PUS C Library for TEC-SW Lab CCN

TEC-ED & TEC-SW Final Presentation Days

12/5/2020

PUS C Library for TEC-SW Lab CCN: Overview

- **Budget: 150k€**
- **Duration: 8 months**
- **Consortium:**
  - Prime: Intelligentia (Italy)
  - Subcontractor: D-Orbit (Portugal)
- **Technical officer:**
  - Marek Prochazka (heavily supported by Piotr Skrzypek)

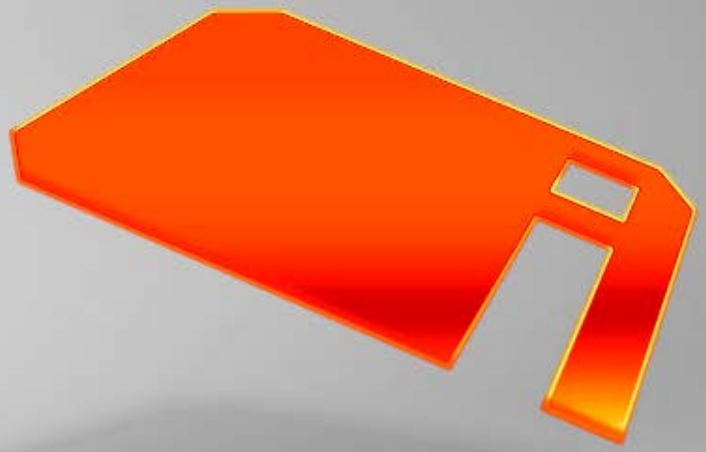
PUS C Library for TEC-SW Lab CCN: Objectives

1. Consolidate the Service 1 implementation.

2. Update the SVT test scripts and corresponding documentation in order to reflect the consolidation of Service 1 implementation.

3. Implement LibPUS C and ATB improvements
    - Implement a file system
    - Update Service 23
    - Update Service 15 to allow for file-based operations
    - Integrate MicroPython based OBCP engine (CFI) and link with Service 18

4. Verify updated PUS library by demonstrating that the upgraded EagleEye CSW with the new PUS library meet the EagleEye mission and performance requirements, i.e. AOCS and other partitions are functional and performance is compliant with previous versions.



PUS C Library for TEC-SW Lab CCN1

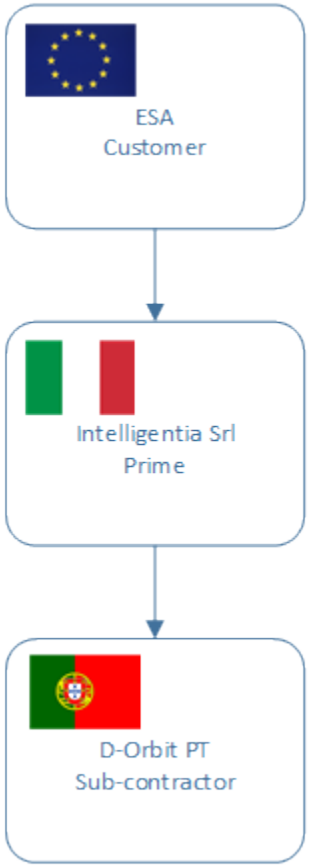
> Final Presentation

ESTEC, Noordwijk, The Netherlands, 12-13 May 2020

Agenda

- Industrial Team
  - Intelligentia Srl
  - D-Orbit PT
- Project Management
  - Key Personnel
  - WBS
  - WPD
  - Work Logic
- Engineering
  - Consolidation of LibPUS-C
  - Improvement of EagleEye CSW
- Integration into EagleEye CSW
- Validation & System Tests
- Conclusion:
  - Lesson Learnt
  - Suggestions for Future developments

Industrial Team

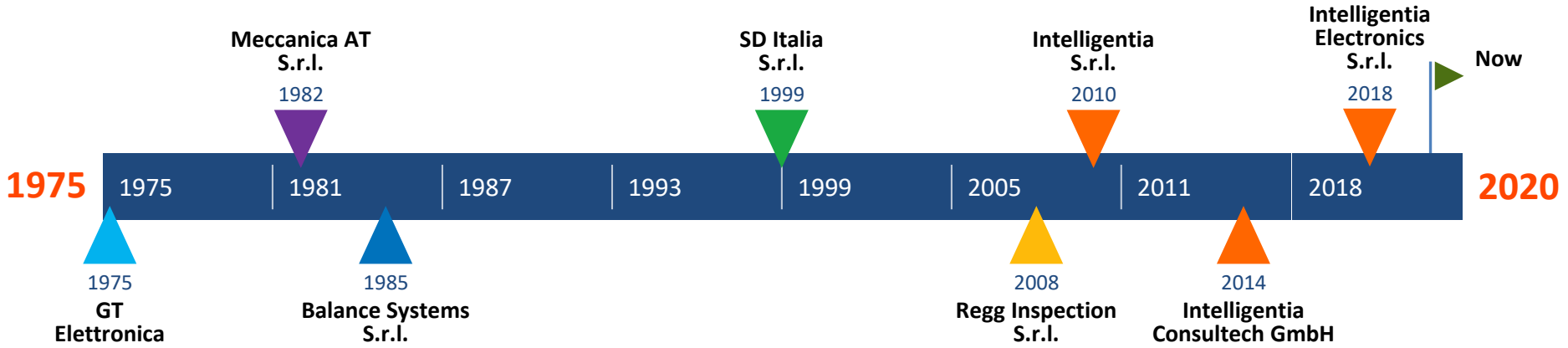


Intelligentia S.r.l.

- Intelligentia S.r.l. was established in 2010 and belongs to Balance Systems group.
- Two main business areas: industry & space projects.
- R&D: new opportunities in the market, interaction with universities, and project synergies.
- Intelligentia S.r.l. is currently listed in the registry of the Italian Chamber of Commerce as Innovative SME.
- Intelligentia has 21 employees in 2 countries
- 1M€ turnover
- 75% turnover from export

Intelligentia S.r.l.
Our Group

- Balance Systems group operates in the manufacturing sector since 1975 and is leader in balancing machines and auxiliary systems for industrial processes.
- Passion and professionality based on a solid history and expertise, with a focus into future opportunities.



Intelligentia S.r.l.
Our Group

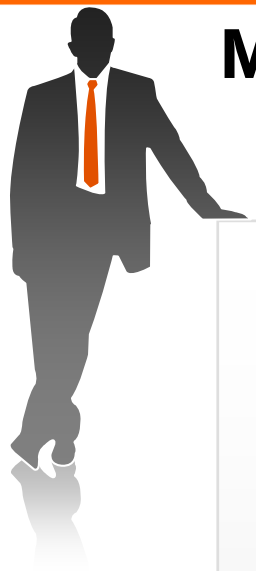
Group in numbers:

- 180 employees in 14 Countries/4 Continents
- 90% turnover from export
- 1000+ customers around the world
- 20+ product lines on the Market
- 100+ patents
- 32M€ turnover of the Group

Group competences:

- Production of mechanical components and electronics
- Design, integration and setup of industrial plants and machineries
- SW & System Engineering
- Development of HW/FW & control systems

Intelligentia S.r.l.



Main projects

MTG Satellite

- SW Unit Test of the Satellite OBSW using the MTG toolchain.
- SW TS Validation of the Satellite OBSW using the MTG toolchain.
- Development of SCCDB SW
- SRDB Data Population
- Platform Functional Chain Validation activities.

Ital-GovSatCom

- Participation into the ItaGovSatCom Industrial Consortium, led by TAS-I:
- Tailoring of the ECSS-E-ST-70-41C to fulfill Ital-GovSatCom Satellite Spec.
- Design, development and testing of the PUS-C library
- Integration of the PUS-C Library into ItaGovSatCom On-board SW
- Complete project documentation following the ECSS standard.

Intelligentia S.r.l.

R&D

R&D

- Preparation of proposals both as Prime and Subcontractor (e.g. ESA Expro+ ITT, ASI ITT).
  - "ANNOUNCEMENT OF OPPORTUNITY FOR TECHNOLOGY TRANSFER PROOF OF CONCEPTS " ITT - Application of ECSS-E-ST-70-41C to Industry 4.0
- Development of tools for automating SW testing activities.
- Application of Industry 4.0 concepts in Space (e.g., Satellite Long Term Storage, Big data).
- Application of AI techniques in Space (FDIR, S/C autonomy, flight data analysis).
- HW embedding solutions & control system design, e.g. sensors, motor driver design and implementation, power electronics.
- Collaboration with universities and research centres (e.g., master thesis).

Intelligentia S.r.l.
Where you can find us

Intelligentia S.r.l.

Via Del Pomerio 7, 82100 Benevento (Italy)
Phone: +39 0824 177 4728
Fax: +39 0824 1811080
Web & Mail: www.intelligentia.eu, info@intelligentia.eu

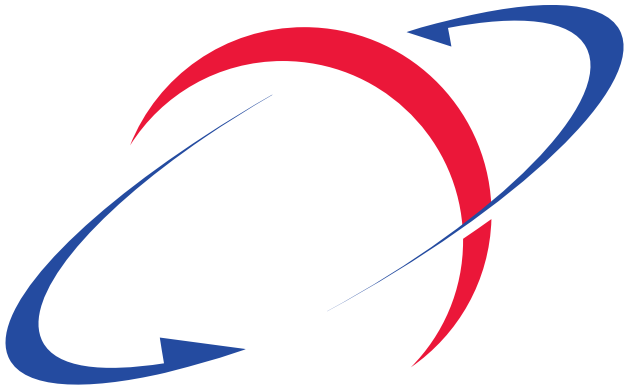
Via Ticino 30/G, 20064 Gorgonzola, Milan (Italy)
Phone: +39 02 4795 2357
Fax: +39 02 4795 1170

Intelligentia Consultech GmbH

Albert-Einstein-Straβe 2, 70806 Kornwestheim (Germany)
Phone: +49 (0) 152 15529228
Mail: info@intelligentia.eu

D-Orbit PT
Company Presentation

IN-ORBIT
TRANSPORTATION
AND DEPLOYMENT
FOR SMALL SATELLITES
AND NEW SPACE
CONSTELLATIONS
COMMISSIONING
DECOMMISSIONING
AND LIFE EXTENSIONS
OF SATELLITES AND
LAUNCHER STAGES

