On-Board Control Procedures: Autonomous and Updateable Spacecraft Operator Onboard and Beyond

Marek Prochazka / Kjeld Hjortnaes
European Space Agency,
ESTEC, Software Systems Division.

FSW-10, Pasadena
8-10 Dec 2010

OUTLINE

- Overview of OBCP
- Overview of the OBCP ECSS standard
- Advanced concepts
- Conclusions

OPERATOR ON-BOARD?

Having an operator on-board spacecraft would be more efficient ??

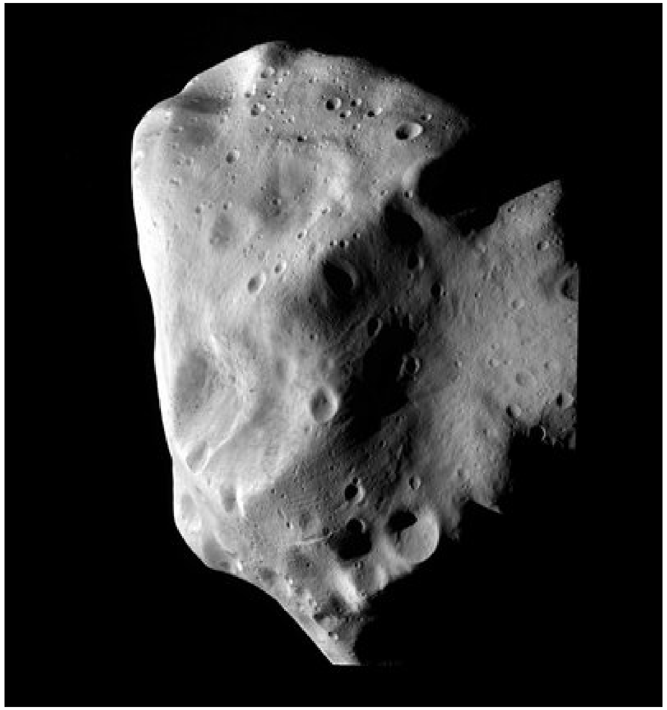
- Reduction of delay
- Reduction of bandwidth need
- Enhance autonomy (no need of 24/7 ground operations support)
    - Useful in situations of reduced spacecraft visibility
    - In deep-space missions with long signal propagation delays
    - In situations when a rapid reaction is needed
- Implementation of on-board solutions in view of unforeseen failures occurring in orbit
- Adaptation to unpredictable environment
- End-of-life operations

ON-BOARD CONTROL PROCEDURES

OBCPs are flight control procedures that can be resident on-board or that can be uploaded to the spacecraft as required by the ground

Key features:

- Often interpreted
- No fault propagates to FSW
  - Isolated in time & space
- Can be uploaded to the spacecraft in advance

