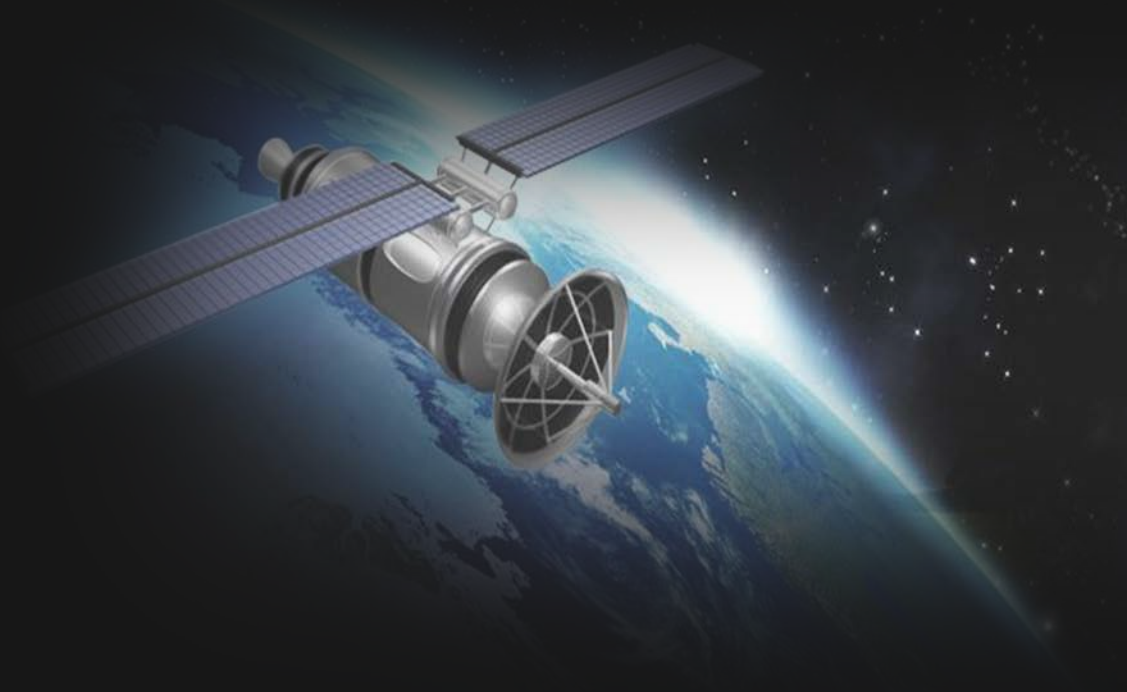


Reaching into space
TOGETHER

PARTITIONING AND MAINTENANCE OF FLIGHT SOFTWARE IN INTEGRATED MODULAR AVIONICS FOR SPACE

J. Hardy, M. Hiller, P. Creten

21st May 2014 – ESA ESTEC

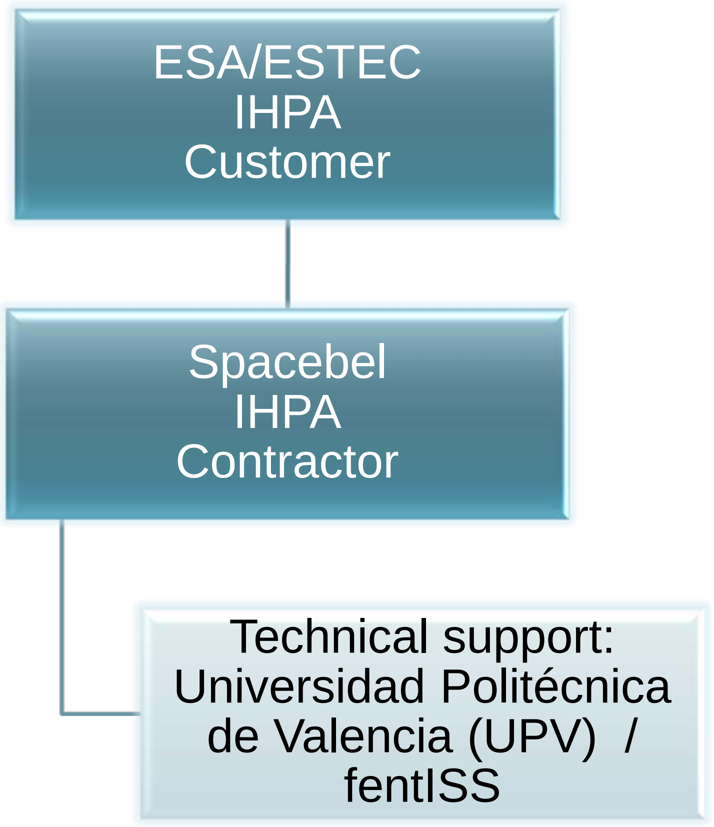
Agenda

- Study Objectives
- Organisation
- Study Logic
- Development Process
- Partitioning Strategy of the flight software (FSW)
- Boot Strategy and OBSW Maintenance
- Assessment, Lessons Learned and Conclusion

Objectives

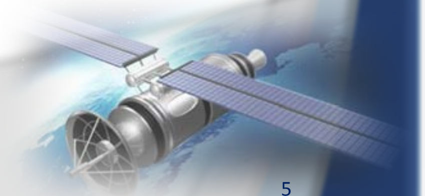
1. Demonstrate how the FSW of a small spacecraft can be partitioned.
2. Demonstrate and assess off-line integration of different applications in a partitioned execution environment.
3. Demonstrate and assess upload and on-line integration of new and/or updated applications in a partitioned execution environment.
4. Demonstrate and assess software development process and roles across different software teams.

Organisation



Study Logic - Scope

- Updating the PROBA-2 SVF based on a LEON2 to LEON3 FT Target Simulator with MMU support
- Partitioning and modifying the PROBA-2 CDMS to execute in this partitioned environment
- Developing and integrating an independent payload ASW1. The integration of a second payload independently developed by an external team could not be performed
- Developing OBSM components and ground tools for this partitioned environment



Study Logic - Deliveries

- SVF: An appropriate simulated and representative execution environment
- FSW: Partitioned flight software executing in this environment
- ASW1: A payload application software integrated in this partitioned flight software
- OBSM: On board Software Maintenance facilities to upload and update such application software

