TSP architectures for OBSW in CNES

Julien GALIZZI

19/10/2016

SOMMAIRE

- LVCUGEN : origins
- IMA and Time and Space Partitioning
- LVCUGEN components : features and maturity
- Process
- Reuse use cases : Mass Memory computer on MERLIN, MAJIS on JUICE
- Fully-centralized OBSW : OBC on nanosatellites
- Avionics architecture optimization use cases : ECLAIRs computer on SVOM
- Communalisation of SW components for different users of a same HW : ECLAIRS and MXT on SVOM.
- Future Boot SW ?

INTRODUCTION : LVCUGEN origins

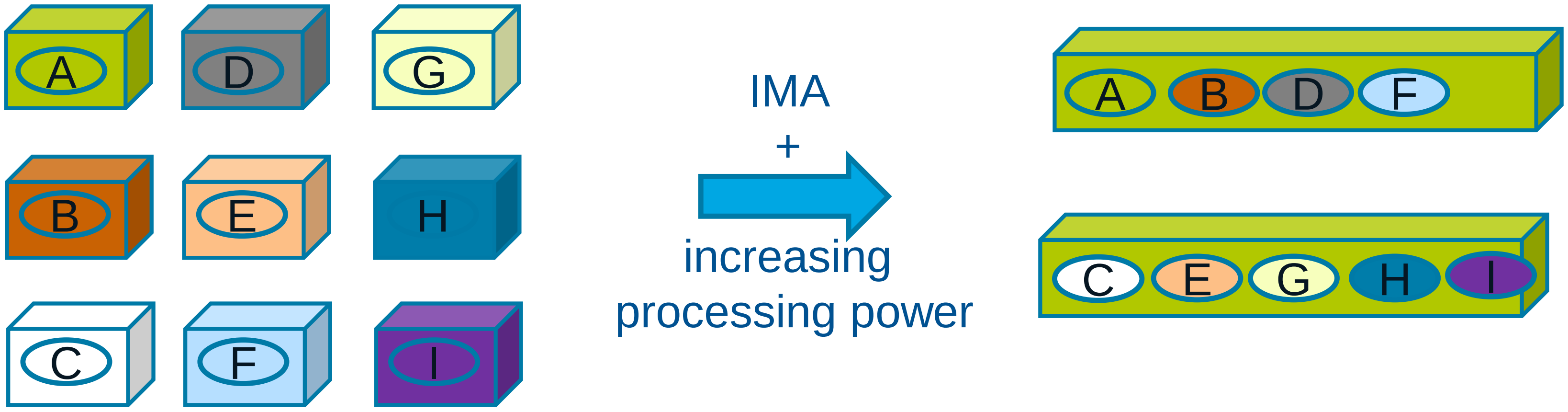
CNES is in charge of securing the developments of French scientific institutes.

- Growing complexity / autonomy of payloads : more and more on-board data processing
- Large variety of hardware architectures depending on mission needs
- Payload / instrument software are developed from scratch in the scope of each mission.
- National (french) Scientific institutes are not well experienced in real time software engineering and space environment and constraints (reliability, availability, engineering standards E40/Q80, monitoring and control).
- Standardisation of monitoring and control (PUS) applicable down to payload / instrument software…
- Cost pressure… Efficiency objectives higher and higher !

=> Always increasing efforts for process and recurrent functions prevent institutes from focusing on their core added value : science !

LVCUGEN – technological context (1) : IMA

- IMA : concept developed by the aeronautical domain and deployed with Airbus and Boeing airplanes since A380 and B777.



IMA
+
increasing processing power

9 SW functions on
9 different computers

9 SW functions on
2 identical computers

- Basics Principles :
  - ✦Capture the needs (in CPU power and I/O) from all the functions of the aircraft
  - ✦Select the appropriate number of computers and switches allowing to cover all the needs
  - ✦Distribute all the functions across the computers
  - ✦Configure the newtork

LVCUGEN – technological context (2) : IMA

- But IMA could be possible only thanks to strong design properties :
  - A deterministic and configurable network (AFDX)
  - An on-board software architecture allowing to guarantee :
    - Strong segregation between applications hosted by the same computer (confinment of anomalies during software execution)
    - An independant development and validation between the different hosted applications (potentially qualified by different suppliers)
  - An architect / integrator in charge of managing all the rationalization :
    - Capture the needs (in CPU power and I/O) from all the functions of the aircraft
    - Select the appropriate number of computers and switches allowing to cover all the needs
    - Configure the distribution of all the functions across the computers
    - Configure the newtork

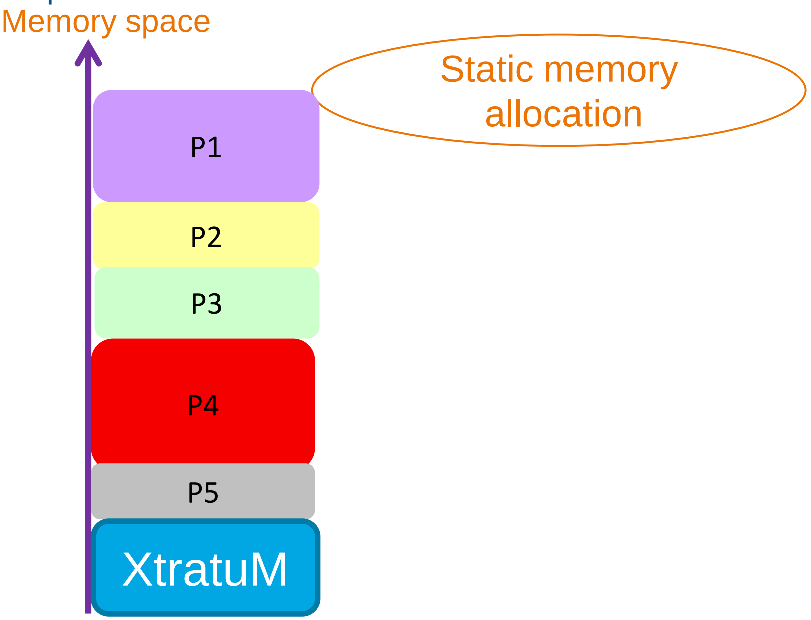
LVCUGEN – technological context (3) : IMA