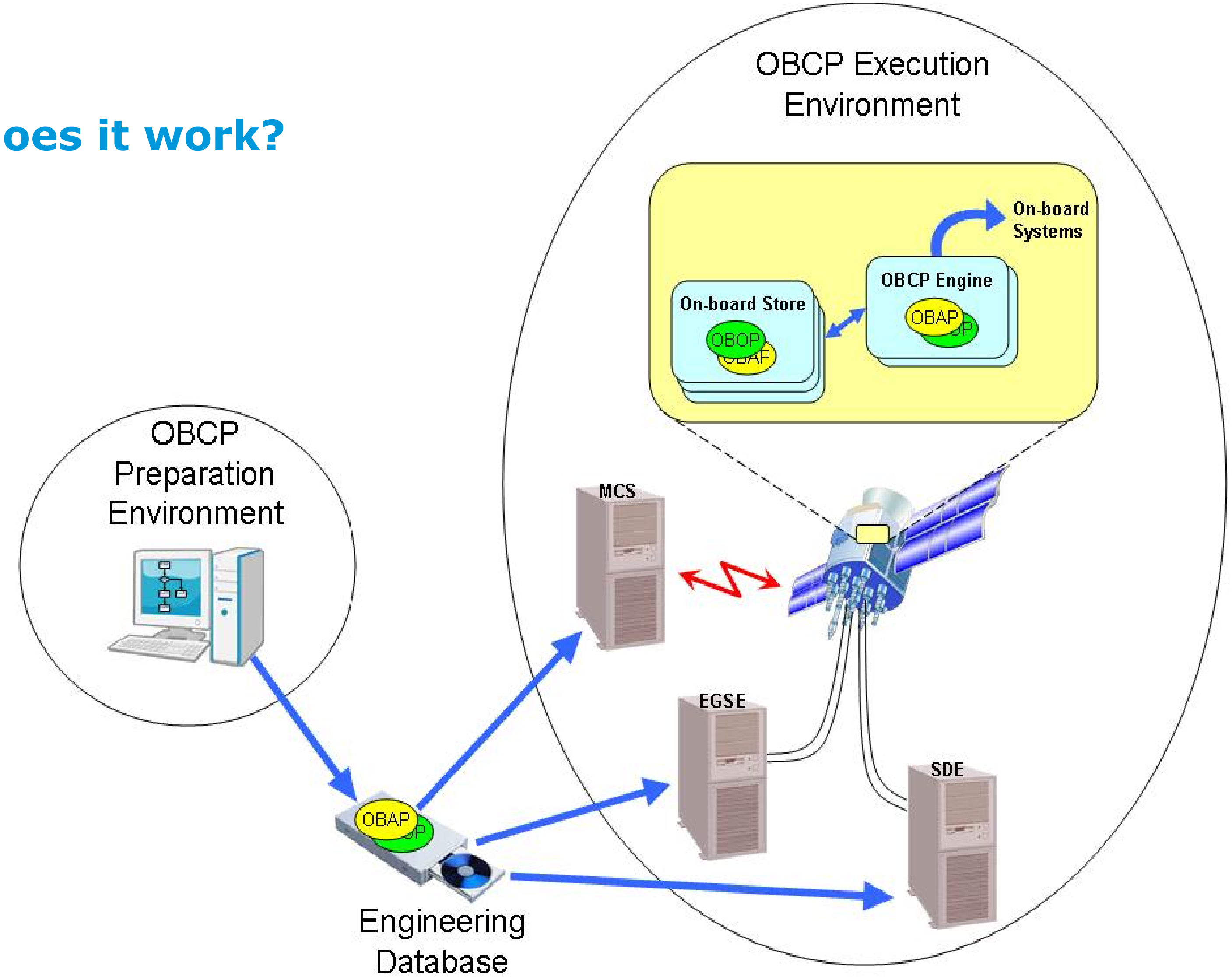


Currently used in a variety of ESA missions

- Deep space missions
  with high degree of autonomy
  Rosetta, Venus Express, LISA PF, Herschel-Planck,
- But also Earth observation missions as missions as GOCE, Cryosat, Sentinel, …

ON-BOARD CONTROL PROCEDURES

How does it work?



The OBCP Standard

ECSS-E-ST-70-01C

www.ECSS.nl

European Space Agency

THE OBCP CONCEPT

European Cooperation for Space Standardisation (ECSS)

ECSS-E-ST-70-01C

- Published in April 2010
- Defines the OBCP concept, system capabilities and the associated engineering process

OBCP system includes:

- OBCP system capabilities
  - Language
  - Preparation environment
  - Execution environment

- OBCP engineering process
  - Lifecycle of development and V&V

OBCP DOMAINS OF APPLICATION (I)

Spacecraft Operations

- Streamline: Reduction of bandwidth, delay
- Enhance autonomy
- Implementation of on-board solutions in view of unforeseen failures occurring in orbit
- Adaptation to unpredictable environment
- End-of-life operations

System design

- Platform functions
    - To isolate mission-specific platform functions of FSW
    - To implement one-shot applications deleted after use
    - To accommodate late specification of detailed FDIR
    - To accommodate late delivery/removal/addition/replacement of equipment
    - Fine tuning configuration sequences without having to modify FSW
- Payload functions
    - To accommodate late definition and tuning of complex configuration sequences and monitoring/recovery actions

OBCP DOMAINS OF APPLICATION (II)

FSW design and development

- Ease of development and validation
- FSW generic, mission-specific functions in OBCP
- Easier maintenance
- Automation of tests
- Debugging
- Short-term workaround solutions

Assembly, Integration and Testing (AIT)

- Fault injection and robustness testing
- Long and complex configuration sequences
- Temporary functions for testing purposes

TYPES OF OBCP

OBOP: On-board Operation Procedures

- To operate spacecraft
  - i.e. these procedures allow to execute a sequence of telecommands with corresponding logic, which would normally have to be executed manually by spacecraft operators

OBAP: On-board Application Procedures

- Part of the spacecraft functionality
- Qualification together with FSW

There are major differences how OBAP/OBOP are treated

- Scheduling
- Resource allocation
- Accessible services

OBCP SYSTEM

Preparation environment

- Editor
- Static checking (constraints, consistency)
- Configuration
- Compilation
- Validation