Study Logic - Reuse

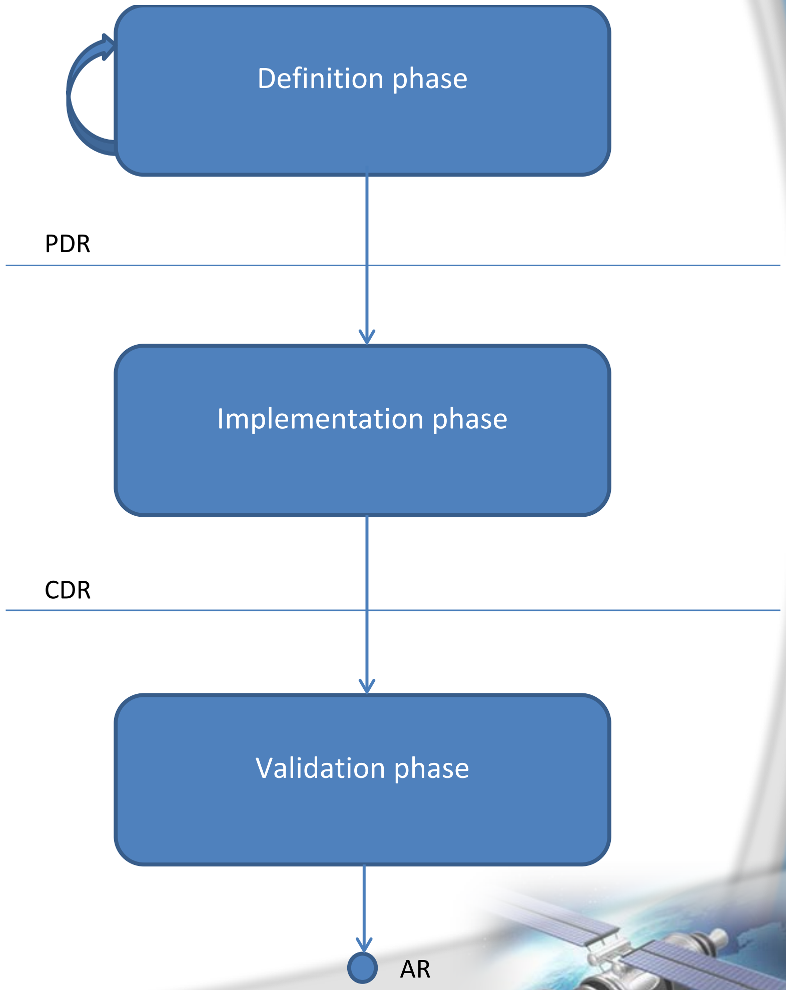
- Reuse of existing components from PROBA-2:
  - The SVF mimics the avionics (includes a LEON3 TS)
  - The FSW consists of the BSW, the DHSW and the AOCS SW
  - The first application software (ASW1) is the SWAP

Study Logic – SW Criticality

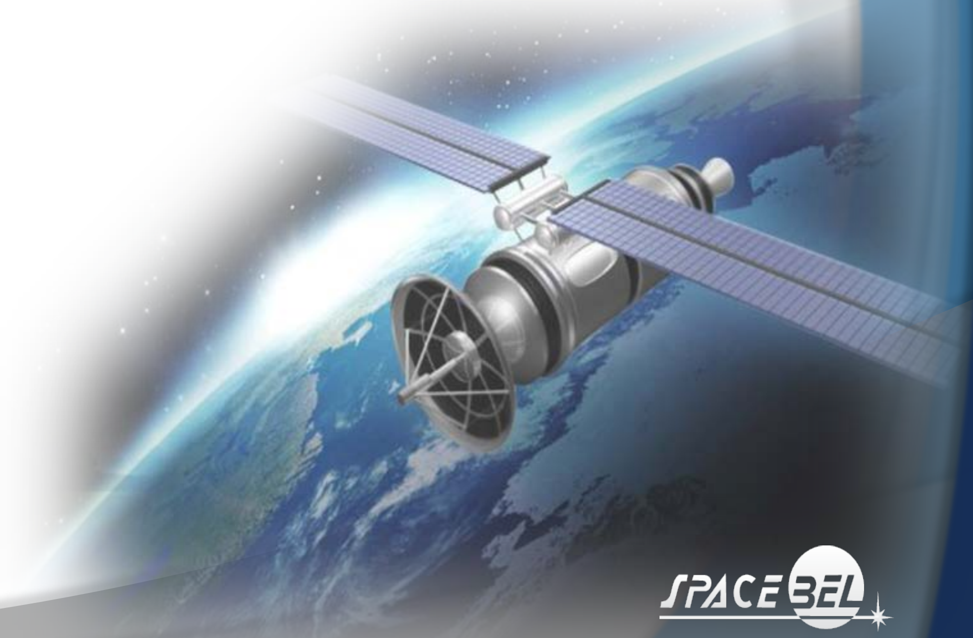
- FSW and ASW: prototype
- OBSM – flight part: level C
- OBSM – ground part: level D

Development Process

- One definition iteration before PDR
- Two development iterations before CDR:
  - First iteration, which executes the complete PROBA-2 OBSW as a single partition
  - Second iteration, with a FSW partition regrouping all PROBA-2 OBSW functions except the SWAP payload application and AOCS (implemented in their own partitions)
- OBSM developed by independent team
- Integration and validation activities were performed at system level



Partitioning & Maintenance details



Partitioning kernel & execution platform

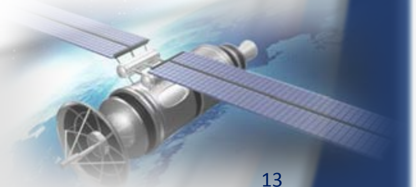
- The partitioning kernel selected for this experiment is XtratuM v3.4.2 (product developed by fentISS)
- The guest OS used in the frame of this study is the Edisoft version of rtems v4.8
- The execution environment is the adapted PROBA-2 Software Validation Facility (SVF)

Partitioning – Iteration 1

- The complete PROBA-2 OBSW in a single partition
- Assumption on the memory map:
  - Virtual memory corresponds to physical memory map (one-to-one mapping)
- No particular issues have been raised during the porting of PROBA-2 Software
- All non-regression test cases (i.e. PROBA-2 validation tests) have been replayed successfully

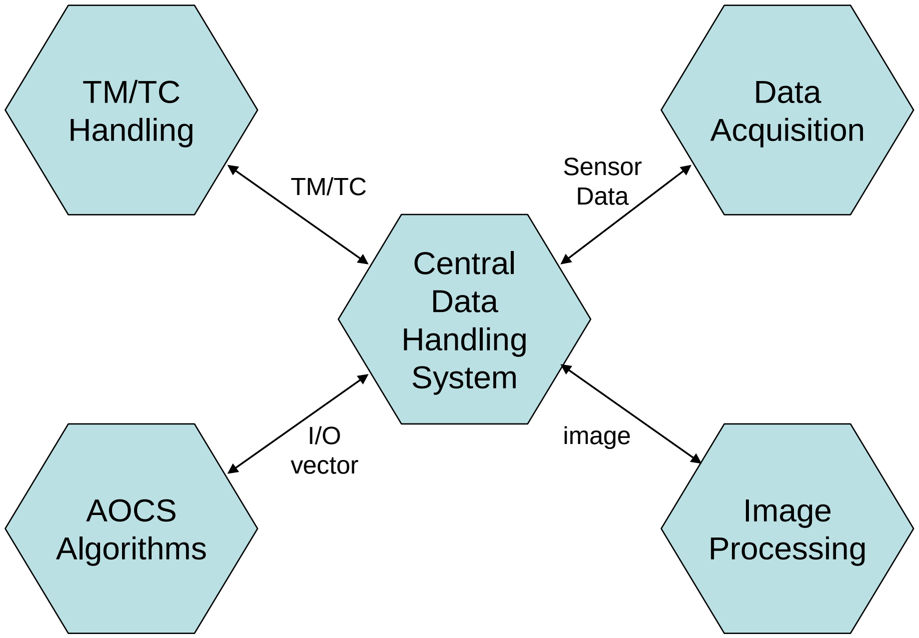
Partitioning – Iteration 2

- PROBA-2 OBSW is divided into several partitions
- The following OBSW functionalities had to be revisited:
    - TM/TC Handling
    - I/O access
    - Scheduling and real-time requirements
    - Interrupts/Traps
    - Memory/Patch management
    - Memory scrubbing
    - Data bank