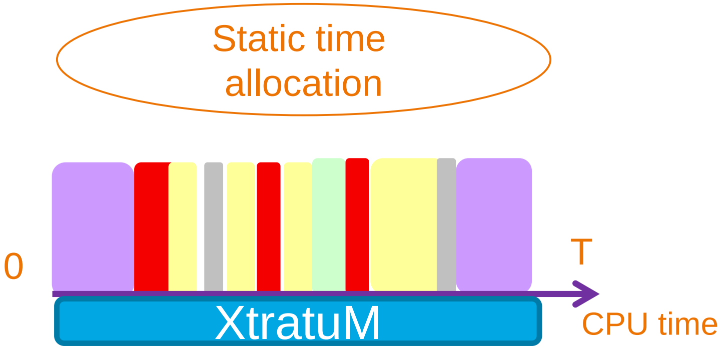
- Benefits :
  - ✦Reduction of the number of HW to qualify (+ SW execution platform)
  - ✦Very fast reliability of the HW (+ SW execution platform) used (intensively used in many different ways by different suppliers).
  - ✦Capability to decrease the quality assurance level of software applications that have a lower criticality rather than always apply to all of them the worst criticality of the computer.
  - ✦Capability to host several SW functions, potentially from different suppliers on the same processor/SoC, while keeping an independent qualification process between these functions.

=> reduction of weight, volume and consumption of electronics

What is TSP ?

- TSP = Time and Space Partitioning
  - Within a given computer, each application software is called a partition and has its own private memory space.
  - Each application software also has dedicated time slots allocated by the RTOS/hypervisor in a predefined scheduling plan. Preemptions or dynamic modifications of the scheduling plans are not possible.
  - All the accesses to HW shared resources (timers, registers, …) are virtualized to avoid conflicts between partitions.





- With such properties, it becomes possible to develop and qualify the applications independently from each other because none of them can modify the execution context of the others.

TSP – advantages and drawbacks

Advantages :

- Allows an independent development and validation of the different partitions (absence of non functional side effects).
- Ease the reuse of recurrent functions.
- Ease the performances analysis (WCET and reactivity) because much less preemptions by other functions (and less functions per partition).
- Capability to decrease the quality assurance level of software applications that have a lower criticality rather than always apply to all of them the worst criticality of the computer.

Drawbacks :

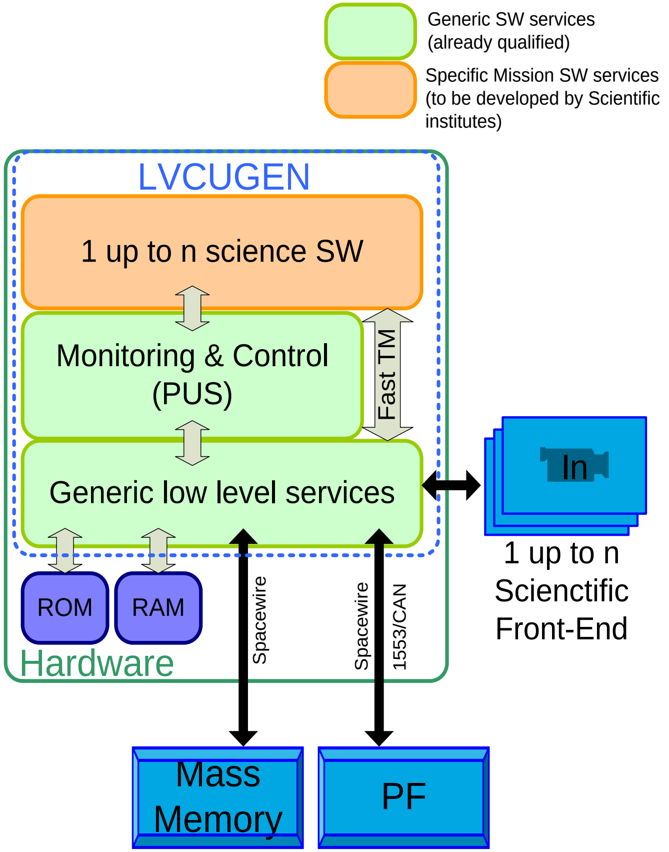
- CPU and memory overhead (but acceptable for space OBSW projects)
- The CPU time that is not consumed by one partition can not be used by the others.
- Incompatible with strong response time (< 10ms) required by HW.

⚠ Crucial role of the resource allocation to guarantee the coverage of Project requirements without impacting the qualification context of the other partitions.

LVCUGEN objectives

Objectives :

- Develop a generic software solution (LVCUGEN = LV Charge Utile GENerique) : infrastructure + basic building blocks
- Transfer to a wide range of users : industry, institutes.