Execution environment

- Ground
  - Control: Uplinking, Monitoring
  - Visualisation
  - Constraint and consistency checking
- On-board
  - OBCP engine
    - Monitoring and control (interfacing the ground)
    - Interaction with FSW, platform and payloads
    - Execution of OBCP
    - Could be multiple of them, “independent”
  - OBCP store
    - Loading, garbage collecting – out of the scope of the ECSS standard

OBCP LANGUAGE CAPABILITIES

- **Domain-specific language or generic programming language**
- **Data types**
  - Based on another ECSS standard “Ground Systems and Operations - Monitoring and control data definition” (ECSS-E-ST-70-31)
  - Structures and arrays
  - Explicit type casting
- **Declarations**
  - Variables of any data type
  - Constants of any data type
  - Local functions
- **Expressions**
  - Assignment
  - Math, time, string bitwise operations
  - Constants, on-board parameters, activity arguments and variables
  - **Constants together with their engineering units**
    - Mix compatible units
    - Automatic conversion of units
    - On-board parameters by names and in engineering units and also **raw form**
  - Refer to events by their names

OBCP LANGUAGE CAPABILITIES (II)

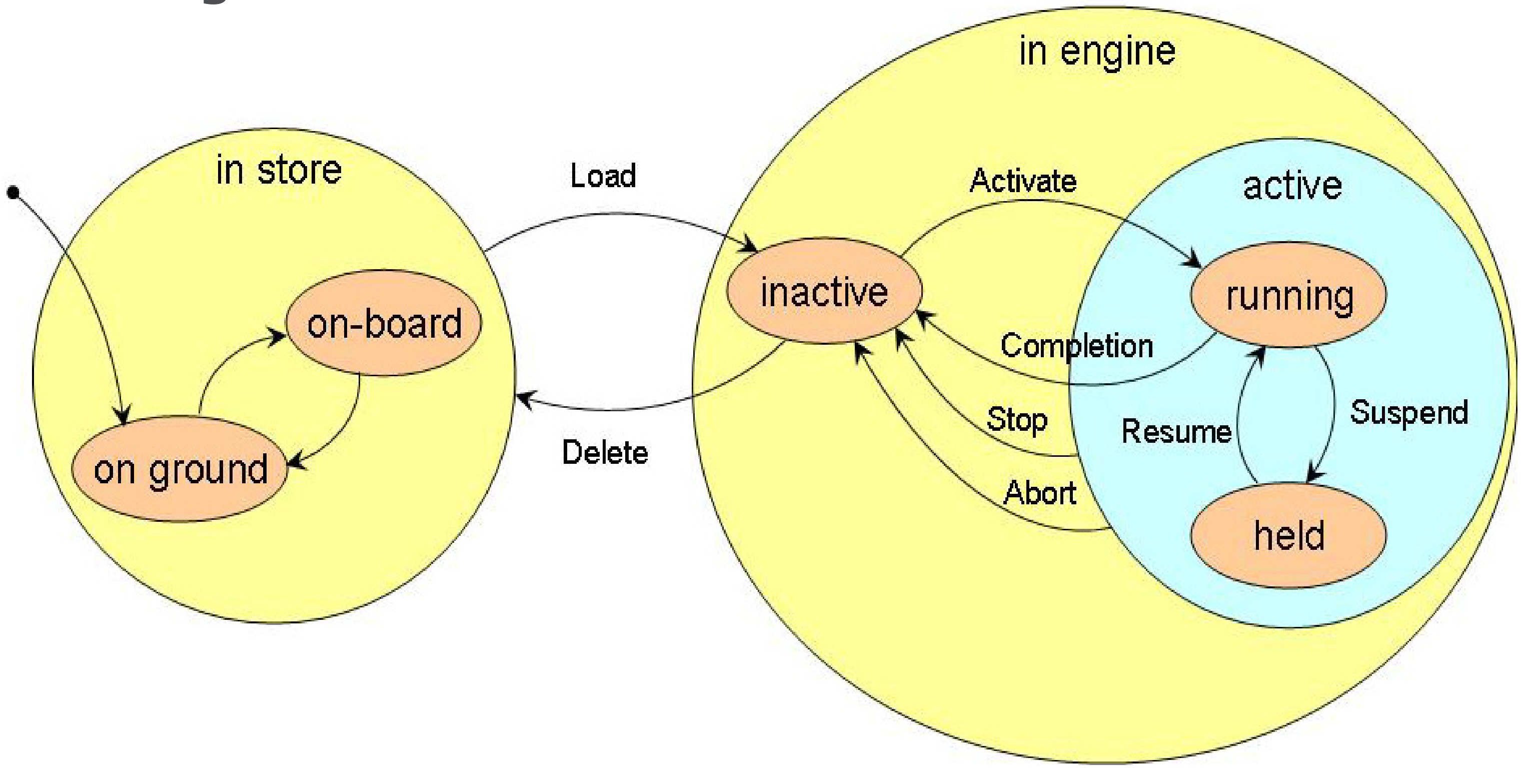
- **Flow control**
  - Branching simple and multiple conditional based on any parameter or variable
  - Repeated execution (for-loop, repeat-until, while-do)
- **Waits**
  - Until a given OBT, or elapsed time
  - Until a condition becomes true
  - Until given event occurs
  - Until an event from a list of events occurs
  - Combination of conditions within wait statement
  - Precondition/postcondition
    - All waits + test a condition
  - Timeouts for wait
    - Should be possible to define
    - Behaviour in case of exceeded timeout