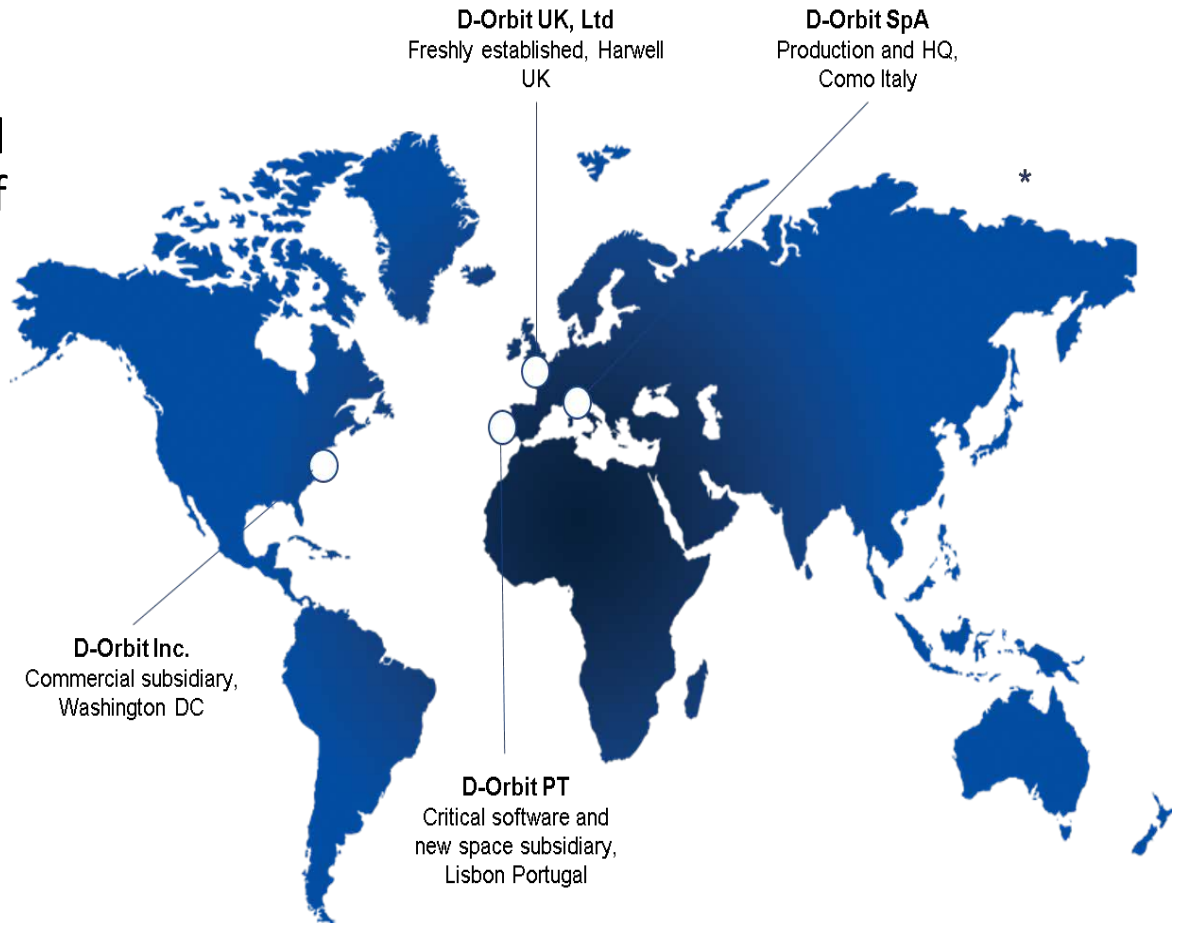


D-Orbit PT
Company Presentation

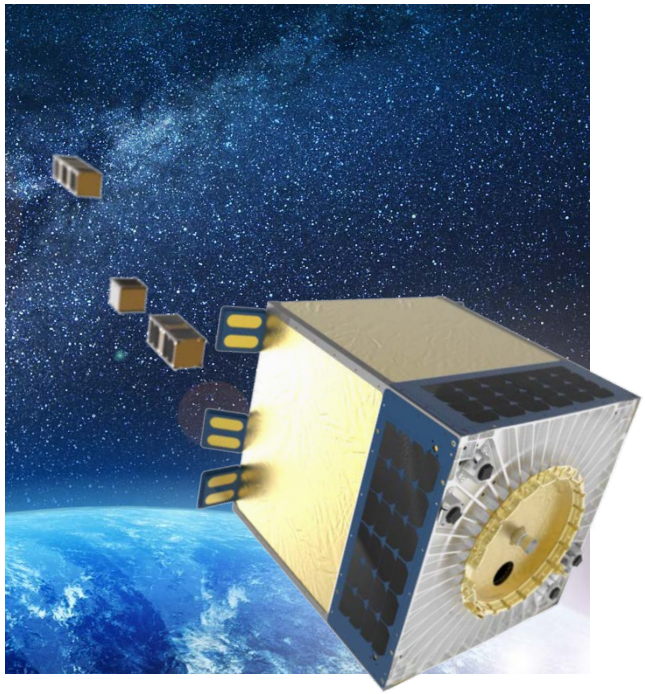
- D-Orbit PT was established in Lisbon in 2014 as part of D-Orbit group
- The Portuguese team is the responsible for the software development for the whole group

Bruno Carvalho
COO D-Orbit PT

Critical SPHEREA ilex space SCISYS



ION Launch Service



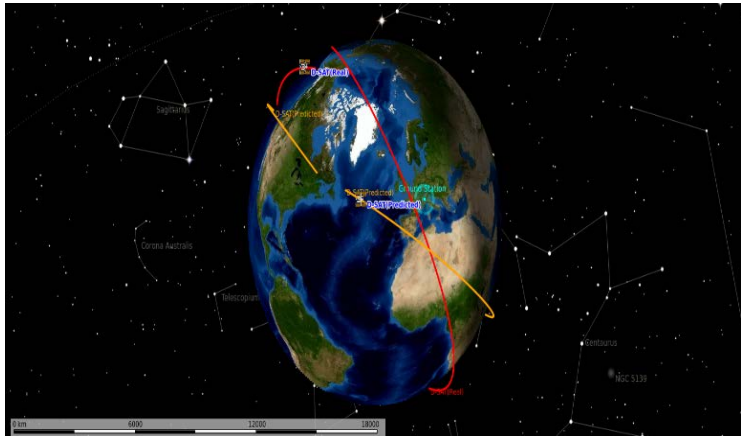
- An **innovative launch service designed to transport satellites to space and release them into precise, independent orbital slots**, enabling to start their space mission quickly and in optimal operational conditions
- ION Satellite Carrier is equipped to launch **a combination of CubeSats of any form factor,** up to 64 CubeSat units in total, or microsatellites.
- **Extra services** include mission analysis and design, platform engineering, software development, acceptance testing, and transportation.

D-Orbit PT
Major Mission/Products

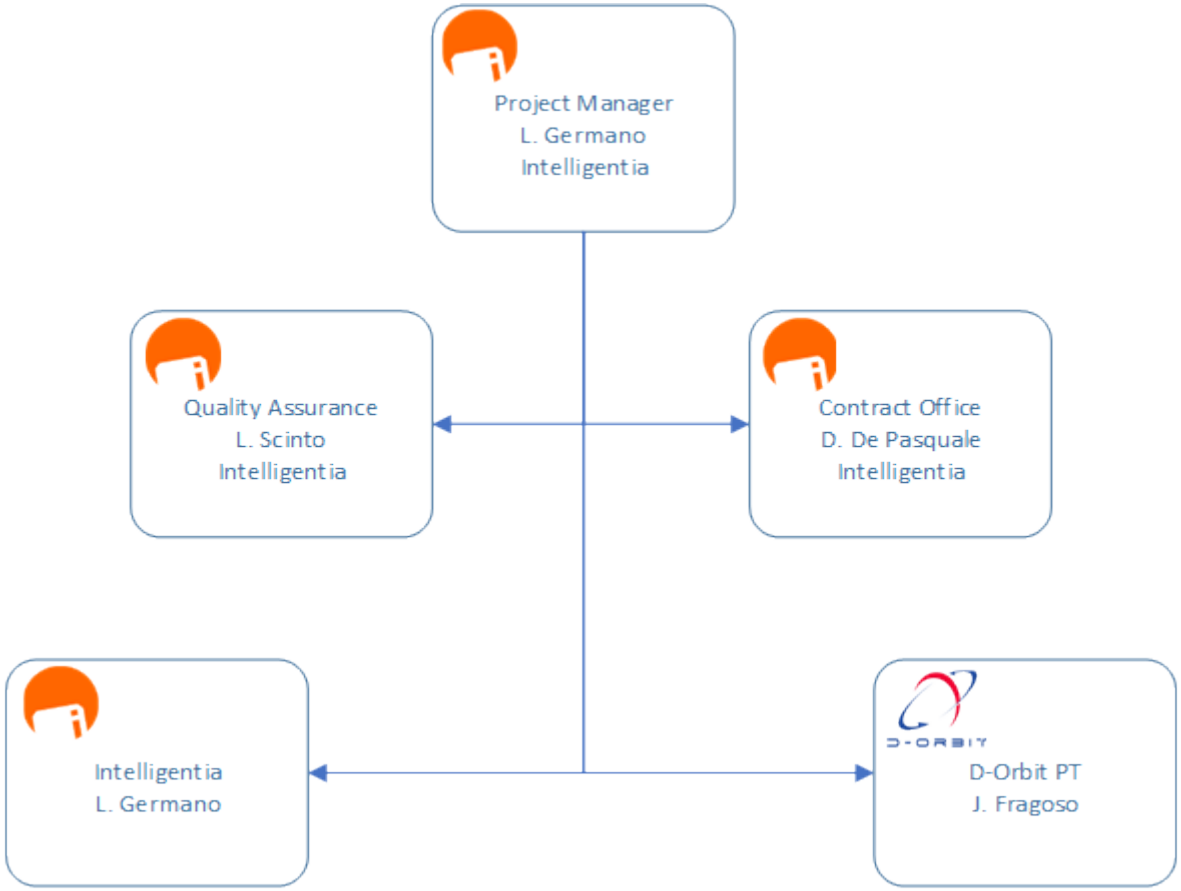
- Aurora is a powerful **cloud-based mission control software** suite designed to control a single satellite or a complete constellation through a user-friendly, fully customizable control interface.