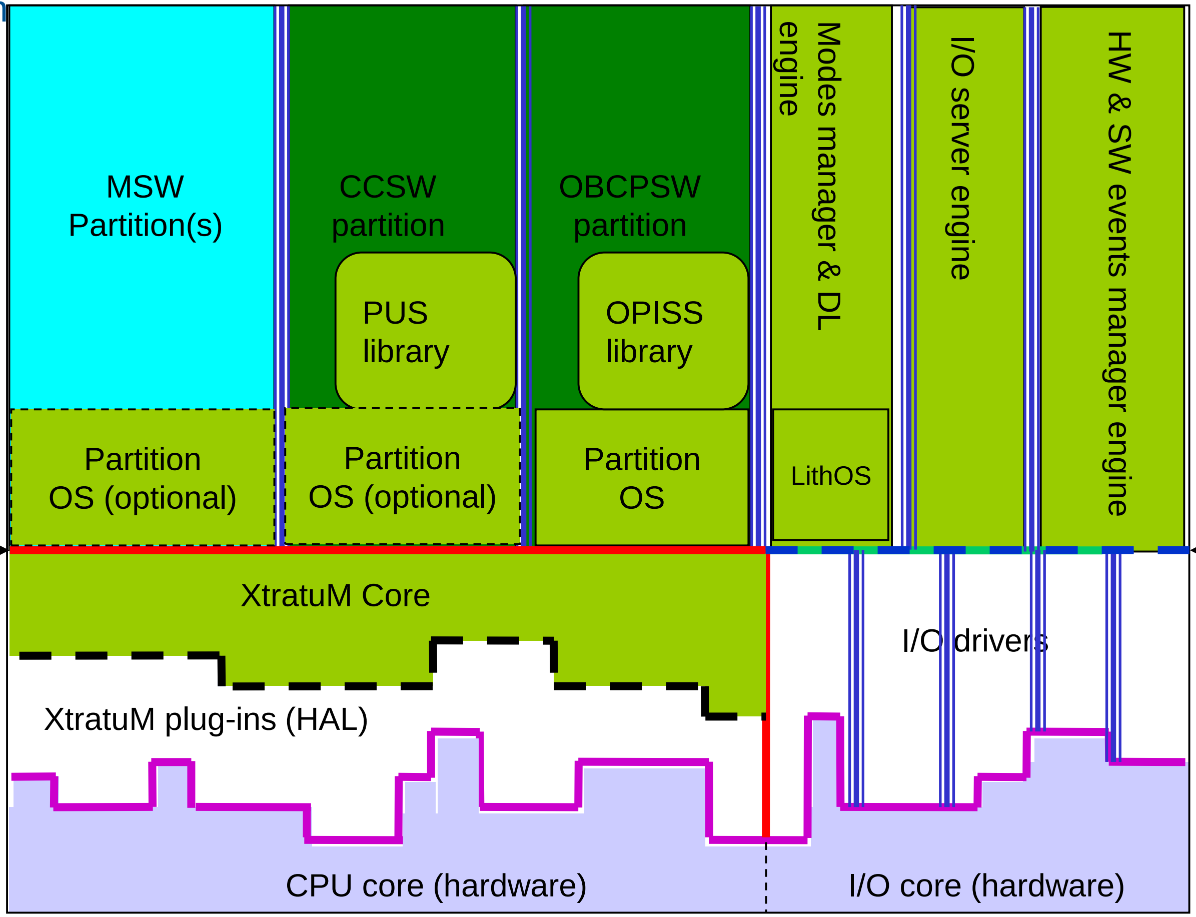


Structuring requirements :

- Use and deploy PUS standard
- Cover a wide diversity of computers while minimizing adaptation efforts.
- Use TSP properties to strictly segregate data handling (generic recurrent functions) and applicative processing (specific to computer)
- Develop and validate once for all generic building blocks complying to the most dimensioning applicable standards (E40/Q80 level B).

OBSW building blocks for payloads : LVCUGEN

🟦	Science specific
🟩	M&C – mission specific
🟩	Generic code
□	HW specific code
🟪	HW – mission specific



Standardised API #1
(SW/SW interface)

Standardised API #2
(SW/SW interface)

BSW

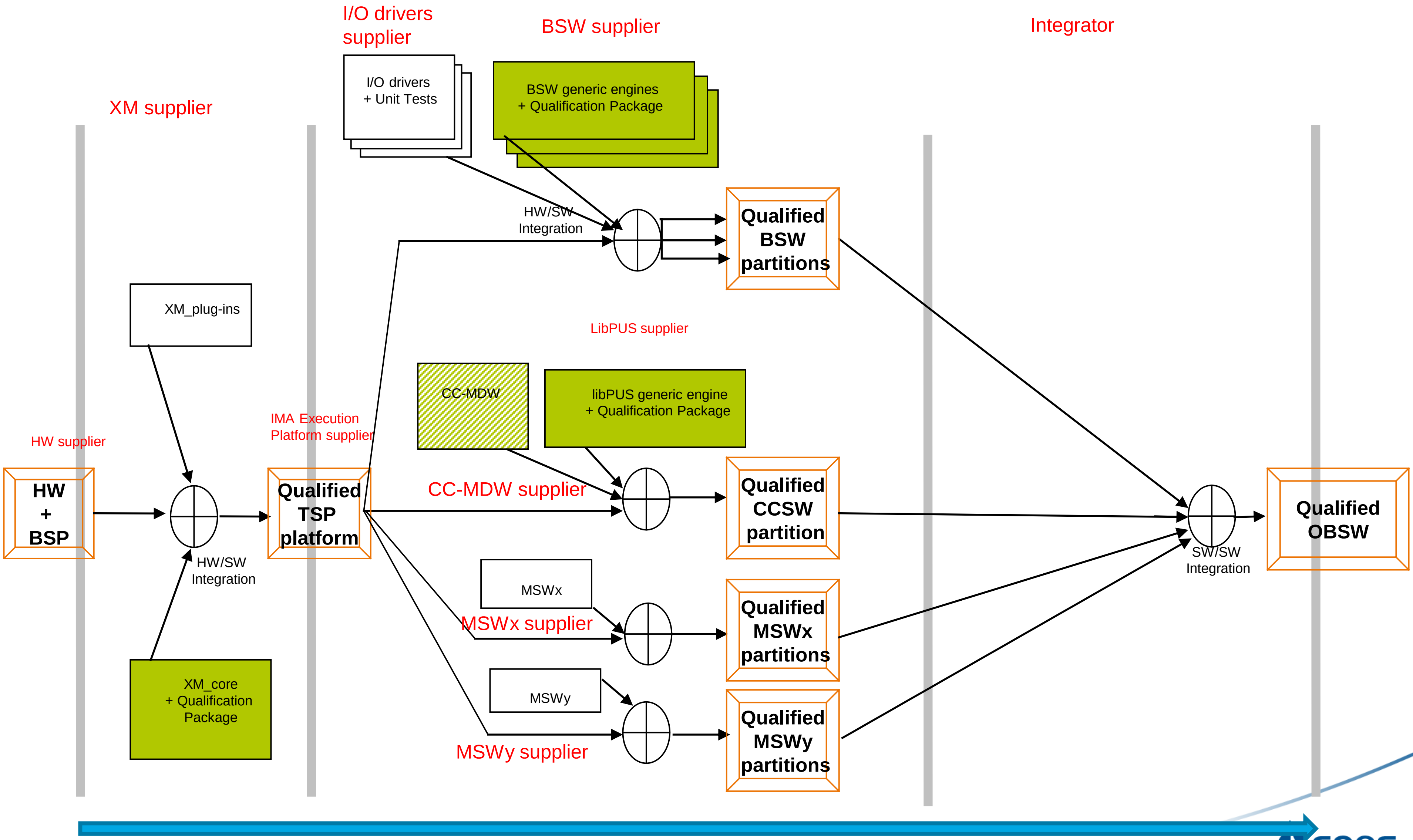
- BSW basic principles :
  - Mutualise and make generic the recurrent low-level services of Payloads Software.
  - Hide the I/O handling to applicative functions and isolate the HW-dependant parts.

