# Generic Operations Interface Requirements Document (GOIRD)

13th ESA Workshop on Avionics, Data, Control and Software Systems

Ignacio Clerigo

12/11/2019

# Agenda

- **Introduction**
- Generic OIRD overview
- Functional requirements
- PUS-C requirements
- Conclusions

# Introduction

- **Operability requirements** are an early input to the spacecraft development
  - Functional requirements for the space segment, including the payload, necessary for the conduction of all mission operations
  - Prepared by **ESA**

- For (most) ESA missions→ **OIRD**: Operations Interface Requirements Document
  - **Mission-specific**
  - Standalone doc or integrated in System Requirements Document (**SRD**)
  - Tightly coupled with CCSDS and ECSS standards

# Introduction

- **The problem**
  - A dedicated Operations Interface Requirements Document (**OIRD**) per mission
  - New mission OIRDs are derived from previous OIRDs
    - Lessons learnt not always propagated to other missions and mission families
  - OIRDs are perceived as significantly different from one mission to another…
    - …but difference not necessarily justified or not as big as perceived
    - A similar problem is perceived by operators for what concerns on-board SW implementations
  - The operability requirements drives the mission data management services on-board and the Mission Control System on-ground
    - The variability of requirements and implementations hamper product orientation
  - OIRDs evolved independently of standards (PUS-C, OBCP, etc.)
    - Partial overlap between OIRD and PUS requirements

# Introduction

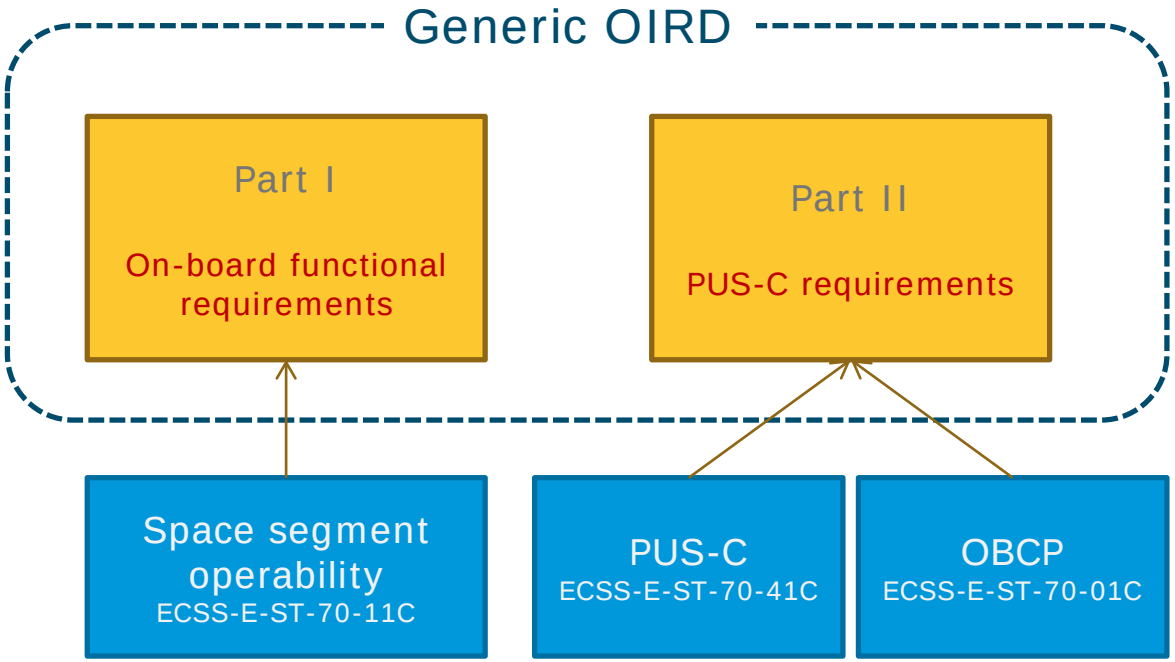
- **Generic OIRD (GOIRD) foundations**
  - SAVOIR entrusted ESOC in June 2016 the task to establish a Generic OIRD
  - It shall form the baseline spec for future missions
  - It shall allow the definition of compatible OBSW libraries