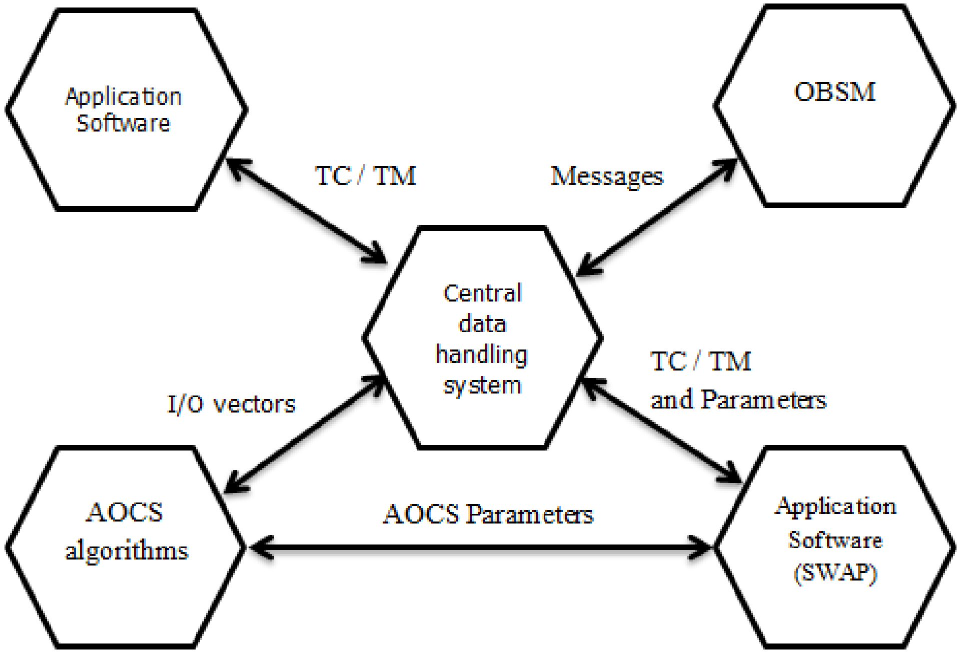
Partitioning – Iteration 2

- Initial partitioning
  - ***TM/TC handling*** responsible for CCSDS TC/TM and RF chain
  - ***Data Acquisition*** responsible for the acquisition of data from various PROBA-2 equipment
  - ***Central Data Handling System*** partition groups central data bank, the PUS services and applications (AOCS units and SWAP instrument)
  - ***AOCS Algorithms*** partition executes the code generated by third party embedded GNC algorithms
  - ***Image processing*** responsible of JPEG/LZW compression of SWAP images



Partitioning – Iteration 2

- Revised and selected partitioning
  - ***Central Data Handling System*** partition groups the **TM/TC Handling** function, **Data Acquisition** function and **Data Handling System**
  - The ***AOCS Algorithms*** partition is exclusively used for the computation of the GNC parameters
  - The ***ASW (SWAP)*** partition, which is the **Image Processing** of SWAP payload
  - The other ***ASW*** partition is a second demonstration application to be integrated within the PROBA-2 software
  - The ***OBSM*** partition, which is responsible for the maintenance of the present system



Boot strategy

Several boot strategies have been envisaged

- Central staged boot
- Distributed boot
- OBSM partition as secondary bootloader

OBSM partition: a privileged partition responsible for the OBSW Maintenance of the IMA-SP system

Primary bootloader: a bootloader that is present in the PROM and which is responsible for deploying the Partitioning Kernel (and possibly for deploying the software image of partitions).

Resident software: primary bootloader of XtratuM

Secondary bootloader: a bootloader that is executed in the frame of a partition and that is responsible for deploying the final software image of the partition (from non-volatile memory) into the partition (virtual) memory

Boot strategy

Several boot strategies have been envisaged

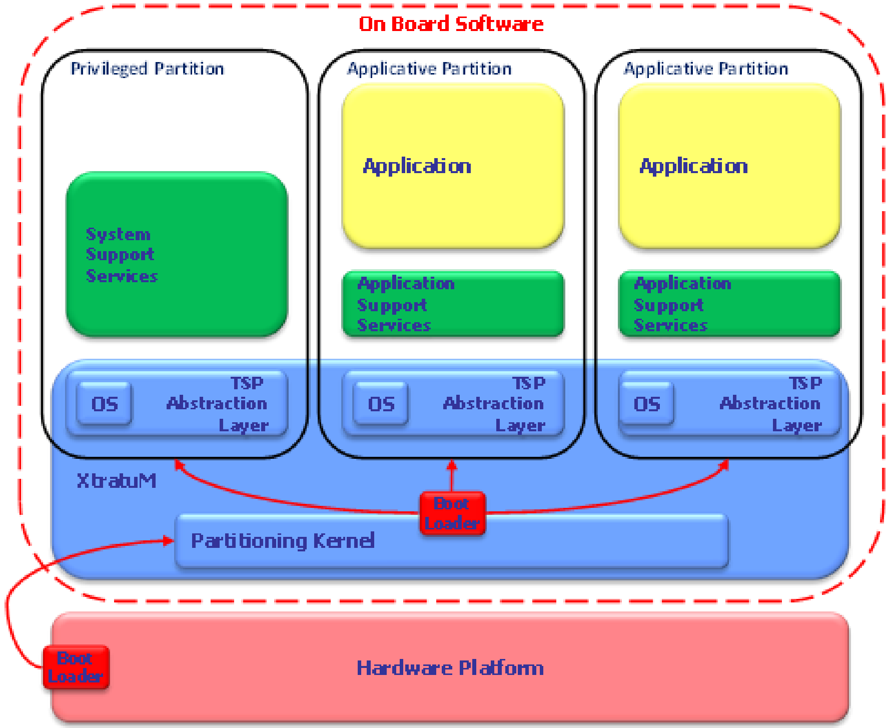
- **Central staged boot**
- Distributed boot
- OBSM partition as secondary bootloader

Pro

- ✓ Modular and independent of partitions
- ✓ Single format for binaries
- ✓ Easy management of non-volatile memories

Con

- Not really a functionality of IMA-SP but rather to space application domain
- Not realistic to modify the Partitioning Kernel (out of the scope of IHPA project)
- Secondary bootloader must be aware of all the settings of the partitions



