



MOSAR

Deliverable Reference : D2.4

Title : Preliminary Design Document

Confidentiality Level : PU

Lead Partner : Space Applications Services

Abstract : This report is the output of the preliminary design activities of the project MOSAR ("WP2 Preliminary Design and Modelling). The document provides the overall system architecture, the description of the system operations in the context of the project demonstration and the preliminary design of the different components and sub-systems, including their interfaces.

EC Grant N° : 821996

Project Officer EC : Christos Ampatzis (REA)



MOSAR is co-funded by the Horizon 2020
Framework Programme of the European Union



Preliminary Design Document

DOCUMENT APPROVAL SHEET			
	Name	Organization	Date
Prepared and Cross-reviewed by:		Space Applications Services Thales Alenia Space France Thales Alenia Space UK SITAEL GMV University of Strathclyde German Aerospace Centre (DLR) MAG SOAR ELLIDISS	11/03/2020



Preliminary Design Document

DOCUMENT CHANGE RECORD				
Version	Date	Author	Changed Sections / Pages	Reason for Change / RID No
1.0.0	01/12/2019	Consortium	All	First release for PDR milestone review
1.1.0	11/03/2020	Consortium	2	Update MOSAR demonstration introduction description, to present mission related aspects and links with demonstration, as feedback to RID OG09-99
			2.2	Clarify faulty detection handling and autonomous network reconfiguration capability, as feedback to RID OG09-98
			3.2	Update Visual Sub-system related description, following re-scoping of demonstration, as feedback to RID OG09-96
			4.1.2.2	Add section about network reconfiguration from faulty detection, as feedback to RID OG09-98
			5.1	Add section 5.1 Operational Scenarios as introduction and link to the demonstrations tests description, as feedback to RID OG09-99
			5.3.4	Add demonstration scenario 4 related to automatic data re-routing, as feedback to RID OG09-98
			6.7.2	Clarification FMC board design, as feedback to RID OG09-76
			6.9	Update section about visual processing system, following re-scoping of demonstration, as feedback to RID OG09-96



Contents

Contents	3
1 Introduction	9
1.1 Purpose and Scope	9
1.2 Document Structure	9
1.3 Applicable Documents	9
1.4 Reference Documents	9
1.5 Acronyms	10
2 MOSAR Demonstrator Introduction and Objectives	12
2.1 MOSAR Mission Overview	12
2.2 MOSAR Demonstrator	14
3 MOSAR Demonstrator Components and System Architecture	17
3.1 Ground Segment	17
3.2 Space Segment	17
3.2.1 MOSAR Space Segment Components	17
3.2.2 MOSAR Space Segment Generic Data and Power Architecture	18
3.3 MOSAR Product Tree Responsibilities	22
4 MOSAR System Reconfiguration	24
4.1 Hardware System Reconfiguration	24
4.1.1 Mechanical Reconfiguration	24
4.1.2 Data – SpW Network Reconfiguration	24
4.1.3 Power Reconfiguration - Distribution	25
4.2 Software System Reconfiguration	28
4.2.1 Types of software reconfiguration for MOSAR	28
4.2.2 Operating systems features to implement node reconfiguration	30
4.2.3 MOSAR scenarios	35
4.2.4 Preliminary design of the software reconfiguration for the MOSAR demonstrators	36
5 Demonstration Scenarios	40
5.1 Operational Scenarios	40
5.2 Spacecraft Modules Description	42
5.3 Demonstration Scenarios	42
5.3.1 Scenario 1: Initial Assembly of SMs from SVC to CLT	42
5.3.2 Scenario 2: Replacement of a failed SM	43
5.3.3 Scenario 3: Thermal transfer between two SMs	44
5.3.4 Scenario 4 (S4): Automatic CLT Network Reconfiguration	45
5.4 Demonstration Configuration Analysis	46



Preliminary Design Document

5.5	Ground Segment Operations	49
5.6	Space Segment Operations	54
5.6.1	Planner Elementary Actions	55
5.6.2	Routines.....	56
6	Components Preliminary Design.....	61
6.1	Design and Simulation Tool	61
6.1.1	Design Tool.....	61
6.1.2	Purpose of Simulator	61
6.1.3	Components of Simulator	63
6.1.4	Integration of the Simulator	66
6.1.5	Simulator Verification.....	67
6.2	Autonomy Agent and Planning.....	69
6.2.1	Autonomy Agent	69
6.2.2	Mission Planning	71
6.2.3	Robotic Arm Motion Planning	72
6.3	Telemetry and Telecommand Service	74
6.4	Spacecraft Modules and Payloads.....	76
6.4.1	Structural concept of a generic Spacecraft module.....	76
6.4.2	SM1-DMS	78
6.4.3	SM2-PWS.....	79
6.4.4	SM3-BAT	81
6.4.5	SM4-THS	83
6.4.6	SM5/6-OSP.....	84
6.5	Walking Manipulator	86
6.5.1	Preliminary Design	86
6.5.2	WM Operations in MOSAR.....	87
6.6	HOTDOCK Standard Robotic Interface	89
6.6.1	Status.....	89
6.6.2	HOTDOCK Control in MOSAR	90
6.7	R-ICU.....	92
6.7.1	Status.....	92
6.7.2	FMC Board Design	92
6.7.3	SpaceWire Router IP	93
6.7.4	R-ICU Operations in MOSAR	94
6.8	SVC/CLT Satellites Mockups	95
6.8.1	Test benches general design concept.....	95



Preliminary Design Document

6.8.2	SVC/CLT Architecture	96
6.9	Visual Processing System	98
6.9.1	Update of the Functional Requirements	98
6.9.2	Physical Organization of the Vision Subsystem	100
6.9.3	Software Organization of the Vision Subsystem	101
6.10	MCC and Demonstration Setup	103
7	Annex	107
7.1	MOSAR Setup Position References.....	107
7.2	MOSAR Sequence of Manipulations Example.....	108
7.3	Software Reconfiguration Documents References	113



List of Figures

Figure 2-1: MOSAR roadmap to space application.....	12
Figure 2-2: Spacecraft design MOSAR-like mission (client and servicer).	13
Figure 2-3: Life-cycle of the MOSAR mission concept.....	13
Figure 2-4: Spacecraft with different mission configurations.....	14
Figure 2-5: On-orbit reconfiguration with substitution and addition of modules	14
Figure 2-6: MOSAR Operational Concept.....	15
Figure 3-1: Main components of the MOSAR space segment.....	18
Figure 3-2: MOSAR generic data and power architecture	21
Figure 3-3: MOSAR demonstrator product tree	22
Figure 4-1: cPDU architecture	26
Figure 4-2: Software reconfiguration process	29
Figure 4-3: Various options for the node	29
Figure 4-4: Reconfiguration mode manager – General architecture.....	38
Figure 4-5: Reconfiguration mode manager – Deployment	38
Figure 5-1 – Different use cases of MOSAR mission concept.	41
Figure 5-2: MOSAR scenario 1 initial and final configuration.....	43
Figure 5-3: MOSAR scenario 2 initial and final configuration.....	44
Figure 5-4: Design, simulation and planning stages	51
Figure 5-5: Sample TASTE model for the CLT software.....	52
Figure 5-6: Re-configuration operation architecture.....	54
Figure 5-7: Routine 1 – WM re-localization between initial SI and Target SI.....	56
Figure 5-8: Routine 2 – SM re-localization	58
Figure 6-1: Simulator application cases within the overall system development scheme in MOSAR ..	62
Figure 6-2: Modelica library prepared for MOSAR.....	63
Figure 6-3: Simulator components and interfaces	66
Figure 6-4: Simulator verification taking into account models from the mechanical, electrical, and thermal domain.....	67
Figure 6-5: Simulator verification inside an orbital scenario.....	68
Figure 6-6: Overview of the ERGO Agent tailoring for MOSAR.....	70
Figure 6-7: OG2/ERGO Ground Control Interface (GCI) Adaptation	71
Figure 6-8: Robotic Arm Main Components	72
Figure 6-9: Initial concept of spacecraft Module structure	76
Figure 6-10: Customized face hosting the internal tray structure.....	77
Figure 6-11: SM1-DMS physical layout and Intel NUC board.....	78



Preliminary Design Document

Figure 6-12: SM2-PWS layout.....	79
Figure 6-13 SM3-BAT layout.....	81
Figure 6-14: OG5 battery controller.....	82
Figure 6-15: SM4-THS layout.....	83
Figure 6-16: SM5/6 optical payload and the stereo ZED camera	84
Figure 6-17: MOSAR Walking Manipulator	86
Figure 6-18: Walking Manipulator avionics / data architecture	87
Figure 6-19: HOTDOCK prototype	89
Figure 6-20: HOTDOCK control states.....	91
Figure 6-21 Components of the R-ICU.....	92
Figure 6-22 Zed Stereo Camera for the optical payload	92
Figure 6-23 Block design of the SpaceWire router hardware	93
Figure 6-24 Spacecraft Bus Overview.....	95
Figure 6-25: SVC structure exploded view	96
Figure 6-26: SVC/CLT data architecture	97
Figure 6-27: Hardware configuration of vision subsystem	101
Figure 6-28: Software architecture of the vision subsystem	102
Figure 6-29: MCC and demonstration setup space - notional layout (numerical dimensions are correct but elements geometry and size are only indicative)	103
Figure 6-30: Monitoring & Control Center (MCC) facing the demonstration setup space	104
Figure 6-31: Technical sheet of an ARRI Fresnel Daylight Compact 2500 (possible lighting solution)	105
Figure 6-32: Space for demonstration setup	106
Figure 7-1: References of the HOTDOCK standard interfaces spot on SVC and CLT.....	107
Figure 7-2: References of the spacecraft module faces (in the initial MOSAR setup configuration) ..	107



List of Tables

Table 3-1: Components design responsibilities	23
Table 3-2: MOSAR software main responsibilities	23
Table 4-1: cPDU parameters	27
Table 4-2: Summary of the software reconfiguration features by execution platform	33
Table 5-1 – Declination of ground demonstrators covering main functionalities identified in real mission scenarios	42
Table 5-2: Standard interface selection for MOSAR components (O=optinal)	48
Table 5-3: System characteristics relevant for configuration	49
Table 5-4: Preliminary list of planner elementary actions	55
Table 6-1: Input and Output Signals of the Simulator	66
Table 6-2: Standard PUS services	74
Table 6-3: Thermal Payload Parameters	80
Table 6-4: Battery payload parameters	82
Table 6-5: Camera payload parameters	85
Table 6-6: WM operations parameters	88
Table 6-7: Supported feature per HOTDOCK declination	90
Table 6-8: HOTDOCK parameters	90
Table 6-9: R-ICU operations parameters	94
Table 6-10: InFuse objectives in MOSAR	99
Table 7-1: Scenarios 1 and 2 step-by-step sequence of operations	108



1 Introduction

1.1 Purpose and Scope

This report is the output of the preliminary design activities of the project MOSAR ("WP2 Preliminary Design and Modelling). The document provides the overall system architecture, the description of the system operations in the context of the project demonstration and the preliminary design of the different components and sub-systems, including their interfaces.

