchestrator + DPM + data flow	19
Figure 7: CDFF product tree: focus on DFPC + DPM + data flow	20
Figure 8: CDFF product tree: All integrated	21
Figure 9: OG3-OG2 interfaces highlight.....	22
Figure 10: OG3-OG4 interfaces highlight.....	23

List of Tables

No table of figures entries found.

D4.3: CDFF Unitary and Integrated Test Plans

Using InFuse Preliminary Design deliverables in Perspective of the H2020 Space Robotics Technologies SRC Call 2

Overview of the Preliminary Design WP deliverables' content and purpose/logics within InFuse

D4.1: The goal of D4.1 is to present the choices of the data processing and data fusion functions that will be developed within InFuse.

After an analysis of operational scenarios which exhibit the required InFuse outputs, the list of these outputs ("data products") is defined. For each data product, a brief analysis of possible solutions is made, and a solution is selected. Each solution is defined by a "Data Fusion Process Compound" (DFPC), and is described with a coarse level of precision. The refinement of the DFPCs into a series of elementary "Data Fusion Nodes" (DFNs, i.e. elementary data processing functions), and the definition of the organisation of the DFNs to compose a DFPC is matter for further work for the InFuse project.

The DFPCs are defined for both the planetary and orbital reference implementations. While both implementations mostly require different DFPCs, many DFNs will be shared between the two targeted contexts.

The document also presents a sets of operational scenarios for each reference implementation. They have the purpose of putting in context the proposed DFPCs.

D4.2: This deliverable focuses on the one hand on software architecture considerations, with the preliminary specification of all the key components that the InFuse CDFF consists of, and on the other hand on the identification of all relevant interfaces, both internal to InFuse and the external ones with respect to other OGs (OG2-ERGO and OG4-I3DS) in particular.

Note that the architecture and ICD material introduced in this document are essentially application independent – application specific considerations will be introduced at a later stage, during the detailed design of the InFuse CDFF (each of the orbital track and planetary track will then be tailored, accordingly).

This deliverable will serve as a starting point for the detailed design work, in the following work package.

D4.3 (this deliverable): The purpose of this document is to define the strategy and overall approach for the testing activities to be carried out internally (OG3), so that to ensure that the developed software is sound while meeting the requirements expressed in the earlier phases of the project. The content is orbital / planetary independent: it is not in the scope of this deliverable to specify the testing approach with facilities and EGSEs specific to either the orbital or planetary scenarios. These aspects will instead be addressed in the detailed design activities that are just starting as this document is being written, and will be released by the CDR milestone.

Relation between WP4 deliverable's content and other "common building block" OGs

The scope of OG3-InFuse lies essentially between OG4-I3DS, that produces raw sensor data, and OG2-ERGO, which controls all the rover activities. The interfaces with these two OGs are defined in the deliverable D4.2:

- The interfaces with OG4-I3DS are defined by sensor data types

D4.3: CDFF Unitary and Integrated Test Plans

- The interfaces with OG2-ERGO are defined on the one hand by the types of data products (mainly terrain maps and localization information related to the rover and the terrain, and to the chaser and target satellites), and on the other hand by requests made by OG2-ERGO to OG-InFuse.

OG2-ERGO is interfaced with OG3-InFuse at the granularity of the DFPCs: OG2-ERGO has no view of the inner mechanisms of OG3-InFuse that assembles DFNs into a DFPC. An OG2-ERGO request triggers the activation of a DFPC (which can either be synchronous or not), along with given parameters, and the DFPC returns the requested data products with an execution report. Within a DFPC, the DFNs are assembled, sequenced and triggered via pre-defined scripts, which are configured according to the parameters associated to the OG2-ERGO requests.

The definition of the DFPCs provided in D4.1 is generic, and makes no hypothesis regarding the integration middleware within which they will be developed. The way the developed functions will be integrated within the OG1-ESROCOS framework (and potentially other target middlewares) is depicted in the document D4.2.

Finally the content of D4.3 is largely centered on InFuse internal testing and validation activities – in that context, “integrated test plans” deal with the joint testing of several sub-parts of the InFuse framework, not InFuse and other OGs. Still, several components of the InFuse CDFF have interfaces with other OGs – mainly OG2-ERGO and OG4-I3DS. For these ones, it is foreseen in the test plan to develop specific components as placeholders of ERGO and I3DS, exposing the interfaces that are assumed to be the ones with which InFuse should integrate in the upcoming Space Robotics SRC projects. We call them M-OG2 and M-OG4 (M standing for Mock). Their purpose again is only to make it possible, internally, to carry out end-to-end tests and ensuring the soundness of the CDFF interfaces.

Applicability to the H2020 Space Robotics Technologies SRC upcoming calls (i.e. OG7 to OG11a/b).

Environment perception, be it to model the environment or to localize the controlled robotic platform within this environment, is at the core of autonomous operations, and is therefore required in all the applications defined by the upcoming calls.

For each of the future Operational Grants, a set of relevant DFPCs identified in D4.1 document is identified below as having particular relevance (note however that this list is based on very preliminary, and still limited knowledge of the content and scope of each upcoming OG).

OG7: (Orbital Support Services): Long/Mid/Close-range Tracking and Detection, 3D Target reconstruction, Point Cloud based Localisation

OG8: (Modular Robotized Assembly): Mid/Close-range Tracking and Detection, Point Cloud based Localisation

OG9: (Satellite Re-configuration): Mid/Close-range Tracking, Point Cloud based Localisation

OG10: (Advanced Autonomy): DEM Building + Soil Type Map, and all the rover localisation DFPCs: Visual Odometry, Visual/LIDAR based SLAM, Scientific Area Localisation, Visual Map-based Localisation, Absolute Localisation.

OG11a: (Advanced Mobility): DEM Building + Soil Type Map and Visual Odometry for extreme terrain mobility, plus all the localisation DFPC for the coordination of multiple platforms.

D4.3: CDFF Unitary and Integrated Test Plans

OG11b: (Robotized Construction): The required DFPCs for this OG are certainly similar to some of the ones required for the orbital OGs: Long/Mid/Close-range Tracking and Detection, 3D Target reconstruction, Point Cloud based Localisation – though in planetary context. DEM building and rover localization DFPCs remain relevant.

Deliverable D4.2 provides the latest baseline about the InFuse CDFF architecture and ICD. In perspective of the next OGs, it is essential to understand the proposed architecture and mechanisms to handle data, and the proposed approach to generate middleware specific reference implementations from the vanilla (middleware independent) CDFF environment.

In perspective of the upcoming OGs, D4.3 has limited relevance as its purpose is essentially to define the InFuse internal strategy to test the various components of the CDFF. It is therefore of little use for what concerns the preparation of bids for the next call, and similarly would have limited relevance for the future implementation of the next OGs.

1 Introduction

1.1 Purpose

The purpose of this document is to define the strategy and overall approach for the testing activities to be carried out internally (OG3), so that to ensure that the developed software is sound while meeting the requirements expressed in the earlier phases of the project.

1.2 Document structure

In brief, the document is structured as follows:

Section 1: This introductory material.

Section 2: Scope and approach for testing activities

Section 3: Individual CDFF components testing

Section 4: OG3 internal integration testing

Section 5: External interfaces testing

Section 6: Concludes this document

1.3 Applicable Documents

AD1 InFuse Grant Agreement (confidential document)

AD2 InFuse Consortium Agreement (confidential document)

1.4 Reference Documents

RD1 Description of Action document (confidential document)

RD2 D3.2: System Requirements and Operational Concept (public document, available on request)

RD3 D3.3: Functional and Physical Architecture Specification (public document, available on request)

1.5 Acronyms

AF: Autonomy Framework (aka. OG2 / ERGO)

API: Application Program Interface

CDFF: Common Data Fusion Framework

CDR: Critical Design Review

DAP: Data Analysis and Performance

DFC: Data Flow Control

D4.3: CDFF Unitary and Integrated Test Plans

DEM: Digital Elevation Map

DF: Data Fusion

DV: Data Visualization

DFN: Data Fusion Node

DFPC: Data Fusion Process Chain

DPM: Data Product Management

EGSE: Electric Ground Support Equipment

IMU: Inertial Measurement Unit

M-OG(X): Mock of OG(X) software, substituting to it as part of the tests

MW: Middleware. In this document is used as synonym for RCOS.