Boot strategy

Several boot strategies have been envisaged

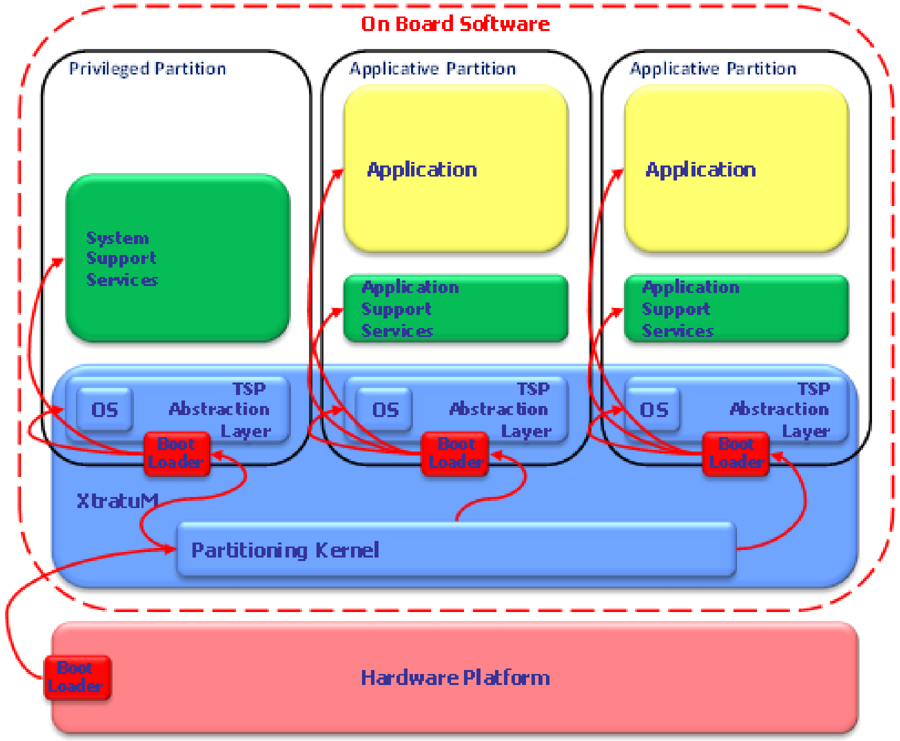
- Central staged boot
- **Distributed boot**
- OBSM partition as secondary bootloader

Pro

- ✓ Each partition can have its own boot and patch strategy.
- ✓ The secondary boot loader is part of the central staged boot. The partitioning kernel remains unchanged.

Con

- Each partition should have an access to the non-volatile memory
- The secondary bootloader is always re-executed when resetting a partition, even if the reload is not required
- The primary bootloader must have a table that indicates where to find the boot information for each partition and for each secondary bootloader
- Since the secondary bootloader is in the same virtual memory as the final application partition image, they should be aware of each other in order not to overwrite themselves while executing



Boot strategy

Several boot strategies have been envisaged

- Central staged boot
- Distributed boot
- **OBSM partition as secondary bootloader**

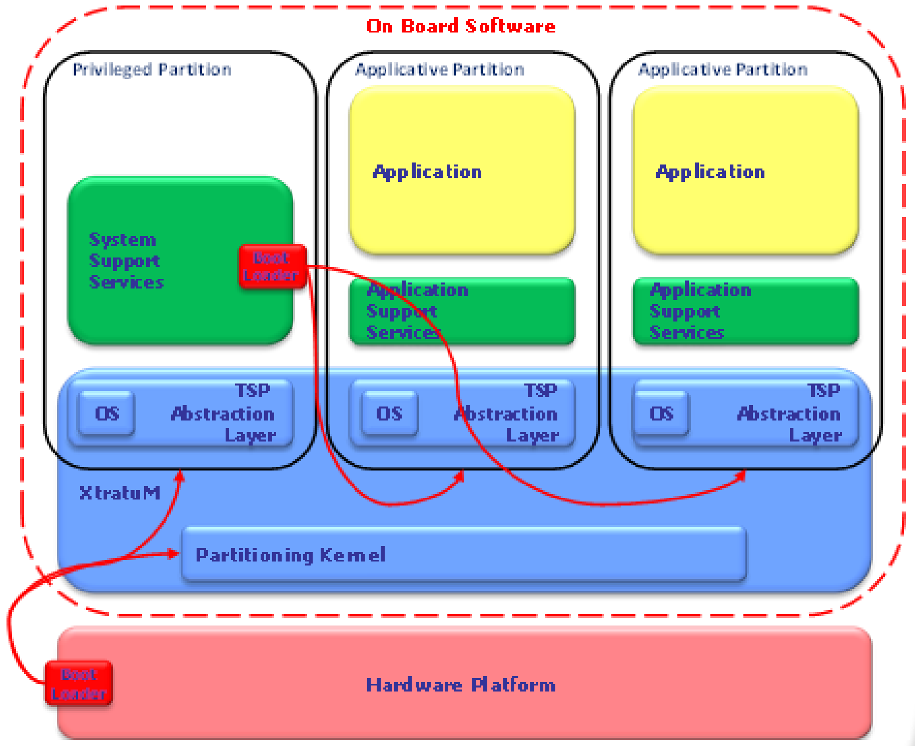
Pro

- ✓ Partitioning kernel remains unaffected by the boot strategy
- ✓ Easy reload and restart of partitions (warm or cold)
- ✓ The boot information is directly available in the privileged partition

Con

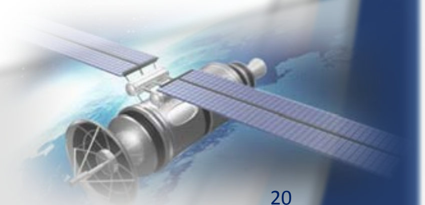
- The reload, patch or maintenance of the privileged partition is not possible. A cold start of the whole system is required (rational for class C software)