- **The goal**
  - Create a **common** OIRD for **all** new missions
    - Maximize commonality among missions
    - Minimize differences and group them by **mission families**
  - Mission OIRDs to be created starting from the generic OIRD
    - Changes limited to mission specific deltas
    - Plus sizing of PUS-C services (e.g. MTL capacity, etc.)
  - Generic OIRD to be evolved with lessons learnt and feedback from new missions

# Agenda

- Introduction
- **Generic OIRD overview**
- Functional requirements
- PUS-C requirements
- Conclusions

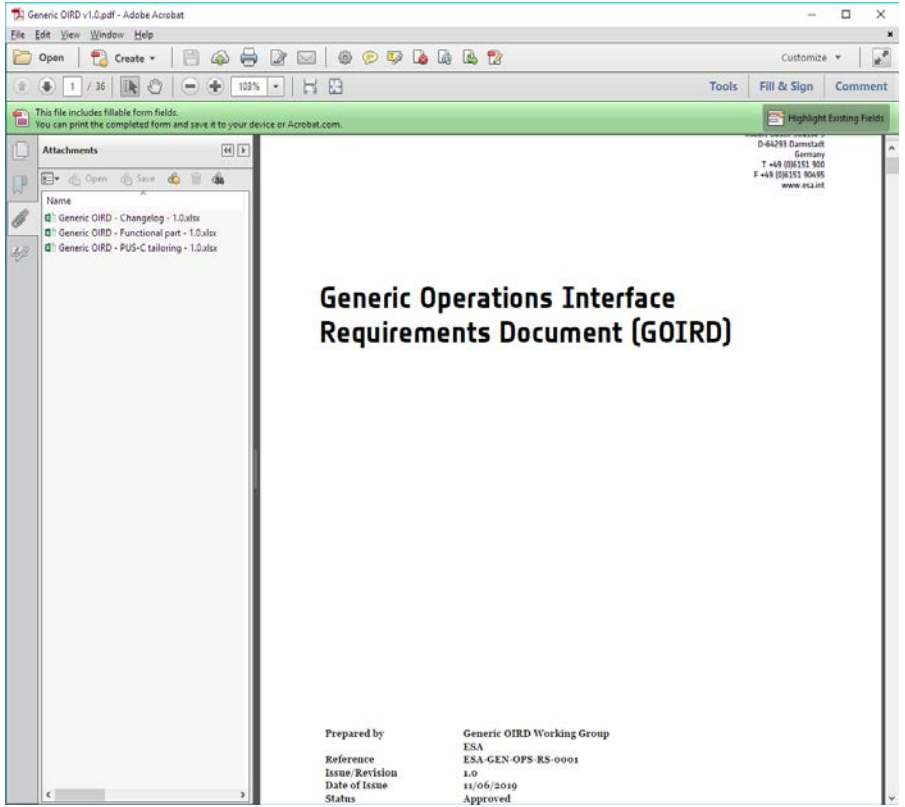
# Generic OIRD - Structure



# Generic OIRD - Format

- GOIRD delivered as a **PDF document**

- 3 Excel files attached → core of the GOIRD
  - **Functional requirements**
  - **PUS-C requirements**
  - Plus changelog
    - All changes to PUS-C & functional requirements since Draft A tracked



# Generic OIRD - Content

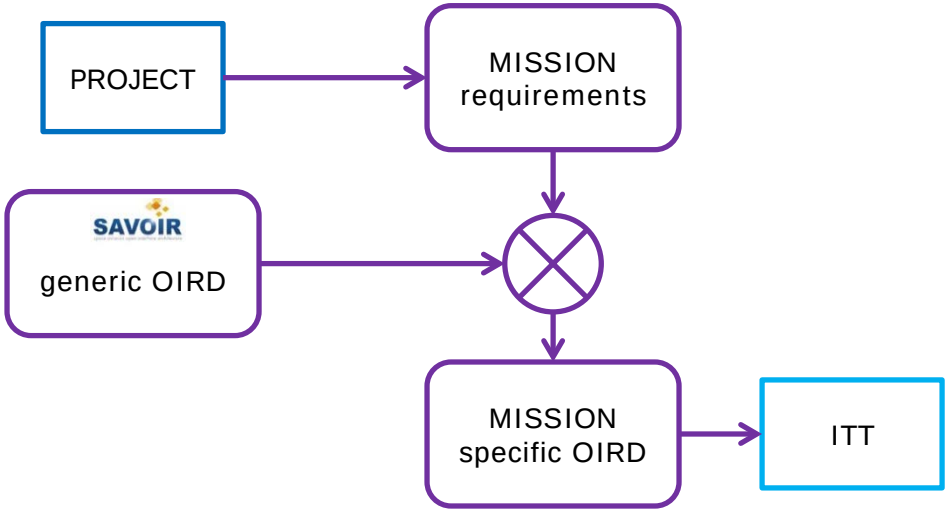
- Generic OIRD **document**
  - Motivation and scope
    - Scope limited to robotic missions with an **avionics-like subsystem in mind**
  - Description of spreadsheet structure and requirements format
    - Including relationship with ECSS standards
  - Mission OIRD generation instructions

