# Faster, Later, Softer: COrDeT – an on-board software reference architecture

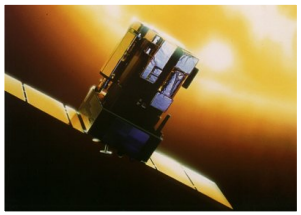
Andreas Jung and Jean-Loup Terraillon
European Space Agency
ESTEC, Software Systems Division

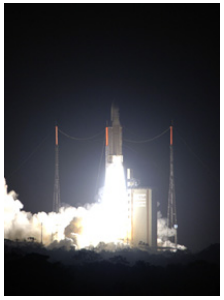
2010 Workshop on Spacecraft Flight Software (FSW-10)
8 Dec 2010, Pasadena

# Spacecraft on-board software landscape – Observations and concerns

Software **size** of the central computer's ESA missions is increasing…

- Science satellites
  - Exosat (launch 1983), RCA1802 – 8K memory, ASM
  - SOHO (launch 1995), 2xMDC281 – 2x64KB , Ada83
  - Rosetta (launch 2004), 2xMA3-1750 – 2x1MB, 170KLoc, Ada83
- Launcher
  - Ariane5, 68020, ~200KLoc, Ada
- Earth Observation
  - Cryosat-2 (launch 2009), ERC32 – 4MB, ~50KLoc Ada95
  - GOCE (launch 2009), ERC32 – 4MB, ~100KLoc, Ada95
  - Aeolus, ERC32 – 4MB, ~160KLoc
- ATV (launched 2008), ERC32--8MB, 1MLoc (650KLoc code), Ada95

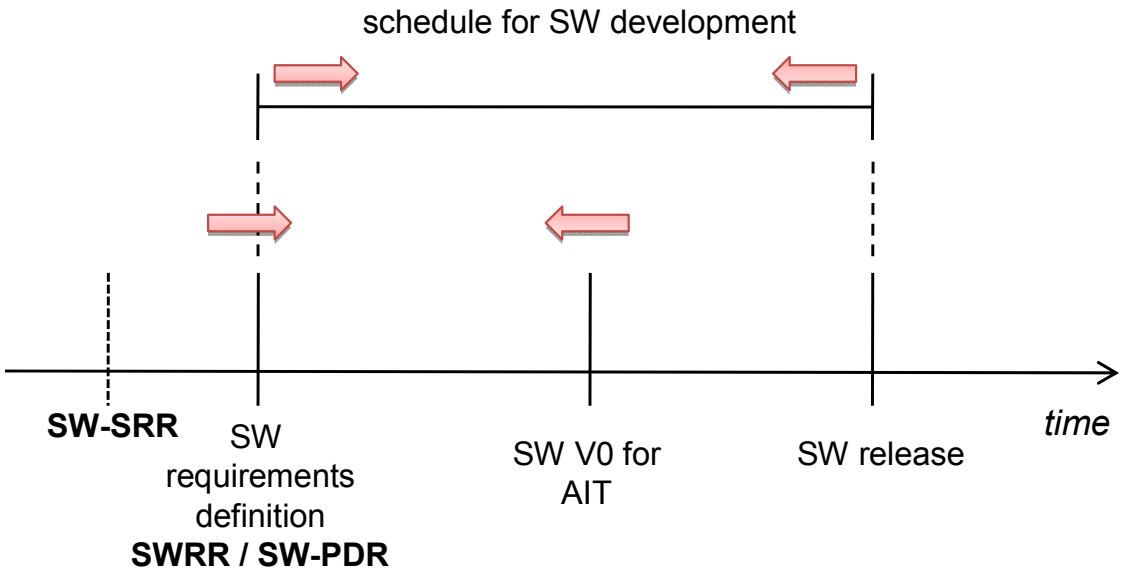



…and their **complexity** is increasing

(see also NASA Study on Flight Software Complexity, 2009).

# Spacecraft on-board software landscape – Observations and concerns

The **schedule** for the software development is getting tighter:




*Nevertheless*:

Spacecraft platforms have **similar functionalities**. There are families of spacecrafts (for science, earth observation, …).

The **platform software is even more similar**… but currently, there are few opportunities to spend effort on advanced functions.

# Software Engineering needs: Faster, Later, Softer

## FASTER (increase productivity)

- Shorter software development time
- Reduce Verification and Validation effort
- Reduce recurring developments (don't redevelop recurring software: about 50% of platform SW)
- Increase cost-efficiency (more requirements same cost)
- Quality of the product (at least same quality)

## LATER (increase reactivity)

- Mitigate the impact of late requirement definition or change
- Optimize flight maintenance
- Simplification and harmonization of FDIR

## SOFTER (increase flexibility)

- Support for various system integration strategies (customer-supplier)
- Industrial policy support
- Role of software suppliers (multi-vendor policy)
- Dissemination activities (concept usable by system engineers)
- Future needs

# Why a reference architecture replies to these needs ?

**FASTER**? → **automation of life cycle**, model driven engineering

*yes, but not enough…*

---

We need also **pre-development of software** for faster configuration, later configuration, softer developments (6 years, 6 months, 6 days…), e.g. missionisation of launchers

*yes, but…*

Predevelopment of what? → Of **building blocks**

Are they Lego? → No, they are flexible (parameterization)

Compose Building Blocks? → Therefore need **interface standardization**

Where are the interfaces? → Therefore a **reference architecture**

---

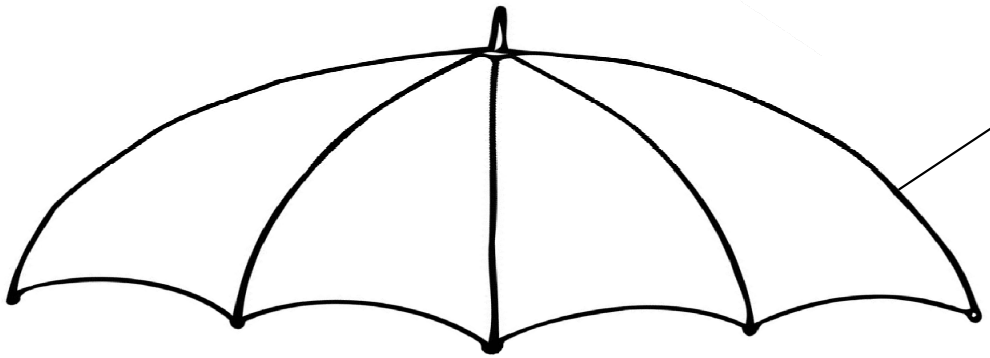
Reduce validation? → Composability and compositionability,

*Separation of concerns, correct by construction, component model*

# SAVOIR – the umbrella of avionics reference architecture related activities

**SAVOIR**

Space Avionics Open Interface Architecture



**SAVOIR Advisory group:**

- Space Agencies (ESA and national)
- Prime industry
- Supplier industry

SW: SAVOIR-FAIRE working group

Other SW / HW groups and activities

SW R&D activities

...