AURORA® Satellite Dashboard Control

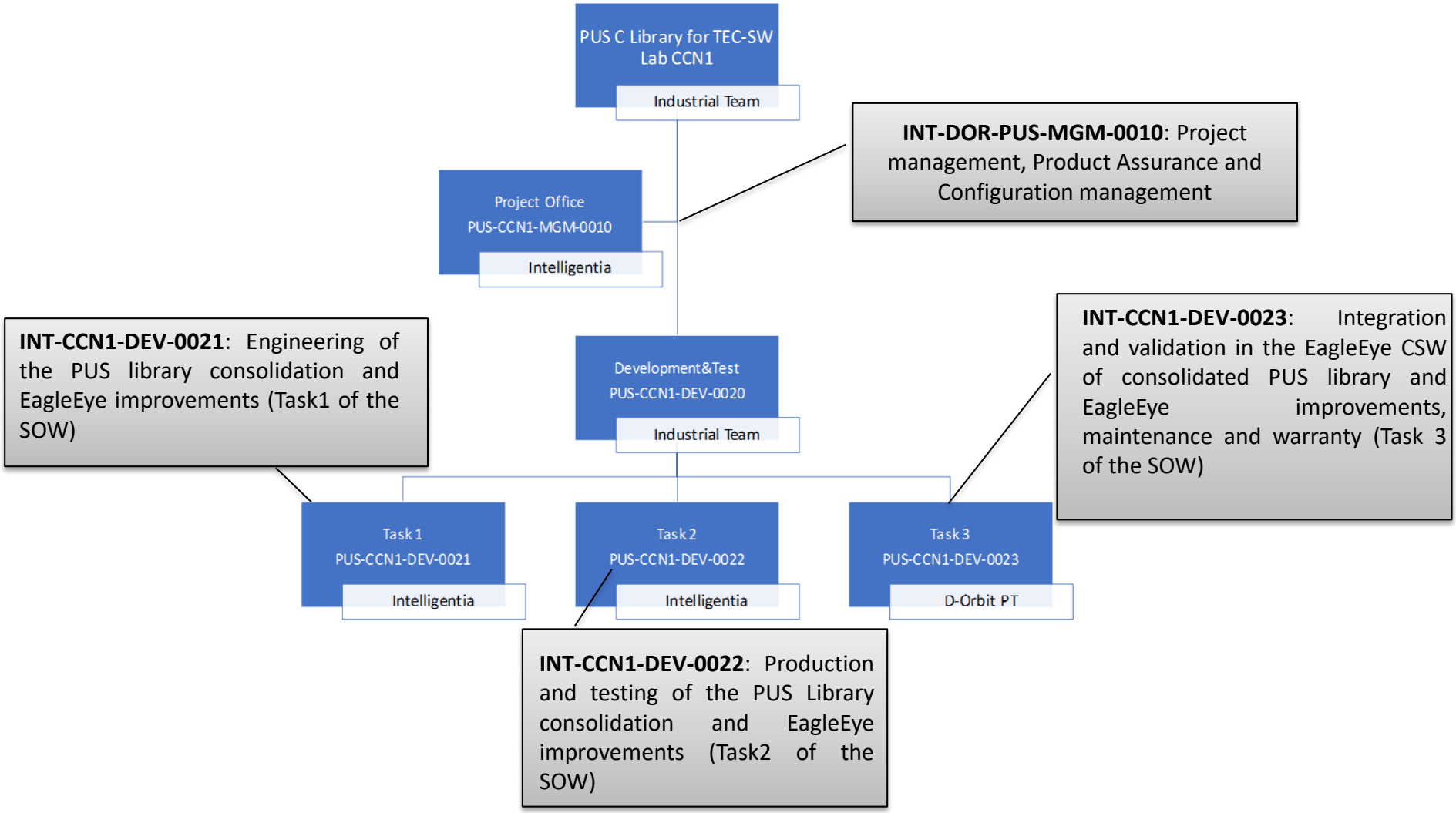
- Graphical representation of orbital position and attitude, with updates on upcoming passages, grants full **control of operations, sub-systems status, and energy and power budget**, and store all command, telemetry, and photographic data for diagnostics.



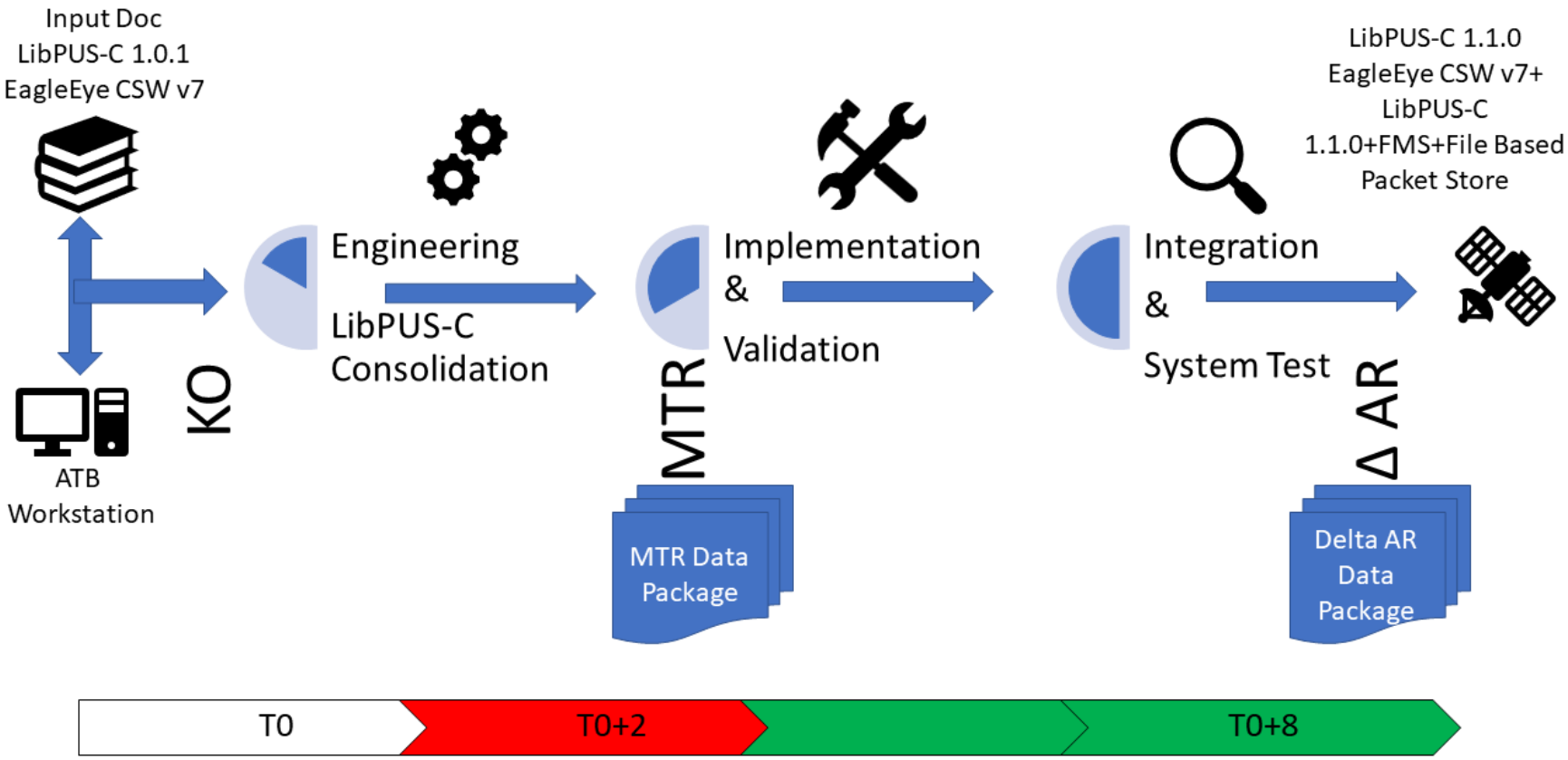
Project Management
Key personnel



Project Management
WBS/WPD



Project Management
Work Logic



Engineering – LibPUS-C Consolidation

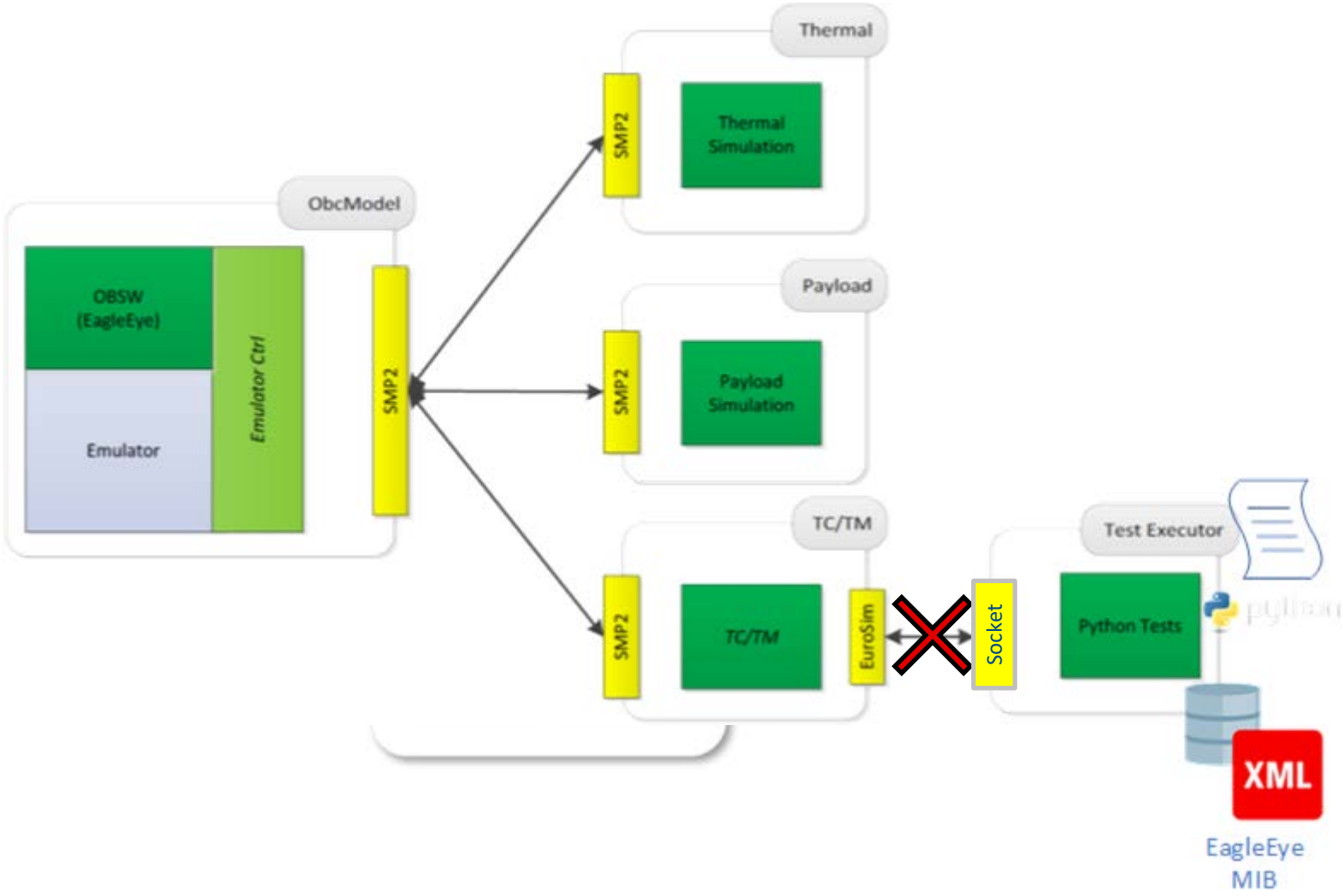
Starting point has been the LibPUS-C 1.0.1 from AR of project phase 1