1.2 Document Structure

In brief, the document is structured as follows:

- | | |
|------------------|--|
| Chapter 1 | Introduction: this section |
| Chapter 2 | MOSAR Demonstrator Introduction and Objectives: gives a general information about the MOSAR operational concept and demonstration purpose |
| Chapter 3 | MOSAR Demonstrator Components and System Architecture: provides the definition of the MOSAR setup components and the global architecture |
| Chapter 4 | MOSAR System Reconfiguration: describes the concept of hardware and software reconfiguration, addressed in the project |
| Chapter 5 | Demonstration Scenarios: describes the foreseen scenarios demonstrated in MOSAR, including the analysis of the system components and the description of the ground/space segment operations |
| Chapter 6 | Components Preliminary Design: describes each MOSAR setup component and their interfaces |

1.3 Applicable Documents

- | | |
|-----|---|
| AD1 | Strategic Research Cluster "Space Robotics Technologies" – Collaboration Agreement |
| AD2 | MOSAR Consortium Agreement, version 1.0 (7-Nov-2018) |
| AD3 | MOSAR Grant Agreement (821996) (18-Jan-2019) |
| AD4 | MOSAR D1.4 System Requirements Document, MOSAR-WP1-D1.4-SA issue 1.0.0 (1-Sep-2019) |
| AD5 | MOSAR FMC TRS - MOSAR-TASU-SYS-RS-0001_Iss_1.0 |

1.4 Reference Documents

- | | |
|-----|---|
| RD1 | European Cooperation for Space Standardization. Space Engineering (2016) — Telemetry and telecommand packet utilization (PUS) Services. ECSS-E-ST-70-41C. |
| RD2 | HOTDOCK Preliminary Design Definition File, MOSAR-WP2-D2.5-SA issue, 1.0.0 |



1.5 Acronyms

API	Application Programming Interface
APID	Application Process Identifier
ASN.1	Abstract Syntax Notation One
CAD	Computer-Aided Design
CAN	Controller Area Network
CLT	Client Satellite
CMD	Command Dispatcher
COTS	Commercial Off-the-Shelf
cPDU	central Power Distribution Unit
CPU	Central Processing Unit
DLR	Deutsches Zentrum für Luft- und Raumfahrt
DMA	Direct Memory Access
DMS	Data Management System
EC	European Commission
ERGO	European Robotic Goal-Oriented Autonomous Controller
ESA	European Space Agency
FMC	FPGA Mezzanine Card
FPGA	Field Programmable Gate Array
GCI	Ground Control Interface
GPIO	General Purpose Input/Output
GUI	Graphical User Interface
HD	HOTDOCK
I3DS	Integrated 3D Sensors
ICU	Instrument Control Unit
ID	Identifier
IEEE	Institute of Electrical and Electronics Engineers
IF	Interface
IP	Intellectual Property or Internet Protocol
JSON	JavaScript Object Notation
LVTTTL	Low Voltage Transistor Transistor Logic
MCC	Mission Control Centre
MOSAR	Modular Spacecraft Assembly and Reconfiguration
MPSoC	Multiprocessor System on Chip
OBC	On-Board Controller
OBC-C	OBC of the Client Spacecraft
OBC-S	OBC of the Servicer Spacecraft
OBCP	On-Board Control Procedures
OBSW	On-Board Software
OG	Operational Grant
OMPL	Open Motion Planning Library
OS	Operating System
OSP	Optical Sensor Payload
PC	Personal Computer-Aided
PCB	Printed Circuit Board
PCI	Peripheral Component Interconnect
PDDL	Planning Domain Description Language
PDU	Power Distribution Unit
PGCI	PUS Ground Control Interface
PRO-ACT	Planetary Robots Deployed for Assembly and Construction Tasks
PUS	Packet Utilization Standard
PWS	Power Subsystem
QoS	Quality of Service
R-ICU	Reduced Instrument Control Unit



Preliminary Design Document

RARM	Robotic Arm
RCOS	Robot Control Operating System
RGB	Red, Green, Blue
RID	Review Item Discrepancy
RMAP	Remote Memory Access Protocol
SI	Standard Interface
SIROM	Standard Interface for Robotic Manipulation of Payloads in Future Space Missions
SM	Spacecraft Module
SoC	System on Chip
SOD	SM Operations Decomposer
SpW	SpaceWire
SVC	Servicer Spacecraft
SW	Software
TASTE	The ASSERT Set of Tools for Engineering
TBC	To Be Confirmed
TBD	To Be Determined
TC	Telecommand
TCP	Transmission Control Protocol
THS	Thermal Subsystem
TM	Telemetry
TSP	Time and Space Partitioning
UART	Universal Asynchronous Receiver-Transmitter
UDP	User Datagram Protocol
URDF	Universal Robot Description Format
USB	Universal Serial Bus
WAN	Wide Area Network
WM	Walking Manipulator



2 MOSAR Demonstrator Introduction and Objectives

2.1 MOSAR Mission Overview

The technologies developed in MOSAR aim to support the development of fully modular and on-orbit reconfigurable spacecraft. This ambitious objective is declined into a demonstrative mission concept that allows the deployment of MOSAR technologies in the near/middle future.

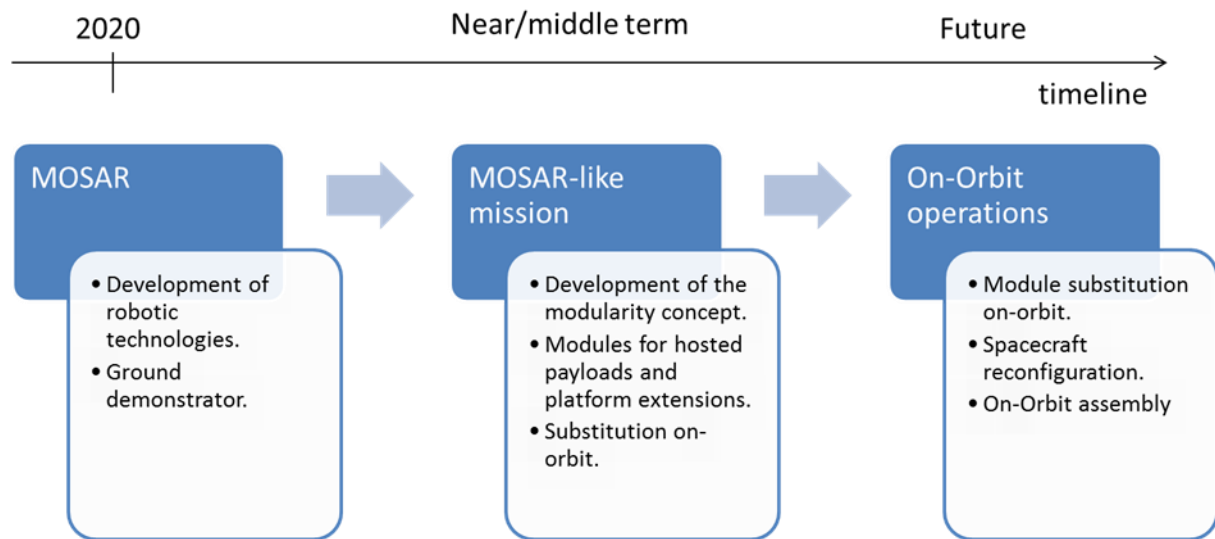


Figure 2-1: MOSAR roadmap to space application

The proposed demonstrative mission concept and spacecraft design are represented in the next figure. The baseline scenario is the one of a Servicer Spacecraft (SVC) transporting a cargo of Spacecraft Modules (SM) and a dedicated Walking Manipulator (WM). This last enables a number of operations with the transfer of SM from and to the Client Spacecraft (CLT), with the purpose of adding, replacing or enhancing client platform or payload functionalities.

The client satellite is built around a standardized common platform (product line), with all the required main functionalities. The platform is designed to cover an envelope of payloads and functions. It can be initially specialized, on-ground, with a primary payload and services modules. Both the platform and the payload are equipped with standard interfaces enabling the connection of the standard modules, for initial configuration on-ground or for addition or replacement of units during on-orbit mission.



Preliminary Design Document

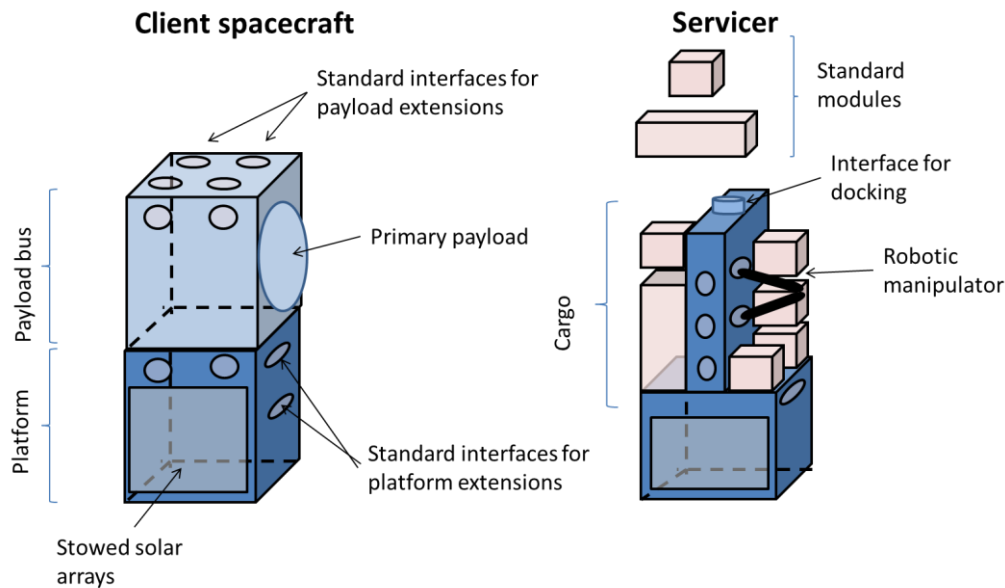


Figure 2-2: Spacecraft design MOSAR-like mission (client and servicer).

This mission concept aims at making use of MOSAR technologies to introduce enhanced functionalities on space missions, mainly on two topics:

- Standardization of design thanks to modularity approach: a common design of the product line allows multiple configurations to meet different mission objectives thanks to the standardized interfaces that could be used to plug additional payloads and service modules. In this way the customization needed to meet specific mission objectives does not need a redesign of the main elements of the spacecraft.
- The spacecraft, equipped with multiple standard interfaces and some modules, performs its original missions for some years with the option of introducing new elements or reconfiguring the mission by replacing or adding new modules brought by a servicer satellite equipped with a robotic manipulator.

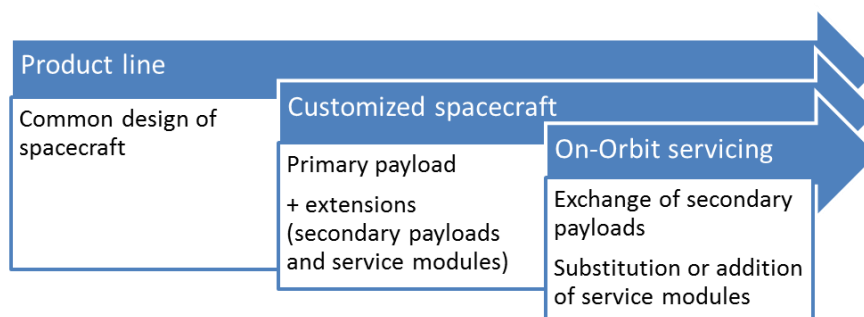


Figure 2-3: Life-cycle of the MOSAR mission concept

Figure 2-4 illustrates examples of different client spacecraft mission configurations.

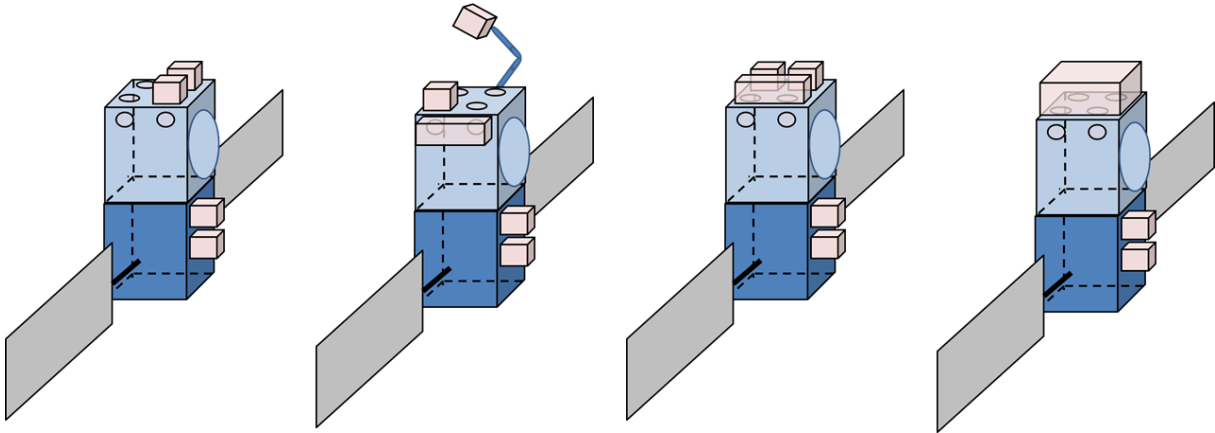


Figure 2-4: Spacecraft with different mission configurations.