OBCP LANGUAGE CAPABILITIES (III)

- **External interactions**
  - Read/write on-board parameters
  - Initiating activities (other OBCPs, possibly also using other FSW services)
    - Including reporting conditions, information on started activities
    - Both blocking and non-blocking (spawn or exec&wait)
    - OBAP: Also execution of activities not accessible from the ground (bus access)
    - Raising and accessing “events” which are reported to the ground and also can trigger appropriate FSW actions
    - Contingency handling based on events/conditions

OBCP EXECUTION ENVIRONMENT CAPABILITIES (I)

- **Ground**
  - Checks (resources, inter-dependecies, state transitions)
  - Uplink
  - Management (PUS 18)
    - Activate request allows to pass parameters
- **Monitoring and control**



OBCP EXECUTION ENVIRONMENT CAPABILITIES (II)

- **OBCP integrity**
  - Checksum generated by translation process
  - Checksum checked during load to engine
  - Checksum on request
- **On-board capabilities**
  - Predefined scheduling policy
  - Static and dynamic memory allocation policy
  - Garbage collection policy
  - Observability of these by ground
  - Engine services
    - Process the language capabilities
    - Global service for all OBCPs
    - Defined behaviour in case of reset, discontinuity of time (affects waits)
    - Housekeeping information
  - Loading policy
    - OBCP stores
    - Different types of memory
    - Reprogrammable? – should be defined
    - Persistency? – should be defined

OBCP EXECUTION ENVIRONMENT CAPABILITIES (III)

- **On-board capabilities (cont.)**
  - Process scheduling
    - Should be defined
    - Minimum allocation time per time intervals
    - Several OBCPs "in parallel"
    - OBAP and OBOP resource allocation is independent
    - Max number of loaded and active OBOPs is defined
    - OBOP resource allocation independent from concurrently executing OBOPs (i.e. context does not change behaviour)
    - OBAP resource allocation to concurrently executing OBAPs should be defined (i.e. definition of priority scheme)
  - Isolation of OBCPs
    - Internal faults do not propagate to OBSW
    - Maximum allocated resources never exceeds
    - Loading, activation and control of an OBCP should not affect active OBCPs
      - How about higher-priority OBCP preempting a lower-priority one?
    - OBCPs are isolated – no fault propagation, no illegal memory access

OBCP EXECUTION ENVIRONMENT CAPABILITIES (IV)

- **On-board capabilities (cont.)**
  - Exception handling
    - Internal errors detected and handled by OBCP or engine
    - Internal errors reported to the ground
    - Run-time error of error handler → Termination of OBCP
    - When condition to run OBCP(s) are not longer provided, actions taken defined
      - Contingency handlers establishing default state of SW/HW
      - All running OBCPs suspended
      - Report to ground
  - Continuity of service
    - The concept should be defined
    - Capability to define default autorun OBCPs at engine startup

TRADE-OFF ANALYSIS: OBOP VS. GROUND-BASED OPERATIONS

OBOP benefits:

- Operations during phases of non-visibility or with long signal propagation
- Loss of ground control
- Synchronise with asynchronous elements (events)
- Coded and up-linked once, used many times
- Atomic operations (critical activities to be performed “at once”)
- Decrease the need for human availability

Ground-based operations benefits:

- Human response in unforeseen scenarios
- Decrease the complexity of FSW & validation
    - Engineering effort required to develop and validated an ops procedure is lower than to develop an OBOP
- Less effort to update a procedure
- Less complex configuration management

TRADE-OFF ANALYSIS: OBAP VS. FSW

OBAP benefits:

- Variability and flexibility (in case of mission requirements change)
- Late definition
- Maintenance

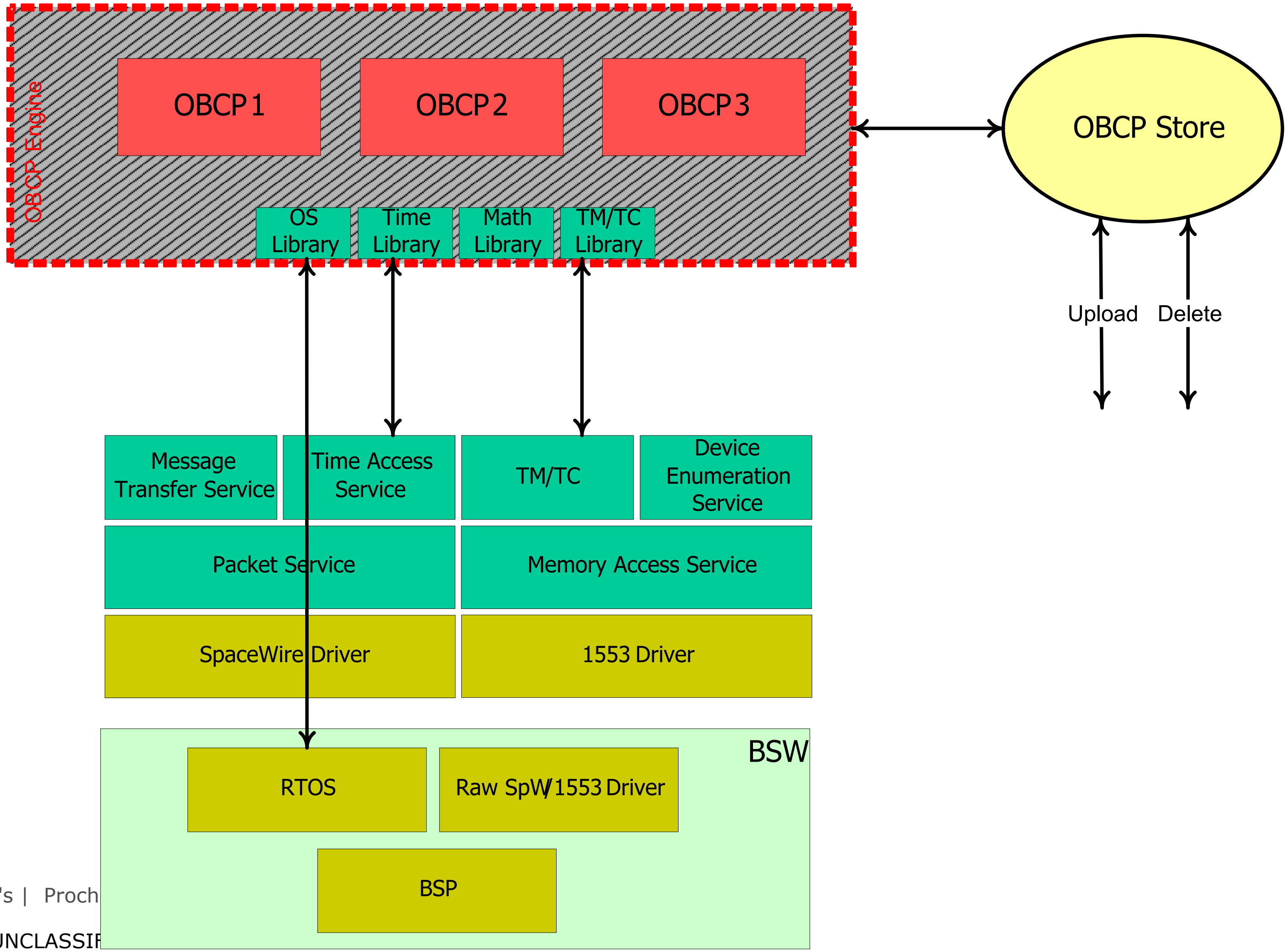
FSW benefits:

- Better performance
- Complex functions (engineering process, techniques)
- Core system with stable reqs. and close to subsystems (e.g. AOCS)
- Functionality for survival modes (no interpreter activated)
- Generic functionality (e.g. PUS, reused subsystems)
- Subsystems to be available early in the lifecycle