MWF: Middleware Facilitator

OG: Operational Grant

PCL: Point Cloud Library

RCOS: Robotics Control Operating System (e.g. ROS, Rock, GenoM)

RI: Reference Implementation

TRR: Tests Readiness Review

2 Approach for Testing Activities

2.1 Scope

It is important to stress that the test plan reported in this deliverable deals with the CDFF components development (including the data fusion features, aka. DFNs) in abstraction of particular application cases.

In the following activities (i.e. detailed design), several tasks (T5.3, 5.4, 5.5) are provisioned to specify detailed test plans in perspective of the Reference Implementations (planetary track, orbital track, and for what concerns EGSEs), and will also take into account the specificity of validation platforms that OG6 will provide to OG3. Those aspects are therefore not in the scope of the present report.

2.2 Approach to testing

For individual components testing within the CDFF, the approach is to define best practices to ensure a proper test coverage (mostly unit testing and functional testing based). A distinction is done between legacy code and new code development, assuming that legacy code comes with legacy unit tests (or that it is otherwise possible to define unit tests at the interface between novel and legacy code). New code development shall anyway systematically come with unit tests.

Functional testing is provisioned: each individual requirement (as identified in deliverable D3.2) should reflect into one or several features or interfaces. The span of each of these features and interfaces should in turn be entirely covered by a suitable combination of test cases. Traceability will be maintained.

For OG3 internal components integration tests, the focus is on test cases allowing to validate interfaces between individual CDFF components, starting from bi-lateral tests and progressively combining more components.

Regression tests based on unit tests will additionally be carried out on a regular basis.

For what concerns OG3 – OG2 and OG3 – OG4 interfaces testing, we will develop mock OG2 (M-OG2) and OG4 (M-OG4) components exposing the non-functional interfaces that we expect to see with the actual OG2 and OG4 frameworks.

M-OG2 will be able to generate fake requests to OG3 and to receive data from OG3 as the actual OG2 should do.

M-OG4 should in particular be able to provide sensors data (either fake or recorded samples).

The inter-OGs ICD will be the reference document to specify and implement M-OG2 and M-OG4 interfaces.

A sample diagram indicates the overall approach planned for InFuse during production and deployment of the system. The diagram is only representative of the similar approach that will be taken for InFuse.

D4.3: CDFF Unitary and Integrated Test Plans

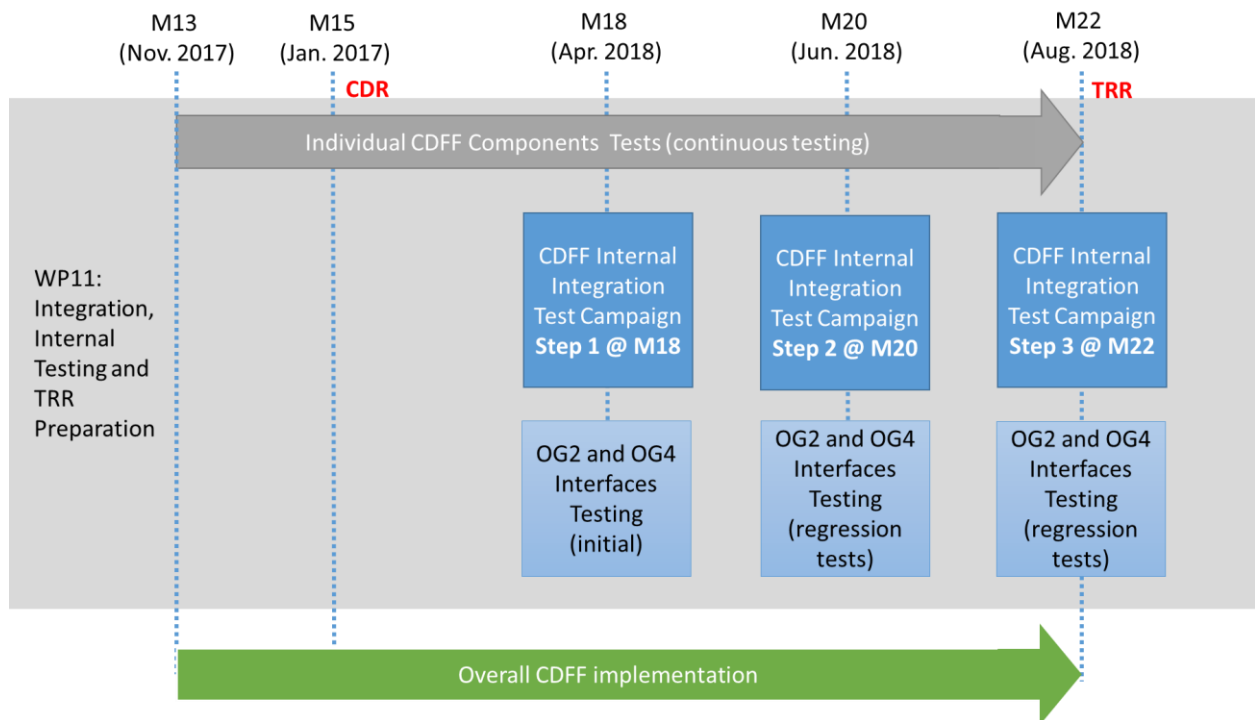


Figure 1: CDFF tests schedule

2.3 Supporting tools

The following criteria will be used to evaluate the choice of specific tools for automating the testing of software components in CDFF:

- Explore possibilities of using automated C/C++ unit testing tools like VectorCast, GoogleTest, BoostTest etc. The applicability or choice of a specific testing tool for InFuse will be done during the development phase to best understand the requirements.
- Static code analysis to check for code quality and other white box testing steps (Ex. CLANG)
- Generate test case code based on scripting test cases.
- Run time code coverage for test cases
- Parametric testing of interface over a wide range of values (complex data structures)
- Python binding of DF-core libs can be leveraged => scripting unit test cases in Python using logs for injecting data or collecting fused data from DFNs and eventually DFPCs
- Use of docker to ease continuous integration, cross compilation and deployment of a subset of the InFuse framework for integrated testing.

Some of the tools being investigated currently are as follows:

D4.3: CDFF Unitary and Integrated Test Plans

CppUnit: CppUnit is a unit testing framework module for the C++ programming language. It allows unit-testing of C sources as well as C++ with minimal source modification. The framework runs tests in suites. Test result output is sent to a filter, the most basic being a simple pass or fail count printed out, or more advanced filters allowing XML output compatible with continuous integration reporting systems.

MockCpp: mock++ is a jmock-like generic easy-of-use C++ Mock Framework, creating a mock object by mock++ is virtually as easy as using jMock.

Boost.Test. It is less convenient to use as the test auto registration does not work very well and sometimes you have to force some headers inclusion. Moreover, it does not have a native mock system. On the positive side it is compatible with CTest and CDash.

Google test looks good on the paper and has a mock system. It is also compatible with CTest and CDash.

MAGVALID (MAG proprietary): allows to generate word document with test description and requirements traceability. Note that this tool will only help the consortium producing tests description in the scope of the project, the tool itself is not intended to be released as part of the CDFF framework (therefore licensing considerations, out of the InFuse consortium, are not relevant).

3 Individual CDFF Components Testing

3.1 CDFF Core

The CDFF core contains all the Data Fusion Nodes, each of them consisting of key data fusion capabilities.

It is essential to ensure a proper test coverage for each of the DFN included in the core. For that purpose, each partner will be requested to follow best practices regarding testing, so that to make sure that only properly tested code is brought to the CDFF.

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.

Besides unit testing, functional testing will be carried out at each integration step, to verify requirements coverage, as far as the “CDFF core as a whole” is concerned. The objective is to successfully “pass” the requirements verification for all the identified CDFF core requirements (i.e. A100, A200, A300, A400, A500, A600, A700, A800, A900 series) for the Mandatory requirements, and for as many Desirable requirements as possible.

Note however that the requirements baseline as elicited in D3.2 may further evolve, as consequences of interactions with other OGs and PSA.

Pass criteria: each individual DFN of the CDFF Core will be considered as successfully tested if all related functional tests are successful (i.e. meeting the corresponding functional requirements) and if the overall unit tests for that DFN are ran successfully.