# Spacecraft on-board software landscape – ESA R&D studies and activities

ESA **R&D studies and activities** on software reference architectures:

- COrDeT-1/2 – Component Oriented Development Techniques
- DOMENG – Domain Engineering
- SAVOIR-FAIRE - SW reference architecture working group

Complementary studies presented in the two following ESA presentations:

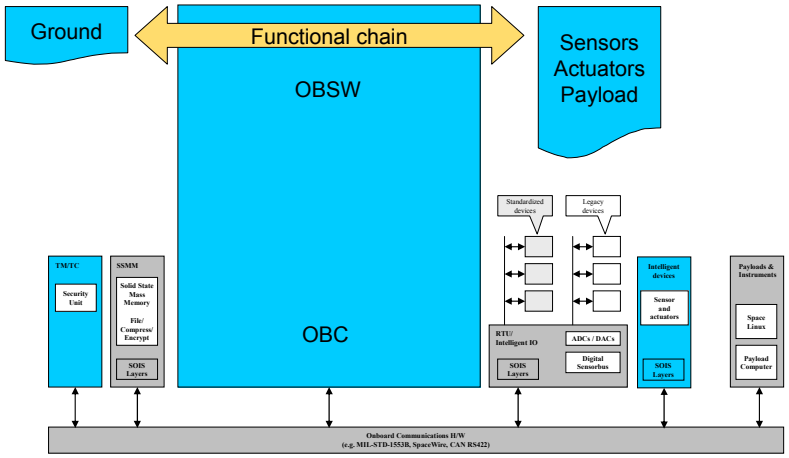
- Time and space partitioning
- On-board control procedures

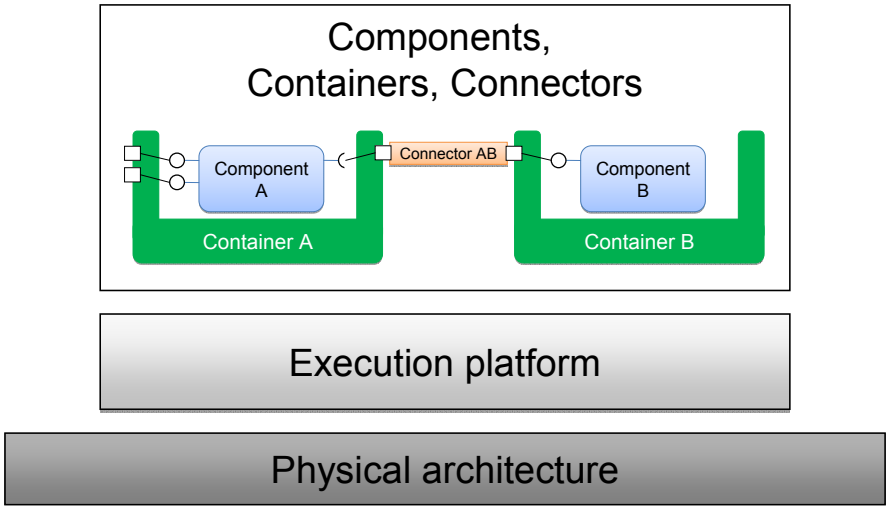
**Results:**

- Result from COrDeT activity: Spacecraft platforms and **software have similar functionalities**, even across the different families (science, earth observation, …).
- Therefore: **Opportunities** to spend more effort on (advanced) **functions** rather than "re-inventing the wheel" for the common elements → REUSE

# How to arrive at and What is a reference architecture?

**Reference architecture** = mapping of **Functional chains & Variability factors** on to **Software Architectural concepts**