2.2 MOSAR Demonstrator

MOSAR demonstrator aims at reproducing the main functionalities needed for on-orbit module assembly and reconfiguration of spacecraft, as described in the mission section above. Figure 2-5 illustrates the concept for the spatial mission, involving a client and a servicer satellite, already docked. The scope of the project covers the phases immediately after on-orbit rendez-vous of the space vehicles and docking; in particular:

- Connection of a new payload.
- Exchange of modules between client and servicer spacecraft.
- Assembly of modules interacting with other modules or with the spacecraft bus

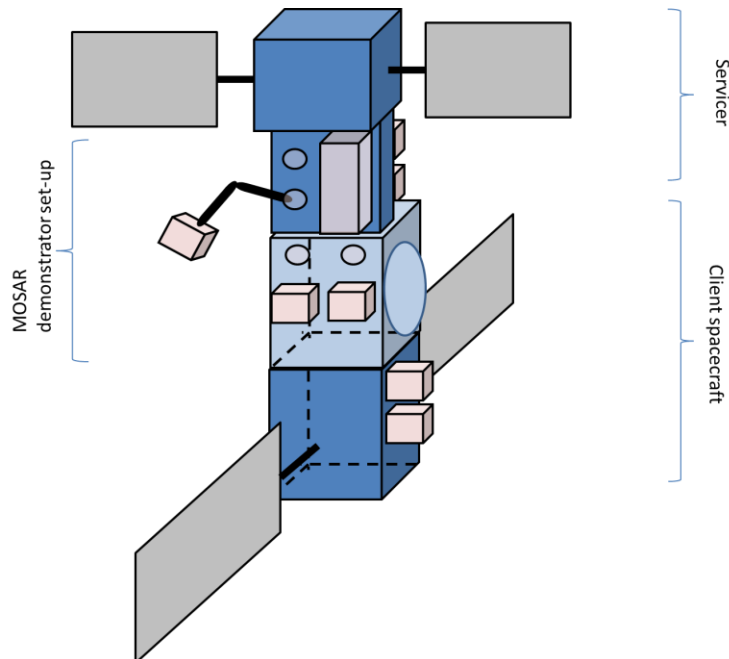


Figure 2-5: On-orbit reconfiguration with substitution and addition of modules



Preliminary Design Document

The ground demonstrator will be focusing in the validation of the sequence of operations needed for the preparation of the operations, the manipulation of the standard modules and the operations of the robotic manipulator. It will address the following topics related to modular and re-configurable spacecraft:

- Hardware: with the possibility to re-configure the physical arrangement of the spacecraft and/or providing means to add/replace/upgrade specific functions.
- Software: with the possibility to re-configure components responsibilities and support the re-configuration operations
- Data: with the possibility to re-route TM/TC and data transmission along the different elements
- Power: with the possibility to re-route and control the power transmission along the elements.

Figure 2-6 represents the main components of the MOSAR demonstrator that is split between the space and the ground segment. The space segment is representative of the middle part of the assembly in Figure 2-5. It includes the following hardware elements:

- A base plate with standard interfaces simulating the client spacecraft payload bus, where the modules are assembled.
- A second baseplate attached to the first one, simulating the cargo part of the servicer, where the modules are stocked until its deployment on client spacecraft.
- A robotic manipulator, originally brought by the servicer, with capacity to move around both satellites and manipulate modules.
- Several modules, simulating payloads and equipment that are exchanged between the servicer and the client.

Due to cost and schedule constraints the modules will not be fully representative of the spatial application in terms of size, shapes and materials. However, the main functionalities derived from the proposed mission scenario and the use of the standard interface will be covered by the different tests.

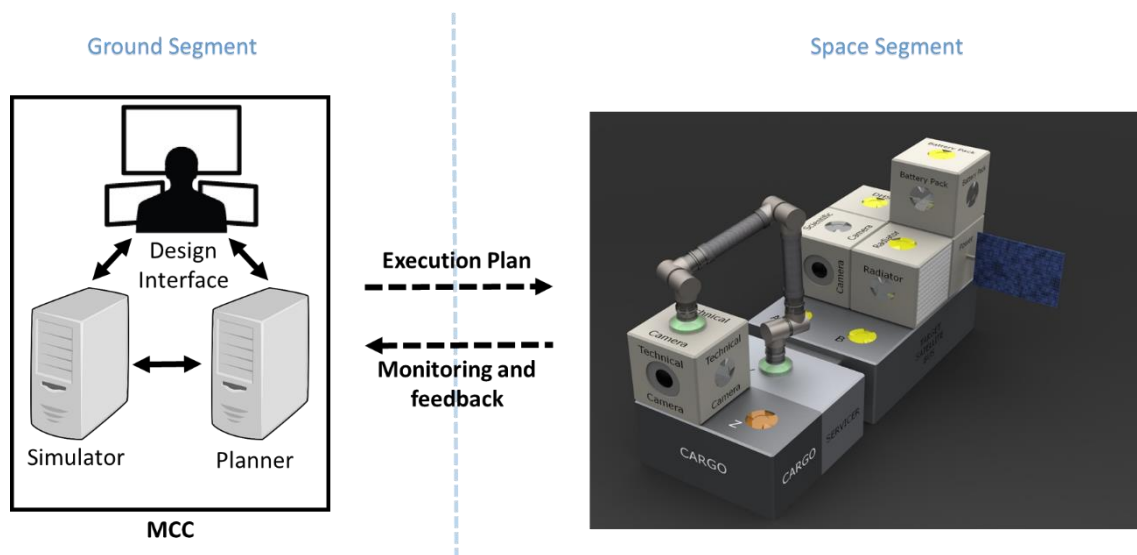


Figure 2-6: MOSAR Operational Concept

The autonomous transfer and configuration of the SM follow an execution plan prepared and validated off-line, in the Monitoring and Control Centre (MCC), on the ground segment. The MCC includes a satellite design, modelling and validation tool, specifically targeting modular satellites applications. It also allows the automatic planning of the assembly or reconfiguration sequence that can be verified with a multi-physics simulator. All these elements are working iteratively together to prepare a valid execution plan that is finally uploaded to the spacecraft for execution. Based on the monitoring and feedback information received from the spacecraft during the operations (e.g. detected failed module), the MCC



Preliminary Design Document

can update the execution plan. The MCC finally includes visualisation front-end to support the design, verification and monitoring activities during sequence execution.

The MOSAR demonstrator shall allow verifying and validating the following functionalities relevant for future modular spacecraft missions (with reference to the MOSAR mission's requirements [AD4]):

- Design and creation of a re-configuration execution plan (FuncR_S105)
- Simulation of the execution plan (FuncR_S106)
- Manipulation and repositioning of SM (FuncR_S101)
- Control and re-location of the WM (FuncR_S104, FuncR_S107)
- Update/upgrade of satellite functionalities (FuncR_S102)
- Data and power transfer between SM
- Resources re-allocation, data and power routing (FuncR_S110)
- Heat management between SM (FuncR_S115)
- Failure detection and handling (FuncR_S111), with automatic client network reconfiguration (without the support of the servicer or ground segment)

These functionalities will be demonstrated in MOSAR through different operational scenarios that are described in section 5.



3 MOSAR Demonstrator Components and System Architecture

3.1 Ground Segment

In the context of the MOSAR demonstrator scenario, the ground segment consists of a design tool, the ERGO Agent (including the planner) and Functional Layer, the MOSAR simulator and the Monitoring and Control interface. The architectural integration of these components is the same as in the Monitoring and Control Centre (Figure 2-6), introduced in the operational concept (see Chapter 2.1). Specific adaptations are required in the planner algorithms and in the libraries of the simulator models to consider the gravity conditions in the ground lab and the actually available MOSAR components in the demonstrator setup. However, these adaptations have no impact on the general system architecture and demonstrator mission procedure.

The simulator replaces the demonstrator space segment as introduced in 3.2 on a functional level. More specific, the simulator is a virtual demonstrator that is

- using the same input and output signal interface as used by the demonstrator space segment,
- using the same system components as used by the demonstrator space segment (see 3.2.1) to represent the system topology and its reconfigurations,
- providing the same manipulator skills and degrees of freedom for spacecraft module assembly as done by the demonstrator space segment,
- providing the same spacecraft module and HOTDOCK functionalities as the demonstrator space segment,
- providing power, thermal, and data network functionalities and logics to let the user analyse the system as done with the demonstrator space segment.

In terms of system architecture, apart from differences in the selected communication protocols, the simulator is fully transparent for the ERGO planner. The goal is to let the planner deliver the same results on algorithm level, independent from the attached system appearance.

3.2 Space Segment

3.2.1 MOSAR Space Segment Components

The MOSAR demonstrator space segment and its main components are illustrated in Figure 3-1. It includes:

- The servicer spacecraft (SVC) has the role to bring the building blocks and tools for the operations of (re)-configuration of the client satellite. It has its own on-board computer (OBC) that will manage all the operations of configurations (following the plan prepared by the ground segment). It also manage the communication with the monitoring and control centre (MCC), on ground, during the assembly operations.
- The client satellite bus (CLT) is the mechanical structure that supports the assembly of the client satellite. During (re)-configuration operations, the CLT is docked to the SVC and has a data interface with it, such that the SVC OBC can operate components on the CLT.
- The spacecraft modules (SM) are individual units proposing a specific function of the satellite, either as sub-system (e.g. OBC, power generator, batteries, thermal management) or as payload (e.g. camera). In the scenario of MOSAR, they take the shape of cubes that can be assembled.



Preliminary Design Document

- The walking manipulator (WM) is a symmetric robotic arm (both extremities can play the role of the robot base or the end-effector). It allows manipulating the SM between the SVC and CLT during assembly operations. It has also the capability to “walk” along the structures to be able to reach the different components.

Standard interfaces connectors (HOTDOCK) are used to inter-connect all the above components. They can provide mechanical, data, power and thermal interface between the modules, the spacecraft/bus and the walking manipulator (all combinations are supported).

The MOSAR demonstrator will also include an external visual 3D processing system that is used to analyse and demonstrate algorithms and techniques, that will support future autonomous re-configuration operations. This will mainly address detection of unexpected configuration, spacecraft deterioration and discrepancies, based on validation of modules, interfaces or manipulator shapes and features detection. At this stage, this feature will not be directly interfaced with the modular spacecraft operations.

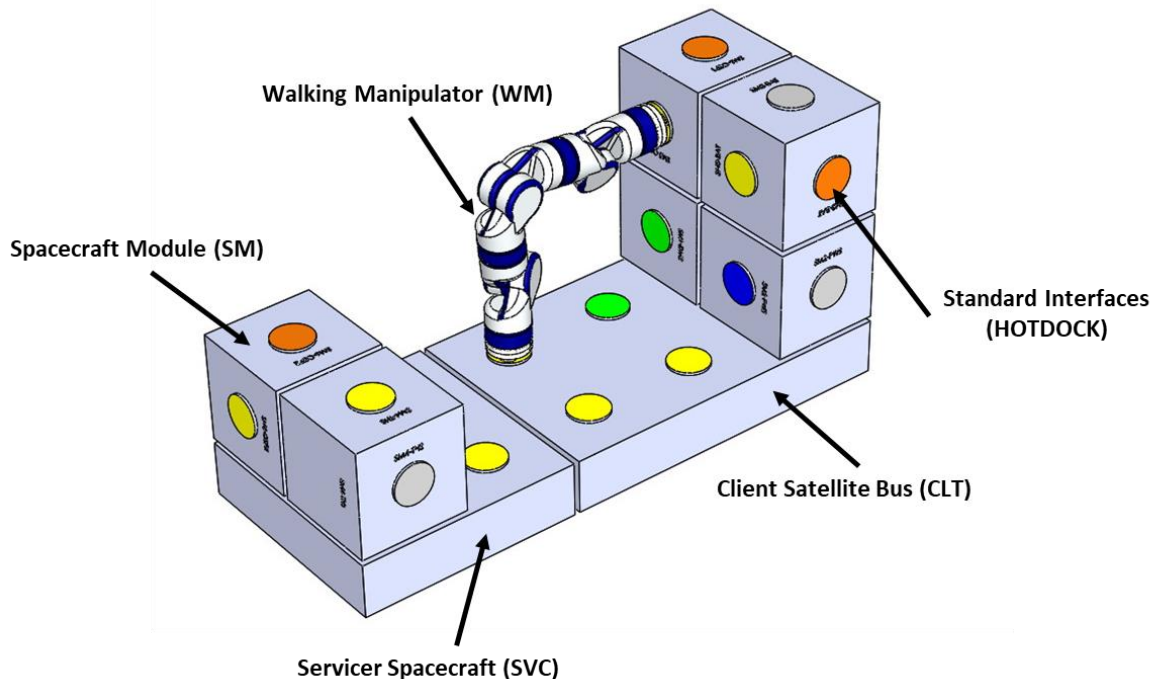


Figure 3-1: Main components of the MOSAR space segment

3.2.2 MOSAR Space Segment Generic Data and Power Architecture

The components of the system (SM, WM, SVC and CLT) present a similar architecture for the powering and control of their elements (SI, payload or PDU). The corresponding data and power architecture is illustrated in Figure 3-2. We can identify the following elements.