Following the recommendation from ESA has been implemented the handling of the TM(1,7) and TM(1,8) as final stage of TC verification for all the TCs implemented in the LibPUS-C.

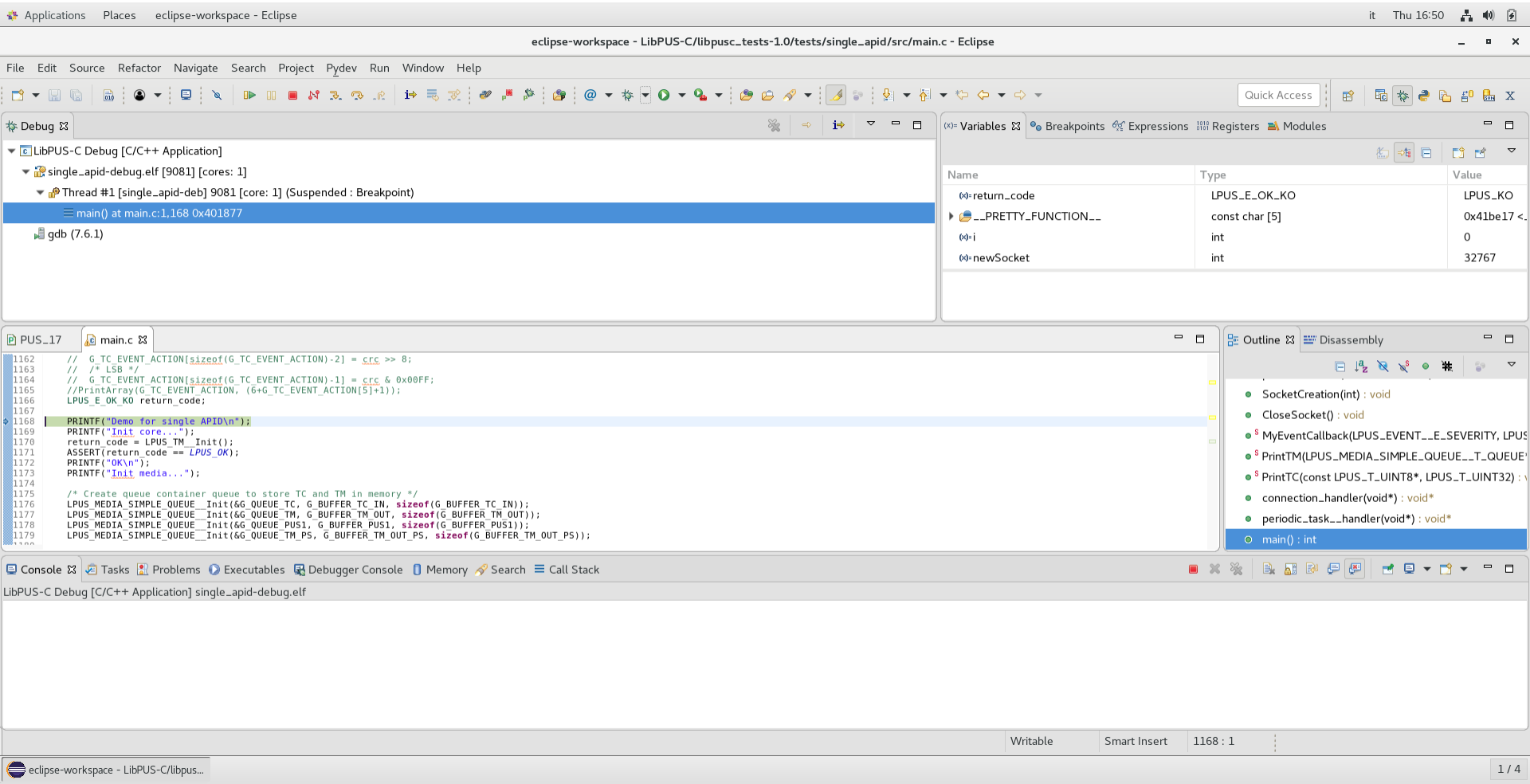
Before integrating the consolidated LibPUS-C into EagleEye CSW v7 all the standalone validation and unit test developed in the previous phase of the project have been re-executed using the test environment created in the previous project phase.

At the end the Industrial Team released the the LibPUS-C 1.0.2

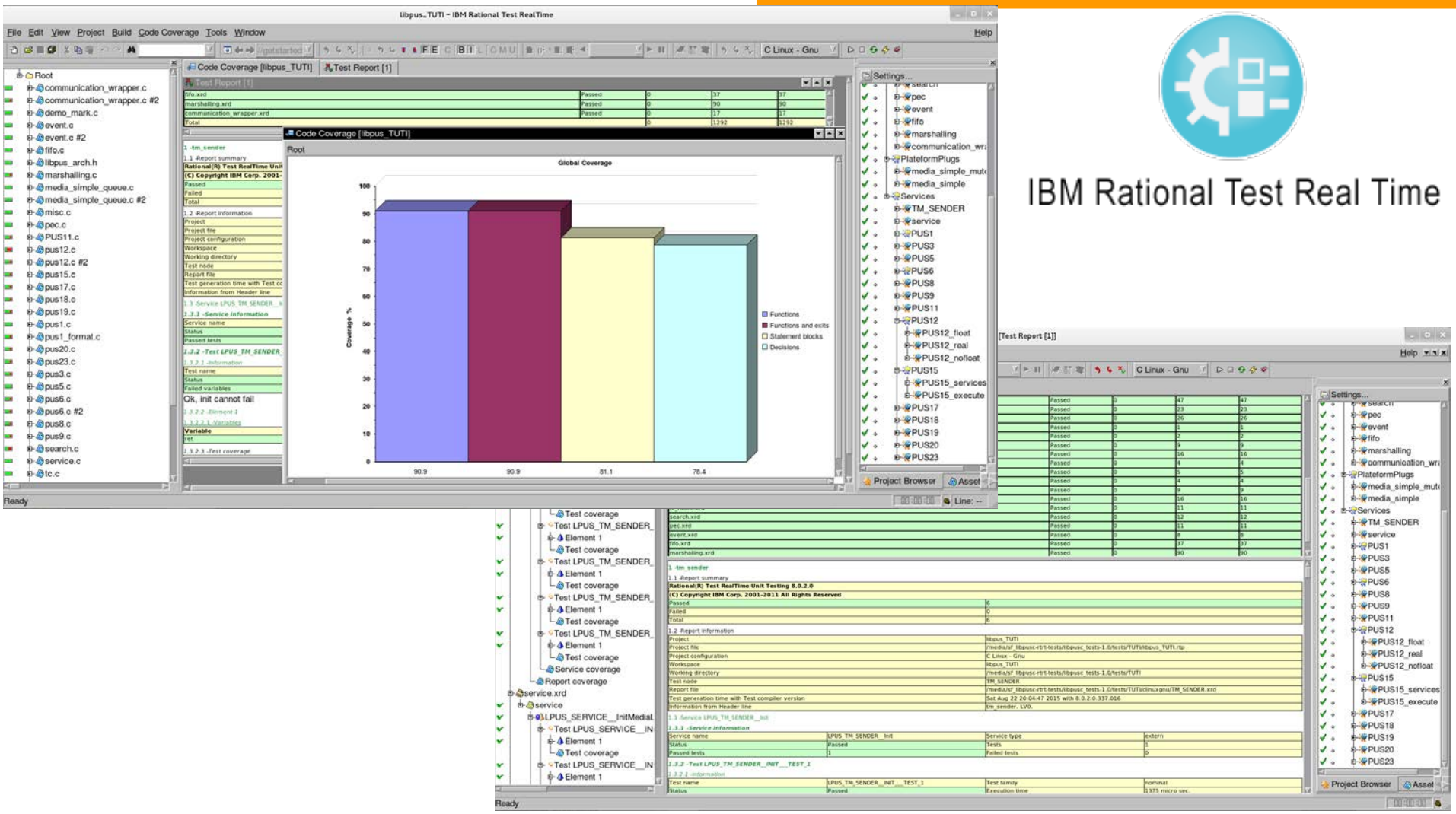
Production & Test
LibPUS-C Test Environment 1/2