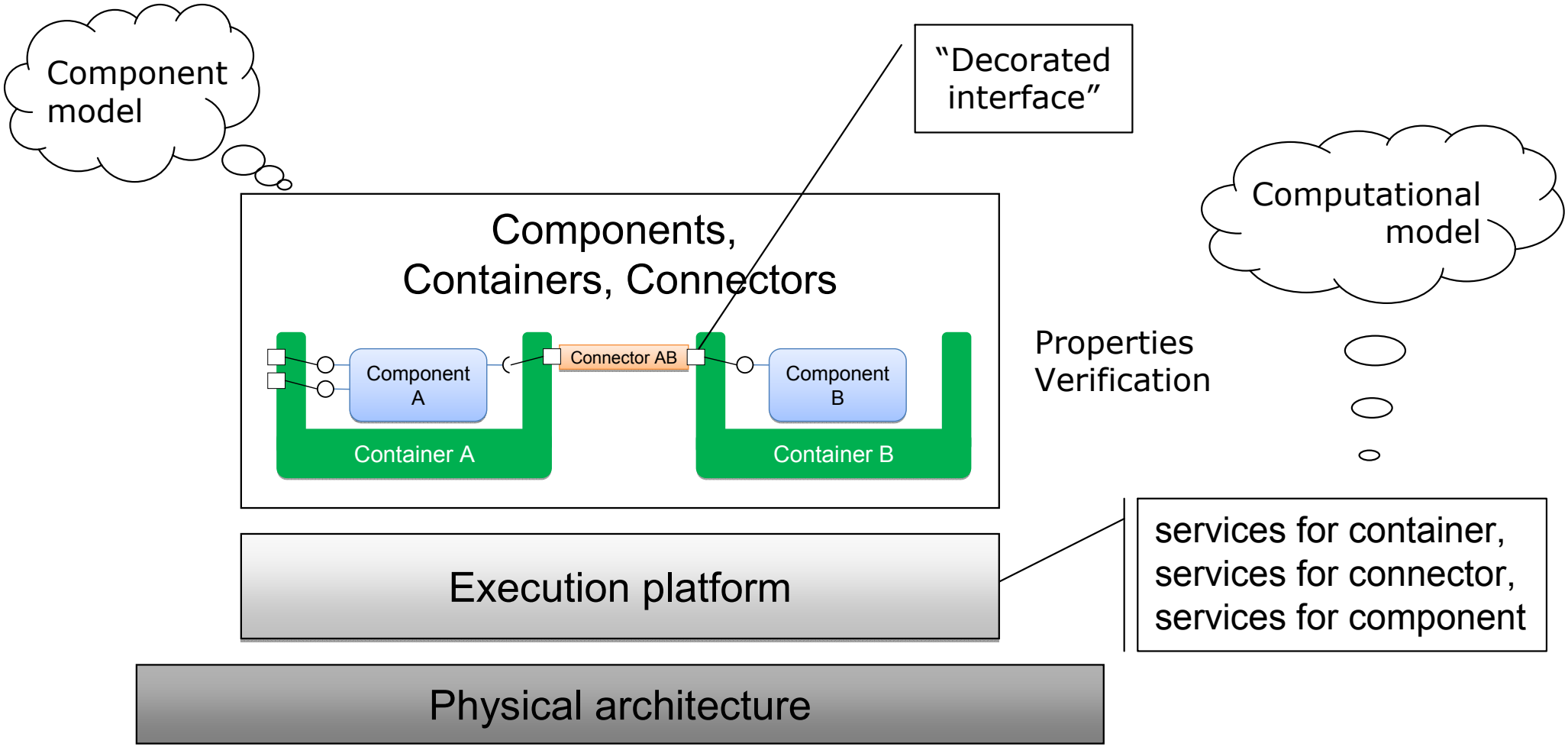


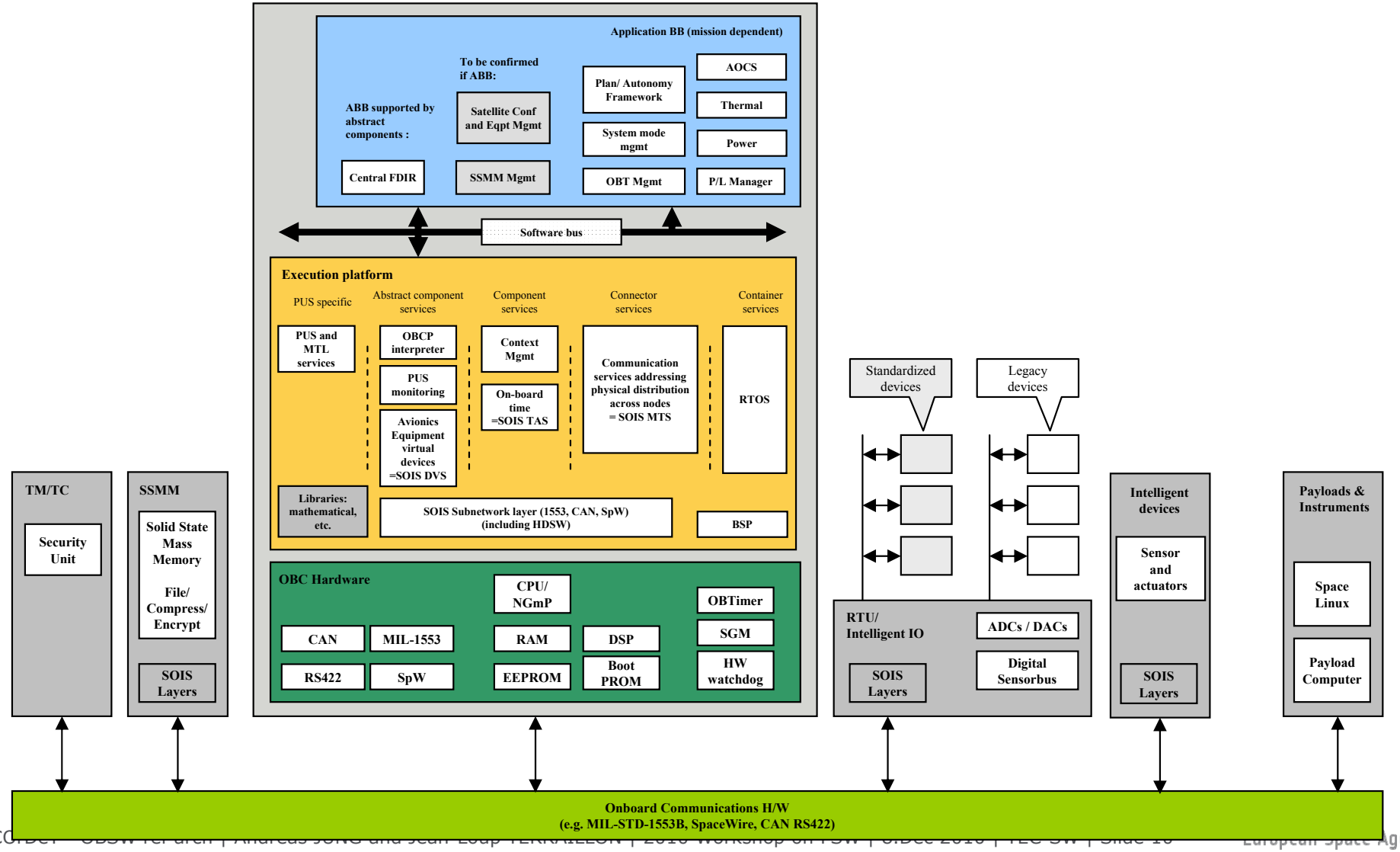

**Building blocks & Interfaces**

# Software architectural concept

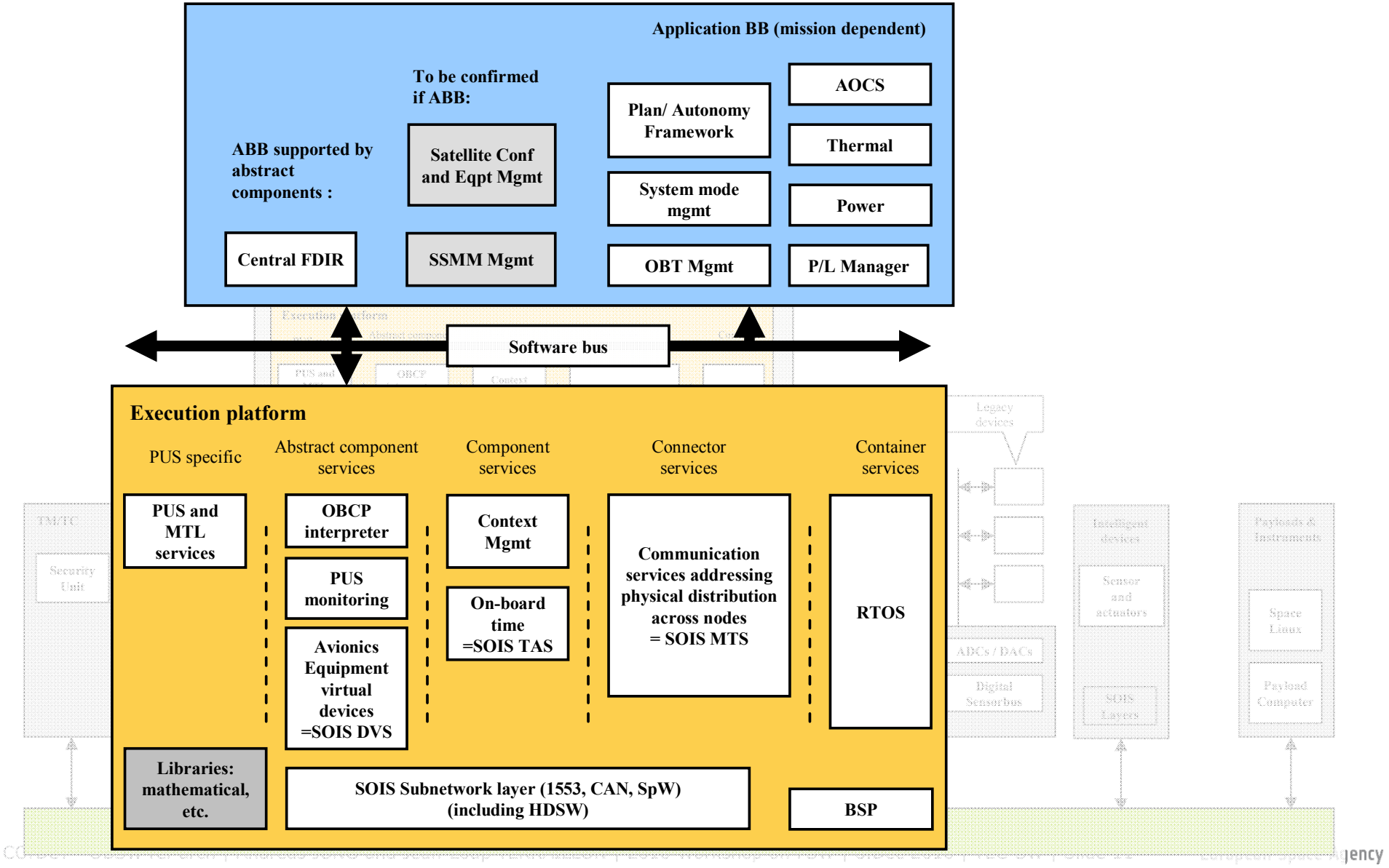