# Mission OIRD generation

- GOIRD is **NOT** a replacement of mission OIRDs
  - Missions are still expected to generate their own mission OIRD
  - GOIRD is only a reference document for the mission

- Mission OIRDs shall be produced starting from GOIRD
  - No copy & paste from previous mission OIRDs
  - Instead take all GOIRD requirements and implement mission-specific changes
  - Clear tracing in mission OIRD of changes/additions with respect to GOIRD
  - Changes shall be truly mission specific:
    - Very few deltas with respect to the GOIRD expected in the mission OIRD
    - Definition of mission constants and sizing of PUS-C services
    - De-scope of PUS-C services and functional requirements not applicable to the mission

# Mission OIRD generation



# Generic OIRD - Status

- Generic OIRD 1.0 formally released in **July 2019**

- **Used already by multiple projects to build their mission OIRDs**
  - Copernicus high-priority candidate missions OIRD
  - ERO / Mars Sample Return OIRD
  - HERA OIRD

- Generic OIRD will evolve over time
  - Feedback from new projects preparing their mission OIRD (e.g. new features)
  - Lessons learnt from flying missions

# Agenda

- Introduction
- Generic OIRD overview
- **Functional requirements**
- PUS-C requirements
- Conclusions

# Functional requirements - Content

- **High-level operability requirements**
  - No assumptions on the underlying system architecture
  - PUS agnostic requirements
    - Which capabilities are required is left to the PUS-C tailoring
  - Linked to ECSS Space Segment Operability Standard
    - Traceability maintained w.r.t. Operability Standard
    - But GOIRD functional requirements are self-consistent

# Functional requirements - Content

- Functional requirements groups:
  - General
  - Modes
  - Operations/mission phases
  - Spacecraft control
  - Telecommands
  - Authentication
    - Only applicable to Earth Observation missions.
  - Telemetry
  - Timing
  - In-flight testing
  - General autonomy
  - Failure detection isolation and recovery
  - Safe mode
  - Subsystems: Attitude Control And Navigation, On-board Processors and Software, Data Storage and File System, Power, Thermal, TT&C, Payloads and Mechanisms.