- Initially 5 generic partitions were planned:
  - IOServer in charge of shared I/O management – strong QoS.
  - MMDL in charge Modes Management and DataLoading.
  - HSEM in charge of the FDIR at computer level.
  - INSTRUM in charge of instrumentation.
  - MMM in charge of context management.

- INSTRUM and MMM are not yet developed.

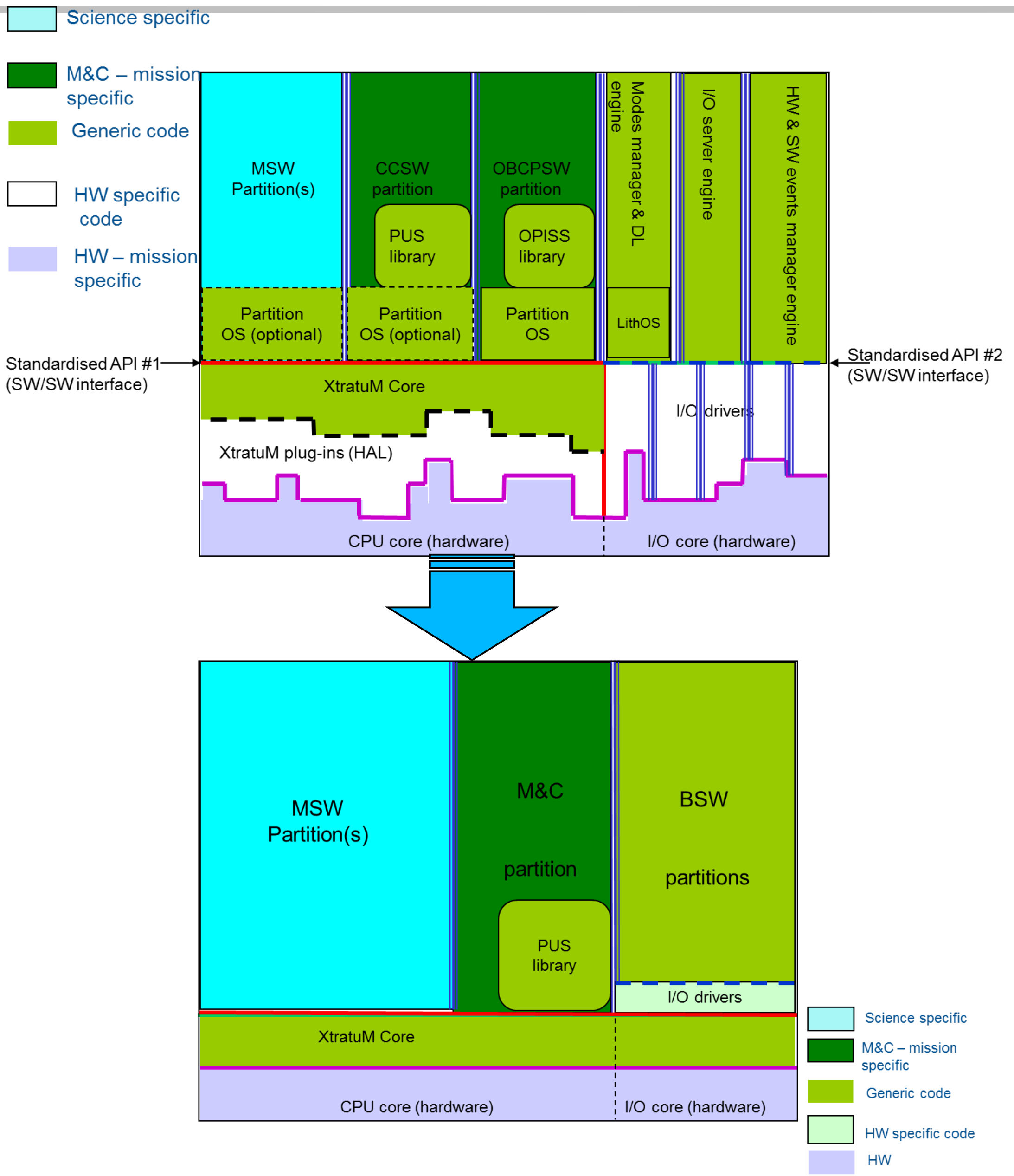
- The 3 other BSW partitions :
  - Are configurable according to the project needs.
  - Are linkable with I/O drivers through a dedicated API.
  - Behave in server mode for client partitions.
  - Are qualified at level B with qualification kits available.