3.2 CDFF Dev

3.2.1 Data Product Management (DPM)

As introduced in D4.3, the DPM will manage an EnviRe Graph in which multiple data products will be stored in a spatiotemporal consistent manner. EnviRe currently provides serialization for various data types related to environment representation (e.g. maps) and it is extendable to enable serialization of further data types.

EnviRe already comes with extensive unit tests that can be used in a large extent as part of the DPM development.

New pieces of code required for DPM, besides EnviRe existing ones, shall come with their own comprehensive unit tests covering specifically the features introduced with that new code. Unit tests shall also cover the connection between novel source code and EnviRe existing one.

Regarding functional testing, a certain number of requirements introduced previously in D3.2 connect to the features expected for the DPM. These requirements are:

D4.3: CDFF Unitary and Integrated Test Plans

{SR_UserR_A112}	CDFF is highly coupled to the AF (OG2) as a toolbox with DF bricks or as a standalone agent
LEVEL	Mandatory
VERIFY_METH	The data fusion chain designed a priori produced the requested data product. The request is coming either from the OG2 framework or from an operator through the by OG3 provided interface.

{SR_UserR_B103}	Tools to handle all reference data within the CDFF for functional assessment
LEVEL	Mandatory
VERIFY_METH	User is able to trigger the visualization of data products during different stages of DF process

{SR_UserR_B107}	CDFF framework is compatible with future adjunction of sensors and DF techniques
LEVEL	Mandatory
VERIFY_METH	Demonstration of the integration of a new sensor and associated core processing function(s), insertion of this function in a data processing chain.

{SR_FuncR_B202}	Tools to handle all reference data required for CDFF functional and performance assessment
LEVEL	Mandatory
VERIFY_METH	CDFF functional and performance assessment can be performed using the reference data handling tools.

{SR_FuncR_B210}	On-board a priori knowledge data retrieval: CDFF ability to retrieve prior knowledge on the environment from the on-board Database
LEVEL	Mandatory
VERIFY_METH	VERIFY_METH: Demonstration of the implementation of a data processing chain that exploits data products stored in the CDFF internal database.

For each of these requirements, clear and measurable pass/fail tests will be expressed in the context of each specific RI (i.e. T5.x), so that to allow verifying whether the DPM adequately addresses these requirements.

D4.3: CDFF Unitary and Integrated Test Plans

For that purpose, relevant data sets covering each of the RI will be required to verify that the DPM can effectively handle all relevant data products.

Pass criteria: will be considered as successfully tested if all related functional tests are suitably addressed (i.e. refined requirements addressed suitably) and if the overall DPM related unit tests are ran successfully.

3.2.2 Orchestrator (Orch)

The orchestrator is an important component of the CDFF-Dev that supervises CDFF activities according to data product requests expressed by the AF (OG2).

It is foreseen to be developed as a new piece of software, though possibly relying on third party libraries providing efficient algorithmic mechanisms for the purpose of tasks supervision. Such relevant libraries exist in ROS, for instance SMACH (which btw. is python based), that conveniently offers finite state machine mechanisms adapted to robotic tasks supervision.

Besides unit tests implementation along with new software development, functional test will be provisioned as with other CDFF components.

As a baseline the following requirements are identified as relevant for that purpose:

{SR_UserR_B101}	A software suite for configuration, prototyping and validation of CDFF-core
LEVEL	Mandatory
VERIFY_METH	The user has an framework available in which data fusion libraries can be connected and orchestrated. The quality of these can be analyzed with tools also provided by the framework.

{SR_FuncR_B211}	Configuration or orchestration utility to assemble units of DF components – offline and during runtime.
LEVEL	Desirable
VERIFY_METH	An orchestrator instance modifies on runtime the data flow of a data fusion process.

Pass criteria: will be considered as successfully tested if all related functional tests are suitably addressed (i.e. refined requirements addressed suitably) and if the overall Orchestrator related unit tests are ran successfully.

3.2.3 Data Visualization (DV)

Data Visualization, although considered a peripheral component of the CDFF, will be instrumental in the verification of the CDFF data fusion capabilities and performances, as well as being a very convenient tool for future potential users. Although not primarily intended to be exploited for operational monitoring

D4.3: CDFF Unitary and Integrated Test Plans

on a target middleware, it can be used remotely on a separate computer to visualize data products from the CDFF during run time.

This visualization tool will rely on a large extent on the existing visualization software used with EnviRe (namely vizkit3d). As with the DPM tool, new pieces of code around the data visualization tool shall come with their own comprehensive unit tests covering specifically the features introduced with that new code. Unit tests will also cover the connection between novel source code and the existing one.

Besides unit tests implementation, functional test will be provisioned as with other CDFF components. As a baseline the following requirements are identified as relevant for that purpose:

{SR_UserR_A116}	Operator interaction with the CDFF is foreseen at least for the functionalities of object detection and tracking.
LEVEL	Mandatory
VERIFY_METH	The visualizations for the algorithms of object detection and tracking can be started by the operator. These should be available on real-time depending on the communication conditions.

{SR_OpR_A602}	Operator Compatibility: Observables delivery with data res and rates compatible with monitoring purpose
LEVEL	Mandatory
VERIFY_METH	The operator can trigger the visualization tools and the data frequency at the different nodes can be observed.

{SR_UserR_B103}	Tools to handle all reference data within the CDFF for functional assessment
LEVEL	Mandatory
VERIFY_METH	User is able to trigger the visualization of data products during different stages of DF process

{SR_UserR_B104}	Graphical tools to enable data visualization for different CDFF processing steps
LEVEL	Mandatory
VERIFY_METH	The user can trigger the launch of visualizations of the data products at the different stages of their processing.

D4.3: CDFF Unitary and Integrated Test Plans

{SR_FuncR_B203}	Graphical tools to enable data visualization throughout CDFF processing steps
LEVEL	Mandatory
VERIFY_METH	User is able to trigger the visualization of data products, at the output of nodes of the DF chain.

{SR_FuncR_B209}	Configurability: Functionality observables
LEVEL	Mandatory
VERIFY_METH	The user and the operator can trigger the execution of visualizations for the data products being generated by the CDFF.

Pass criteria: will be considered as successfully tested if all related functional tests are suitably addressed (i.e. refined requirements addressed suitably) and if the overall DV related unit tests are ran successfully.

3.3 CDFF Supp

CDFF Supp deals with the supporting tools of the CDFF, that are relevant in support to the development of new Reference Implementations, but that are not expected to be deployed on target robotic platforms.

3.3.1 Data Flow Control (DFC)

The DFC deals with middleware agnostic solution to be used during RI development phases, to conveniently test data transfers between components of the CDFF – in particular, within (and between) DFNs.

It will rely on Python Pandas, and will implement all the custom features required to enable CDFF internal data transfers (once more, for development support only). That novel custom code will come with comprehensive unit tests. This is also the case of the software covering specifically the interface between that new source code and the PythonPandas libraries.

As a baseline, the following requirements will be taken into account as part of the functional tests:

{SR_UserR_A113}	CDFF implementation based on best trade off in terms of performance, flexibility, development and validation
LEVEL	Mandatory
VERIFY_METH	Exploit performances analysis tools in terms of resource consumption and quality and precision of the data products. Validate the development workflow by adding new core functions and defining a data processing chain that use them.

D4.3: CDFF Unitary and Integrated Test Plans

{SR_FuncR_B201}	Emulation tools of robotic system components interacting with CDFF-core (Functional Layer of AF) – for implementation and demonstration
LEVEL	Mandatory
VERIFY_METH	Realistic and representative sensor data streams are provided by the emulation modules.

Pass criteria: will be considered as successfully tested if all related functional tests are suitably addressed (i.e. refined requirements addressed suitably) and if the overall DFC related unit tests are ran successfully.

3.3.2 MW Facilitators (MWF)

The facilitators are CDFF-SUPP components introduced as means to ensure good portability of the CDFF data fusion capabilities.

The facilitators consist of a set of tools that should ease the porting of CDFF data fusion nodes (DFNs) from the “vanilla” version available as a baseline in the CDFF framework, towards different target “middlewarees” (or RCOS). This includes primarily ESROCOS, but also ROCK, ROS and GenoM. Each time, the MW facilitators will support users through the automatic generation of MW specific templates (with a certain amount of MW specific “codes”). The automatic character of the porting of new DFNs to specific MW, relying on facilitators, will depend upon the advancement of these tools within the time frame of the project. Due to the complexity and effort required to develop advanced automatic porting tools, it is anticipated that only partially automatic porting will be enabled, reducing manual effort but not entirely substituting to it.