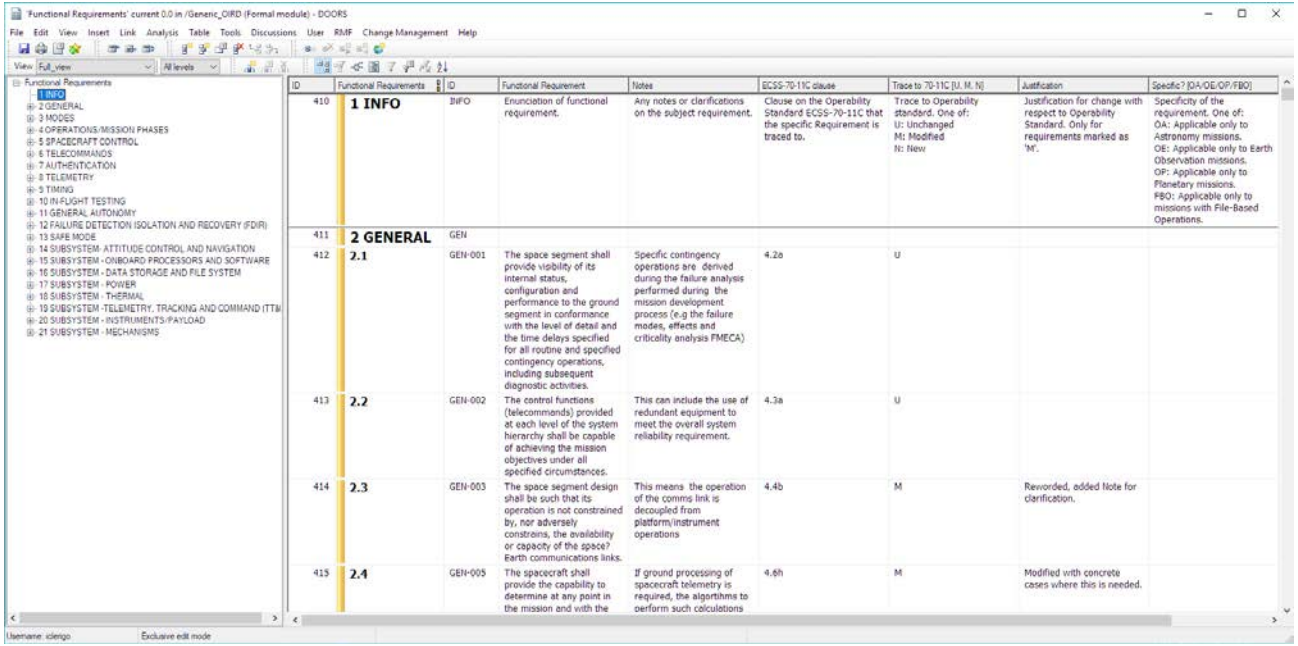
# Functional requirements - Format

- Functional requirements maintained in **DOORS**
  - Requirements can be provided as DOORS modules on-demand

- First module contain all the requirements
  - ID
  - Requirements and notes
  - Relationship to ECSS-E-ST-70-11C (unchanged, new or modified) and clause
  - Justification for modified requirements wrt ECSS
  - Mission-family specific flag to report requirements specific to interplanetary, astronomy or Earth Observation

- Second module contains list of Operability Standard requirements tailored out

# Functional requirements - Format



# Agenda

- Introduction
- Generic OIRD overview
- Functional requirements
- **PUS-C requirements**
- Conclusions

# PUS-C requirements

- **PUS-C** is a major step forward compared to PUS-A
  - Solid PUS foundational model
  - Clear and extensive system requirements
    - Behavior of each service type and related TCs is (almost) unambiguous
  - Clear interface definition separated from system requirements
  - New service types and sub services (file management, S3 func. Reporting, etc.)