3.2.2.1 On-Board Computers (OBC)

The OBC runs the software that will manage the high-level functions of the spacecraft during assembly/reconfiguration or during nominal operations. The setup will include two OBCs, the OBC-S located in the SVC (mainly dedicated to the reconfiguration operations), and the OBC-C as payload of one of the SM on the CLT (more dedicated to the spacecraft nominal operations).



Preliminary Design Document

The OBC-S has the following software components:

- The Autonomy Agent that manages the execution of the other components according to the plan execution.
- The Telemetry and Telecommand (TCC) Service that provides the capability to command the Agent from the MCC and sent back telemetry of the operations. The TTC Service also allows for commanding at individual component level, for testing purposes and anomaly resolution.
- The HOTDOCK Management that provides high level control and telemetry of the SI during reconfiguration (with the purpose to provide more intelligence in the local components, as described below).
- The Data and Power Management to support, respectively, data (by commanding R-ICU) and power (by commanding cPDU) re-configuration during the assembly process.
- The Client Management to manage the states of the CLT during reconfiguration. Before starting the robotic operation, the CLT must transition to a safe mode and the OBC-C will hand over control of the functions needed for by reconfiguration to the OBC-S. At the end of the operation, the process is reversed and the OBC-C restarts nominal operations in the new configuration. The Client Management component arbitrates these transitions.
- The Walking Manipulator Management for the high-level control and telemetry of the WM during the reconfiguration operations (the WM is always considered as part of the SVC, although it can be re-localized physically on the CLT).

The OBC-C, more oriented to the CLT nominal operations, has the following software components:

- The Telemetry and Telecommand (TCC) Service for commanding and monitoring of the SMs during nominal operations.
- The Reconfiguration Management that manages the modes of the software, according to which SMs are available.
- The Data and Power Management to support data and power routing during nominal operations (e.g. for FDIR management, as faulty node isolation).
- The Payload Management for the operations of the SMs payloads (e.g. thermal, battery,...)
- The Payload relay to support the transfer of large size data (e.g. video images).

3.2.2.2 R-ICU

The R-ICU is derived from the ICU developed in I3DS. Located in each SM and in the two spacecraft (SVC and CLT), it has mainly two roles:

- The R-ICU controller will interface and control the different elements of the SM or spacecraft bus. That includes the HOTDOCK SI, the Power Distribution Unit (see below) and the SM payloads. The R-ICU receives high-level commands from the OBCs and translate them in low level control sequence. Having this local intelligence in the SM reduces the number of data transfer and allows the OBC to stay at a higher abstraction level in regards to the command control, improving the independence from the SM possible evolutions.
- The R-ICU SpaceWire Router manages the SpW routing and data communication functions (see below for the description of the SpW bus).

It has to be noted that the Walking Manipulator doesn't include an R-ICU due to volume constraints. The R-ICU is then replaced by a local controller associated with a SpW conversion unit (from USB to SpW), to be able to connect to the MOSAR SpW bus.

3.2.2.3 cPDU

The central power distribution unit (cPDU) is interfaced with the main power bus of the spacecraft through the HOTDOCK. It has the function of routing of the power bus with the other HOTDOCK of the



Preliminary Design Document

component. It also converts and provides the required power/voltage to the other elements (R-ICU, payload, WM joints...).

3.2.2.4 HOTDOCK Standard Interface

Each component has a set of HOTDOCK interfaces that provide a pass-through for the main power bus and the SpW data bus, respectively connected to the cPDU and the R-ICU. On top of that the SI ensures the mechanical connections between the different components.

3.2.2.5 Payloads

The payload is specific to each component and is interfaced with the R-ICU for command, telemetry and data transfer. In the case of the WM, this corresponds to the Joint Controllers. For the SM, in MOSAR battery, thermal and camera payloads will be integrated.

3.2.2.6 Communication Links

The MOSAR demonstrator implements a SpaceWire network between the different components (modules, WM, spacecraft), passing by the HOTDOCK interfaces. The communication is enabled by the SpaceWire routers in the R-ICUs. They direct the SpW packets according to the topology of modules as defined by the OBC and the planner. The Figure 3-2 highlights the SpW network between the OBCs and the R-ICU modules. In the case of the WM, it is not, by nature, permanently attached to one element, but will always connect through one of the R-ICU router to be integrated inside the network (the WM himself can play the role of a SpW relay between two SM). The two OBCs are directly connected through SpW to represent the data link of the docking interface.

In order to enable the communication between the SM and the OBC, the RMAP protocol is proposed. RMAP enables memory access over a SpaceWire network with a defined packet structure. The commands sent by the OBC through RMAP are directly written to the R-ICU memory, and the data to be sent back will be placed in memory and retrieved by the OBC. The control software, running on the R-ICU will get access to the memory to enable the control and telemetry of the connected elements (SI, cPDU and payloads).

The CAN protocol is proposed as the standard connection between the R-ICU and the elements, unless the connected item has specific interface (e.g. with a camera). This is the nominal interface for the control of HOTDOCK. For the other elements developed in the course of the project, this approach will simplify the development and integration of them, as a common CAN compatible controller can be developed (also required for local control of the element).

The communication between the Monitoring PC (Ground) and the OBC uses generic mechanisms:

- For general monitoring and command purposes, the TASTE middleware (PolyORB-HI) is used. The TM/TC services of the Client spacecraft (including the SMs) are based in the ESROCOS PUS Services library.
- For large data transfer, a specific mechanism is needed. The ZeroMQ protocol is selected for this purpose. The ZeroMQ library is selected for commonality with the I3DS framework. I3DS uses ZeroMQ for communication with the ICU and transport of the data (both raw sensor data and metadata encoded in ASN.1). The deployment of ZeroMQ in the ICU and the encoding of the data from the optical sensors (which are selected from the I3DS sensor suite) have therefore been tested in the previous project.



Preliminary Design Document

3.2.2.7 Main Power Bus

The main power bus voltage has been currently selected at 28V, as it is the closest ECSS standard voltage (for systems up to 1.5kW) and is above all elements required voltage (TBC for the walking manipulator).

3.2.2.8 Generic Data and Power Deployment Diagrams

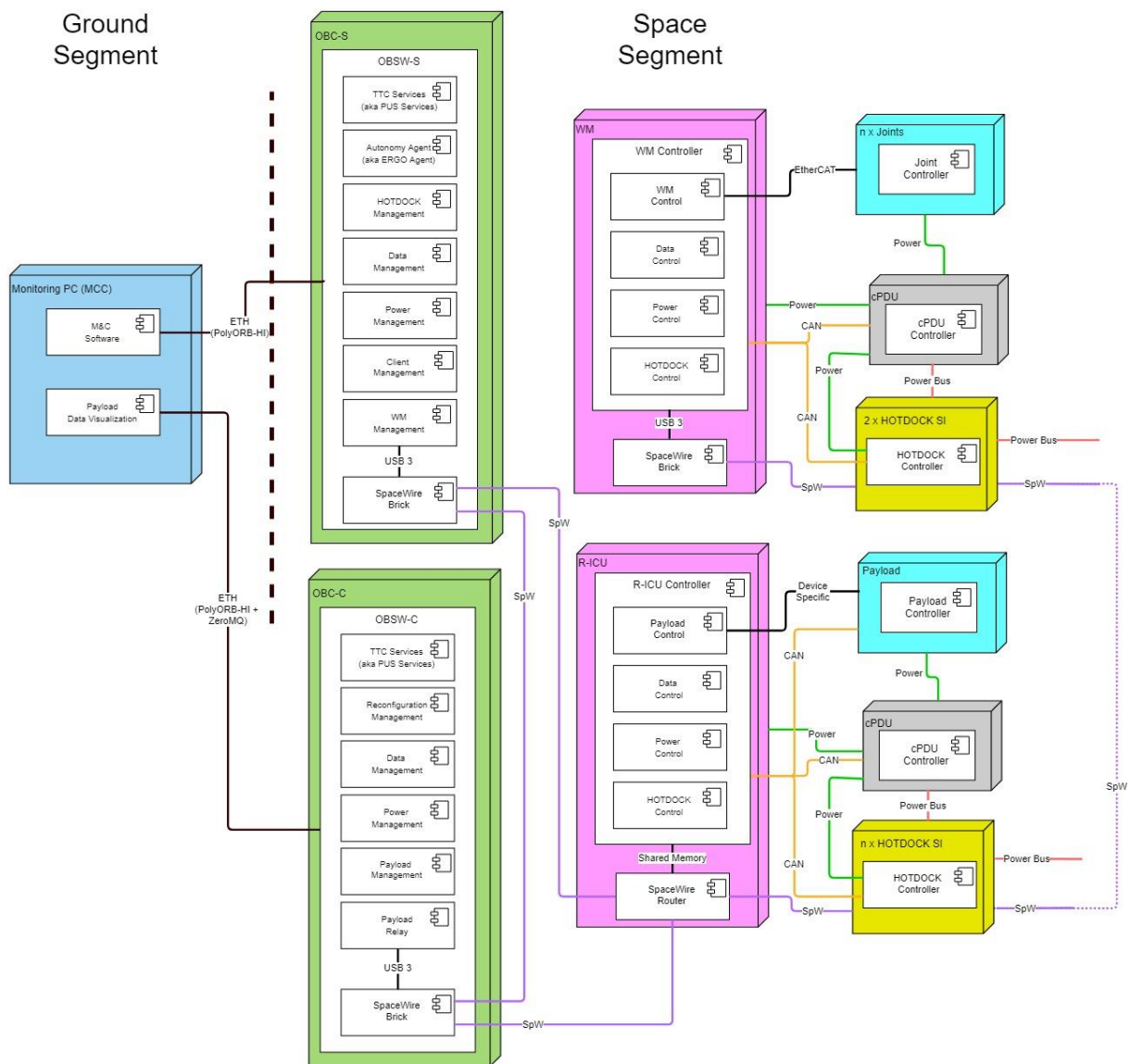


Figure 3-2: MOSAR generic data and power architecture



3.3 MOSAR Product Tree Responsibilities

Figure 3-3 illustrates the global MOSAR demonstrator product tree, highlighting the main components of the setup. Table 3-1 provides for the different components the design responsibilities among partners. Table 3-2 defines the main software responsibilities. Specific contributions from other partners can be required but are not displayed in the table.