Production & Test
LibPUS-C Development and Test Environment

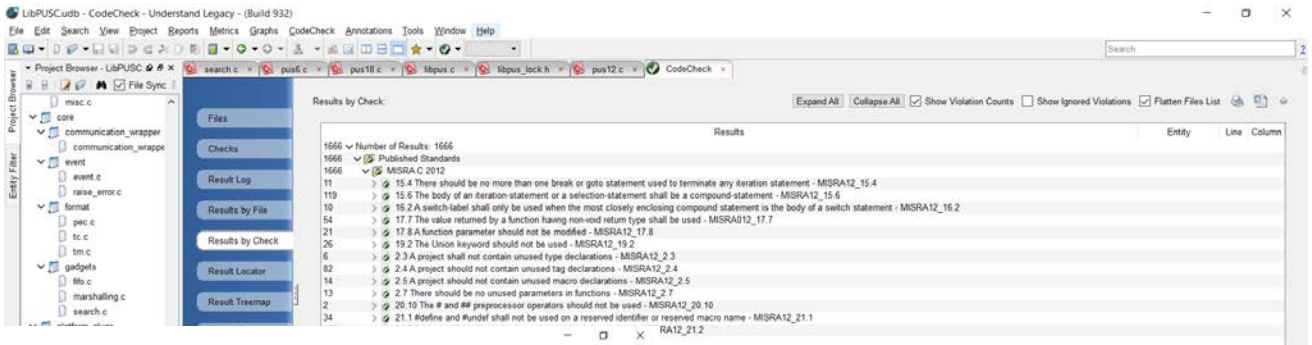


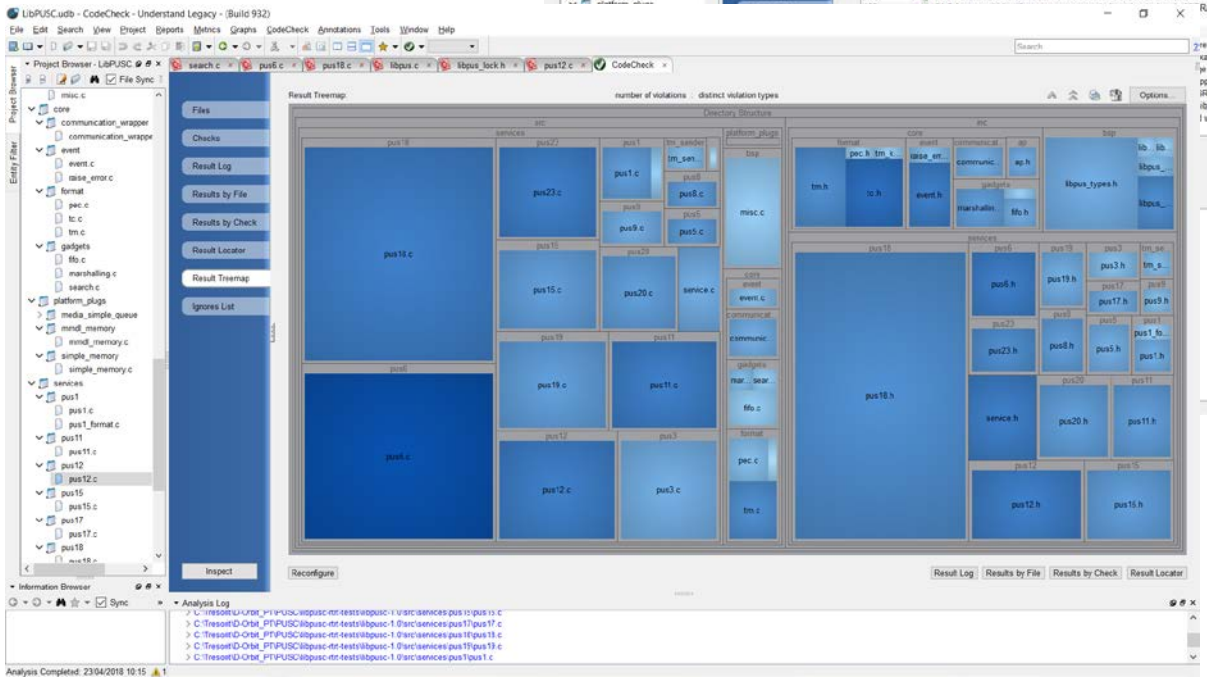
Production & Test
LibPUS-C Unit Test



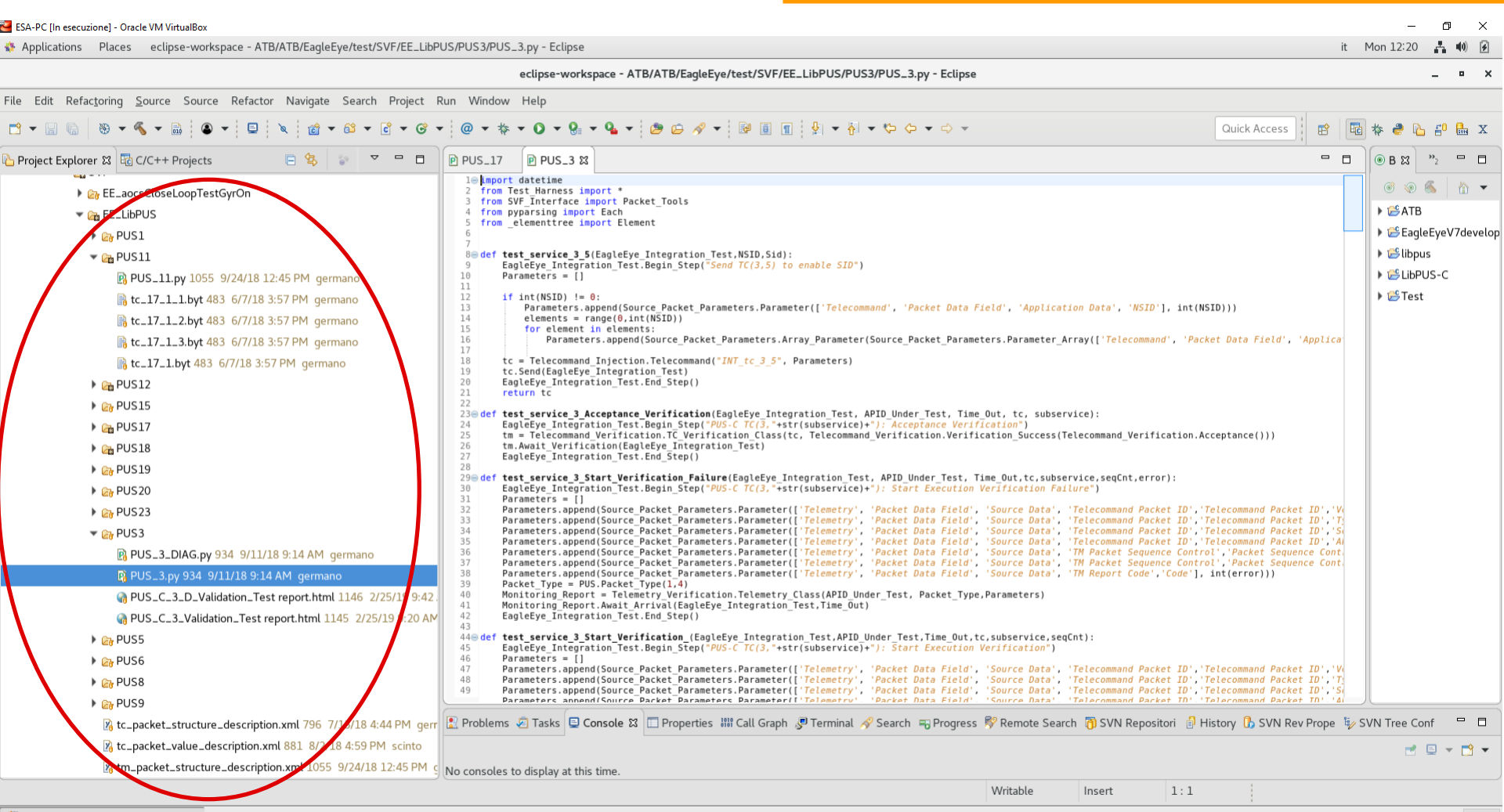
Production & Test
LibPUS-C Static Analysis

scitools Understand





Production & Test
LibPUS-C Validation Test 1/2



Production & Test
LibPUS-C Validation Test 2/2

PUS-C EagleEye

Test Case 1.7: PUS-C TC(3,25) - Wait Periodic HK Report generation for SID 2

Test Steps

ID	Description	Result
1.7.1	**Wait Housekeeping report (3,25)**	
1.7.1.1	TM(Mission Manager , (3,25)) -> [['Telemetry', 'Packet Data Field', 'Source Data', 'SID'] = 2]	OK

Test Case 1.8: PUS-C TC(3,6) - Disable Periodic HK Report generation SID 2

Test Steps

ID	Description	Result
1.8.1	**Send TC(3,6) to disable SID**	
1.8.1.1	TC(INT_tc_3_6)-> [['Telecommand', 'Packet Data Field', 'Application Data', 'NSID'] = 1, ['Telecommand', 'Packet Data Field', 'Application Data', 'SID'][0].[] = 2]	OK
1.8.2	**PUS-C TC(3,6): Acceptance Verification**	
1.8.2.1	Verification(TC(INT_tc_3_6)-> [['Telecommand', 'Packet Data Field', 'Application Data', 'NSID'] = 1, ['Telecommand', 'Packet Data Field', 'Application Data', 'SID'][0].[] = 2]): Ack(Acceptance) = Success	OK
1.8.3	**PUS-C TC(3,6): Start Execution Verification**	
1.8.3.1	TM(Mission Manager , (1,3)) -> [['Telemetry', 'Packet Data Field', 'Source Data', 'Telecommand Packet ID', 'Telecommand Packet ID', 'Version Number'] = 0, ['Telemetry', 'Packet Data Field', 'Source Data', 'Telecommand Packet ID', 'Telecommand Packet ID', 'Type'] = 1, ['Telemetry', 'Packet Data Field', 'Source Data', 'Telecommand Packet ID', 'Telecommand Packet ID', 'Secondary Header Flag'] = 1, ['Telemetry', 'Packet Data Field', 'Source Data', 'Telecommand Packet ID', 'Telecommand Packet ID', 'APID'] = 2, ['Telemetry', 'Packet Data Field', 'Source Data', 'TM Packet Sequence Control', 'Packet Sequence Control', 'Sequence Flag'] = 3, ['Telemetry', 'Packet Data Field', 'Source Data', 'TM Packet Sequence Control', 'Packet Sequence Control', 'Packet Sequence Count'] = 3]	OK