## Component Based Software Engineering



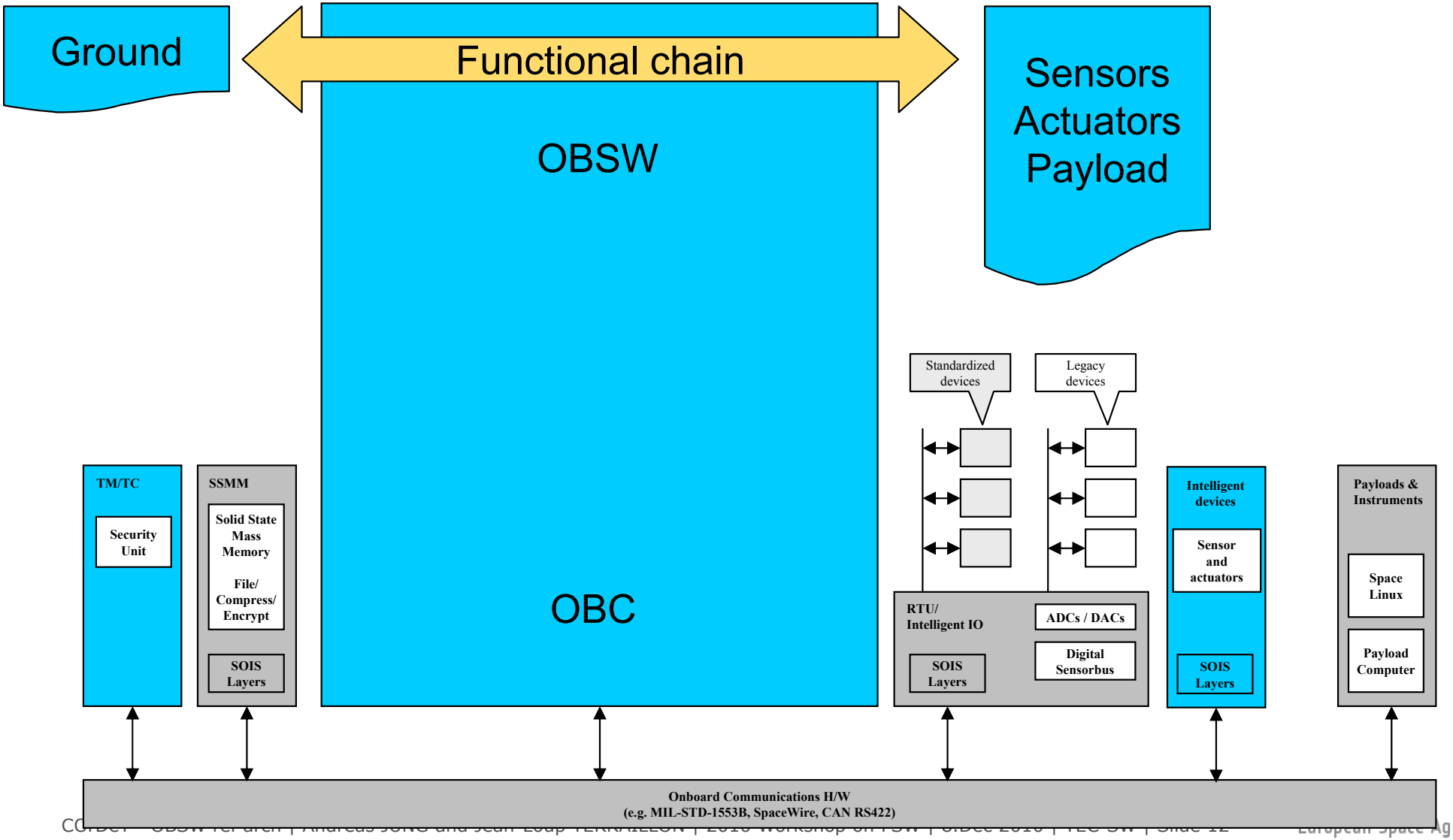
# The SAVOIR avionics reference architecture (HW + SW)



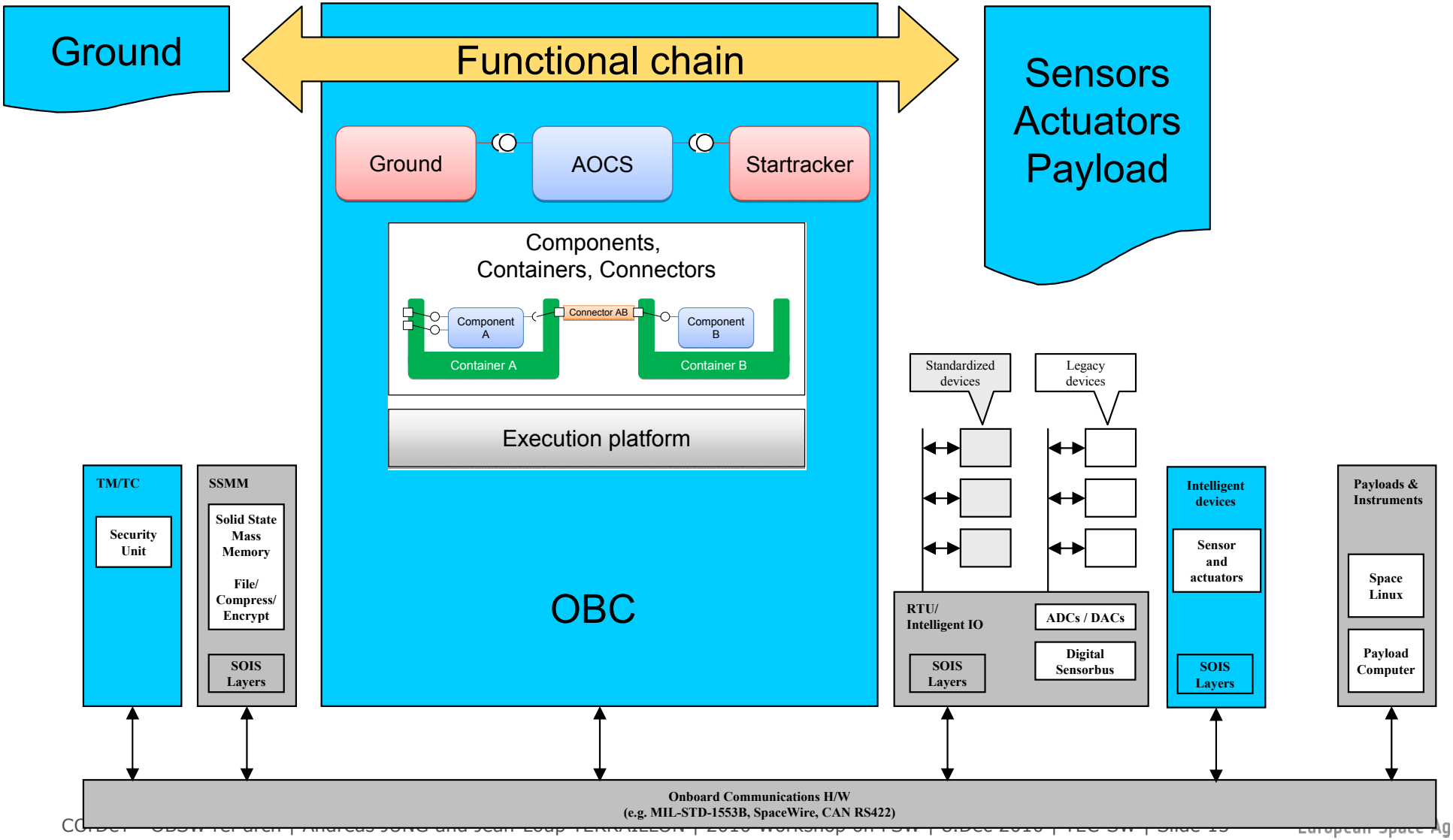
# The SAVOIR avionics reference architecture (HW + SW)