MW facilitators software will be come with relevant unit tests, although being of lesser criticality than e.g. DFNs (that shall eventually run on target robotic platforms).

Note also that there are no specific requirements in relation to the MW facilitators: this is due to the fact that this objective was defined by the consortium, as a will to allow a high level of porting of the developed CDFF software, and therefore to maximize the uptake by the wide robotics community.

Pass criteria: will be specified on a case by case basis, depending on the target MW. As a baseline, a time reduction of at least 30% for the porting of CDFF DFN to the target MW, could be used as a measurable target.

D4.3: CDFF Unitary and Integrated Test Plans

4 OG3 Internal Integration Tests

4.1 Overall strategy and timeframe

The approach to perform the integration of the different CDFF components rely on a progressive, bi-lateral and then multi-lateral integration steps.

This will culminate with an OG3-internal test and verification campaign including all the CDFF components and ensuring that all expected features and requirements are suitably addressed - and that the CDFF is operational.

Note that OG3 internal integration tests will be possible only after involved individual components were successfully tested (with evidences, typically under the forms of logs) as individual components (i.e. unit testing and functional testing).

We go in more details below with each of the intermediary steps that are deemed necessary before the final comprehensive integration and verification step.

4.2 DFPC (+ DAP Tools) + Data Flow

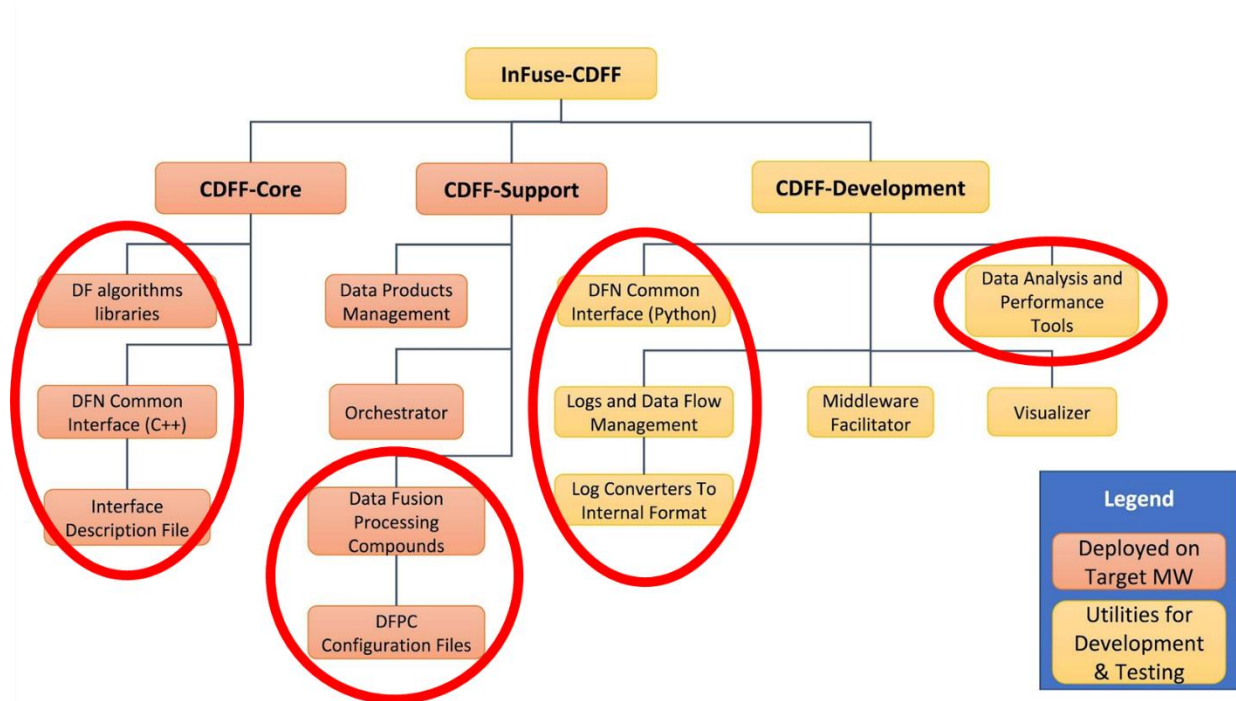


Figure 2: CDFF product tree: focus DFPC + data flow

This part of the integration of the CDFF components is an essential one: it consists in testing together DFN, connected within compounds (i.e. DFPC), so that to verify that the behaviour and outputs are as expected.

All DFNs shall be tested at least one in such a fashion, as part of a DFPC.

D4.3: CDFF Unitary and Integrated Test Plans

In most cases, DFNs will be integrated and tested within a certain number of different DFPCs, resulting in different outputs.

Such DFPC oriented tests will be carried out at each of the 3 integration and testing steps, as DFNs will progressively be released by project contributors.

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.

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.

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.

A draft of a mock DFPC test report is provided in appendix 1, for the reader interest.

Pass criteria: the test will be considered successful if it can be verified that each DFN could be used at least once within a DFPC, could receive and process relevant data inputs, and could generate relevant data outputs as expected for that DFN. If certain DFNs have several operational modalities, each of them shall be separately covered as part of this test, to be successful.

4.3 DPM + Data Flow

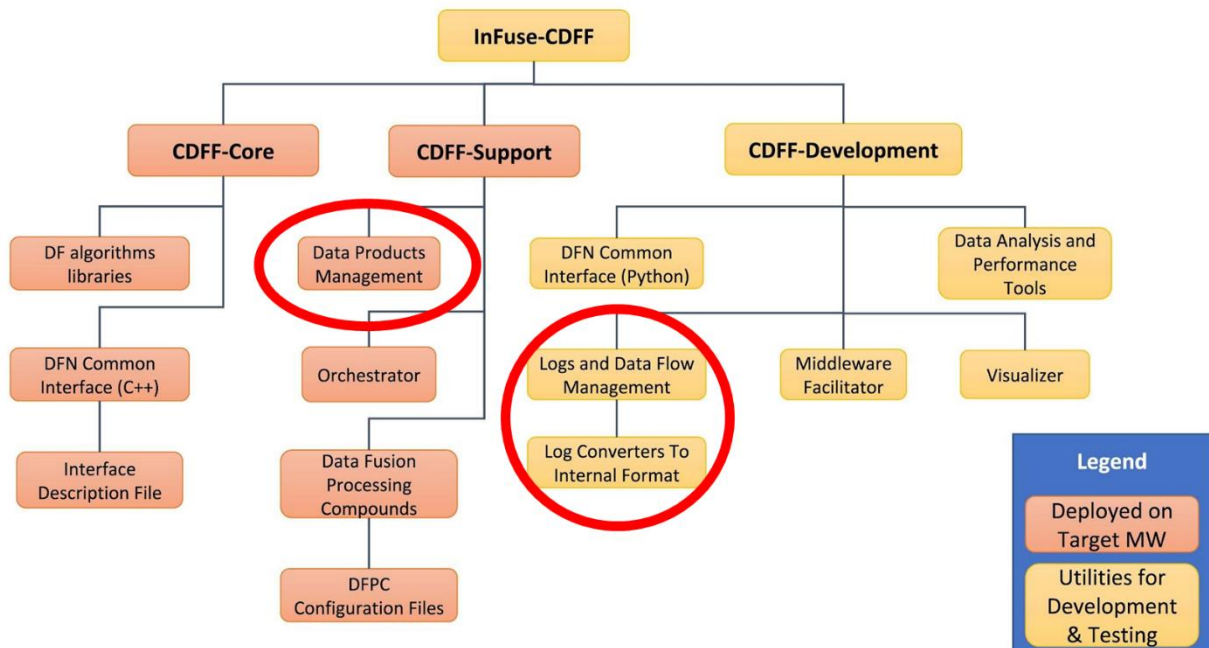


Figure 3: CDFF product tree: focus on DPM + data flow

D4.3: CDFF Unitary and Integrated Test Plans

The purpose of this test is to ensure that (i) the DPM is properly interfaced with the Data Flow solution developed in the CDFF, and (ii) can properly receive (and handle) representative data samples of all the relevant data types.