Process



Integrator : TSP and BSW configuration management

LVCUGEN – simplified view



Science specific

M&C – mission specific

Generic code

HW specific code

HW – mission specific

MSW Partition(s)

CCSW partition

PUS library

OBCPSW partition

OPISS library

Modes manager & DL engine

I/O server engine

HW & SW events manager engine

Partition OS (optional)

Partition OS (optional)

Partition OS

LithOS

Standardised API #1 (SW/SW interface)

Standardised API #2 (SW/SW interface)

XtratuM Core

XtratuM plug-ins (HAL)

I/O drivers

CPU core (hardware)

I/O core (hardware)

MSW Partition(s)

M&C partition

PUS library

BSW partitions

I/O drivers

XtratuM Core

CPU core (hardware)

I/O core (hardware)

Science specific

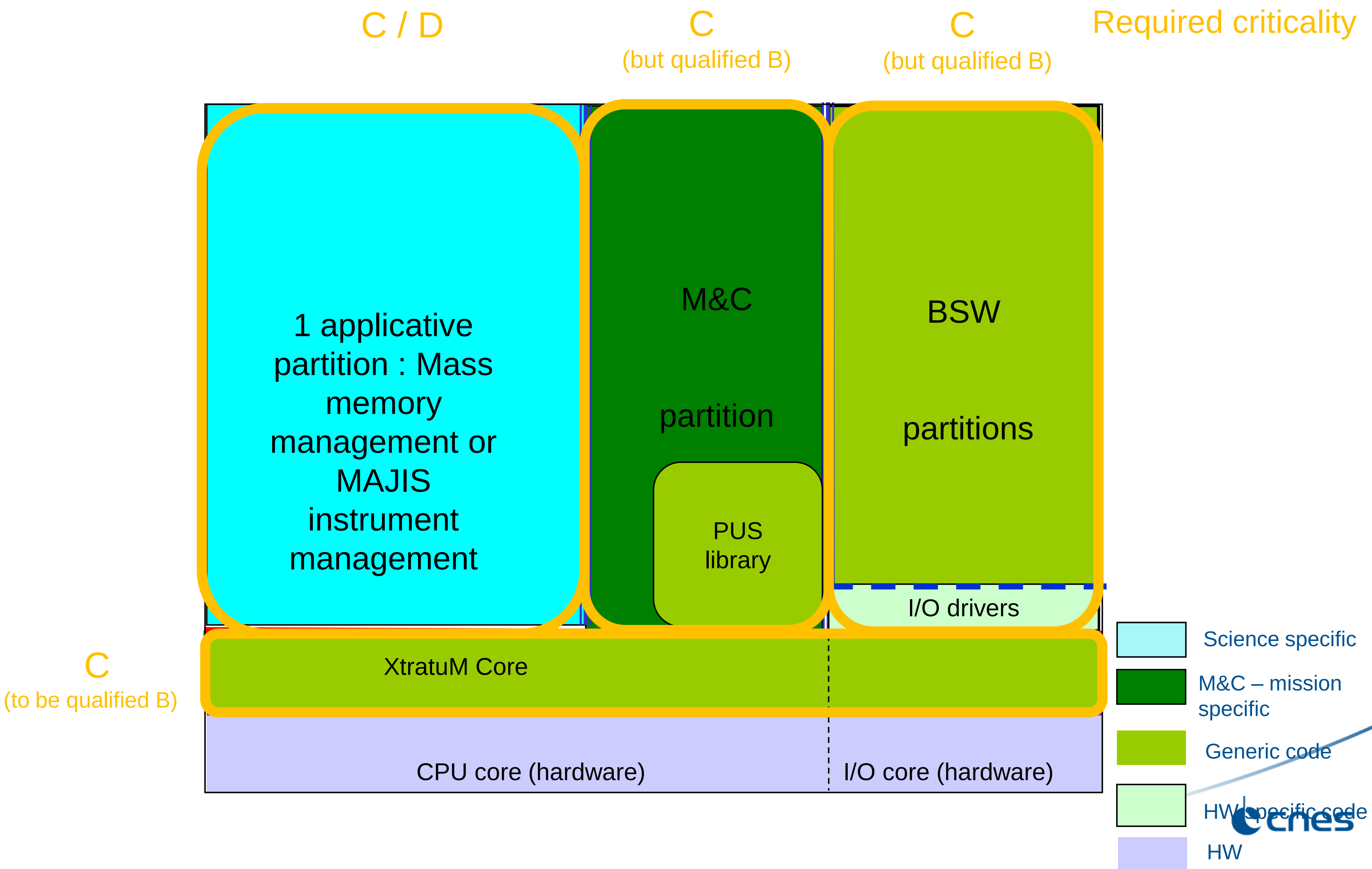
M&C – mission specific

Generic code

HW specific code

HW

Reuse use cases : Mass Memory computer on MERLIN, MAJIS on JUICE



C / D

C
(but qualified B)

C
(but qualified B)

Required criticality

1 applicative partition : Mass memory management or MAJIS instrument management

M&C

partition

PUS library

BSW

partitions

I/O drivers

C
(to be qualified B)