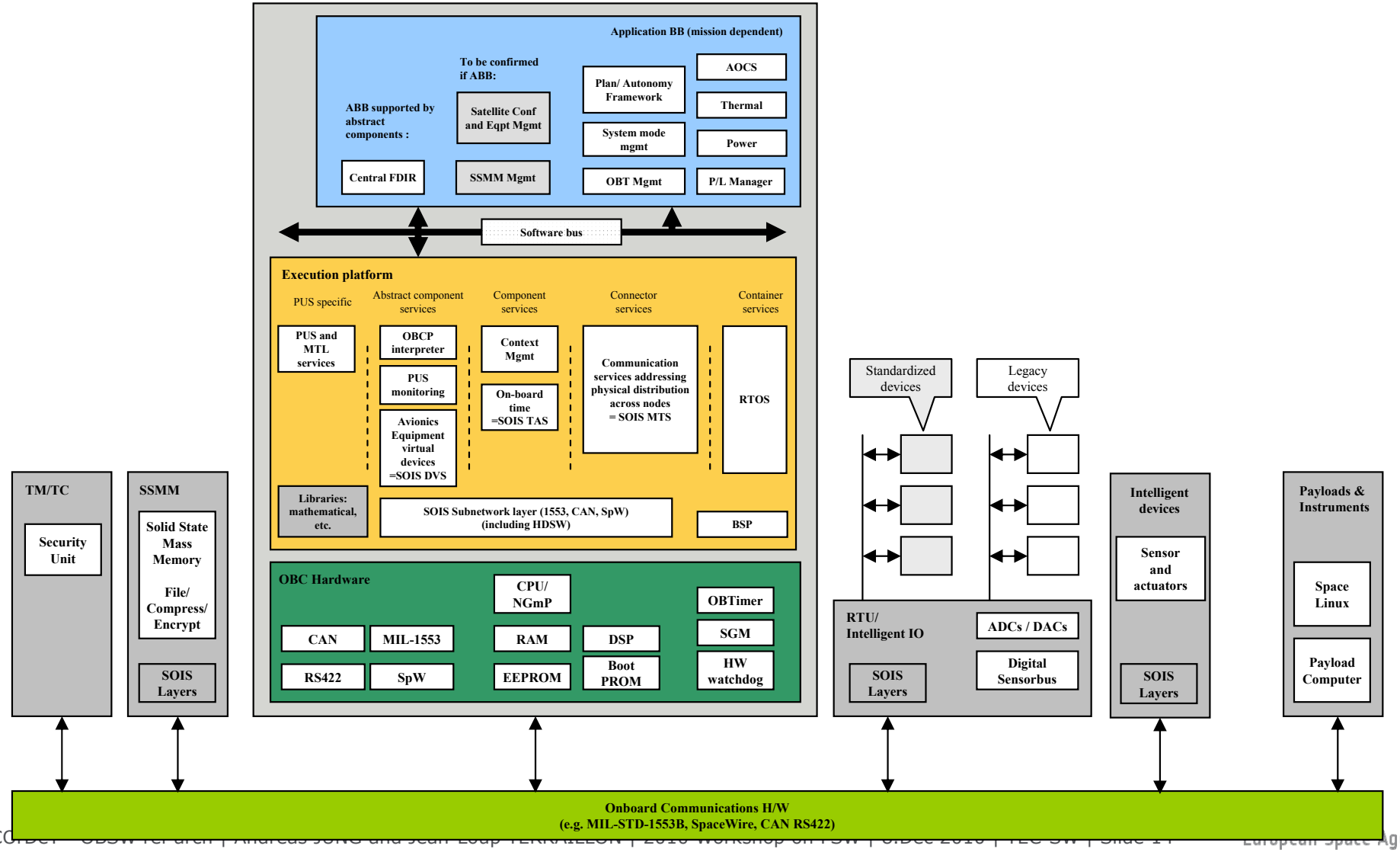
# The Function Chain



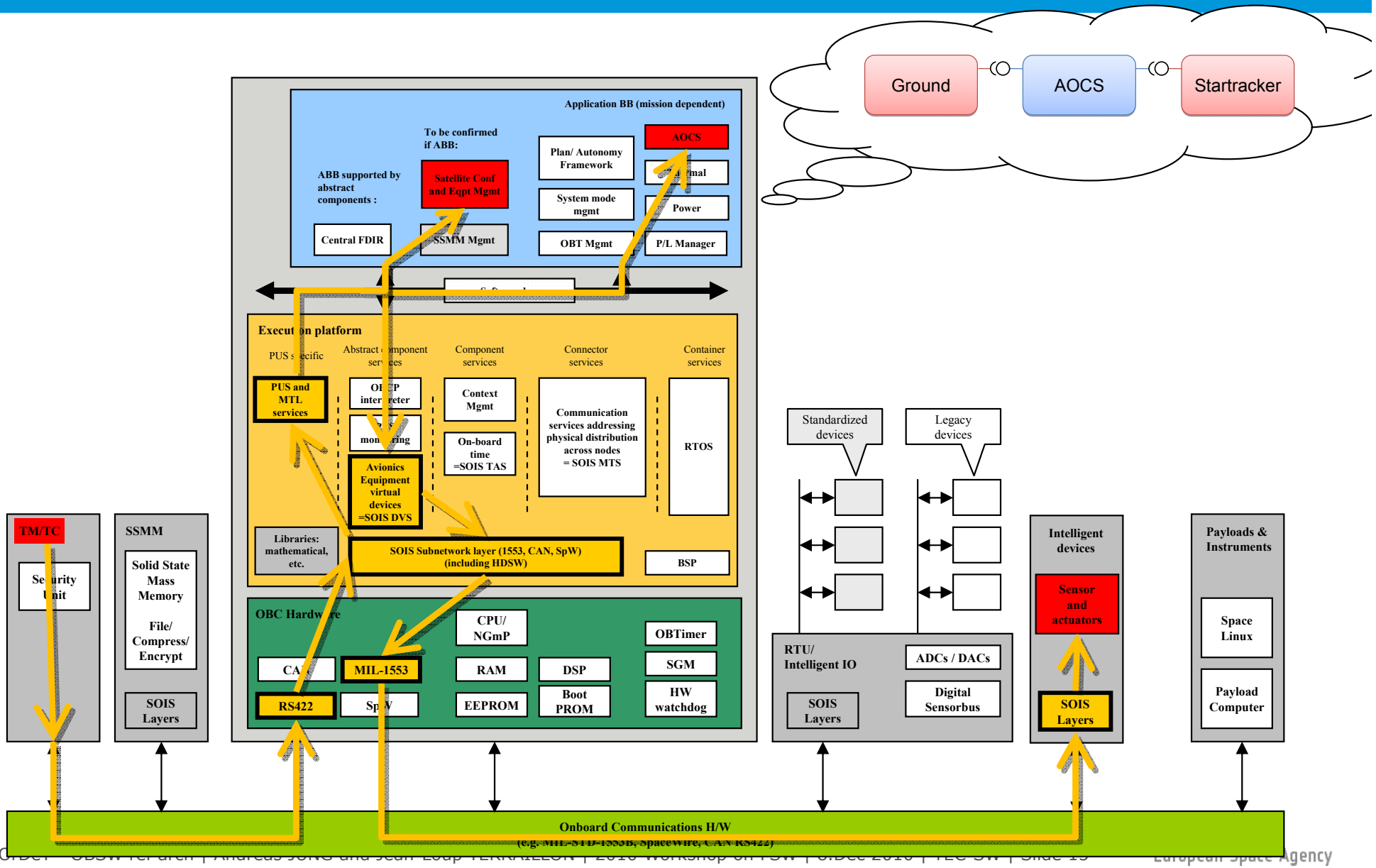
# Mapping Functional Chain onto SW architectural concept



# The SAVOIR (Software) reference