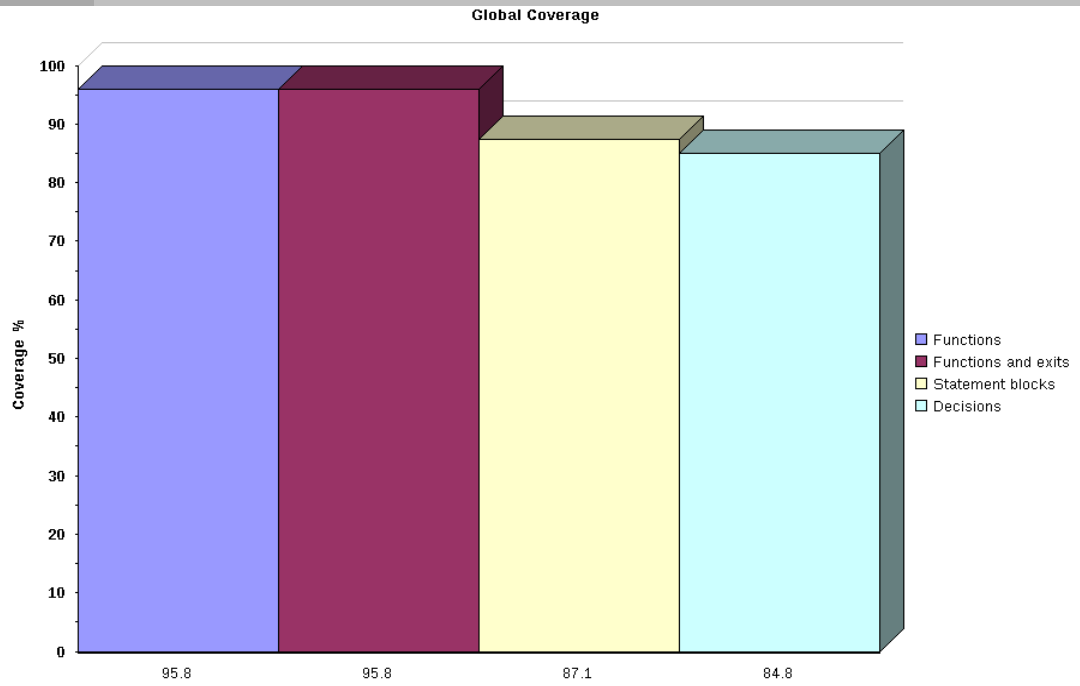
Test Case 1.9: PUS-C TC(3,5) - Enable Periodic HK Report generation SID 1 and 2

Test Steps

ID	Description	Result
1.9.1	**Send TC(3,5) to enable SID**	
1.9.1.1	TC(INT_tc_3_5)-> [['Telecommand', 'Packet Data Field', 'Application Data', 'NSID'] = 2, ['Telecommand', 'Packet Data Field', 'Application Data', 'SID'][0].[] = 1, ['Telecommand', 'Packet Data Field', 'Application Data', 'SID'][1].[] = 2]	OK
1.9.2	**PUS-C TC(3,5): Acceptance Verification**	
1.9.2.1	Verification(TC(INT_tc_3_5)-> [['Telecommand', 'Packet Data Field', 'Application Data', 'NSID'] = 2, ['Telecommand', 'Packet Data Field', 'Application Data', 'SID'][0].[] = 1, ['Telecommand', 'Packet Data Field', 'Application Data', 'SID'][1].[] = 2]): Ack(Acceptance) = Success	OK
1.9.3	**PUS-C TC(3,5): Start Execution Verification**	
1.9.3.1	TM(Mission Manager , (1,3)) -> [['Telemetry', 'Packet Data Field', 'Source Data', 'Telecommand Packet ID', 'Telecommand Packet ID', 'Version Number'] = 0, ['Telemetry', 'Packet Data Field', 'Source Data', 'Telecommand Packet ID', 'Telecommand Packet ID', 'Type'] = 1, ['Telemetry', 'Packet Data Field', 'Source Data', 'Telecommand Packet ID', 'Telecommand Packet ID', 'Secondary Header Flag'] = 1, ['Telemetry', 'Packet Data Field', 'Source Data', 'Telecommand Packet ID', 'Telecommand Packet ID', 'APID'] = 2, ['Telemetry', 'Packet Data Field', 'Source Data', 'TM Packet Sequence Control', 'Packet Sequence Control', 'Sequence Flag'] = 3, ['Telemetry', 'Packet Data Field', 'Source Data', 'TM Packet Sequence Control', 'Packet Sequence Control', 'Packet Sequence Count'] = 4]	OK
1.9.4	**PUS-C TC(3,5): Start Execution Verification**	
1.9.4.1	TM(Mission Manager , (1,3)) -> [['Telemetry', 'Packet Data Field', 'Source Data', 'Telecommand Packet ID', 'Telecommand Packet ID', 'Version Number'] = 0, ['Telemetry', 'Packet Data Field', 'Source Data', 'Telecommand Packet ID', 'Telecommand Packet ID', 'Type'] = 1, ['Telemetry', 'Packet Data Field', 'Source Data', 'Telecommand Packet ID', 'Telecommand Packet ID', 'Secondary Header Flag'] = 1, ['Telemetry', 'Packet Data Field', 'Source Data', 'Telecommand Packet ID', 'Telecommand Packet ID', 'APID'] = 2, ['Telemetry', 'Packet Data Field', 'Source Data', 'TM Packet Sequence Control', 'Packet Sequence Control', 'Sequence Flag'] = 3, ['Telemetry', 'Packet Data Field', 'Source Data', 'TM Packet Sequence Control', 'Packet Sequence Control', 'Packet Sequence Count'] = 4]	OK

Production & Test
LibPUS-C Unit test and Validation Test Result

Test	Passed	Failed	Total	Status
Unit Test	1292	0	1292	Passed
Validation Test	389	0	389	Passed



- **Decision coverage** reached 84,8%.
- **MISRA** compliance - the source code was tested against 84 rules and 61 of them were satisfied, meaning a 73% rate for MISRA 2012 standard adherence

Engineering – EagleEye CSW Improvements
File Management System (FMS)

After a preliminary study phase the Industrial team decided to use as FMS in EagleEye the RTEMS in memory file system.
The choice of RTEMS in memory file system was because:

- DMS partition is the partition in which the FMS needs to be integrated;
- DMS is based on RTEMS;
- RTEMS in memory file system is available out of the box;
- RTEMS in memory file system can be used simply configuring it by using the RTEMS configuration define.

Once enabled in RTEMS, at the start-up it mounts a RAM based file system known as base file system. The root directory of this file system tree serves as the logical root of the directory hierarchy.
A RAM based file system draws its management resources from memory. File and directory nodes are simply allocated blocks of memory. Data associated with regular files is stored in collections of memory blocks. When the system is turned off or restarted all memory-based components of the file system are lost

Engineering – EagleEye CSW Improvements
PUS Service 23

FMS has been delegated to RTEMS so only the functionalities available in the RTEMS base file system have been used in PUS 23, and, because of these limitations, only the following set of sub-services are supported:

- TC[23,1] create a file
- TC[23,2] delete a file
- TC[23,3] report file attributes
- TC[23,9] create a directory
- TC[23,10] delete a directory
- TC[23,12] directory content report
- TC[23,14] copy a file

As RTEMS base file system does not manage the file lock/unlock functionality, the following sub-services will not be supported:

- TC[23,5] lock a file
- TC[23,6] unlock a file

Engineering – EagleEye CSW Improvements
File Based Packet Store 1/6

A File Based Packet Store has been designed and developed as a library component, to allow its integration in the EagleEye software.

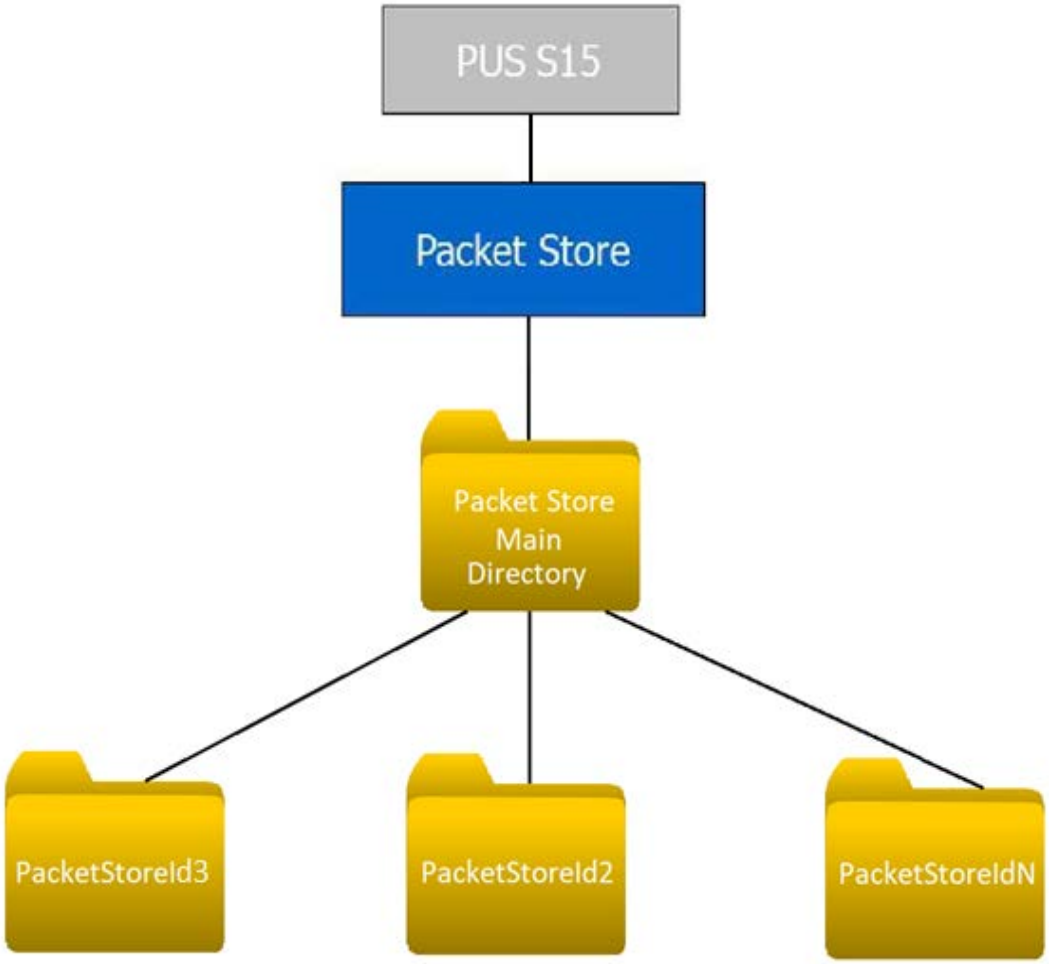
The library uses the underlaying RTEMS in memory file system to create\delete directory and files. A wrapper layer in the Packet Store has been designed to make the library independent from the used file system

In the PS library, a directory is dedicated to each packet store ID.