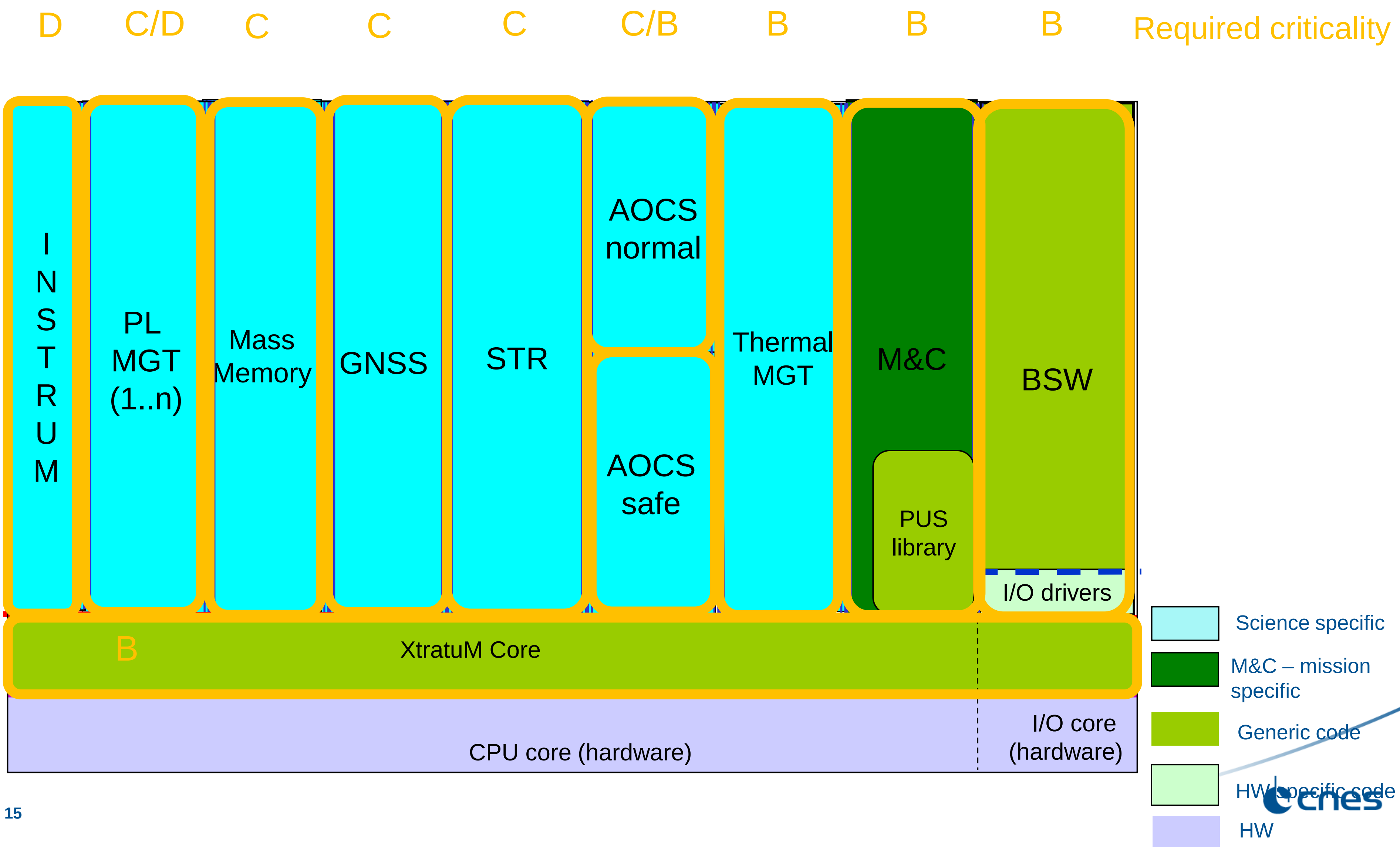
XtratuM Core

CPU core (hardware)

I/O core (hardware)

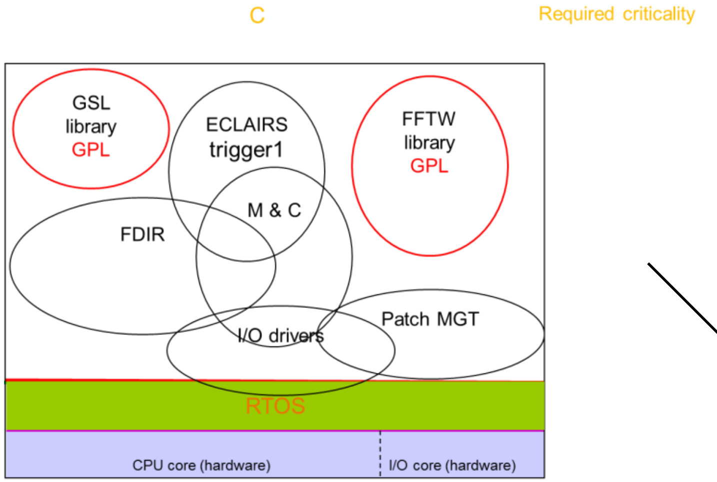
- Science specific
- M&C – mission specific
- Generic code
- HW specific code
- HW

Fully-centralized OBSW use cases : nanosatellites project

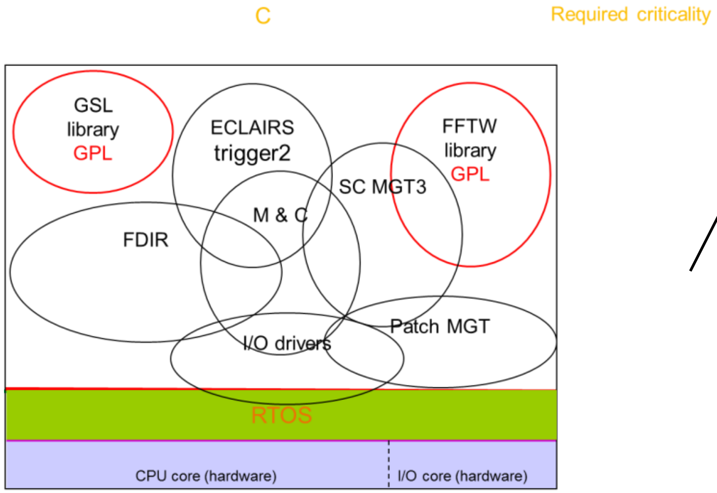


Optimization of avionics architecture use cases : ECLAIRS instrument on SVOM project