architecture



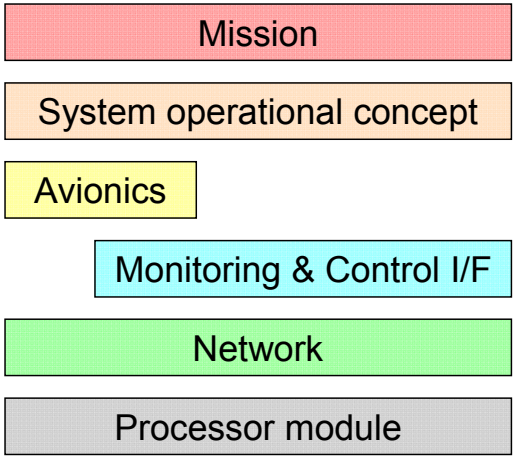
# Mapping of functional chain on to the SW architecture



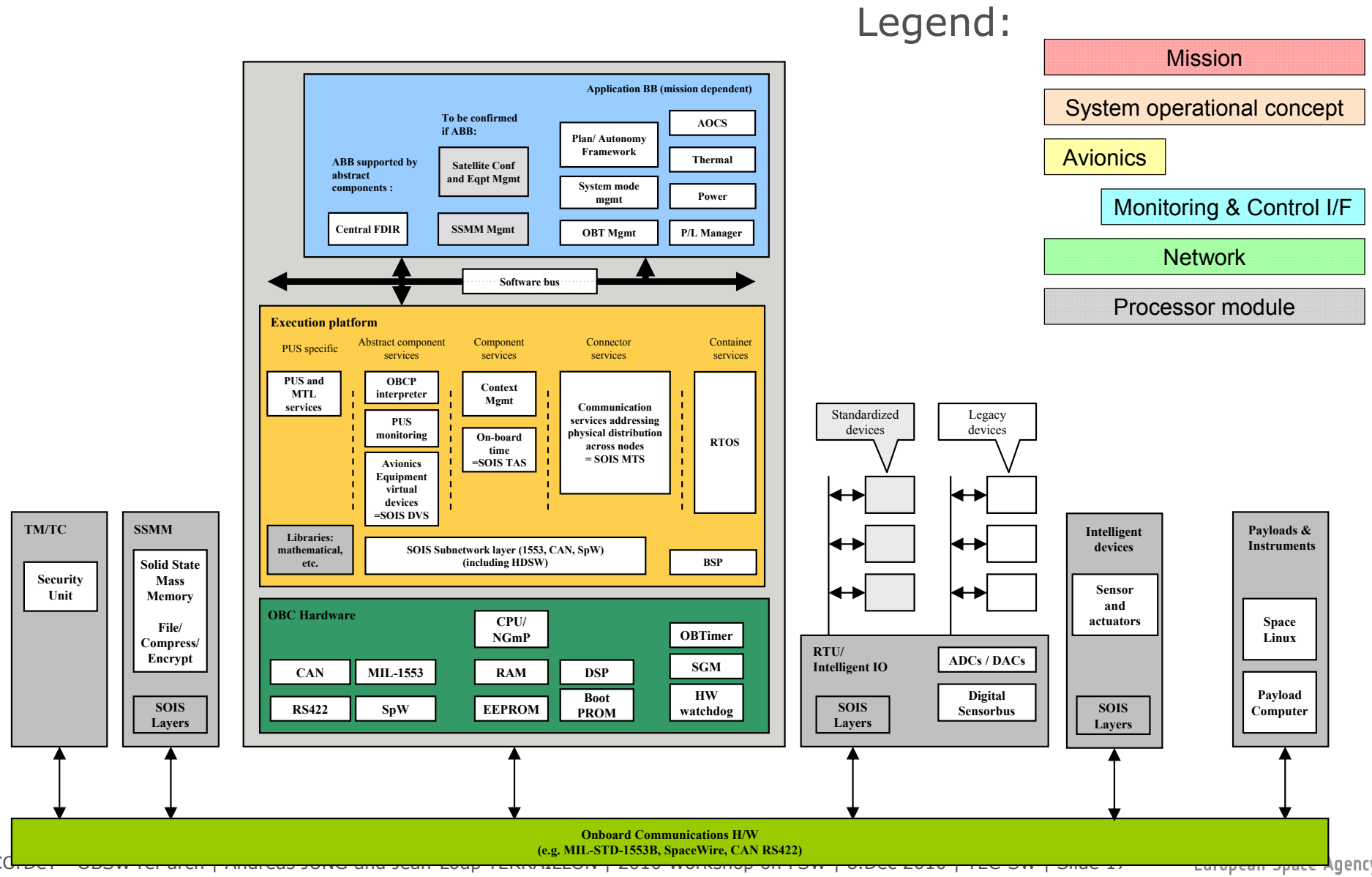
# How is a Block reusable?