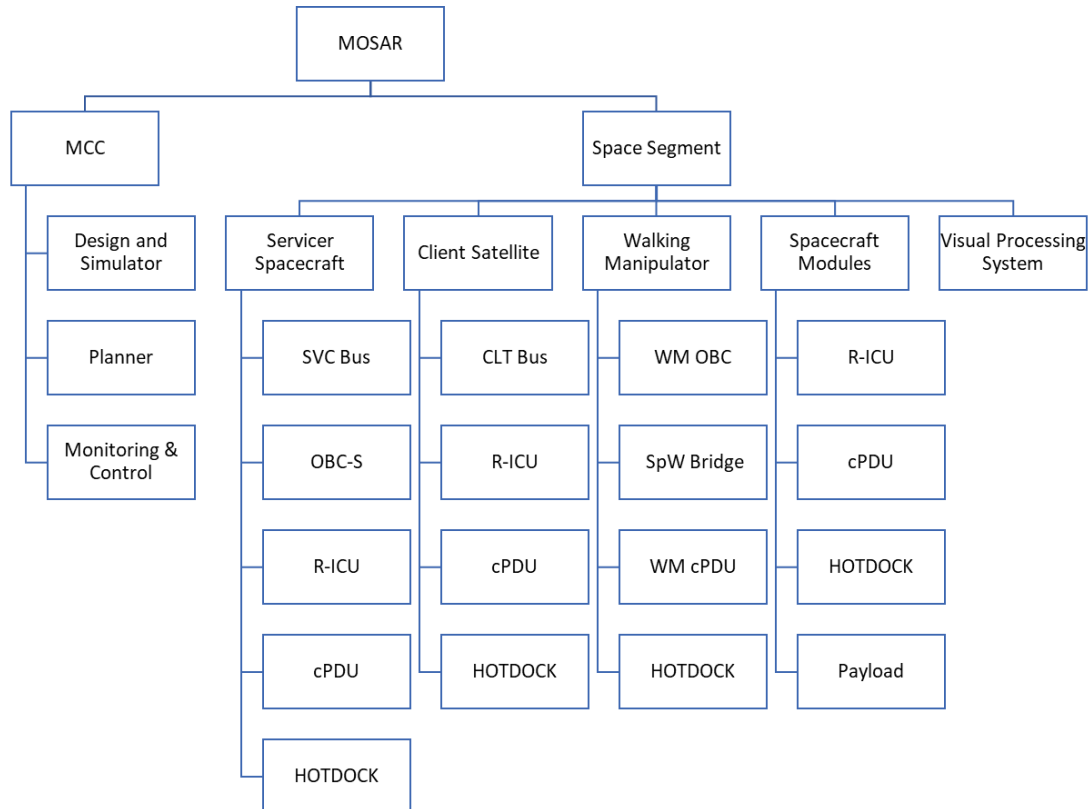


Figure 3-3: MOSAR demonstrator product tree



Preliminary Design Document

Table 3-1: Components design responsibilities

Component	Responsible Partner
MCC Design and Simulator Tool	DLR
MCC Planner	GMV
Monitoring and Control	GMV / DLR
SVC and CLT buses	SITAEL
OBC-S	SPACEAPPS
R-ICU	TAS-UK
cPDU	SPACEAPPS
HOTDOCK	SPACEAPPS + MAGSOAR (thermal interface)
Walking Manipulator	SPACEAPPS
Spacecraft Modules	SITAEL
Spacecraft Modules Payloads	DMS – OBC: SPACEAPPS SM2 – PWS: MAGSOAR (thermal subsystem) SM3 – Battery: SPACEAPPS SM4 – THS: MAGSOAR (thermal subsystem) SM5/SM6 – Optical: TAS-F
Visual Processing System	USTRATH

Table 3-2: MOSAR software main responsibilities

Software	Responsible Partner
MCC Design and Simulator Software	DLR
MCC Planner Software	GMV
Monitoring and Control Software	DLR + GMV (TTC service)
OBC Software	GMV + ELLIDISS (TTC service, Autonomy Agent, Component Management)
R-ICU Software	TAS-UK + SPACEAPPS (support for components control)
cPDU Controller software	SPACEAPPS
HOTDOCK Controller software	SPACEAPPS
Walking Manipulator High-Level software	DLR
Walking Manipulator Low-Level software	SPACEAPPS
Spacecraft Modules Payloads software	See above
Visual Processing System software	USTRATH



4 MOSAR System Reconfiguration

4.1 Hardware System Reconfiguration

4.1.1 Mechanical Reconfiguration

The standard interfaces, embedded in each spacecraft module, ensure the mechanical re-configuration capability of the spacecraft. This includes the mechanical connection as well as the physical harnessing for data, power and thermal transfer.

Mechanical re-configuration allows placing, theoretically, each module in different positions and orientations. This is of course still limited to module constraints, including payload specific face (e.g. camera) or relative required connections between two modules.

4.1.2 Data – SpW Network Reconfiguration

4.1.2.1 SpW Network Reconfiguration (Servicer attached)

SpaceWire network reconfiguration is done from the SVC OBC using RMAP to update the routing tables when nodes are added/removed. There are several distinct scenarios to cover:

1. *Initial network discovery:* The SVC needs to find out the structure of the SpaceWire network and what devices are attached. To achieve this, a SpaceWire based network discovery service will be supported. The SpaceWire-PlugNPlay extension and its follow up project SPACEMAN have added this capability to SpaceWire in the past and will be evaluated for its suitability to MOSAR. Both of these projects added device identification and connect/disconnect event notification to the router design. For MOSAR, during the construction phase, the intention is for the OBC to have full control of the network traffic using RMAP commands. The OBC will query and configure each router to build a network map of device capabilities across the SpaceWire network and manage new devices being attached or detached to the network.
2. *Update of routing tables on addition/removal of modules:* Each unique node on the SpaceWire network will be allocated a unique SpaceWire Logical address as part of the network discovery process. Any node will be able to communicate to any other node using RMAP commands and the Logical address of the destination node. When a Spacecraft Module is added or removed, the routing tables in each R-ICU's routing table will be updated to reflect this change. By updating all routing tables on device connection/disconnection, the SpaceWire network will know the path to any other node. The SVC-OBC will update the router tables via RMAP and the updated network tree will be maintained by the SVC-OBC. SpaceWire network segmentation makes this approach scalable to networks significantly larger than the one required by MOSAR with 1000's of nodes supported.
3. *Support for walking manipulator:* As the walking manipulator moves across the SVC and CLT the SVC-OBC control for it will need to ensure that the correct SpaceWire port is used on the walking manipulator (i.e. the docked port). As this movement is planned it will be known at all times which port on the WM is currently docked. Use of SpaceWire path addressing in conjunction with the network map and WM planning will allow the SVC-OBC to address the correct WM port. Path addressing would also remove the need to continuously update the routing tables as the WM moves across the spacecraft and as only the SVC-OBC needs to access the WM there is no advantage to having a routing table entry that all nodes can use to address the WM.



4.1.2.2 SpW Network Online Reconfiguration (Servicer not attached)

Once the spacecraft is assembled and the servicer has disconnected from the spacecraft there should be no reorganisation of the SMs and so no need for reconfiguring the SpaceWire network to detect new modules or modules that may have been moved.

However there is a need to reconfigure the network routing to provide autonomous fault tolerance by exploiting the redundant paths inherent in the MOSAR grid-style architecture. This process will be driven by the CLT-OBC and follows the following sequence:

1. Fault detection – The CLT-OBC initiates all communications on the spacecraft and uses RMAP commands with a reply requested. If an RMAP command is stalled, lost or corrupted by a link failure then the reply will not be received correctly and the CLT-OBC will assume that a link error has caused this fault.
2. Network rediscovery – Once a fault has occurred the network topology will need to be rediscovered to find the faulty link and remove it from the network topology. This will also use SpW PnP as used by the SVC-OBC.
3. Network mapping – The updated network topology is then analysed to produce valid routing paths between the OBC and each R-ICU. A basic graph traversal algorithm (e.g. Dijkstra's) for finding shortest paths is used as the network size is small.
4. Routing table update – The routers of each R-ICU are updated with the outcome of the network mapping stage. Once this is complete the system can then return to nominal operation.

It can be seen that here will be scenarios which the system cannot recover from e.g. failure of the link on a single network path SM. For the purpose of the MOSAR demonstrators this is considered out of scope of the project as it would likely require intervention from a servicer spacecraft to remedy.

4.1.3 Power Reconfiguration - Distribution

In MOSAR, power distribution is achieved by the transfer of the electrical energy between the SM through the connected HOTDOCK interfaces. In order to configure the power distribution, each SM and the WM is equipped with a centralized power (and conditioning) distribution unit (cPDU), that provides the following functions:

- Power distribution between the HOTDOCK interfaces of the SM / WM
- DC/DC conversion from the main power bus to power the internal components of the SM/WM

The cPDU is the main component implicated in the power re-configuration operations that are used during the spacecraft re-configuration (managed by the OBC-S) or during nominal operations (managed by the OBC-C, e.g. for SM isolation in case of failure). The cPDU architecture is illustrated in Figure 4-2.



Preliminary Design Document

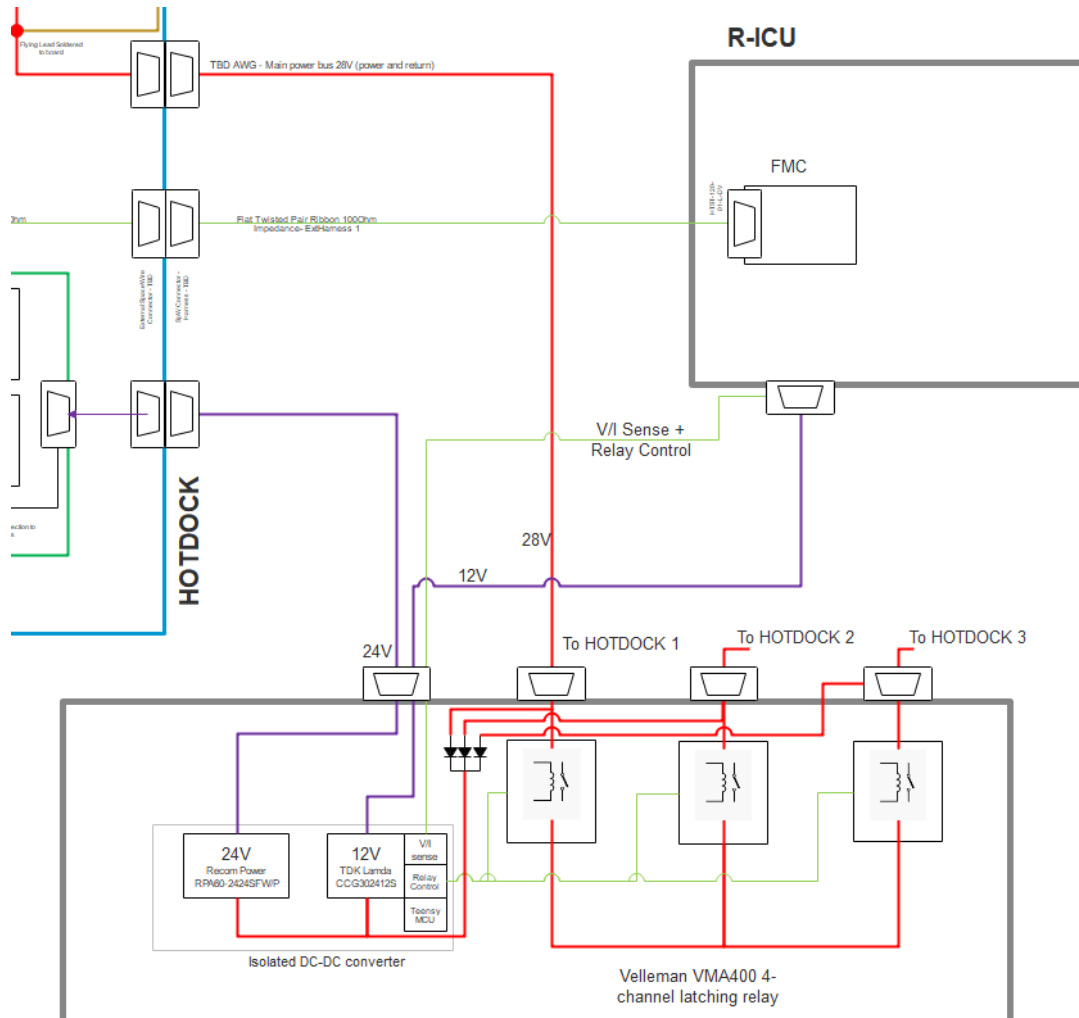


Figure 4-1: cPDU architecture

The power going inside from one of the HOTDOCK SI is distributed via COTS relays. These relays are controlled by the cPDU controller, that is interfacing the R-ICU Control software by CAN bus. The R-ICU receives the commands from the OBC-S/C, using the RMAP protocol on the SpW bus (see section 5.6). By this way, each SM is capable to redirect the power distribution, as planned in the system configuration.

In order to be able to immediately power-on the R-ICU and HOTDOCK controller, all HOTDOCK power input lines are directly connected to the DC/DC convertors to feed the internal components. However, set of diodes avoid propagating directly the power to the outputs of the other HOTDOCKs in the spacecraft module.

It is also envisaged to add voltage and current monitoring on the power lines. These telemetry signals will be sensed by cPDU controller ADCs and forwarded as CAN messages to the R-ICU that can further distribute them over SpW to the OBC-S/C and MCC software (for data monitoring and visualization).

The MOSAR main bus voltage has been selected to 28V, which is the lowest ECSS standard bus voltage for systems lower than 1.5kW. This is compatible with the power transmission on HOTDOCK



Preliminary Design Document

(tested up to 100V). Moreover, this removes the strain from electronics interfacing the bus and eases preliminary design process.

All power converters are isolated to improve the grounding scheme of connected spacecraft modules. Isolating the input power rail (HOTDOCK 28V) from 12V/24V lines will remove the dependence of return paths between main 28V bus and local power. In addition, signal return paths are routed separately and connected through dedicated pins on HOTDOCK interface. This achieves better signal integrity and operation of SpaceWire which has low common mode voltage specifications.