Initial baseline
2 SC functions : 1 computer for each

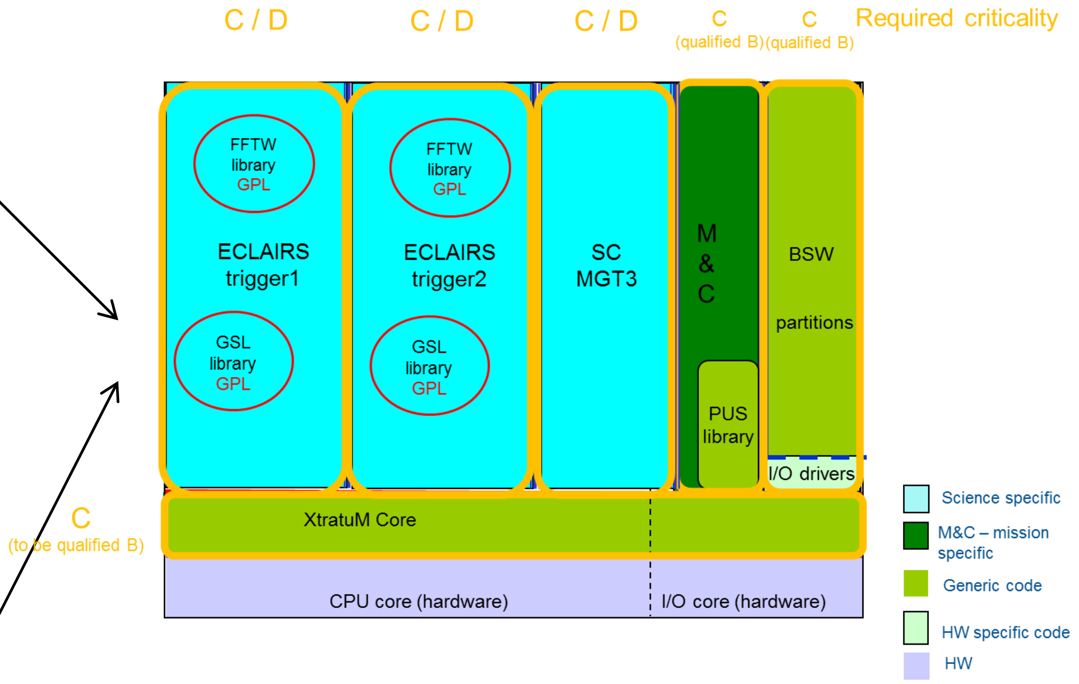


LEON2@100MHz



LEON2@100MHz

Mass / consumption constraints
=> 2 SC functions : 1 computer for all



GR712 : 2 cores LEON3@80MHz

Communalisation use cases : MXT and ECLAIRS instruments on SVOM

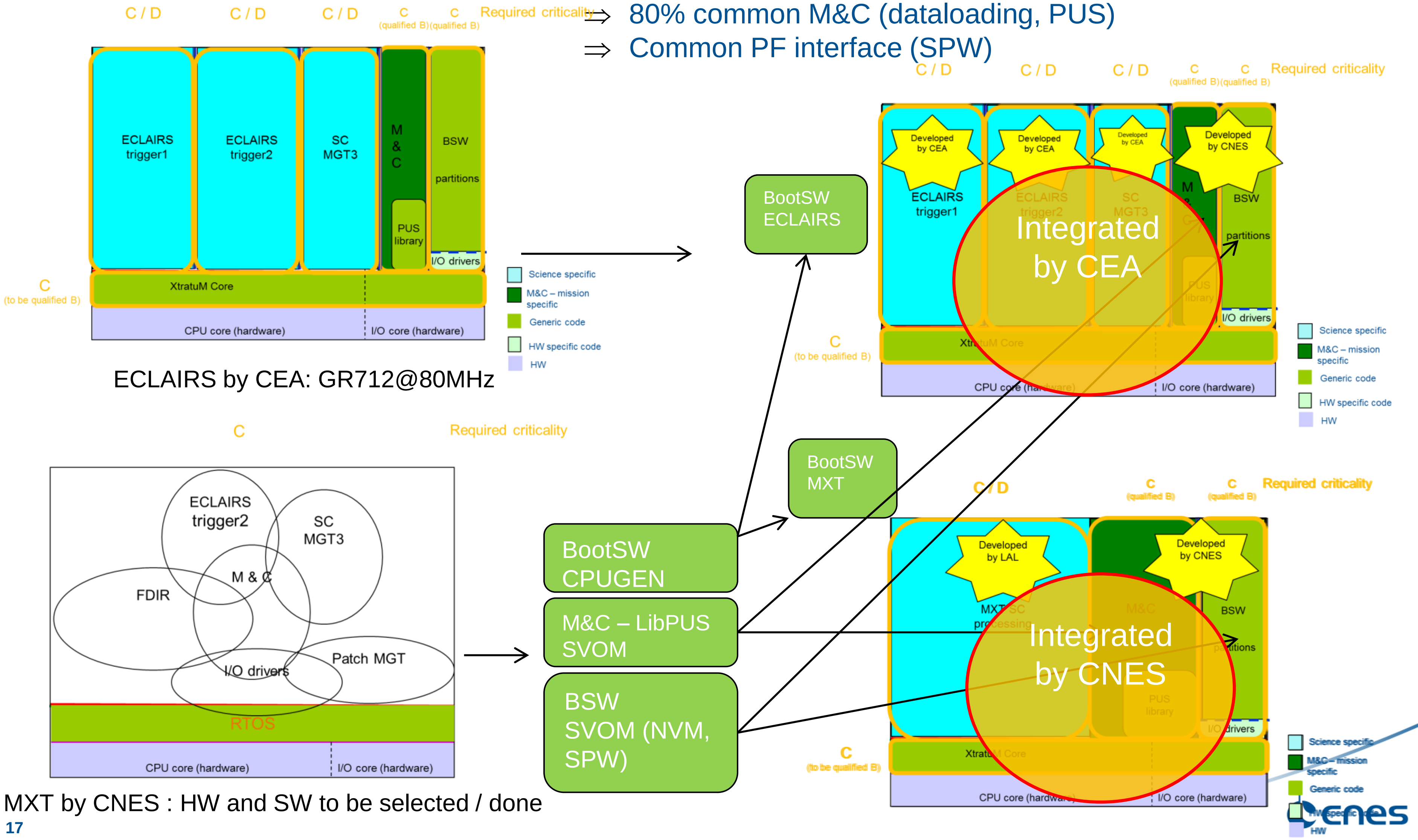
Initial baseline :
2 different SC instruments

Same HW selected : CPUGEN (GR712, SPW, MRAM, FPGA)

⇒ common BootSW

⇒ 80% common M&C (dataloading, PUS)