Advanced Concepts Using OBCP

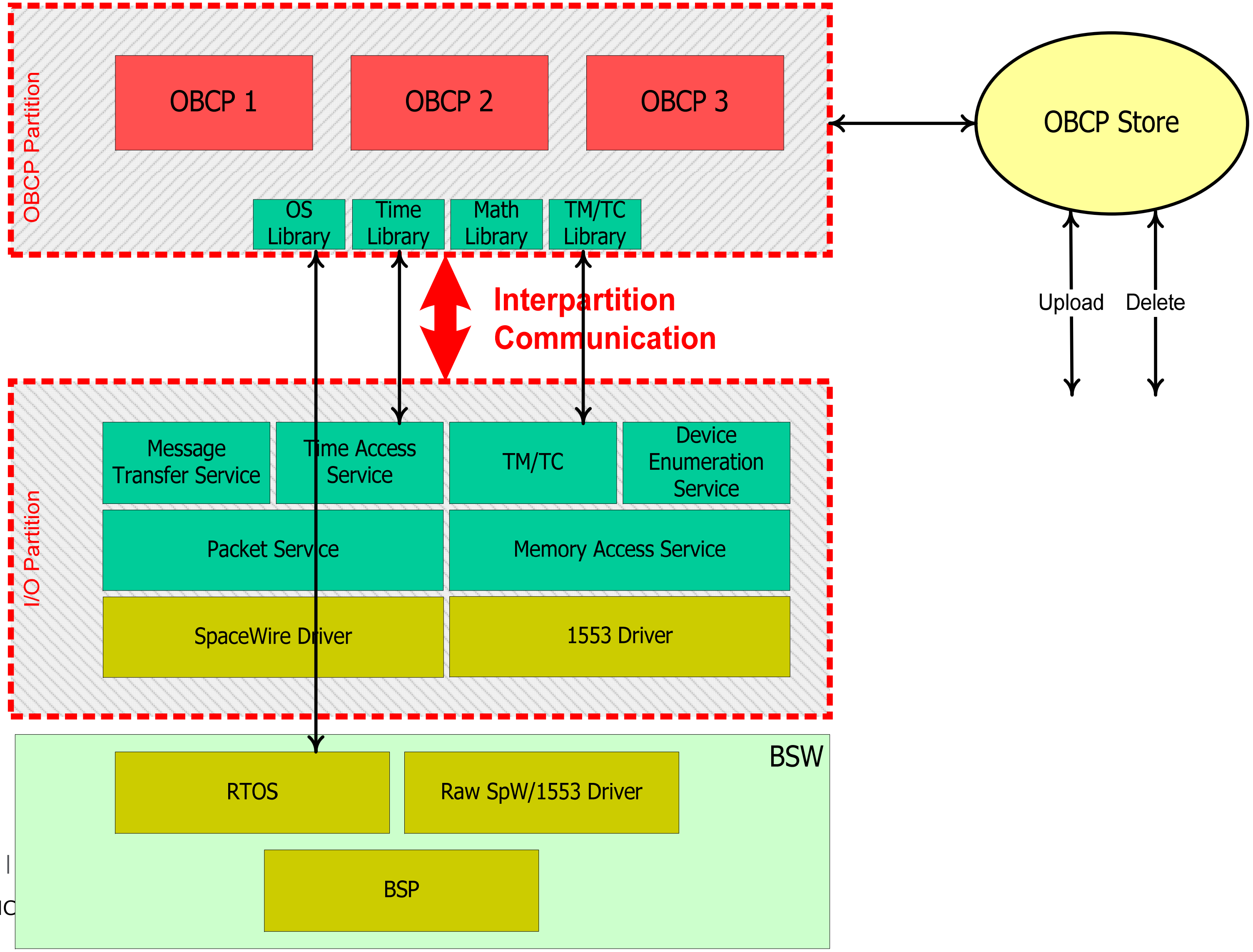
OBCP IN CURRENT EXECUTION ENVIRONMENT