Table 4-1 provides the software components and TM/TC related to the cPDU operations. The software components are defined according to the components used in Figure 3-2.

Table 4-1: cPDU parameters

cPDU Operations	Context: Spacecraft reconfiguration Spacecraft nominal operations
Software Components	
<ul style="list-style-type: none">• <u>MCC</u>: M&C Software (including GUI interface of the SM)• <u>OBC-C</u>: Power Management• <u>R-ICU / WM Controller</u>: Power Control• <u>Payload Controller (Component)</u>: cPDU Controller	
TM	
<ul style="list-style-type: none">• cPDU Control / Mode Status (Idle, Operational, Error)• cPDU Control / Power Line Voltage [6]• cPDU Control / Power Line Current [6]	
TC	
<ul style="list-style-type: none">• cPDU Control / Mode Command (Idle, Operational)• cPDU Control / Power Switch (Channel, On/Off)	
Data	
<ul style="list-style-type: none">• N/A	



4.2 Software System Reconfiguration

In this section, we investigate software reconfiguration of OBCs for modular satellites as the MOSAR architecture and beyond. Here, we assume satellites composed of one or several OBCs connected by a network such as SpaceWire [18]. We assume that OBCs can be moved from one location to any position in the network. Furthermore, we also assume that functions hosted by any OBC can be moved to another OBC at reconfiguration time.

We investigate reconfiguration mechanisms provided by operating systems (OS for Operating Systems) that stakeholders use when implementing OBCs. The benefits and drawbacks of each OS are discussed. This analysis allows us to derive possible software reconfiguration options for TASTE and some preliminary design elements for the MOSAR demonstrators.

The remainder of this part is organized as follows. First, we provide possible scenarios for software reconfiguration under MOSAR. In section 4.2.1, we detail the node reconfiguration process in the context of the MOSAR project. Section 4.2.2 discusses reconfigurations options, tools and mechanisms provided by existing operating systems. We actually illustrate in 4.2.3 by a reconfiguration scenario with RTEMS and AIR and we give elements for the preliminary design of the MOSAR demonstrators in 4.2.4.

Document references are provided in annex Software Reconfiguration Documents References (section 7.3).

4.2.1 Types of software reconfiguration for MOSAR

MOSAR architecture is composed by a set of nodes connected by a switch. Nodes can be either payloads or OBCs. We assume the system may be composed of several OBCs node. Under MOSAR architecture, there are two possible scenarios for software reconfiguration:

- The MOSAR 1 scenario. In this scenario, stakeholders may want to improve performances of the satellite. Improving performances can be achieved either by adding a new OBC or a new payload. Such MOSAR extension may require to reconfigure software components of the nodes but also to change network parameters. Furthermore, this scenario also requires to start and to stop properly the services/applications of the node before and after software reconfiguration.
- The MOSAR 2 scenario. In this scenario, stakeholders may want to recover failures of an OBC or a payload. Once detected as faulty, OBC or payload may be replaced in the same network location or at any place to enforce the same level of satellite services. Then, this scenario:
 - 1) may not require to software reconfiguration of the node since the software replacing is the same than the faulty one
 - 2) requires to start and to stop properly the services/applications of the node before and after reconfiguration
 - 3) may require to change network parameters if the node is replaced on a different network location during reconfiguration.

Figure 4-2 presents the steps of a reconfiguration process under MOSAR architecture. The two MOSAR scenarios above require that the MOSAR architecture is able to both make software reconfiguration for concerned nodes and also to update network parameters. When a reconfiguration starts, MOSAR first cope with the node by either reconfiguring its software or by adding the new node. Then, as shown by Figure 4-2, it also requires an update of network parameters. Node reconfiguration is described in the sequel. For the network parameters update, it may consist to change routing table, MAC or transport addresses depending on the new locations of the updated nodes. On reconfiguration end, the MOSAR system is considered as reconfigured and should operate as expected.



Preliminary Design Document

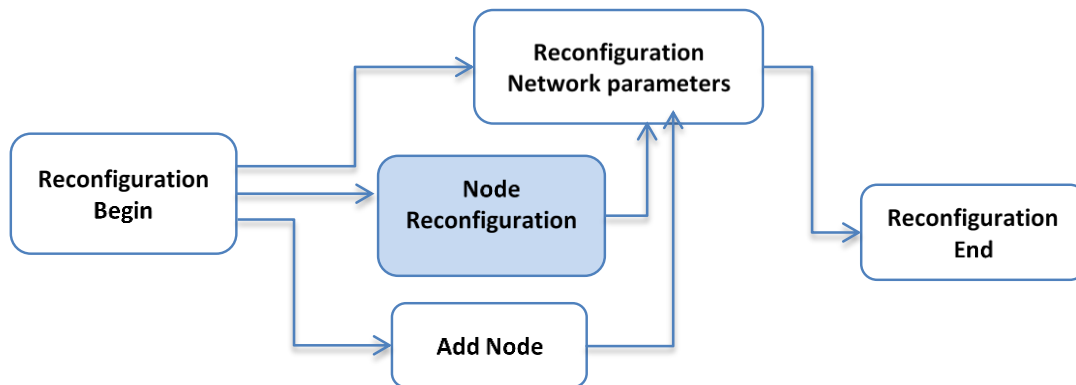


Figure 4-2: Software reconfiguration process

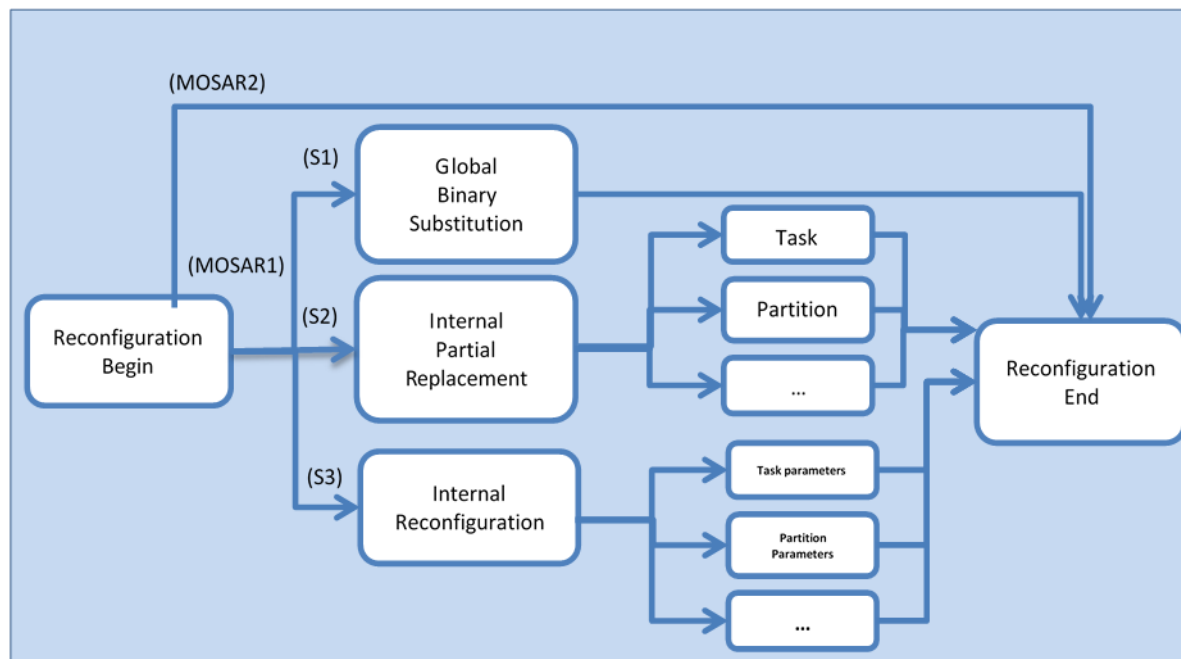


Figure 4-3: Various options for the node

We call a node software reconfiguration an update of the software component of the node, which can be either an OBC or a payload. Figure 4-3 presents 3 possible options for software reconfiguration of a MOSAR node. For each software reconfiguration option, we investigate if the proposed solutions always allow designers to do early verification or/and to enforce at runtime nonfunctional properties of the services/applications such as safety, performance or schedulability [18]. Software reconfiguration can be achieved with the three following ways:

1. Global Binary Substitution (or GBS in the sequel, see scenario S1 in Figure 4-3). In this option, the binary code of the OBC is fully replaced by a new version. In this case, we need to shut down all services and applications of the node, which leads to stop all flows of control and releasing any of the resources used by them. Depending on the technology stakeholders use



to implement MOSAR nodes, services/applications can be composed of POSIX threads [8], RTOS tasks [10] (RTOS for Real-Time Operating systems) or Unix processes [9]. Shared resources can be semaphores or queues such as Unix IPC. With such reconfiguration option, all services/applications managed by the node must be stopped during reconfigurations: then tasks/processes and their shared resources must be stopped and released properly. Once the new binary is monolithically loaded in the target, Global Binary Substitution is completed by the system switched on. With this reconfiguration option, verification of the nonfunctional properties of the MOSAR services/applications can be fully investigated offline, before runtime.

2. Internal Partial Replacement (or IPR in the sequel, see scenario S2 in Figure 4-3). In such option, only a part of the software components is replaced. Replaced parts can be dynamic libraries, TSP partitions, bunches of data, pieces of code or any elements of the services/applications run by the MOSAR node. Again, before updating software components, services/applications of the node must be stopped but comparing to the GBS option, with IPR, there is no need to stop all services/applications of the satellite. Then, some of the satellite services do not have to be interrupted during reconfiguration with such option. As with scenario 2, once software components are loaded, applications/services that were stopped on the node have to be launched again. The IPR option is more flexible and reduces unavailability time of the services/applications of the node, however, analysis of the nonfunctional properties (e.g. schedulability [18]) is more complex to achieve.
3. Internal Reconfiguration (or IR in the sequel, see scenario S3 in Figure 4-3). In this last option, no new software component has to be reloaded on the node during reconfiguration since we do not update the software components but we only change configuration of them. With IR, we assume that a reconfiguration is limited to the modification of software parameters of the node. We assume that all parameters of the software component of the node can be grouped as a concept called operational mode. An operational mode is the configuration parameters of all the components of the system. An operational mode may specify the list of the active tasks, their priorities and how the tasks will be scheduled, the list of needed resources such as the buffer accessed by the tasks, ... At reconfiguration time, the MOSAR node may have to move from a given configuration/operational mode to another. All configurations/operational modes are supposed to be known prior execution time and then, nonfunctional properties can be assessed before execution time as with the GBS option.

We note that the 3 options described above can be applied to the 2 MOSAR scenarios.

4.2.2 Operating systems features to implement node reconfiguration

Software reconfiguration will require operations at various system levels depending on the execution platforms used to implement TASTE nodes. We introduce in this section options, tools and mechanisms of execution platforms that are available to implement software reconfiguration. Those mechanisms, e.g. task management or memory management, introduce different concepts such as modularity. For example, from a computing point of view, software modularity required by MOSAR concepts can be brought by the notions of task, process or partitions as illustrated with the following execution platforms:

- Bare-metal technologies [16], such as MicroPython, have no operating system. Applications directly access to the hardware resources of the execution platform. Such technologies have limited services to abstract hardware resources but lead to high predictable systems. Then, they usually provide few reconfiguration features. Modularity in such execution platform is usually represented by pieces of executable source (i.e. C functions, scripts ...) that is compiled and linked with the overall system.
- Real Time Unix systems, such as QNX [20], Lynx or RT Linux [21]. As classical Unix systems, the execution platform is organized in two levels: the privileged level and the user level. Privileged level hosts critical features of the system, manages resources and enforces



Preliminary Design Document

application isolation while user level hosts applications. The behavior of such execution platforms is usually less predictable, but they provide safer and easier means to implement reconfiguration policies at user level. Modularity is brought here by the concept of process, which is composed of a program and its execution context. Process is also the mean to enforce safety by isolation. The same considerations apply to plain Linux systems, which do not provide the guarantees needed by critical software but are often used in laboratory setups.