1. Architecturally reusable
   → ensured by interface standards and component model (composability, compositionality)

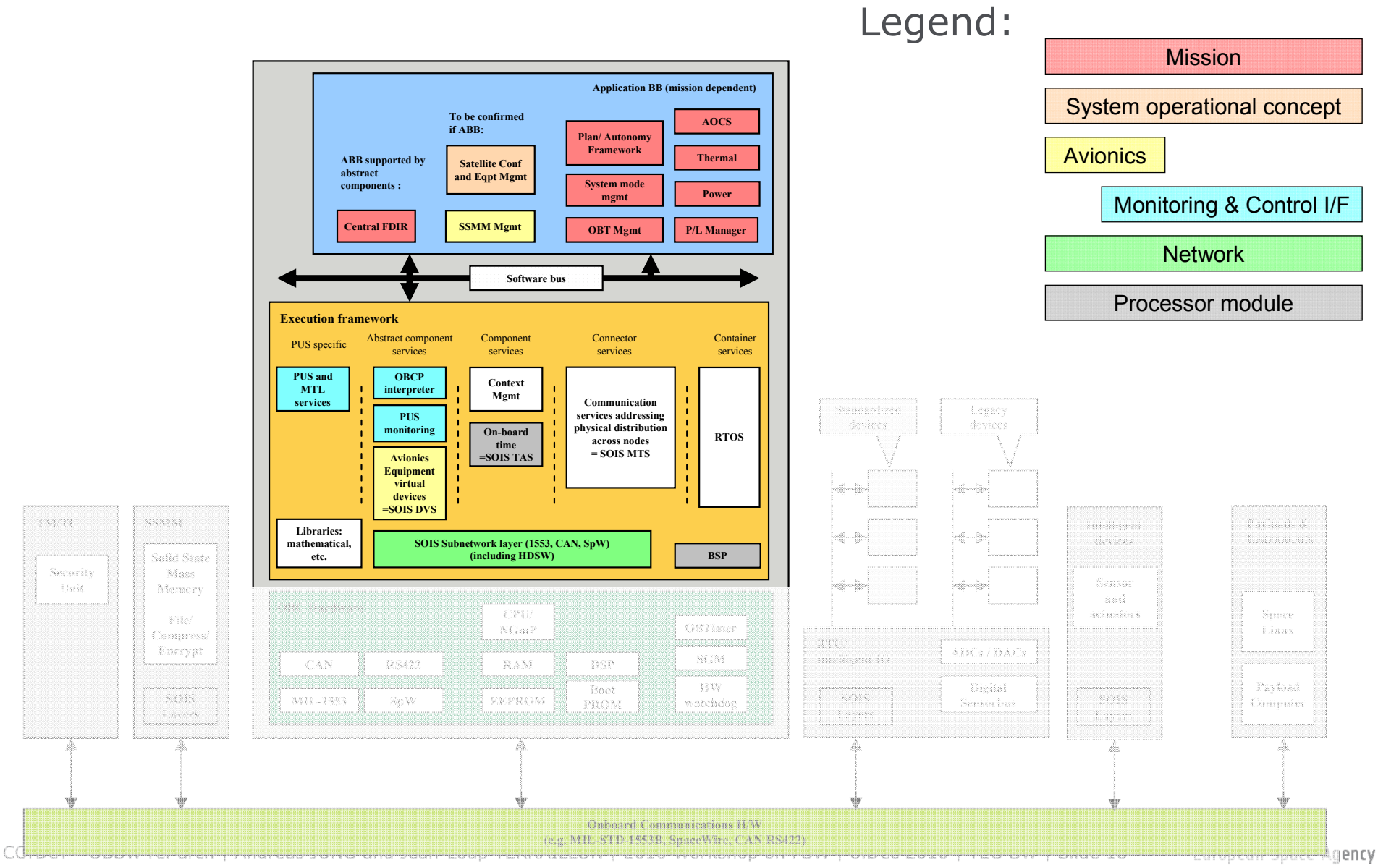

2. Functional reusable
   → Domain engineering
   → Variability factors (characterize the domain of reuse)



# Mapping variability factors



# Mapping variability factors



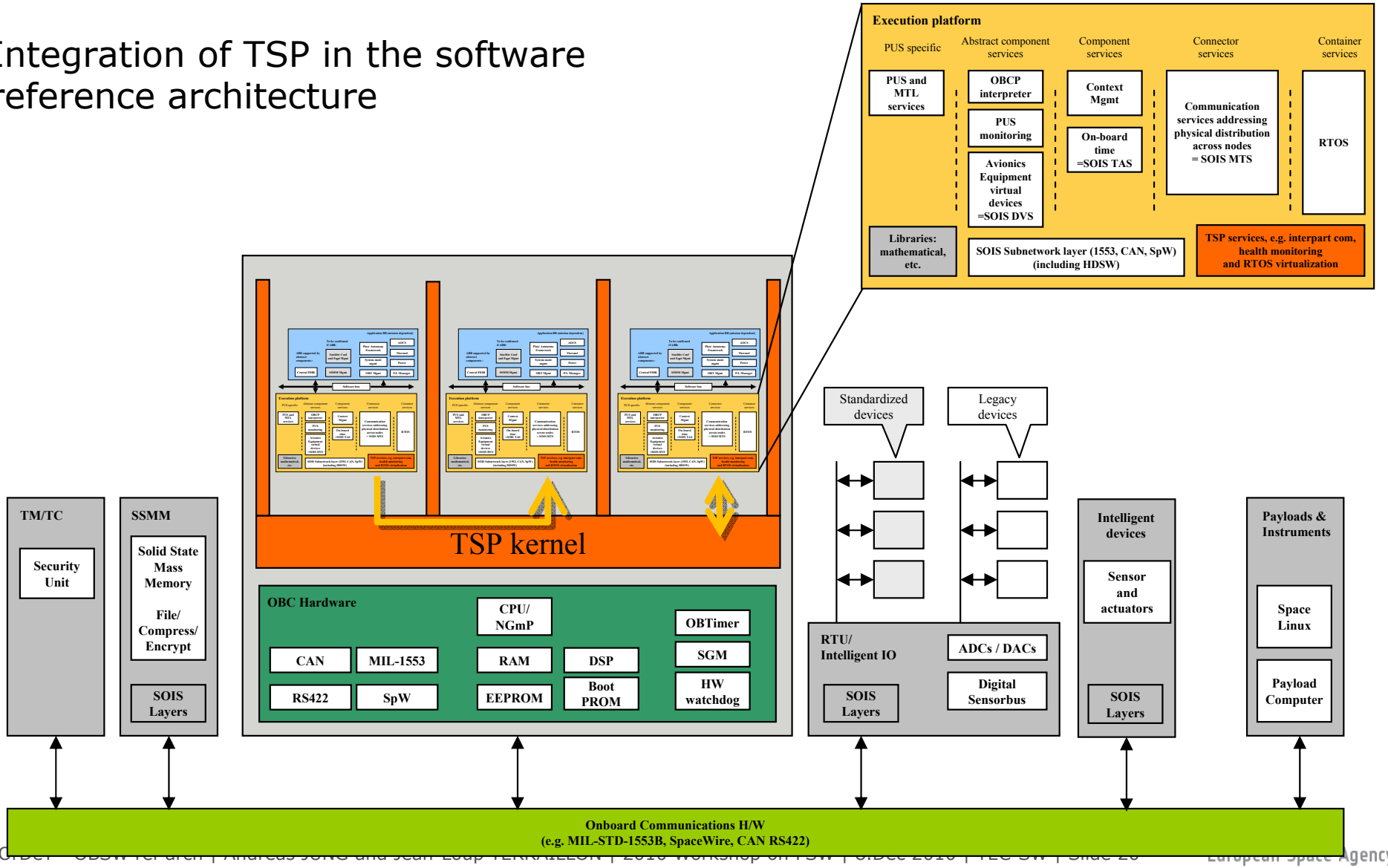
# From Blocks to Building blocks

Based on the **SAVOIR definition**, a **building block**:

1. Has a *clear, open, well-defined, specified, documented* function and interfaces
2. Is *worth developing*, i.e. utilization is envisaged at least for the bulk of the ESA missions
3. *Meets* defined performance, operation and other *requirements*
4. Is *self-contained* so as to be compatible with utilization at higher integration levels, e.g. board, equipment, subsystem
5. *Composability and Compositionality* of its properties shall be guaranteed
6. Has a *TRL and quality level* which can be assessed
7. Is applicable in an *envelope* of well defined physical and software environment
8. Results from a process that can be *repeated with guarantees*
9. Is *designed for reuse* by different users, in different projects (it may be configurable depending on the variability factors)
10. Can be made *available off-the-shelf, under defined conditions*

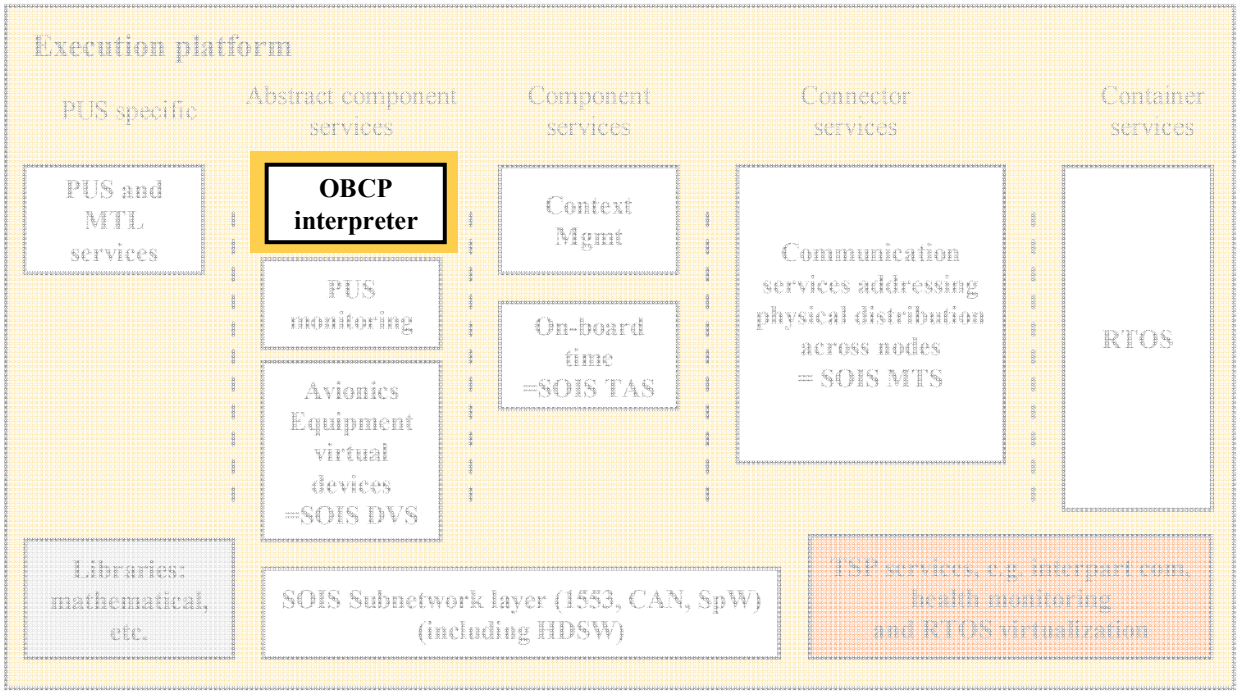
# Complementary activities: 1st: Time and space partitioning

Integration of TSP in the software reference architecture

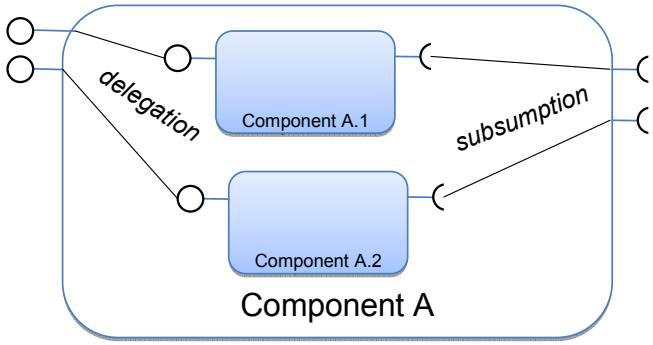


# Complementary activities: 2nd: On-board control procedures

On-board control procedures (**OBCP**) interpreter is part is of the execution platform.



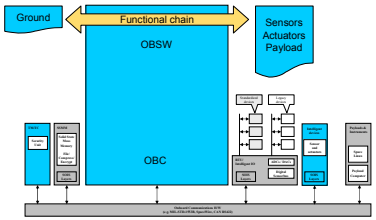
# Open points for the software reference architecture

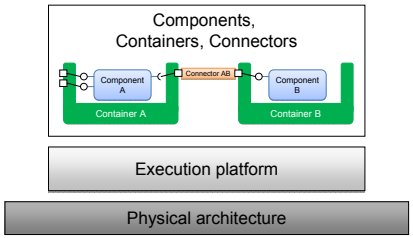



- **Hierarchical** components
- **Architectural decisions**:
  - Fault Detection Isolation Recovery
  - Monitoring, On Board Control Procedure interpreter: common mechanisms in several components
- A **new validation process** also reusable; validate functional and non functional separately; reuse tests suites
- **Methods and tools**:
  - UML? HRT-UML? CHESS? Marte? TopCased/Opees? Spacify? LightWeightCCM? AADL? Metamodel editor generator? Domain Specific Language? Profile or metamodel? …

# Summary

| **Reference architecture** | = mapping of | **Functional chains & Variability factors** | on to | **Software Architectural concepts** |
|---|---|---|---|---|




COrDeT – OBSW reference architecture:

- Software architectural concept
  → Component based software engineering
- Functional chains & variability factors
  → **(building) block & interfaces**

# THANK YOU

Andreas Jung and Jean-Loup Terraillon
European Space Agency
Andreas.Jung@esa.int
Jean-Loup.Terraillon@esa.int