Selected Boot strategy



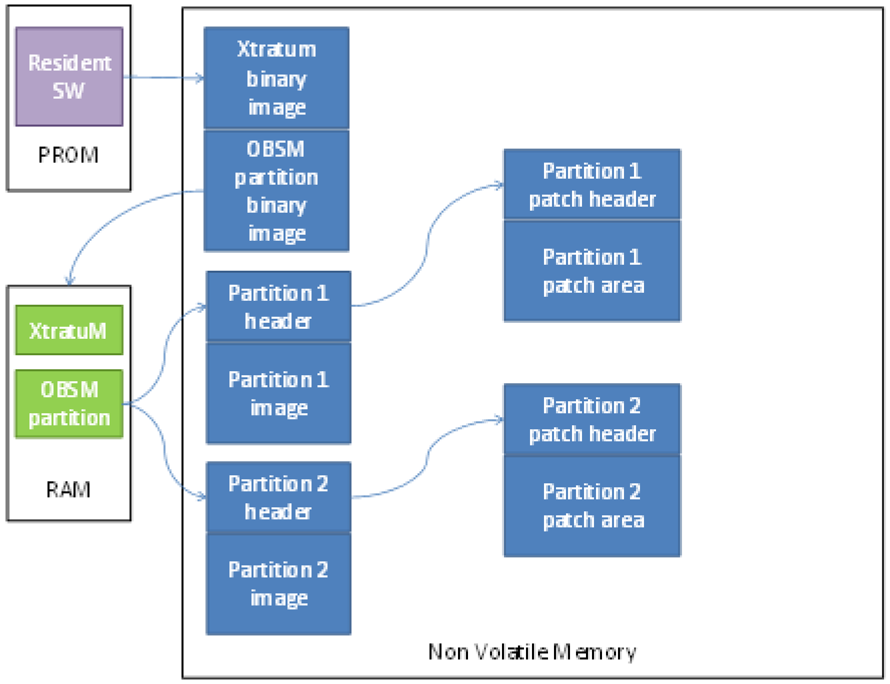
Boot strategy – Trade-off

	Central staged boot	Distributed boot	OBSM partition
Modularity	🟩	🟩	🟥
Binary format	🟩	🟥	🟥
Non-volatile memory	🟩	🟥	🟩
Diversity	🟥	🟩	🟥
Modifications	🟥	🟥	🟩
Easy reload and restart of partitions	🟥	🟩	🟩
Availability of boot information	🟥	🟥	🟩
Capability of secondary bootloader	🟩	🟥	🟩
Complexity of primary bootloader	🟥	🟥	🟩



Selected boot strategy

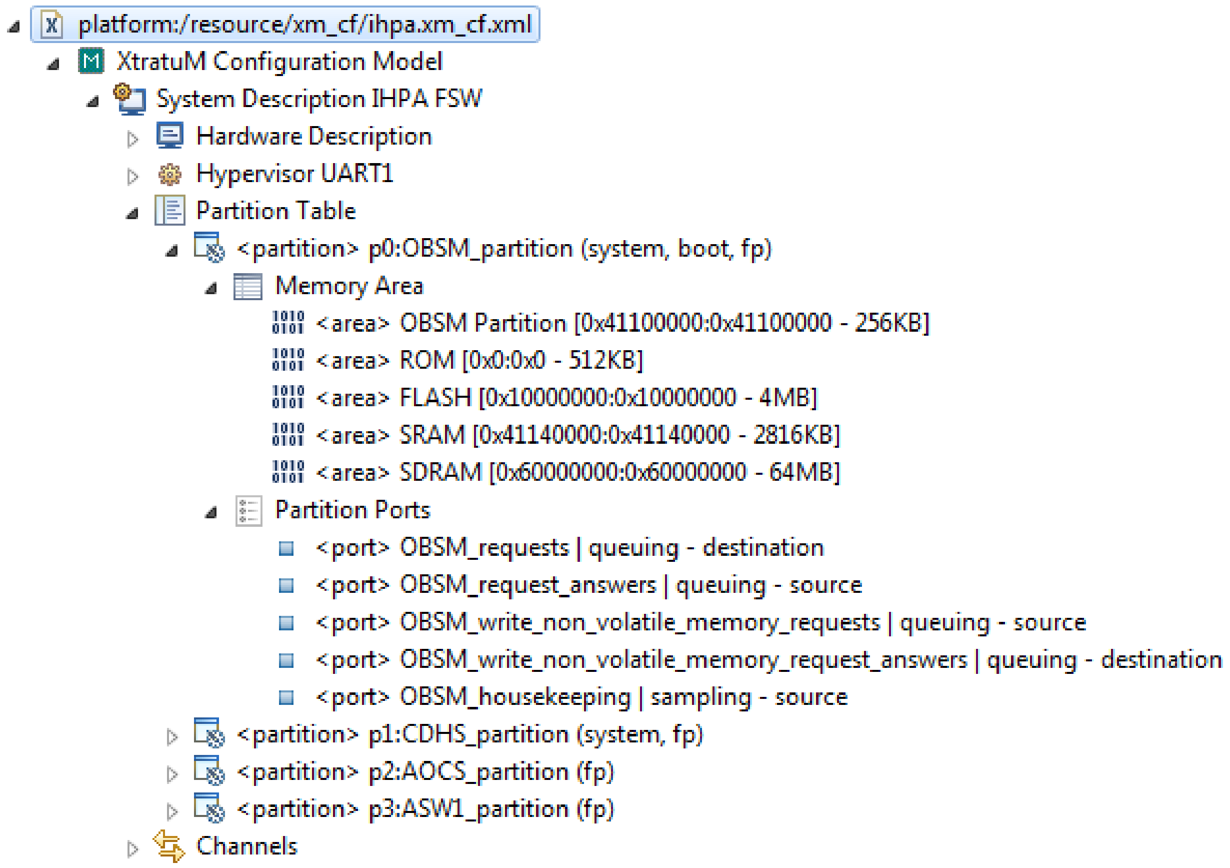
In summary, the primary bootloader is in charge of loading the Partitioning Kernel and of the OBSM partition; the OBSM partition is responsible for loading and starting the remaining partitions.



OBSM partition

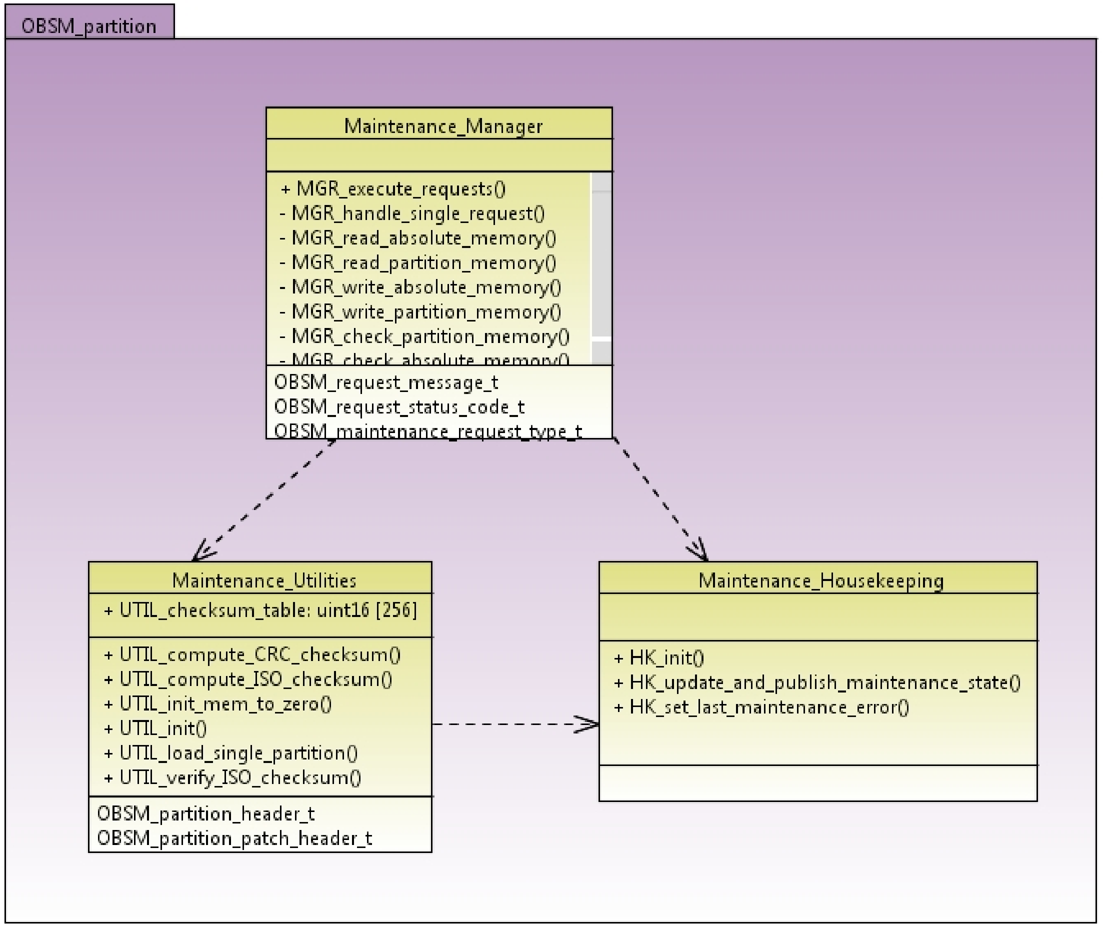
- The current approach was to have one **OBSM partition** that implements both the **bootloader** and the **maintenance services**
- The **OBSM partition** is coupled with a **Ground Tool** in charge of handling user requests corresponding to maintenance of the IMA-SP system and translating them into actual **OBSM partition** function
- As an XtratuM partition in an IMA-SP system, it has a statically allocated time and memory (time and space partitioning). However, due to its role of managing and maintaining the other partitions of the system, it has privileged rights granting access to the complete physical memory (including the virtual memory of other partitions and XtratuM own memory) and to the complete XtratuM API
- The OBSM partition does not have any guest-OS.
- The OBSM is portable in the sense that it has no dependencies with the partitioning kernel