- RTOS (Real-Time Operating System), such as VxWorks [2] or RTEMS [19]. These execution platforms have only one address space hosting both the applications and the operating system functions, i.e. they do not have user and privileged levels. It is a tradeoff between Unix and bare-metal technologies: RTOS provide a higher level of predictability but do not enforce isolation of the applications. Such execution platforms may provide mechanisms to implement reconfiguration policy. The modularity concept here is the task or the thread.
- Finally, systems based on « hypervisor » such as Micro-kernel [17], virtualized based systems [14,15] or Time and Partitioned Systems such as AIR [3] or Deos [13] are usually composed of more than 2 levels. Each level provides its concepts of modularity. For example, TSP has a hypervisor level which enforces isolation of higher level containers called “Partitions”. Partitions are both timely and spatially isolated: each partition has its own address space and is run at different times thanks to a partition scheduling computed off-line. At partition level, each partition has an operating system that manages processes located within the partition. Then, the concept of modularity in TSP systems is brought by the notions of both partition and process.

In the sequel, we give a short overview of some mechanisms that are involved during a reconfiguration: task management, memory management, power management and device management.

4.2.2.1 Task management

This part of the execution platform controls the execution of one or more programs and their corresponding flows of control within the computer.

4.2.2.2 Memory management

The memory management is the part of the execution platform that keeps track of the status of each memory location (i.e. allocated or free). It determines how memory is allocated among competing flows of control and may isolate memory access to enforce safety.

4.2.2.3 Power management

This part of the execution platform controls all the devices within the system from a power consumption point of view. It decides which ones are needed by the applications and then, which must be switched on or switched off.

4.2.2.4 Devices and input/output management

This part of the execution platform manages all the I/O hardware devices of the onboard computer. For example, it manages the storage device as well as network units.

4.2.2.5 Fault management: fault detection, isolation, and recovery (FDIR)

Some execution platforms may provide services to monitor, analyze and actually identify faults in the system. Once detected, the faults may be recovered. Those mechanisms are usually called FDIR service, for Fault detection, isolation, and recovery service.



4.2.2.6 Summary and analysis

In Table 4-2 we summarize mechanisms involved in software reconfiguration for several execution environments: Micro-Python, RTEMS, QNX and AIR.

Micro-Python does not provide ready to use mechanisms which are useful for software reconfiguration [1] contrary to RTEMS and AIR. Most of reconfiguration operations must be implemented by hand by the programmers. Modularity in Micro-Python is Python script.

RTEMS is a real-time operating system (RTOS) [19]. RTEMS is used to implement computing systems in the space and aircraft domains. Modularity concepts of RTEMS are tasks sharing resources such as semaphores or queueing buffers. RTEMS schedules tasks using priority pre-emptive scheduling. Several kinds of semaphores allow RTEMS programmers to implement synchronizations between tasks. Modularity is C functions and tasks.

ARINC 653 is a software specification for space and time partitioning in safety-critical real-time operating systems [12]. AIR ARINC 653 is an operating system designed according to ARINC 653 standard. It includes the AIR Partition Management kernel (PMK), which is a simple microkernel that efficiently handles partition scheduling, dispatching, and inter-partition communication. AIR architecture fully implements the ARINC 653 application executive interface (APEX) [3] including an implantation of the ARINC 653 Health monitoring which can be used to implement a FDIR service. With AIR, granularity reconfiguration is the partition and possibly the process.

To conclude, RTEMS and AIR provide most of the required features to implement node software reconfiguration of a satellite system based on the MOSAR concepts. In the next section, we illustrate MOSAR scenarios with two examples of execution platforms: RTEMS RTOS and AIR.



Preliminary Design Document

Table 4-2: Summary of the software reconfiguration features by execution platform

Execution Platform			Concepts for modularity	Task/Partition Management
Real Time Unix	QNX		Process Thread	Create thread (pthread_create()) States: Ready, Running, Blocked, Waiting
Time and Spatial partitioning	Air ARINC 653		Partition, Application, Process	Processes are created with CREATE_PROCESS service and started with START service. Each partition is activated with NORMAL mode using SET_PARTITION_MODE service. Process states: dormant, ready, waiting or running Partition state : COLD_START; WARM_START; NORMAL; IDLE
RTOS single Address Space	RTEMS		Process Thread	Create, start, suspend tasks States : Pended, Ready, Running, Blocked
Bare metal	Micro Python		Script	Built by designer. Handmade concurrency



Preliminary Design Document

Task/Partition Scheduling	Memory Management	Power Management	Device management
POSIX scheduling : preemptive fixed priority scheduling APS (adaptive partition scheduling) for group of threads	MMU and Unix memory management	Your own policy	Drivers and OS modules that can be loaded or unloaded during execution time
Spatial and Temporal Partitioning between partitions: Off Line for Partition Scheduling (AIR PMK Partition Dispatcher & Scheduler) On Line for Process Scheduling (priority preemptive scheduling)	Spatial and Temporal Partitioning between partitions: memory is statically allocated for each partition	AIR Health Monitor	AIR Health Monitor
POSIX scheduling : preemptive fixed priority scheduling User-defined scheduling	With or without MMU	I/O manager	
Built by designer. Handmade scheduling	Built by designer	API available but handmade power management	



Preliminary Design Document

Local communications	Network services
Unix Inter-Process Communications services (shared memory, queues, semaphores) Exchanging message through the micro kernel Inter-partition communication uses sampling port services and Queuing port services via messages. Intra-partition communication uses APEX buffers (shared message queues) and APEX blackboards (shared variables).	Kernel services Hypervisor services
Semaphores, message queues, pipes and signals.	System services
No concurrency = no service Protocols between drivers & applications: One Wire and Two Wires device	API

4.2.3 MOSAR scenarios

In this section, we illustrate one MOSAR demonstrator scenario with RTEMS and AIR ARINC 653. We illustrate how such operating systems can be used for node reconfiguration in the case of the Global Binary Substitution option. We assume in these examples that the loading of the new binary image is performed before reconfiguration.

With RTEMS, GBS reconfiguration can be achieved as follow:

GBS option (RTEMS) (Loading monolithic new binary before reconfiguration)

1. At design time: an initial configuration is defined, and configuration breakpoints are looked for.
2. At start time: the initial configuration is loaded and started. In this step, all tasks and IPC resources are created and started.
3. When a reconfiguration request occurs (required either by an operator or by FDIR service):
 - The new configuration binary is loaded into mass memory.
 - The boot configuration, indicating which image will be booted, is changed to point to the new configuration.
4. A configuration breakpoint is then waited.
5. At breakpoint:



Preliminary Design Document

- The system suspends the tasks and releases the resources.
- The system reboots.
- The system loads the new image.
- The tasks are initialized, and all resources allocated, e.g. semaphores, queues ...

This sequence avoids that resources are loaded at the same time for the old and new configurations, which may reach the resource limits. The following paragraph illustrates the same option but with a TSP ARINC 653 execution platform:

GBS option (Air ARINC 653) (Loading monolithic new binary before reconfiguration)

1. At design time: an initial configuration is defined, and configuration breakpoints are looked for.
2. At start time: the initial configuration is loaded and started. In this step, all *processes* and *intra or inter-communications resources* are created and started.
3. When a reconfiguration request occurs (required either by an operator or by FDIR service):
 - The new configuration binary is loaded into mass memory. Note that a TSP system would permit loading individual partitions instead of the whole system as a monolithic image.
 - The boot configuration, indicating which image will be booted, is changed to point to the new configuration.
4. A configuration breakpoint is then waited.
5. At breakpoint:
 - The system suspends all *processes* and releases *intra and inter communication resources*. *ARINC configurations parameters are switched: inter-partition scheduling, intra-partition scheduling, Inter-partition communication, Intra-partition communication, memory ...*
 - The system reboots.
 - The system loads the new image.
 - The *processes* are initialized and all *intra and inter communication resources* allocated, e.g. *sampling and queueing port resources ...*

4.2.4 Preliminary design of the software reconfiguration for the MOSAR demonstrators

The MOSAR demonstrators will be restricted to the following constraints on the MOSAR/spacecraft hardware. The following assumptions hold on the OBC of the MOSAR demonstrators:

- Only 3 computers in the MOSAR system will be supported by TASTE: the Monitoring PC, the OBC-Client and OBC-Servicer.
- All 3 computers will be x86/Linux and cannot be moved in the network.



Preliminary Design Document

- Any other movable nodes don't contain computers running TASTE. Furthermore, today, R-ICU is not supported by TASTE, i.e. there is no TASTE component running on the R-ICU.
- Only one OBC node will be reconfigurable in the spacecraft: OBC-Client. Then, during reconfiguration, no software component can be moved between OBCs.
- OBC-Client cannot be moved in the network during reconfiguration by the arm. Furthermore, communications between nodes are designed as follows:
 - Communications are made either by a SpaceWire network or a Ethernet network:
 - OBC-Client and OBC-Servicer communicate via SpaceWire (STAR-Dundee)
 - Both OBCs communicate via Ethernet with the Monitoring PC
 - The SpaceWire link between OBC-Servicer and OBC-Client is permanent
 - OBC-Servicer and OBC-Client control the movable nodes using a software library at application level. Movable nodes are controlled using the SpaceWire RMAP protocol [11]. Movable nodes are "slaves" of the OBC-Client or OBC-Servicer.

4.2.4.1 MOSAR demonstrator software reconfiguration main design elements

According to the above constraints, the software reconfiguration for the MOSAR demonstrators will be based on Unix x86 OBCs, i.e. the reconfiguration option will be not based on Bare-Metal, TSP and RTOS technologies. Since no TASTE node can be moved in the network, the only requirement about network reconfiguration is to change network address or routing table. For such a purpose, we will investigate the use of RMAP communications with the other components of the MOSAR architectures, including its switch. We finally choose the IR reconfiguration option for the TASTE node, because not enough knowledge for options 1 or 2; 1 and 2 are alternate solutions for future projects. With the IR option, the MOSAR software has to be designed as a set of configurations, called "operational mode", and the path between sequences of modes, named "transition". To offer software reconfiguration to MOSAR with the IR option, we need means to express operational modes and transitions. We also need to drive mode transition at runtime. In the sequel, we propose with TASTE how such mode transition can be designed.

4.2.4.2 Mode and mode manager

We propose a reconfiguration mode manager in order to manage mode transition of the software at reconfiguration time of MOSAR movable payloads. In order to describe and early verify the design of this mode manager, its system logical architecture has been prototyped with TASTE. For such prototyping activity with TASTE, we have:

1. Described the system logical architecture and interfaces.
2. Generated code skeletons and write the required applicative code in order to run a simulation of this mode manager.
3. Captured the system hardware and deployment.

Figure 4-4 presents the general architecture of the proposed mode manager. It presents a screenshot of the interface view of TASTE. The proposed manager will run on an OBC. We remind that in MOSAR, OBCs are Unix x86 machines and cannot be moved contrary to the MOSAR payloads. As shown in



Preliminary Design Document

Figure 4-5, all created functions are bounded on x86_Unix. Figure 4-5 presents a screenshot of the deployment view for such model.