OBCP = Fault Containment + Dynamic Code Updates



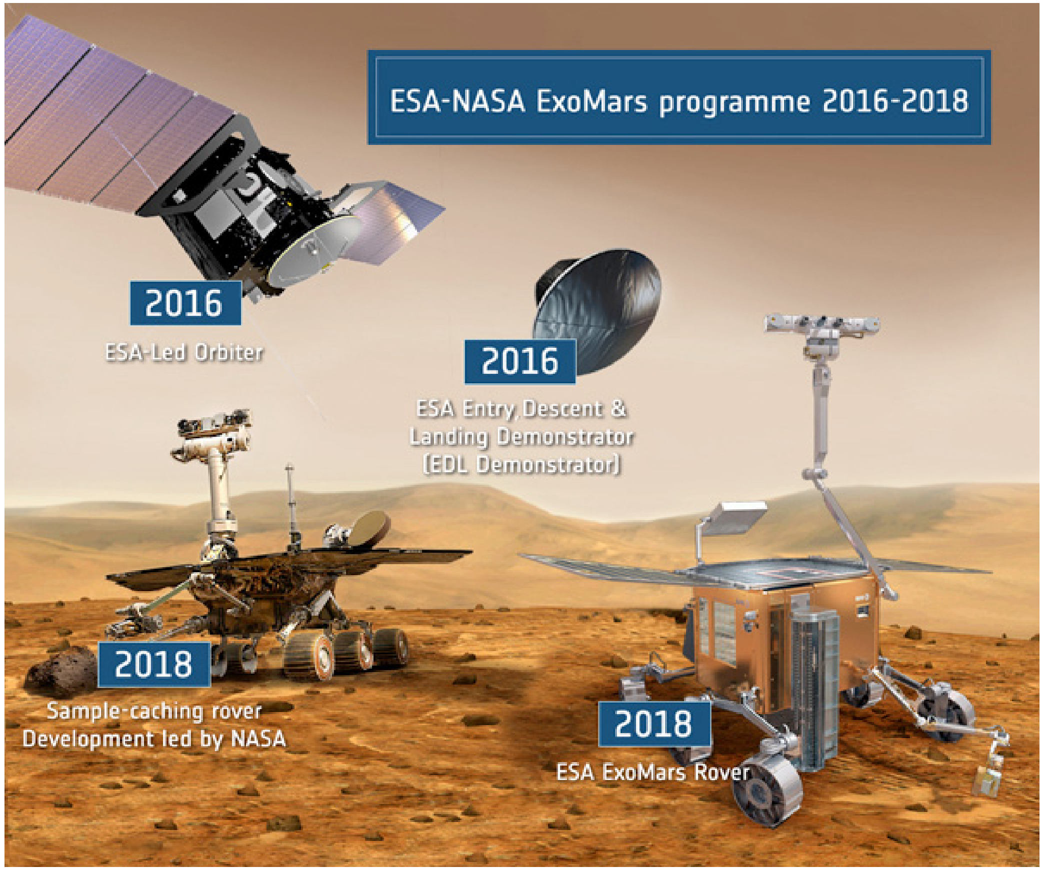
OBCP IN IMA TIME & SPACE PARTITIONED ENVIRONMENT