OBSM partition – Setup & Interfaces



```
◢ platform:/resource/xm_cf/ihpa.xm_cf.xml
  ◢ XtratuM Configuration Model
    ◢ System Description IHPA FSW
      ▷ Hardware Description
      ▷ Hypervisor UART1
      ◢ Partition Table
        ◢ <partition> p0:OBSM_partition (system, boot, fp)
          ◢ Memory Area
              <area> OBSM Partition [0x41100000:0x41100000 - 256KB]
              <area> ROM [0x0:0x0 - 512KB]
              <area> FLASH [0x10000000:0x10000000 - 4MB]
              <area> SRAM [0x41140000:0x41140000 - 2816KB]
              <area> SDRAM [0x60000000:0x60000000 - 64MB]
          ◢ Partition Ports
              <port> OBSM_requests | queuing - destination
              <port> OBSM_request_answers | queuing - source
              <port> OBSM_write_non_volatile_memory_requests | queuing - source
              <port> OBSM_write_non_volatile_memory_request_answers | queuing - destination
              <port> OBSM_housekeeping | sampling - source
        ▷ <partition> p1:CDHS_partition (system, fp)
        ▷ <partition> p2:AOCS_partition (fp)
        ▷ <partition> p3:ASW1_partition (fp)
      ▷ Channels
```

OBSM partition - Design


OBSM_partition
Maintenance_Manager
+ MGR_execute_requests()
- MGR_handle_single_request()
- MGR_read_absolute_memory()
- MGR_read_partition_memory()
- MGR_write_absolute_memory()
- MGR_write_partition_memory()
- MGR_check_partition_memory()
- MGR_check_absolute_memory()
OBSM_request_message_t
OBSM_request_status_code_t
OBSM_maintenance_request_type_t
Maintenance_Utilities
+ UTIL_checksum_table: uint16 [256]
+ UTIL_compute_CRC_checksum()
+ UTIL_compute_ISO_checksum()
+ UTIL_init_mem_to_zero()
+ UTIL_init()
+ UTIL_load_single_partition()
+ UTIL_verify_ISO_checksum()
OBSM_partition_header_t
OBSM_partition_patch_header_t
Maintenance_Housekeeping
+ HK_init()
+ HK_update_and_publish_maintenance_state()
+ HK_set_last_maintenance_error()

Ground tool

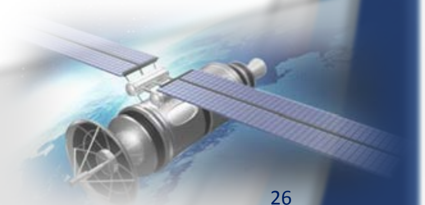
- The **OBSM Ground Tool** is a toolset implemented in Python scripts, XSD/XML but also EMF Editor (Ecore)
- The **OBSM Ground Tool**:
  - ✓ Verifies the coherency of the online IMA-SP system (against versions, checksum, hypervisor version, settings, etc.)
  - ✓ Maintains the IMA-SP system maintenance database with various partition/XtratuM images
  - ✓ Has access to the configuration of the IMA-SP system
  - ✓ Maintains the organization of the non-volatile memory (XtratuM image, partition images, patch locations…)
  - ✓ Auto-generates maintenance commands (new image or deferred patches)
  - ✓ Auto-generates verification commands