In practice, two test support tools will be developed to help carrying out these tests:

- (1) A “mock DFPC output generator”, as a mean to inject well known data (properly relying on the Data Flow). The DPM shall be able to receive these data and handle/store them adequately.
- (2) A “mock Orchestrator” with the capability to issue requests for specific data products, and with the ability to receive (through the Data Flow) and verify the validity of the returned data.

A test report will be produced after each test session accordingly.

Pass criteria: the test will be considered successful if it can be verified that (i) the DPM is able to receive and properly handle any relevant data samples and data product types, and (ii) is able to receive and properly interpret data product requests and accordingly to return (through Data Flow) expected data product material – data product requests shall cover all possible types of data products.

4.4 Orch + Data Flow

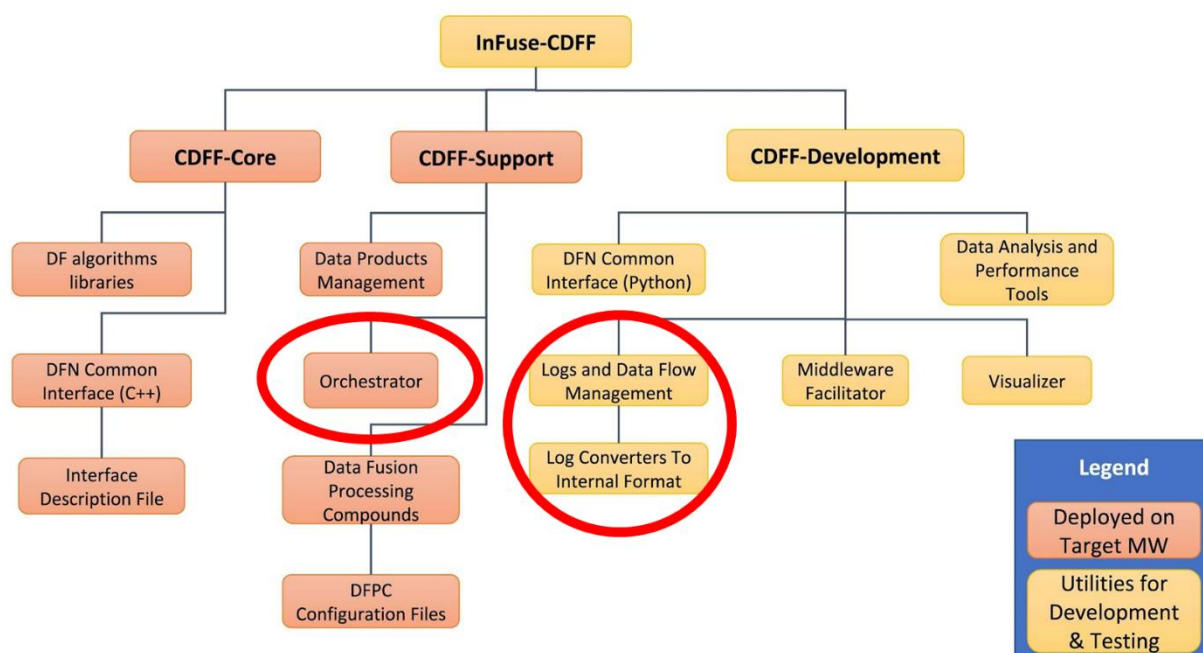


Figure 4: CDFF product tree: focus on Orchestrator + data flow

This test aims to ensure that the Orchestrator is capable of (i) handling the DFPCs appropriately, (ii) to correctly transmit data product requests (originating from OG2) to the DPM, and (iii) to interface towards both OG2 and OG4 API. Not that the external interfacing aspects are covered separately in section 5 and 6.

In practice, two test support tools will be needed to help carrying out these tests:

- (1) A “mock DPM” with the ability to receive (through the Data Flow) and verify the validity of requests for data products, and the ability to return (through the Data Flow) representative data

D4.3: CDFF Unitary and Integrated Test Plans

products, that the Orchestrator shall be able to handle suitably.

- (2) A “DFPC simulator”, as a mean to ensure that the interactions the Orchestrator shall have with DFPCs are suitably implemented.

A test report will be produced after each test session accordingly.

Pass criteria: the test will be considered successful if it can be verified that (i) the Orchestrator generates data product requests with expected format and content for all possible data products types, and (ii) the Orchestrator generates DFPC aimed requests with expected format and content for all possible types of interaction between the Orchestrator and DFPC.

4.5 DV + Data Flow

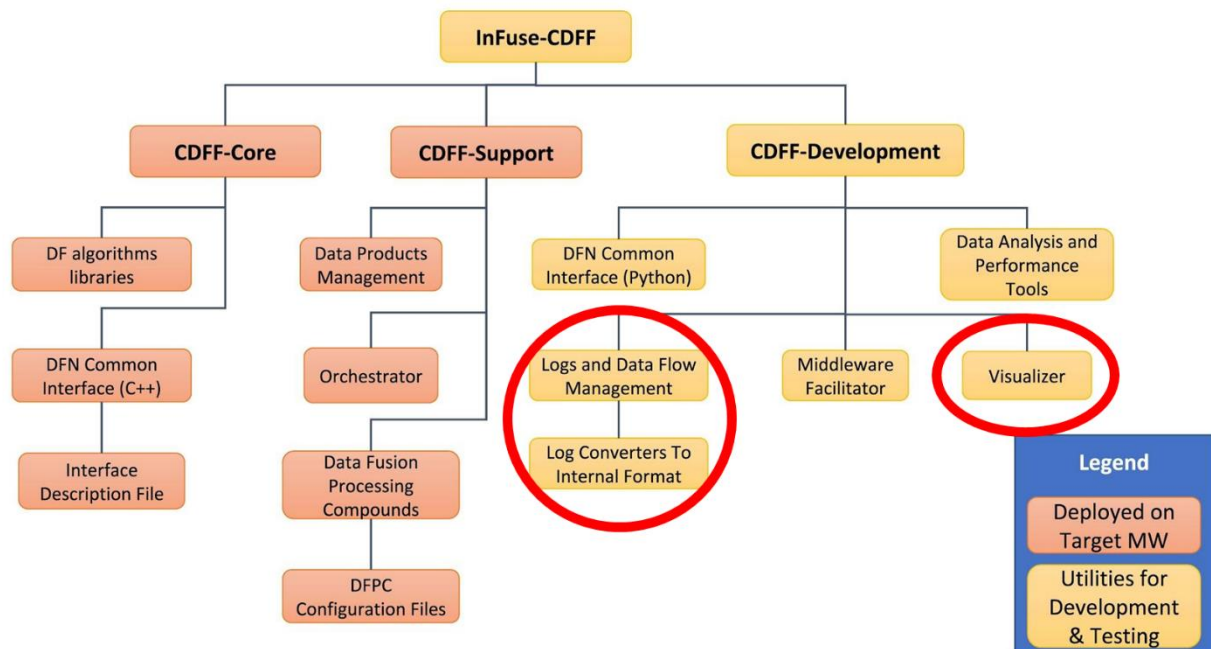


Figure 5: CDFF product tree: focus on DV + data flow

The Data Visualizer comes as a tool supporting the CDFF developers and users.

Its integrated testing will essentially consist in ensuring that it can properly receive representative data samples/products through the Data Flow, and get them suitably rendered for the user.

In practice a mock “DFPC output generator” will be developed as a mean to inject well known data (properly relying on the Data Flow), that the DV shall then capture and render suitably.

A test report will be produced after each test session accordingly.

Pass criteria: the test will be considered successful if it can be verified that all relevant data samples and data products can be properly received (through the Data Flow), handled and visually rendered by the DV.