OBCP = Dynamic Code Updates

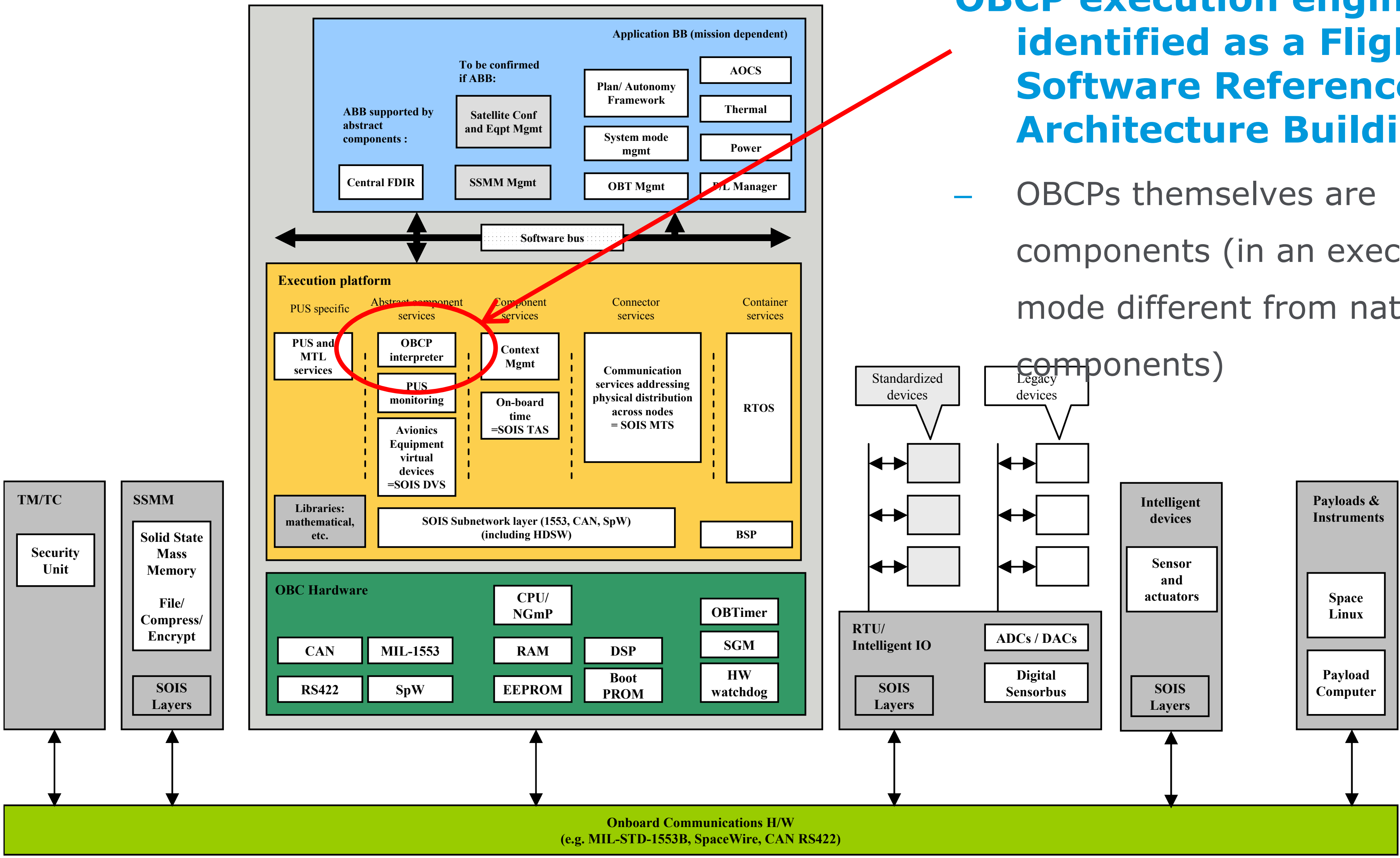


OBCPs in ExoMars Rover

- OBCP engine to manage more complex activities (support for autonomy)
- Most likely together with
  - Mission Timeline Service (time-triggered actions)
  - Event-Actions service triggering TC files or OBCPs upon an event occurence
- An option to use “Interlocks” in the OBCP engine
  - Conditional execution flow of OBCPS
  - Depending on the result of an OBCP another OBCP is allowed to start



OBCPs IN ESA FSW REFERENCE ARCHITECTURE



OBCP execution engine identified as a Flight Software Reference Architecture Building Block

- OBCPs themselves are components (in an execution mode different from native components)

European Space Agency

CHALLENGES OF THE OBCP BUILDING BLOCK

- **Interface of the OBCP engine**
  - **Provided interface**
    - To the flight software / To the Ground
  - **Required interface**
    - To flight software services (e.g. PUS services)
    - To system resources (memory, CPU scheduler, I/O)
- **On-board Operations Procedures (OBOPs) vs. On-board Application Procedure (OBAPs)**
  - Different verification approaches
  - Different capabilities
    - Different resource usage profiles (scheduling, memory utilisation)
    - Different interfaces?
- **Position of the OBCP BB in the reference architecture**
  - An application component or a part of the execution platform?
- **Time and Space Partitioning**
  - Addressed on previous slides
- **Source language**
  - Domain specific language, scripting language, Java?
  - To be standardised? To be generated from different languages or MDE diagrams?
- **Execution mode i.e. what is the target language**
  - Interpretation (i.e. source language = target language)
  - Intermediate language (e.g. Java bytecode)
  - Native code (Ahead-of-Time compilation, Just-in-Time compilation?)

CONCLUSIONS

A new interesting paradigm

- Scripting/Interpreter
- Temporal & spatial isolation
- Uploading new SW components and updating existing ones at runtime
- Specific functionality
  - Telecommands, telemetry, events
  - Flow control (branching, loops)
  - Domain-specific language

Interesting areas of application

  - Autonomy
  - Software maintenance

Challenges remain

THANK YOU

Marek Prochazka, Kjeld Hjortnaes

European Space Agency
Marek.Prochazka@esa.int

Kjeld.Hjortnaes@esa.int