Ground tool

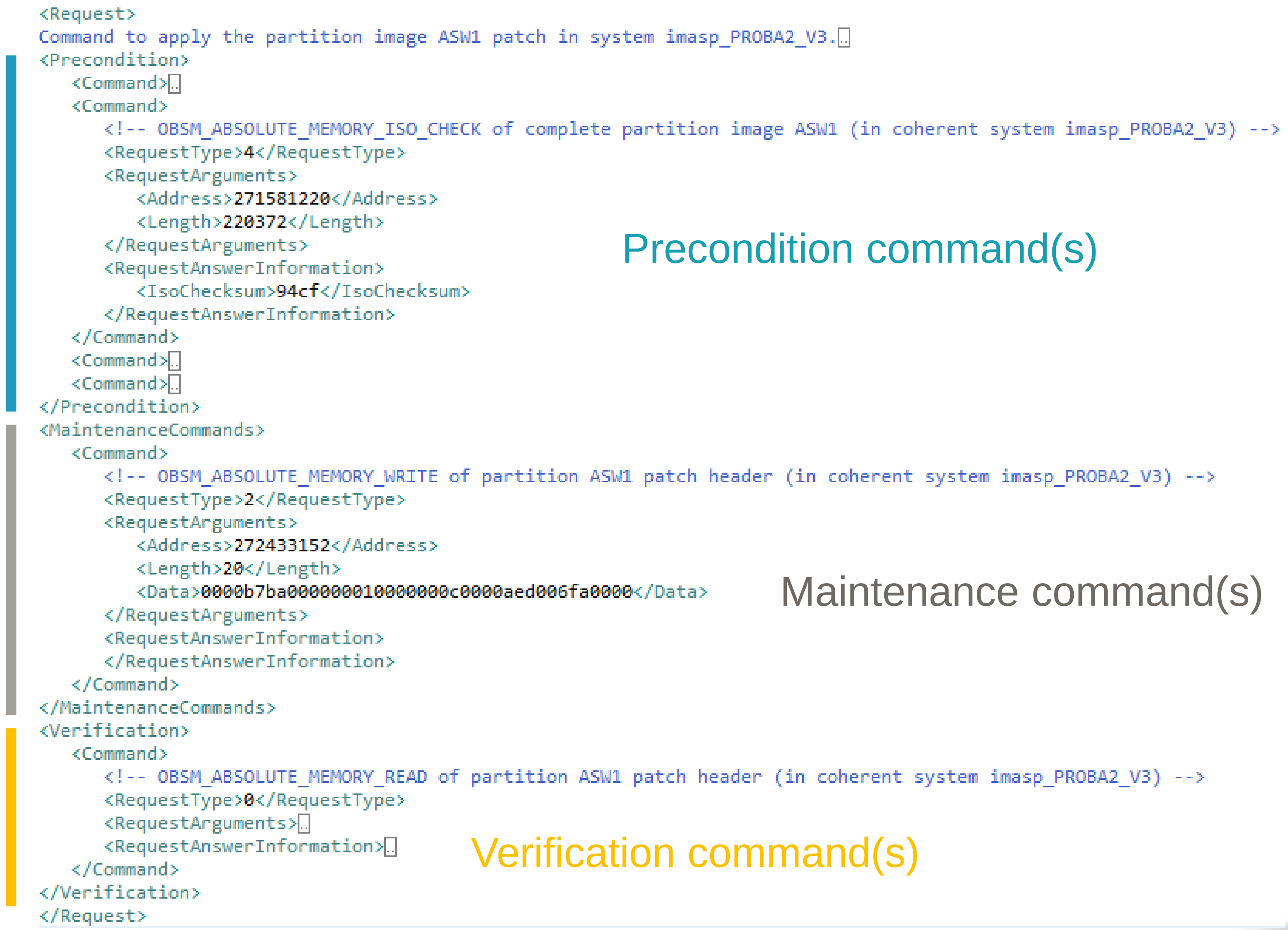
This table presents the different commands made available by the ground tool.

Exceptions are raised in case errors/incoherencies/wrong parameters are detected. The verification is done statically base on the OBSM database referencing the various patches, images, …

List of Commands
Generate the C configuration file
Dump partition configurable parameter
Patch partition configurable parameter
Upload a complete XtratuM image
Upload a partial XtratuM image
Upload a complete partition image
Upload a partial partition image
Upload a partition image patch
Apply a partition image patch
Remove a partition image patch



Ground tool



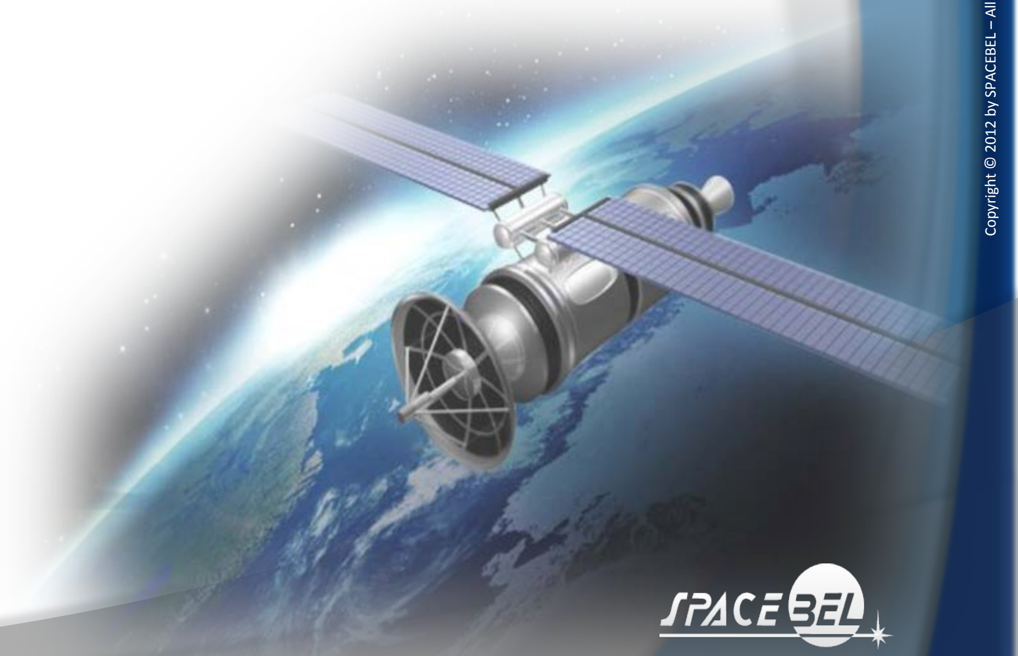
```
<Request>
Command to apply the partition image ASW1 patch in system imasp_PROBA2_V3.…
<Precondition>
   <Command>…
   <Command>
      <!-- OBSM_ABSOLUTE_MEMORY_ISO_CHECK of complete partition image ASW1 (in coherent system imasp_PROBA2_V3) -->
      <RequestType>4</RequestType>
      <RequestArguments>
         <Address>271581220</Address>
         <Length>220372</Length>
      </RequestArguments>
      <RequestAnswerInformation>
         <IsoChecksum>94cf</IsoChecksum>
      </RequestAnswerInformation>
   </Command>
   <Command>…
   <Command>…
</Precondition>
<MaintenanceCommands>
   <Command>
      <!-- OBSM_ABSOLUTE_MEMORY_WRITE of partition ASW1 patch header (in coherent system imasp_PROBA2_V3) -->
      <RequestType>2</RequestType>
      <RequestArguments>
         <Address>272433152</Address>
         <Length>20</Length>
         <Data>0000b7ba000000010000000c0000aed006fa0000</Data>
      </RequestArguments>
      <RequestAnswerInformation>
      </RequestAnswerInformation>
   </Command>
</MaintenanceCommands>
<Verification>
   <Command>
      <!-- OBSM_ABSOLUTE_MEMORY_READ of partition ASW1 patch header (in coherent system imasp_PROBA2_V3) -->
      <RequestType>0</RequestType>
      <RequestArguments>…
      <RequestAnswerInformation>…
   </Command>
</Verification>
</Request>
```

Precondition command(s)

Maintenance command(s)

Verification command(s)

Assessment, Lessons learned & Conclusion

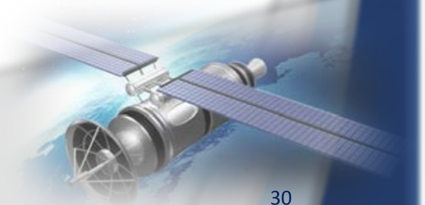


SVF assessment

- One bug reported : related to the reboot/reset of the OBC.
- Sensible drop of performance of the SVF due to the simulation of the MMU (LEON-3 Target Simulator needs to be optimized).
- One risk of the project was the debugging of partitions:
  - *The debugging of partitions is successful thanks to sparc-rtems-gdb and the simulation core.*
  - *Debugging of multiple partitions is currently not feasible*
  - *The debugging is only possible in the case where virtual addresses are mapped on physical addresses (one-to-one mapping).*
  - *The debugging of partition with MMU and virtual addressing would be a great improvement for IMA.*