D4.3: CDFF Unitary and Integrated Test Plans

4.6 Orch + DPM + Data Flow

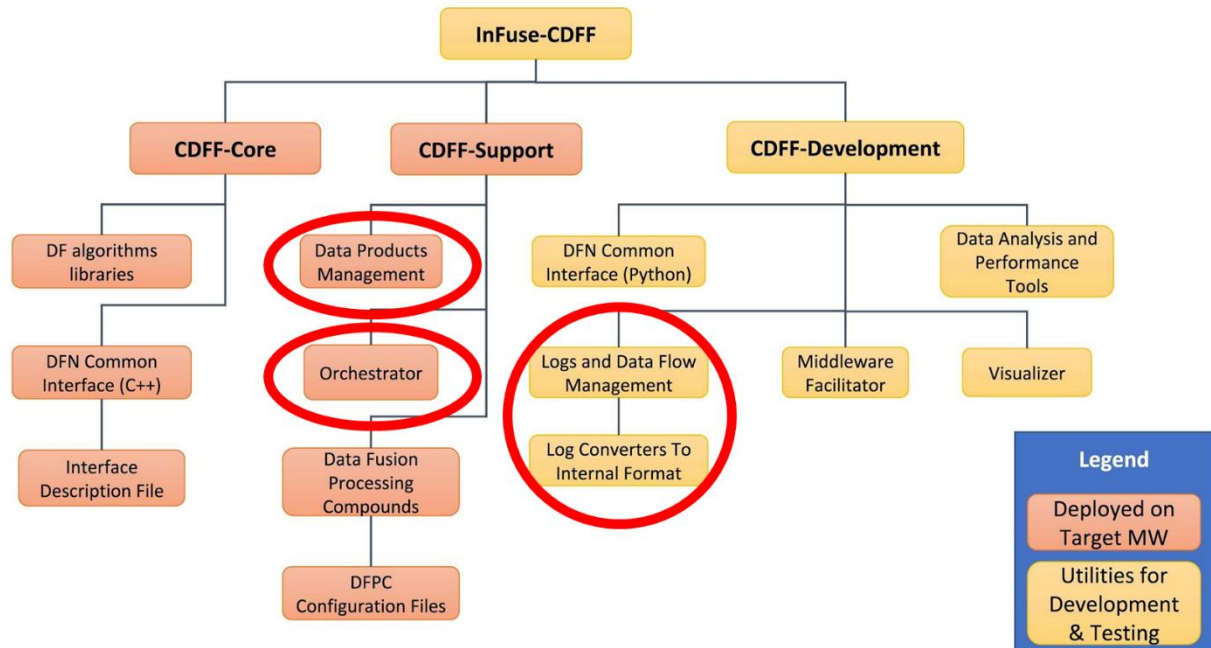


Figure 6: CDFF product tree: focus on Orchestrator + DPM + data flow

This test combines the DPM and the Orchestrator, meaning that mock elements for the DPM or the Orchestrator are no longer needed with that respect.

Still, a “DFPC simulator” is deemed required:

- as a mean to inject representative data samples (properly relying on the Data Flow). The DPM shall be able to receive these data and handle/store them adequately.
- as a mean to ensure that the interactions the Orchestrator shall have with DFPCs are suitably implemented.

The integration tests will in particular focus on the interactions between the Orchestrator and the DPM. A baseline test scenario would consist in the following steps:

- (1) The Orchestrator selects and enable a DFPC, as a reaction to an assumed request from OG2 (will in the test be triggered directly at the Orchestrator level).
- (2) Mock DFPC simulator is used to produce and dispatch a representative data fusion product.
- (3) The DF product is received by the DPM, and the Orchestrator is notified of its availability.
- (4) The Orchestrator requests the resulting DF product from the DPM.
- (5) The DPM passes the DF product material to the Orchestrator.

Other interactions between the Orchestrator and the mock DFPC may be envisaged, as required to ensure that the test coverage is suitable.

D4.3: CDFF Unitary and Integrated Test Plans

Pass criteria: the test will be considered successful if it is verified that the entirety of possible interactions between the DPM and the Orchestrator could be nominally covered through one or several test scenarios (as needed to ensure a comprehensive coverage).

4.7 DFPC (+ DAP Tools) + DPM + Data Flow

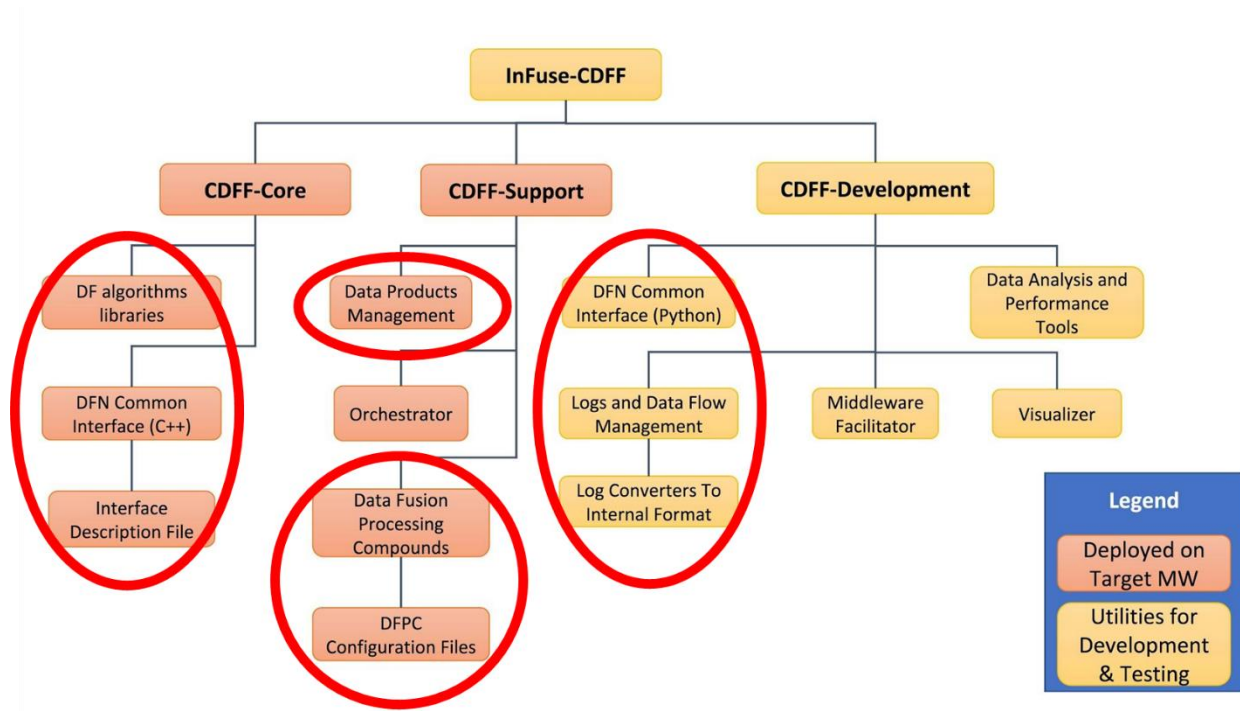


Figure 7: CDFF product tree: focus on DFPC + DPM + data flow

In this test, the focus is on the interactions between the DPM and the DFPCs, all of that through the Data Flow.

The baseline is the one of the “DFPC + Data Flow”, except that the focus will be on ensuring a suitable data format coverage for the DPM.

A selection of DFPCs allowing to produce all required data product types will therefore be elaborated, combining each time relevant DFNs, for the purpose of the test.

Pass criteria: The test will be considered as successful if it is verified that the DPM can handle all the expected data product types, as produced by the genuine DFPCs (and DFNs).