There will be a main directory containing a directory for each defined PacketStoreID.

The user can configure the path in which create the main directory and its name.

Engineering – EagleEye CSW Improvements
File Based Packet Store 2/6



Engineering – EagleEye CSW Improvements
File Based Packet Store 3/6

It is possible to set each packet store in two different ways:

- **BOUNDED**: upon reaching the maximum configured size, a packet store full error will be raised;

- **CIRCULAR**: upon reaching the maximum configured size, the oldest file related to the PacketStoreID will be delated and the current telemetry will be stored in a new file.

Each TM packets is stored in file up to the maximum file size configured by the user for each packet store. When the maximum file size is reached a new file is created. Each file is named in the following way in order to guarantee a temporal order. The name of each new created file is the data and time of creation (YYYY-MM-DDTHH_mm_ss), for example:

2019-11-06T15_08_50

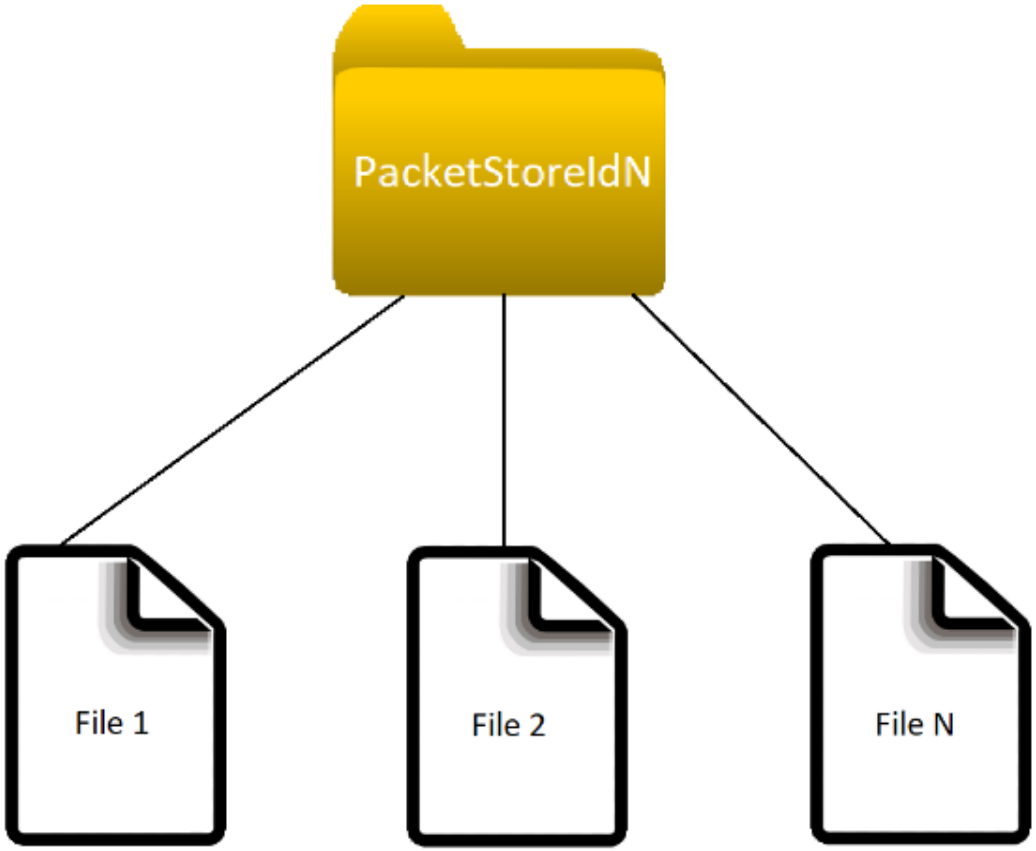
Engineering – EagleEye CSW Improvements
File Based Packet Store 4/6

The structure of a Packet Store file is as indicated below. For each TM packets the following information are stored:

- Storage timestamp in CUC format - 7 bytes;
- Open retrieval status (0 – TM retrieved, 1 – TM not retrieved) - 1 byte;
- TM size – 4 bytes;
- TM Packets - TM size bytes.

The open retrieval status indicates if the telemetry has been sent to ground in open retrieval mode. This field has been added for future purpose, considering that the current version of LibPUS-C service 15 service does not implement the open retrieval sub-service.

Engineering – EagleEye CSW Improvements
File Based Packet Store 5/6



Engineering – EagleEye CSW Improvements
File Based Packet Store 6/6

To match the mission needs and to guarantee the communication with the PUS15, the Packet Stores Library has been integrated in to EagleEye CSW DMS partition.

Into EagleEye CSW two Packet_Store IDs have been configured:

- Packet Store ID related to DMS TMs partition:
  - packet_store_id = “packet_store__DMS”
  - store_type = BOUNDED
  - packet_store_size_mb = 1

- Packet Store ID related to PAYLOAD TMs partition:
  - packet_store_id = “packet_store_PYLD”
  - store_type = BOUNDED
  - packet_store_size_mb = 1

Engineering – EagleEye CSW Improvements
File Upload/Download 1/4

To match the mission needs and to guarantee the upload/download of files, private sub-services 128, 129, 130 have been added to service 23 together with standard sub-service 12 to collect the information about files and directories present in the On-Board file system. These new subservice lead to a new major release of the library named

LibPUS-C 1.1.0

The concept of the **upload** service is to split the file in several segments and send a TC for each of them. The first TC[23,130] shall have a type field equal to 0 and shall contain the path of the on-board destination folder and name the file will have on-board.

tc type	**file path**	
	repository path	**file name**
unsigned integer (value = 0)	variable character string	variable character string

Engineering – EagleEye CSW Improvements File Upload/Download 2/4

The others TC[23,130], with type field equal to 1, shall contain the number of the current file segment is being uploaded, the total number of segments the file has been split into, and the file data segment, whose maximum size depends on the size of the TC Application Data field. According to the EagleEye configuration, the maximum file segment size shall be 993 bytes

tc type	segment number (n)	number of segments (N)	file data (n)segment
unsigned integer (value = 1)	unsigned integer	unsigned integer	variable character string

It is worth pointing out that is up to the ground to split the file in the correct number of segments and with the correct size. A TC(23,12) can be used to check the successful result of the upload process

Engineering – EagleEye CSW Improvements
File Upload/Download 3/4

The file download service allows to download 5 files simultaneously from 5 different on-board source folders. Each download process is identified with an Operation ID, which represents also the priority of the corresponding RTEMS task which implements the process.
This ID, whose value shall be included in the range [1,5], is the first application data field of the TC[23,128].

operation id	file path	
	repository path	file name
unsigned integer	variable character string	variable character string

It is not possible to start two or more downlink processes with the same operation id. The other TC fields contain the path of the source folder on-board and the name of the file to download.

Engineering – EagleEye CSW Improvements
File Upload/Download 4/4

If the requested file exists in the source on board directory, then PUS service 23 starts to generate a sequence of TM[23,129], each of them containing the operation id of the process, the number of current file segment, the total number of segments the file has been split into, and the file data segment.

operation id	segment number (n)	number of segments (N)	file data (n)segment
unsigned integer	unsigned integer	unsigned integer	variable octet string

Since a maximum of 5 downlink processes can be started simultaneously, the operation id contained in the TM is important to rebuild properly the files on ground, because it allows to understand to which file the data segment belongs. It is up to the Ground to implement the application that properly re-build the downloaded files, collecting the data segment coming from TM with the same operation id

Engineering – EagleEye CSW Improvements
OBCP Micropython Engine

The OBCP Micropython engine, CFI from the Agency , was expected to be ready to be integrated and used in XtratuM and RTEMS 4.11.

The Micropython engine was never ported on the EagleEye environment based on XtratuM and RTEMS 4.11. The EagleEye environment has to be considered as completely new.

The spirit of collaboration of the Industrial Team lead it to try with maximum effort possible to use the µpython engine onboard of EagleEye also if the porting activities was not included in the CCN1.

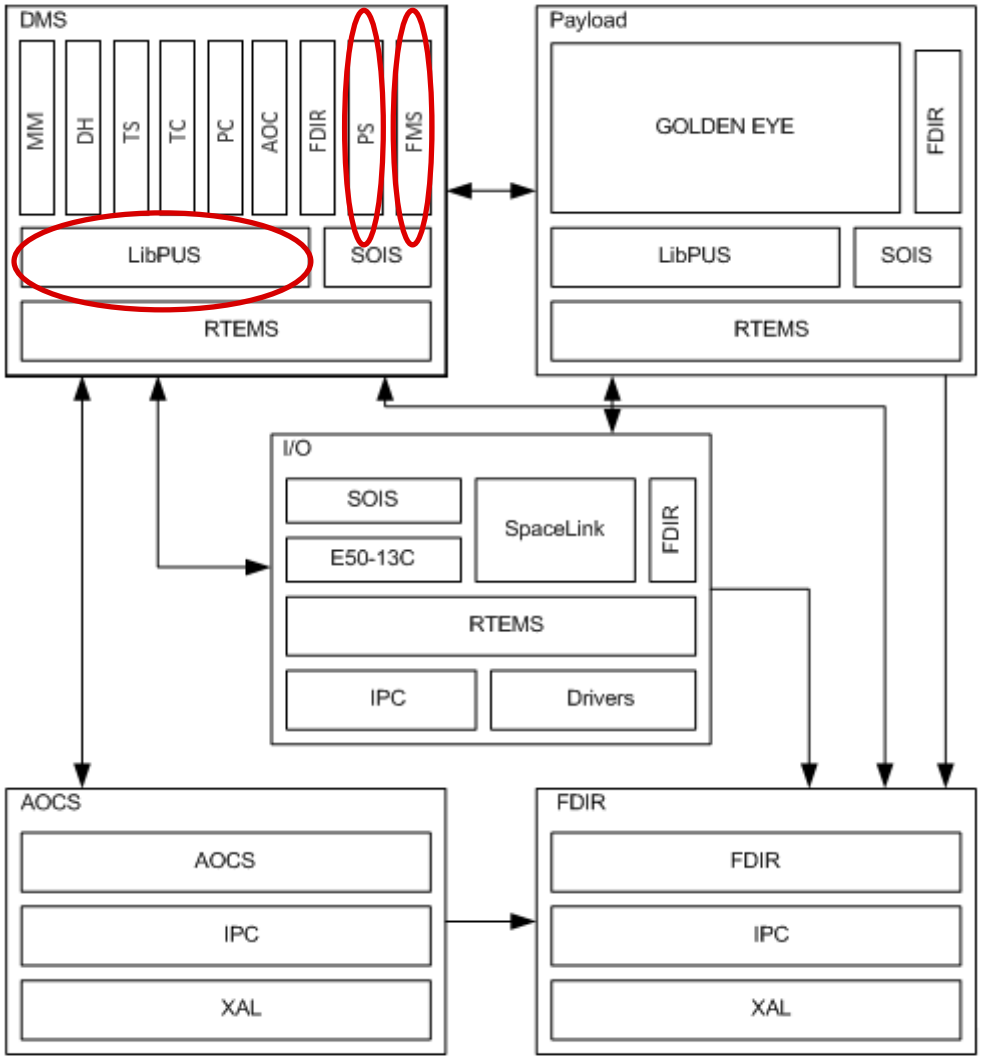
The Industrial team performed a deep investigation on it, spending all the time and budget foreseen for this activity, but unfortunately the Industrial team reached a point over that, without investing more time and support from ESA expert, was difficult to go.