Hypervisor assessment

- Several adaptations have been performed on the hypervisor in order to cope with OBSW maintenance requirements:
  1. On the XM configuration file a new flag is now available in order to specify whether partitions will be loaded by XtratuM or not.
  2. The partitions that are not loaded by XtratuM should be loaded manually by a system partition.
  3. The boot loader for partitions is maintained only for those partitions to be loaded by XtratuM.
  4. The hypercall reset partition contains optional arguments indicating the entry point and XM header partition. These new arguments are only useful for the partitions not loaded by XM.
  5. The functionalities for COLD and WARM reset modes are maintained for the partition loaded by XM, i.e., COLD reset restores the partition image from container and WARM reset jump to entry point of the partition. Additionally, these reset modes modify internal states in XM, such as: reset numbers, etc.
  6. For the partitions not loaded by XM, the only difference with the previous functionalities for COLD and WARM reset modes are in the COLD reset. COLD reset jumps to the entry point specified by the hypercall and does not restore the partition image from container.
  7. The board 'GR712RC' has been added to the list of supported board by XM
  8. …



Hypervisor assessment

Minor issues

- The configuration of XtratuM is not really user-friendly.
- It is not possible to exercise a stand-alone RSW (i.e. a RSW separated from the container).
- Every time the XM configuration is updated, a new distribution needs to compiled and forwarded to stake holders
- The mathematical library is currently not available.
- Several problems have been encountered with the PSR. The virtualisation of LEON registers does not seem to be correctly and completely implemented in XtratuM.

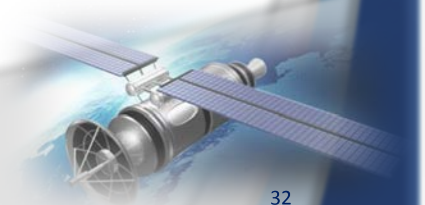
Major issue:

- ➢ Untimely division by 0 that occurs when using FPU. The issue is propagated to other partitions (partitions are halted).

After analysis, the registers of the floating point unit appear to be not correctly reloaded at the context switch

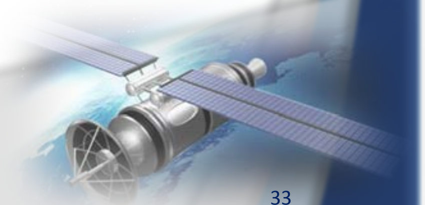
Guest OS assessment

- Two major issues have been identified in rtems Edisoft:
  1. The first one is an issue relevant to the semaphores (configurations of mutex are not compatible with binary semaphores).
  2. The second one is relevant to the function ‘rtems_task_delete’ (results in an unpredictable scheduling).
- According to ESA and fentISS, both issues are fixed in the latest version of RTEMS Edisoft. However, this version still needs to be paravirtualised for XtratuM.
- The POSIX layer is not available anymore.



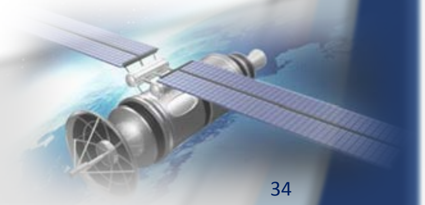
PROBA-SW assessment

- Easy integration of XtratuM tooling in the existing PROBA tool chain.
- PROBA DHS revealed to be flexible enough and generic enough against partitioning
- The partitioning of PROBA-2 software raised no major issue
- However , the partitioning had negative impacts on the global performance of the PROBA SW
- Should the Proba-2 software be developed and partitioned “from scratch”, its design would probably be different



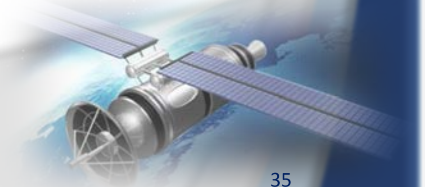
OBSW Maintenance assessment

- The chosen solution appears at to be adequate as it allows a wide variety of strategies
- In particular, the OBSM solution would allow coping with:
  - Partitions or systems with critical boot/reboot time;
  - Functional chain management;
  - Partition or system boot/reboot on multi-core or multi-processor computers;
  - Maintenance of system using different hypervisors (not only XtratuM).
- The current OBSM solution has however a few limitations:
  - Compression/decompression of binary images is foreseen but is not yet supported.
  - OBSM partition has no access to XtratuM memory (due to XtratuM design).
- The validation and the unit testing of the complete OBSM partition are successful
- The unit testing of the OBSM ground tool is also successful and its integration with the actual test environment is successful too



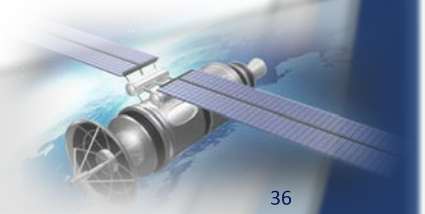
Conclusion

- The SVF clearly covered the requirements of the project.
- The OBSM partition and OBSM Ground Tool have been successfully validated and demonstrated.
- The OBSM solution has been developed with success according to ECSS-E-ST-40C standard: category C for OBSM partition and category D for OBSM Ground Tool.
- Concerning XtratuM, the hypervisor is continuously improved. Surely the maturity of the product must be consolidated but it is encouraging and satisfactory.
- The most constraining issue is the one relevant to floating point. The issue is considered major and at some point blocking.
- At the scale of this project and people involved, it was quite hard to fit exactly the process and roles defined by IMA-SP. Despite this fact, Spacebel is convinced that the process is in line with a real industrial organisation.



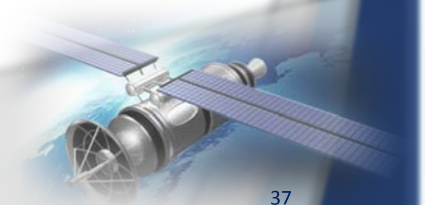
Conclusion

Finally, the development and the design of the partitions generate a lot of problems but adequate solutions (or sometime workarounds) have been figured out. Small applications have already been partitioned in the past but software with the scale of PROBA-2 was really a live action experiment. This study will surely contribute to ESA and Spacebel for future IMA developments.



Conclusion

Objectives	Status	Comments
1. Demonstrate how the FSW of a small spacecraft can be partitioned.	✓	
2. Demonstrate and assess off-line integration of different applications in a partitioned execution environment.	✓	
3. Demonstrate and assess upload and on-line integration of new and/or updated applications in a partitioned execution environment.	✓	
4. Demonstrate and assess software development process and roles across different software teams.	Partialy covered	The ASW2 (developed by a third party) has not been addressed yet. FSW, OBSM and AOCS has however been developed by independent teams.



Thank you for attention