D4.3: CDFF Unitary and Integrated Test Plans

4.8 All integrated: DFPC (+ DAP Tools) + DPM + Orchestrator + DV + Data Flow

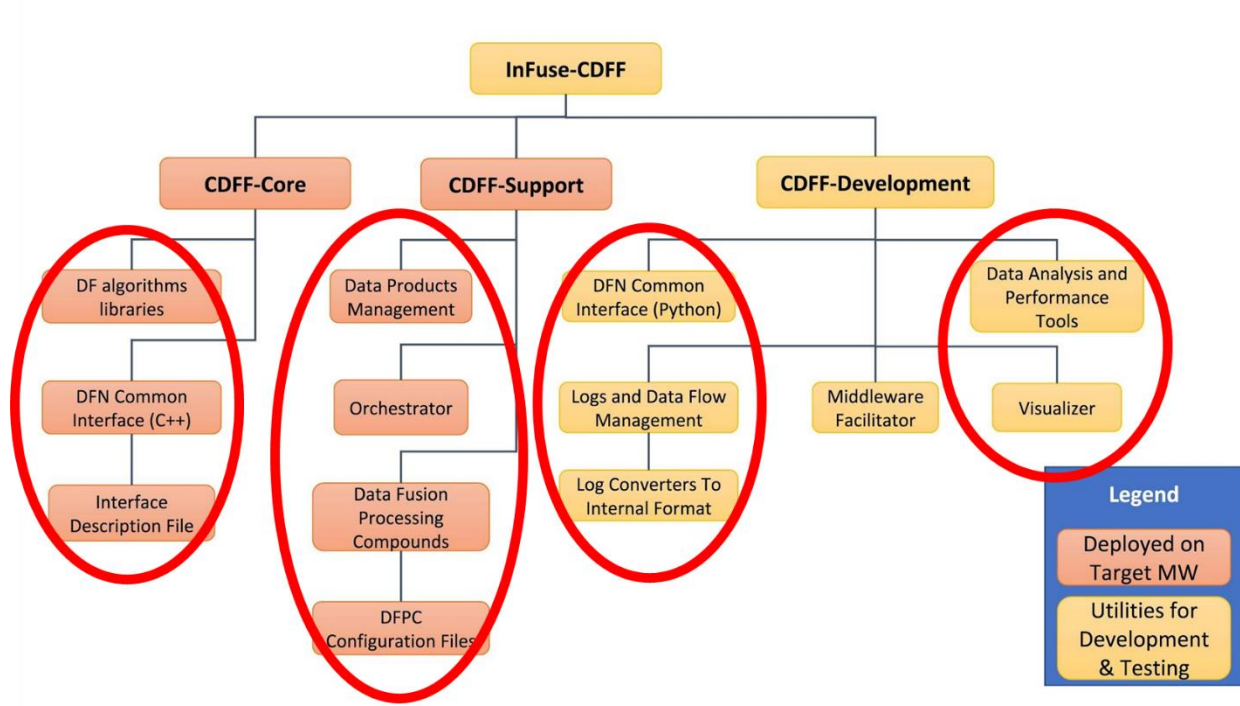


Figure 8: CDFF product tree: All integrated

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.

A baseline test scenario would consist in the following steps:

- (1) The Orchestrator selects and enable 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, either 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 DF product is received by the DPM, and the Orchestrator is notified of its availability.
- (6) The Orchestrator requests the resulting DF product from the DPM.
- (7) The DPM passes the DF product material to the Orchestrator.
- (8) During all the process, the Data Visualizer is able to render relevant data samples and data products, on request.

Pass criteria: The test will be considered as successful if it is verified that the all the CDFF components behave nominally, and that multiple (and representative) DFPCs could be successfully solicited during the execution of scenarios comparable to the baseline test scenario above.

5 External Interfaces Testing

5.1 OG3-OG2 interfaces

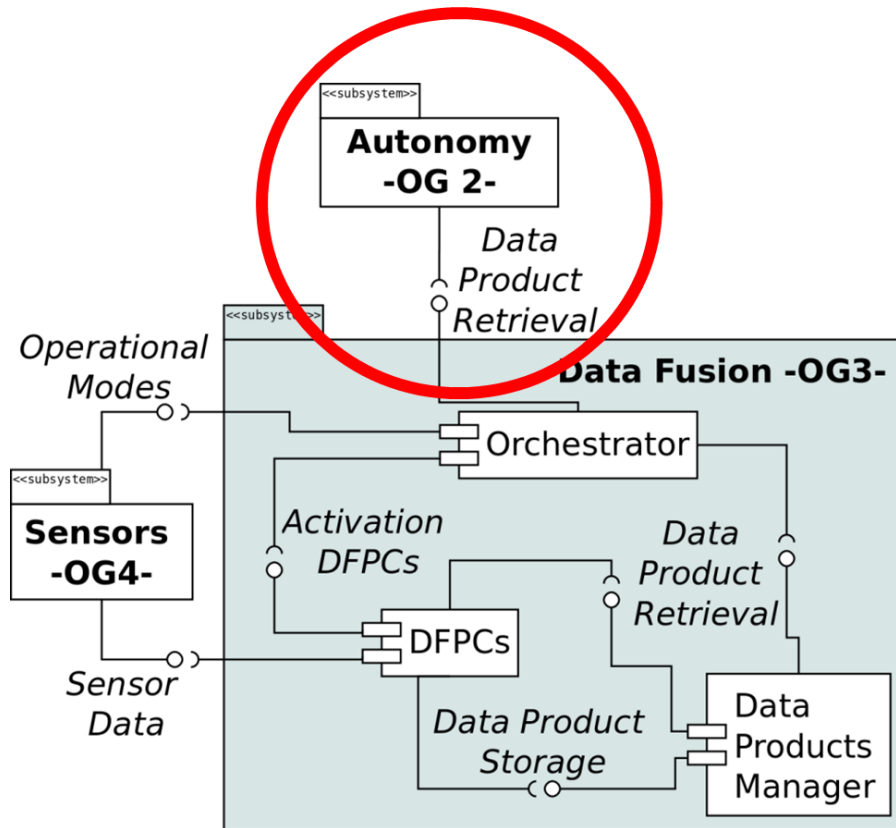


Figure 9: OG3-OG2 interfaces highlight

The interfaces concerned are essentially the ones allowing OG2 to request data products to OG3, and allowing OG3 to return requested data products.

This interface is implemented at the Orchestrator level. In order to test the identified interfaces, a “mock OG2” software (aka M-OG2) will be purposely developed as a mean to issue requests towards the Orchestrator and to ensure that received data products match expectations (structure, content). In order to provide the Orchestrator with relevant data products, a mock DPM will also be used to provide the Orchestrator with the required data products (as pre-recorded samples) – possibly through the CDFF Data Flow, for the sake of convenience.

These interfaces tests will be carried out in SARGON or ESROCOS environment (depending on readiness and availability of OG1 material), and the data products will be expressed in (extended) ASN.1 formalism as defined by OG1. This will require the Orchestrator to be modelled as an ESROCOS component.

Pass criteria: The test will be considered as successful if it is verified that the Orchestrator can handle requests for all possible types of Data Products, 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..).

D4.3: CDFF Unitary and Integrated Test Plans

5.2 OG3-OG4 interfaces

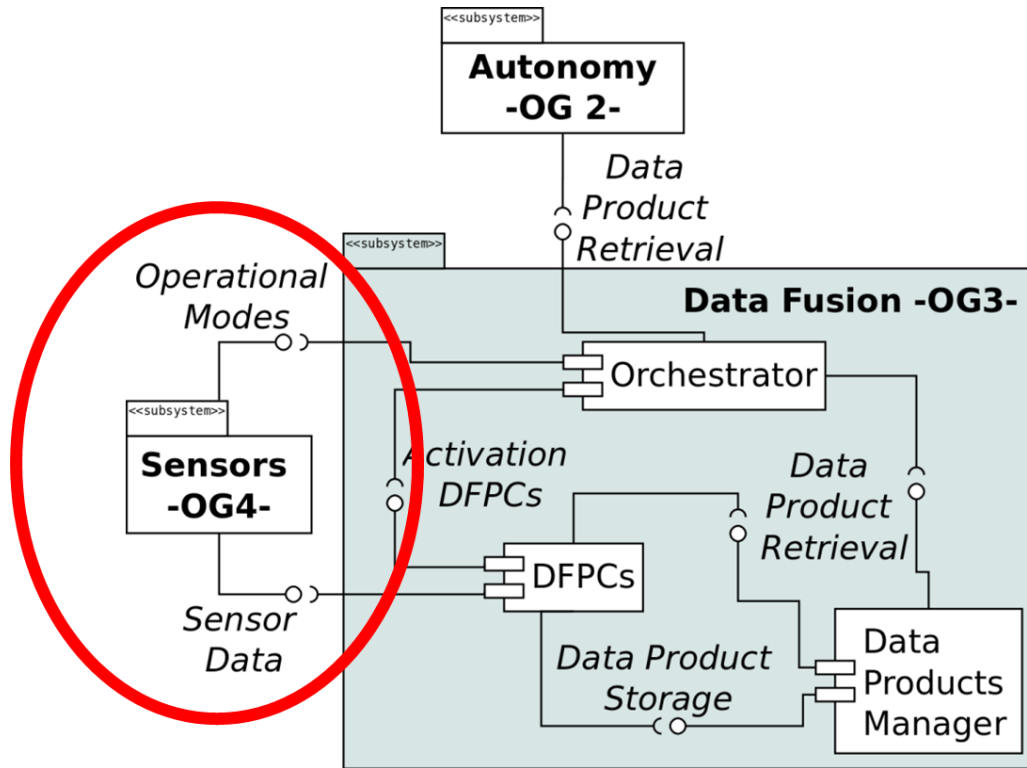


Figure 10: OG3-OG4 interfaces highlight

The interfaces concerned are essentially the ones allowing OG4 to transmit sensor data to OG3 as well as metadata information, and those allowing OG3 to transmit requests about expected operational mode of the sensors to OG4.