⇒ Common PF interface (SPW)



ECLAIRS by CEA: GR712@80MHz

MXT by CNES : HW and SW to be selected / done

Communalisation use cases : MXT and ECLAIRS instruments on SVOM

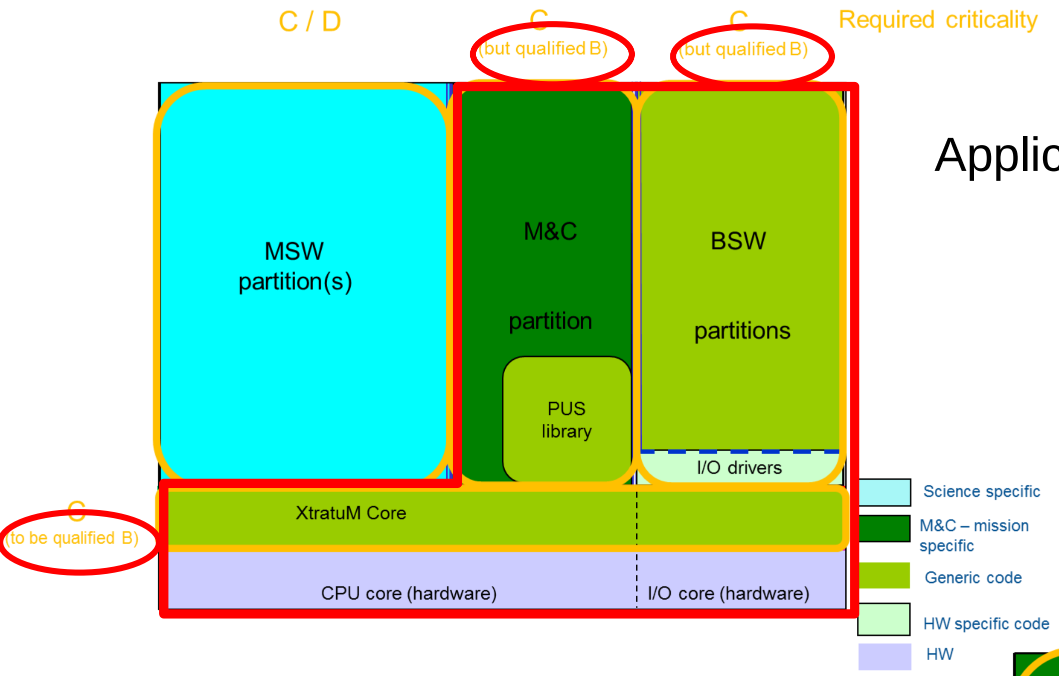
Benefits for the SVOM Project :

- Only 1 HW board to develop for both instruments
- Only 1 BootSW to develop for both instruments
- Only 1 implementation of SVOM PUS services for both instruments
- Only 1 integration / validation of XtratuM on the board for both instruments
- Only 1 development of I/O, NVM drivers and their integration / validation in BSW partitions for both instruments

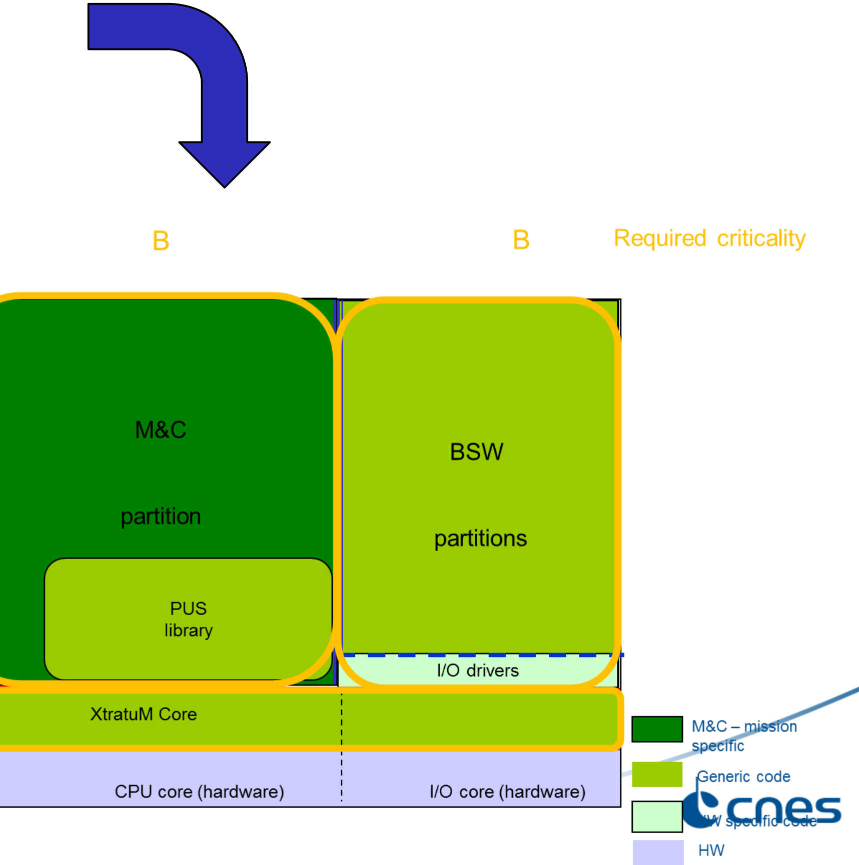
- Problems experienced on one partition are shared with others.
- Parallelization of developments in differents facilities with different actors.

- **Scientifics can focus on science !**

Future BootSW for equipments / Payloads ?



Application SW



A safe mode BootSW is functionnaly « just » a subset of the ApplicationSW…

and this subset will be (soon) qualified at level B…

Conclusion

- TSP is a powerful tool.

- CNES currently uses it to :
  - Promote and ease reuse
  - Reduce costs
  - Parallelize developments with different SW suppliers
  - Avoid contamination from GPL libraries to all the OBSW
  - Optimize avionics architecture

Thanks for your attention…

Any question ?

?

?

?

?