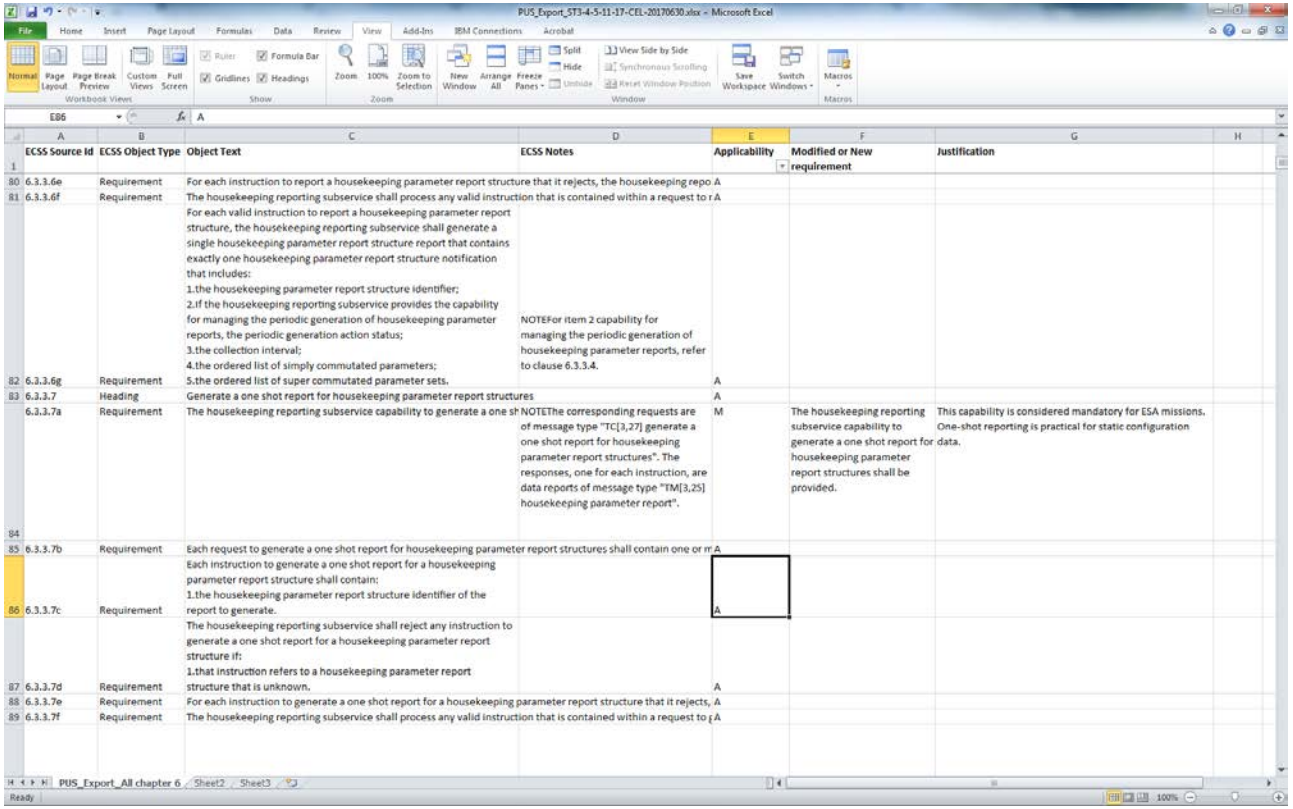
# PUS-C requirements

- **Why do not use the PUS-C standard as is?**
  - 20 services with 33 subservices defined but no mapping to a specific architecture
    - Which services to implement and where?
  - **316** TCs and TM reports (if all subservices implemented) but only **88** are **mandatory**
    - Tailoring required to define functionality required by the mission
  - 382 "shall be declared when specifying“ statements
    - Large number of capabilities and options
    - Without tailoring, 2 PUS-C implementations will result in very different interfaces
  - Some important functionalities for ESOC missions are not covered
    - Critical Event Log/System Log, file transfer, backup MTL…
  - Feedback and lessons learnt from 50+ years of operations
    - A few important requirements from mission OIRDs not captured in PUS-C

# PUS-C requirements - Format

- PUS-C implemented in **DOORS**
  - Against the original PUS-C standard in DOORS format
  - Additional columns
    - Applicability flag (unchanged/modified/deleted/new)
    - Justification for each change
  - Multiple export possibilities
    - Full tailored standard or deltas only
    - In Excel, PDF, DOORS module…

# PUS-C requirements - Format



PUS_Export_ST3-4-5-11-17-CEL-20170630.xlsx - Microsoft Excel

| | ECSS Source Id | ECSS Object Type | Object Text | ECSS Notes | Applicability | Modified or New requirement | Justification |
|---|---|---|---|---|---|---|---|
| 80 | 6.3.3.6e | Requirement | For each instruction to report a housekeeping parameter report structure that it rejects, the housekeeping repo | | A | | |
| 81 | 6.3.3.6f | Requirement | The housekeeping reporting subservice shall process any valid instruction that is contained within a request to r | | A | | |
| 82 | 6.3.3.6g | Requirement | For each valid instruction to report a housekeeping parameter report structure, the housekeeping reporting subservice shall generate a single housekeeping parameter report structure report that contains exactly one housekeeping parameter report structure notification that includes:<br>1.the housekeeping parameter report structure identifier;<br>2.if the housekeeping reporting subservice provides the capability for managing the periodic generation of housekeeping parameter reports, the periodic generation action status;<br>3.the collection interval;<br>4.the ordered list of simply commutated parameters;<br>5.the ordered list of super commutated parameter sets. | NOTEFor item 2 capability for managing the periodic generation of housekeeping parameter reports, refer to clause 6.3.3.4. | A | | |
| 83 | 6.3.3.7 | Heading | Generate a one shot report for housekeeping parameter report structures | | A | | |
| 84 | 6.3.3.7a | Requirement | The housekeeping reporting subservice capability to generate a one sh | NOTEThe corresponding requests are of message type "TC[3,27] generate a one shot report for housekeeping parameter report structures". The responses, one for each instruction, are data reports of message type "TM[3,25] housekeeping parameter report". | M | The housekeeping reporting subservice capability to generate a one shot report for housekeeping parameter report structures shall be provided. | This capability is considered mandatory for ESA missions. One-shot reporting is practical for static configuration data. |
| 85 | 6.3.3.7b | Requirement | Each request to generate a one shot report for housekeeping parameter report structures shall contain one or m | | A | | |
| 86 | 6.3.3.7c | Requirement | Each instruction to generate a one shot report for a housekeeping parameter report structure shall contain:<br>1.the housekeeping parameter report structure identifier of the report to generate. | | A | | |
| 87 | 6.3.3.7d | Requirement | The housekeeping reporting subservice shall reject any instruction to generate a one shot report for a housekeeping parameter report structure if:<br>1.that instruction refers to a housekeeping parameter report structure that is unknown. | | A | | |
| 88 | 6.3.3.7e | Requirement | For each instruction to generate a one shot report for a housekeeping parameter report structure that it rejects, | | A | | |
| 89 | 6.3.3.7f | Requirement | The housekeeping reporting subservice shall process any valid instruction that is contained within a request to g | | A | | |