Integration into EagleEye 1/3

All the features foreseen by the CCN1 have been then integrated into EagleEye CSW with the exception of the OBCP micropython.

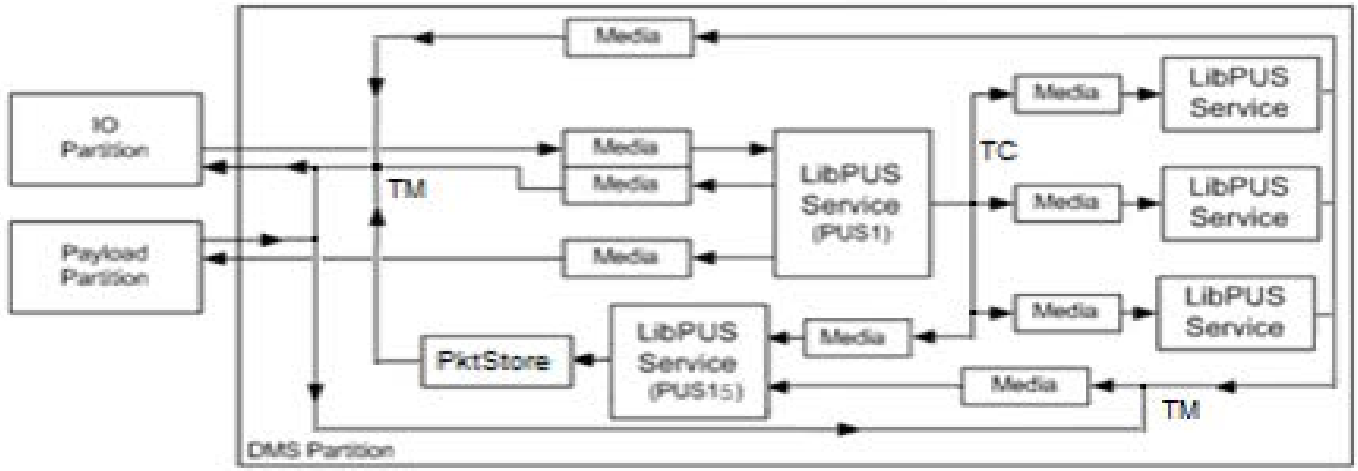
The integration concerned only the DMS partition and the modification performed to EagleEye CSW have been reported into the following figure

Integration into EagleEye 2/3



Integration into EagleEye 3/3

The following figure shows the DMS packet handling and introduce the PUS15 service and File Based Packet store



The PUS15 is configured to check all DMS TMs and store them, if enabled, into the packet store based on filesystem.
The DMS partition also includes a File Management System whose functionalities are accessed via PUS Service 23. It allows, in addition to manage files and directories, to uplink/downlink files to/from the spacecraft.

Validation & System Test 1/3

Once the integration of the new features has been ended the first activity done by the Industrial team has been the re-execution of all system tests to verify that the old features and performance of EagleEye CSW were still the same.

To test the new features a set of system tests have been added to EagleEye. They cover all the SoW requirements and verify the correctness of the implementation/integration into EagleEye CSW.

For each new feature a dedicated test procedure has been designed.

Of course, as agreed, the system tests have been re-executed on both SPARC and ARM architecture

Validation & System Test 2/3

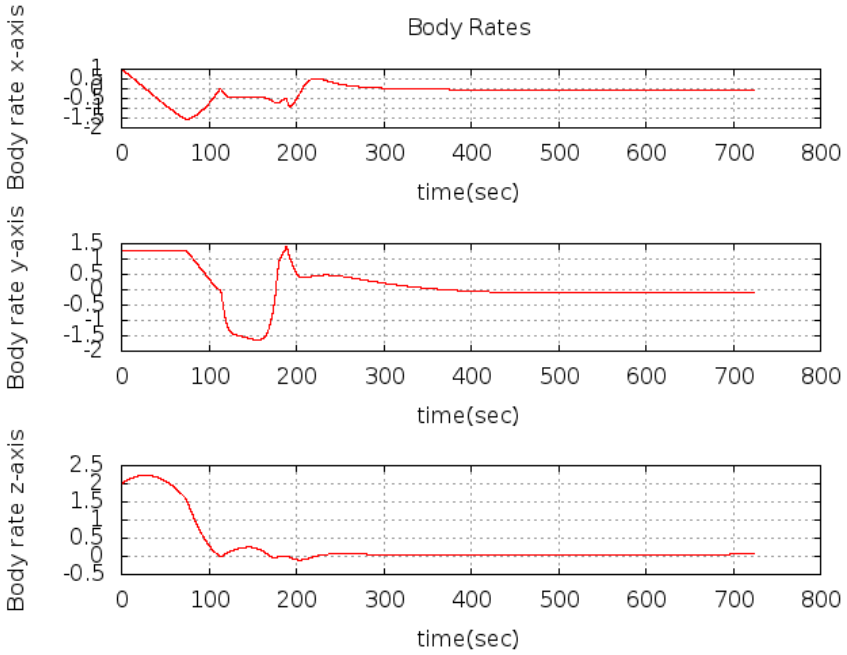
All the system tests were executed on the HW-CFI with EUROSIM provided by ESA

The following table summarize the results obtained executing all the tests on EagleEye CSW v7 integrated with LibPUS-C 1.1.0, LibPacketStore 1.0 and FMS.

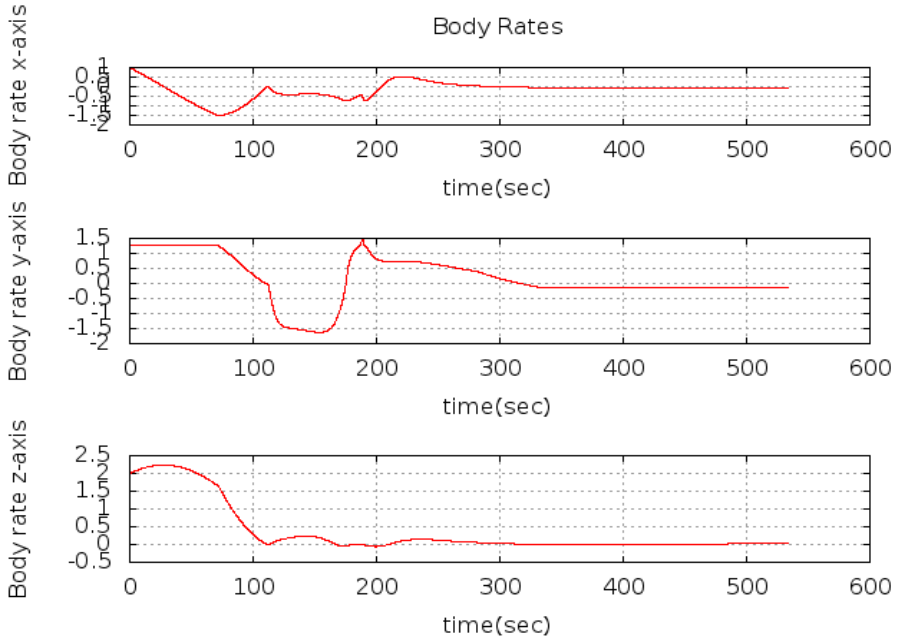
Test	Passed	Failed	Total	Status
Unit Test	39	0	39	Passed
Integration Test	52	0	52	Passed
System Test	2014	0	2014	Passed
AOCS Test	227	0	227	Passed

Validation & System Test 3/3

AOCS performances obtained, re-executing AOCS Tests on both ARM and SPARC, are in line with the previous EagleEye CSW version, so the integration of LibPUS-C 1.1.0, LibPacketStore 1.0 and FMS in the EagleEye software can be considered completed and successful.



SPARC



ARM

Conclusion: lesson learnt

All the things foreseen in the CCN1 SoW have been introduced into EagleEye CSW v7.

Unfortunately, the OBCP Micropython has been cut off as it was not ready to be integrated on EagleEye TSP architecture based on XtratuM and RTEMS.

The Industrial team performed the maximum effort possible to try to integrate it in to EagleEye, but the time and budget foreseen for the CCN1 was not enough for the activity of porting the OBCP micropython on top of this new execution environment.

Moreover, all the documentation provided in bundle with the OBCP was related to the usage of the OBCP on LEON2 and RTEMS 4.8 and so completely out of date respect to usage foreseen by the CCN1

Conclusion: suggestions for future developments

Dedicate a project to the porting of the OBCP Micropython engine on top of EagleEye TSP architecture should be considered.

The time and the budget allocated on this project made hard for the Industrial team to reach this target.

Summary and Conclusion

The CCN1 project has been very challenging for the Industrial team looking the budget and time allocated to the project.
Nevertheless, all the activities foreseen have been carried out.

The only activity not completed has been the integration of the OBCP Micropython engine despite the commitment of the Industrial team.

The activity required for the OBCP Micropython engine integration into EagleEye went over the simple integration, but it required more a porting activity on a completely new execution environment.

Despite the problems found the Industrial team was able to consolidate and enhance the LibPUS-C and improve the EagleEye CSW functionalities with all the desiderata by the Agency.

Q&A