This interface is implemented partly at the Orchestrator level, and partly at the DFPC level – though an interface component allowing to receive OG4 sensors data and convey them to relevant DFPCs.

For the needs of the tests, a fake OG4 software (aka. M-OG4) will be purposely developed as a mean to provide sensors data samples (and associated meta-data information) with similar characteristics (data format, frequency, content...), as the data to be produced by OG4. This M-OG4 software shall also be able to receive (and verify the content of) operational mode requests issued by the Orchestrator.

The interface tests will be carried out relying on ESROCOS (or SARGON, tbc) – this will require proper wrapping of the Orchestrator and a selected number of DFPCs (among all the implemented DFPCs) external interfaces into ESROCOS.

Pass criteria: The test will be considered as successful if it is verified that (1) the operational mode requests sent by the Orchestrator are properly received, in the expected format, by M-OG4 (all possible operational mode variations shall be covered), and (2) relevant samples of all possible sensor data are properly received by the DFPCs interface, in the proper assumed format.

6 Conclusion

This document highlights the unit level testing that has to be carried out at the DFN and DFPC level, integrated testing planned between the CDFF internal components and finally testing method of the external interfaces of InFuse with other OGs. This will serve as a baseline during the implementation phase of InFuse, between the CDR and TRR milestones of the project.

D4.3: CDFF Unitary and Integrated Test Plans

Appendix 1: Mock DFPC test report template

DFPC test reference:		Date:	
DFPC name:			
Responsible partner:		Other involved partners:	
DFNs involved:	DFN ref:	DFN owner:	DFN version:
DFN 1			
DFN 2			
...			
DFN n			
Test protocol description:			
Name of person(s) performing the test:			
Test setup (hardware and software):			
Data used:	Data type:	Origin:	Record location:
Data 1			
Data 2			
...			
Data n			
Test results:			
Comments:			

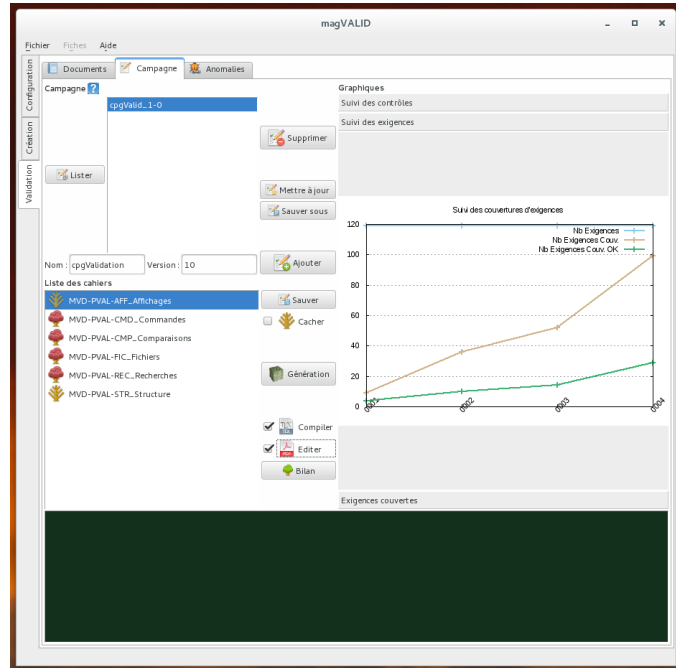
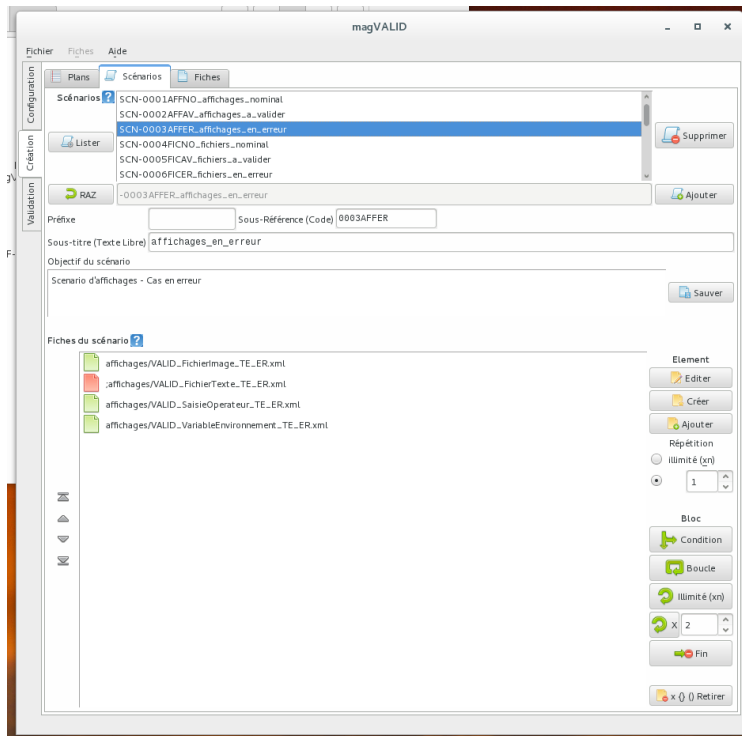
D4.3: CDFF Unitary and Integrated Test Plans

Appendix 2: Illustration of Magellium MAGVALID tools

Sample report images from the automated unit testing tool MAGVALID for a sample DFPC.

6.1.6 VALID_Compare_TE_NO

Typologie : Nominal		Compare	
VALID_Compare_TE_NO			
Méthode : Test		Nature : Fonctionnel	
Objectif : Tests de la fonction magVALID			
Couverture des exigences : NO_Comp			
Actions		▼ Bilan	
act1 - Description du test correct			
Exécuter la commande			
diff /home/philippe/Developpements/MagVALID/Demonstration/demoDocuments/-DonneesProdRef/Donnee1/S11200.bmp /home/philippe/Developpements/MagVALID/-Demonstration/demoDocuments/DonneesProd/VALID_Compare_TE_NO/test2.bmp			
▶ Contrôler la valeur de retour.		Valeur retournée	▼
Valeur attendue : 0		2	NOK
Rapporter la sortie d'exécution.			
diff: /home/philippe/Developpements/MagVALID/Demonstration/demoDocuments/-DonneesProdRef/Donnee1/S11200.bmp: Aucun fichier ou dossier de ce type			
Contrôle			
Bilan	<input type="checkbox"/> Ok	<input type="checkbox"/> A Valider	<input checked="" type="checkbox"/> Non Ok
Date		23/06/2017	<input type="checkbox"/> Non Testé
Suivi			
Références des Faits Techniques ouverts		Réserves éventuelles	
act1 REF_FT-000011			
Références des Faits Techniques re-ouverts —sans objet—		Réserves éventuelles	
Références des Faits Techniques à clore —sans objet—		Réserves éventuelles	

The screenshot shows the magVALID interface for configuring a scenario. It includes fields for 'Préfixe', 'Sous-Référence (Code)', 'Sous-titre (Texte Libre)', 'Objectif du scénario', and 'Scénario d'affichages'. A list of files for the scenario is shown, including 'affichage/VALID_FichierImage_TE_ER.xml', 'affichage/VALID_FichierTexte_TE_ER.xml', 'affichage/VALID_SaisieOperateur_TE_ER.xml', and 'affichage/VALID_VarianteEnvironnement_TE_ER.xml'. The interface also features buttons for 'Supprimer', 'Ajouter', 'Sauver', and 'Retirer'.