# PUS-C requirements – Classification

- **Services, subtypes and capabilities selection**
  - Agreed in a common set of services and subservices for all future ESA missions
    - Same set of TC and TM packet types and subtypes
    - Fix set of capabilities
  - Interface to satellite is fairly independent of the mission → A **single** tailoring for all ESA missions
  - About 60% of the total GOIRD requirements for PUS-C

# PUS-C requirements - Classification

- **Additional observables**
  - PUS-C heavily relies on (on-demand) reports
    - Few mandatory parameters to be provided in HK
  - Additional observables added to most services
    - Based on observables typically available on modern spacecraft
  - About 2% of the total GOIRD requirements for PUS-C

# PUS-C requirements - Classification

- **New capabilities**
  - Features required by ESA missions not covered by PUS-C standard
    - Critical event log / System log
    - Parameter extraction from HK
    - Backup MTL
    - **File based operations**
  - Covered as **new sections**
    - High level requirements only → no PUS-C style text, TC and TM interface not defined
    - Implementation details left intentionally open
  - To be raised as change requests to the standard to be considered in future versions
  - About 19% of the total GOIRD requirements for PUS-C

# PUS-C requirements - Classification

- **Avionics architecture definition**
  - PUS-C does not make any assumption about the avionics architecture
    - But architecture definition has a huge impact on operability
  - A few services required to be implemented in a centralized way
    - E.g. Service 11 (Mission Timeline)
  - A few services mandated to be implemented in all packetized units
    - Services 1, 3, 5, 6 (if applicable) and 17
  - Less than 1% of the total GOIRD requirements for PUS-C

# PUS-C requirements - Classification

- **Clarifications and refinements**
  - Lessons learnt from previous missions and mission OIRDs
  - About 12% of the total GOIRD requirements for PUS-C

- **Changes to the standard**
  - Changes to PUS-C behavior limited to the **absolute minimum**
    - No modification of defined TCs and TMs interfaces
    - Clear justification (typos or omissions, strong operational needs)
  - Less than 1% of the total GOIRD requirements for PUS-C
  - To be raised as change requests to the standard to be considered in future versions

# PUS-C requirements - Problems

- **PUS-C implementation is not straightforward**
  - Key low level aspects have not been standardize in PUS-C
    - Example: number of bytes to encode certain packet fields
    - Very important for **inter-operability of PUS-C units!**
  - Interface and detailed implementation of new features proposed in GOIRD is open

  → **Coordination across PUS-C library implementers desired!**

# OBCP requirements

- No tailoring of the ECSS OBCP standard in the GOIRD
  - The current version of the standard satisfies most ESA needs
  - → Few extra requirements added directly in the PUS-C requirements for service 18 (OBCP)

# Agenda

- Introduction
- Generic OIRD overview
- Functional requirements
- PUS-C requirements
- **Conclusions**

# Conclusions

- Generic OIRD
  - New approach at ESA for operability requirements
  - Common set of functional and PUS-C requirements for future ESA missions
  - **Goal is to reduce variability and cost**
    - Definition of compatible OBSW libraries
    - Further harmonisation of ground segment

- Long term vision
  - Collaboration among PUS-C libraries developers?
    - Interoperable PUS-C implementations and generic TM/TC ICD?
  - Further harmonization of other elements: procedures, databases…


THANK YOU!
QUESTIONS?