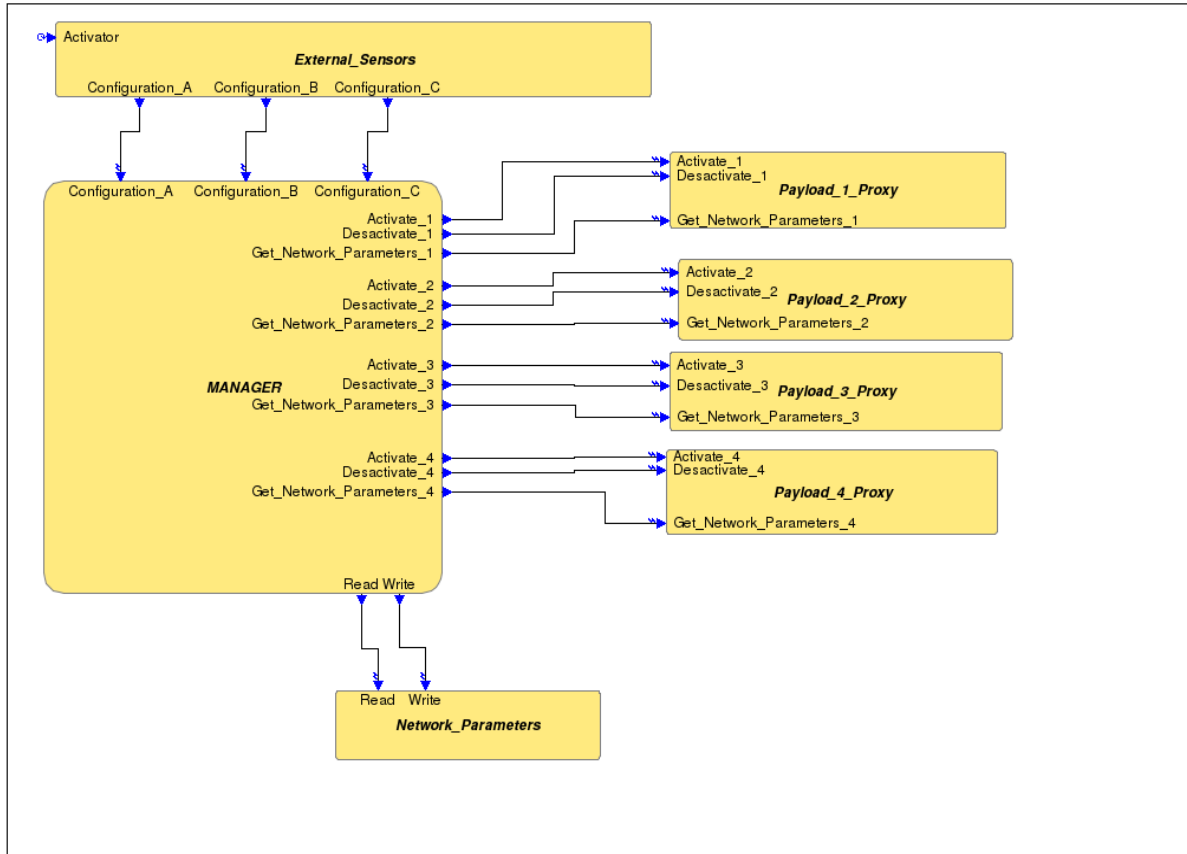


Figure 4-4: Reconfiguration mode manager – General architecture

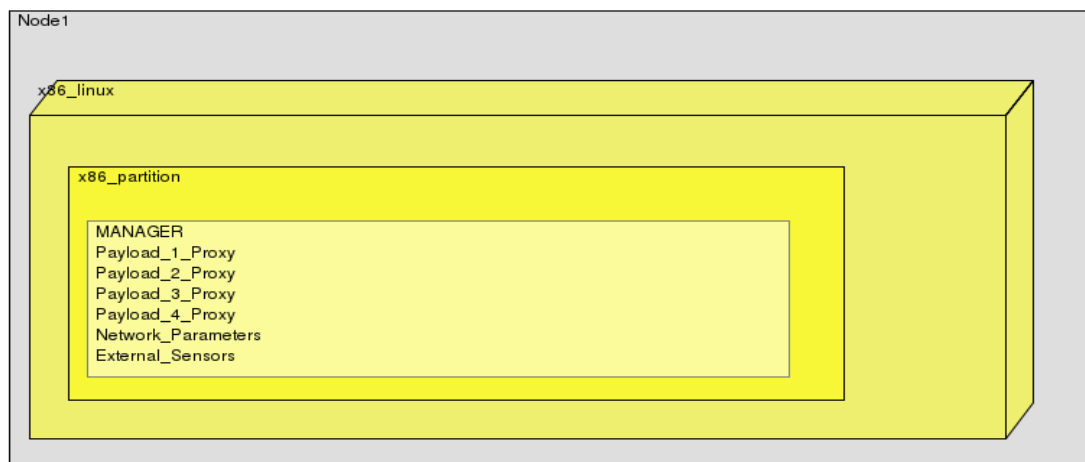


Figure 4-5: Reconfiguration mode manager – Deployment



1. Mode manager

The role of the mode manager is to control the reconfiguration of any component in the MOSAR network. This entity ensures the mode transition when reconfiguration requests are received from external sensors signals. On reconfiguration request, the manager sends signals to any of the concerned payload proxies, to either activate, deactivate or get network parameters of the corresponding MOSAR payload.

After activation of new payload or deactivation of existent payloads, the manager needs to update the network parameters to guaranty that the payloads stay reachable.

2. Payload proxies

A payload proxy represents a payload on the network for the mode manager and interacts with both the payload on the network and the mode manager. A payload proxy forwards any reconfiguration requests from the node manager to the corresponding payload on the network. Depending on received signals from the mode manager, payload proxy may forward activation or deactivation order to the concerned payload.

3. Network parameters

The role of the network parameters function is to update network data at reconfiguration time. Network parameters may concern routing table, address of the payloads ...

4. The external sensors

The role of the external sensors is to receive signals sent by the environment of the node manager. Such signals triggers reconfiguration and has also been introduced for prototyping concern of the node manager.



5 Demonstration Scenarios

5.1 Operational Scenarios

Based on the mission concept presented in section 2, different use cases could be imagined representative of real mission operations. Following an incremental approach, the following scenarios are considered:

- Basic scenario: assembly of a secondary payload. The secondary payload is not very demanding in terms of resources, and only needs data connection and power interface to be provided by the client spacecraft.
 - As an example, let's consider an optical camera.
- Enhanced payload: in this case the secondary payload set-up is more complex. In the first scenario the payload only needed the spacecraft bus to provide the resources needed. In this case, the payload is more demanding, and the capacity available on the client spacecraft is not enough to support its operation. In this case the payload will be added together with some additional modules that provide power, thermal or data capabilities.
 - An example of this could be an infrared camera, usually very demanding in terms of thermal management.
- Full functionality: following the incremental approach, in this case a payload is added to a satellite already on-orbit, but in this case the payload needs much more resources that are obtained through the addition of payload and platform extensions. Complexity is increased, having numerous modules interconnected and connected to the spacecraft bus. In terms of data and power routing, several paths are available, requiring smart management by the client OBC in order to optimize the system combined resources (spacecraft bus plus extensions).
 - An example of this scenario is a radar payload requiring high power that is provided by service modules that are brought by the servicer satellite.
- Substitution of modules: in case of failure of a certain module, or following a preventive maintenance strategy; some modules of the spacecraft could be substituted. This supposes that the spacecraft implements some equipment in a modular shape, enabling on-orbit substitution.
 - For instance battery packs or radiator surfaces.

The following figure presents a schematic view of the different scenarios. The modules are represented by its functionality; each module implementing a basic function:

- SPL: secondary payload.
- PICU: payload interface control unit, module managing power and data distribution to the payload.
- THS: thermal subsystem: module implementing some dissipative surface and a fluidic circuit enabling thermal management of a neighbour module.
- BAT: for battery.
- SA: for solar array.

Note that the figure does not intend to be representative in terms of architecture or shape of the spacecraft or the module, it only presents each element by its main functionality.



Preliminary Design Document

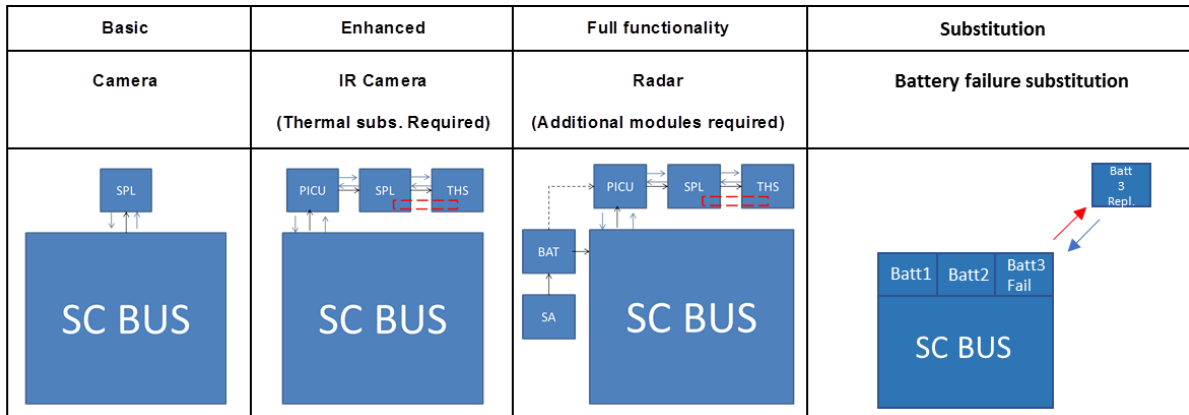


Figure 5-1 – Different use cases of MOSAR mission concept.

As the final goal of the project, the purpose of the demonstration is to illustrate the full sequence of operations representative of the different use-cases described above:

- Assembly of the CLT with new SM from the servicer
- Detection and replacement of a damaged SM
- Re-routing of power/data bus through the SM
- Thermal transfer between SM

This will be done through four demonstration scenarios. It has to be noted that the two spacecraft buses are connected during all the demonstrations.

The success of this demonstration should be built on top of the sub-systems and validations tests presented above.

Real mission scenarios		Basic	Module substitution	Enhanced payload	Full functionality
		Optical camera	Battery pack	IR camera	Radar payload + service modules
Main functionalities needed	Assembly of the CLT with new SM from the servicer	X	X	X	X
	Detection and replacement of damaged SM		X		(X)
	Re-routing of power/data bus through the SM				X
	Thermal transfer between SM			X	(X)



Preliminary Design Document

Demonstrators	S1 – Initial assembly of SMs	S2 – Replacement of failed SM	S3 – Thermal transfer between two SMs	S4 – Automatic CLT network reconfiguration
----------------------	------------------------------	-------------------------------	---------------------------------------	--

Table 5-1 – Declination of ground demonstrators covering main functionalities identified in real mission scenarios

5.2 Spacecraft Modules Description

The demonstrations will be performed around a mock-up satellite composed of six spacecraft modules, divided in four Active System Modules (ASM) and two Active Payload Modules (APM):

- **Data Management Subsystem (SM1-DMS)** hosts the main on-board computer of the CLT that runs management software responsible for the management of the modular spacecraft mission and for the interfacing with the other subsystems and payload modules of the spacecraft. The DMS SM hosts an OBC running an adapted ESROCOS software stack of OG1.
- **Power Subsystem (SM2-PWS)** is providing the electrical power on the CLT. It regulates and balances the electrical power between the different sources of power (Solar Panel power conditioning and its own internal battery charge and discharge management). It converts and distributes the electrical power on the power lines of the CLT. The PWS is a heating element. It also provides a dedicated SI with a thermal interface to perform forced heat exchange with the Thermal Subsystem module (see below).
- **Battery ORU module (SM3-BAT)** is a SM that comprises a set of Lithium-ion batteries and the electronics to manage the balance and regulation of their charge and discharge and to convert their voltage to the CLT buses voltages. The design will be based on the OG5-SIROM battery modules.
- **Thermal Subsystem module (SM4-THS)** is responsible for the thermal management of the CLT SMs. It provides a SI with a thermal interface to allow heat exchange by fluid loop. The THS dissipates heat through ventilators. Interfacing the THS to the PWS will allow extracting heat from the PWS via a liquid cooled loop, and radiate this heat in ambient air through the radiator of the THS. The THS and PWS modules payload developed for the demonstration setup will be adapted from the MAGSOAR thermal laboratory demonstrator that has been developed in OG5.
- **Optical Sensor Payload module (SM5-OSP1, SM6-OSP2)** is a SM integrating an optical sensor payload to support the mission operation. It can be either for space observation or for proprioceptive visualisation of the spacecraft. These SM will typically have a constrained position/orientation on the spacecraft to ensure the position of the optical element as function of its purpose. The SM will include a computing unit to process locally the sensors data.

Three demonstrations scenarios are foreseen to be performed in MOSAR, as representative illustration of modular spacecraft applications.

5.3 Demonstration Scenarios

5.3.1 Scenario 1: Initial Assembly of SMs from SVC to CLT

Objective: demonstrating the assembly of several SMs originally mounted on the SVC onto the CLT spacecraft bus, including both the placement of SMs on the CLT itself and on other SMs.



Preliminary Design Document

Initial Conditions: the initial configuration of the SMs is represented in Figure 5-2-left

- two SM are already installed on the CLT (fixed position for the demonstrations), the SM1-DMS (OBC) and SM2-PWS (power module)
- the four other SM are stored on the SVC
- the WM is stowed in parking position on the SVC
- the system is ready for assembly operations

Success Conditions:

- the desired SMs are mounted onto the CLT, as illustrated in Figure 5-2-right
- the newly mounted SMs are powered on and operational
- it should be possible to receive data / telemetry from each deployed SM and send commands to each of them, including video/picture rendering from the optical payload modules

Based on the execution plan prepared and simulated by the MCC, the whole operations requires successive manipulation operations of the SM by the WM. It can also include relocation of the WM on the spacecraft bus or on SM for reachability reasons. That then requires also re-routing of the power and data communication through the robot. The SI connection between the SM2-PWS and SM4-THS (thermal module) validate also the mating of the fluid interface.

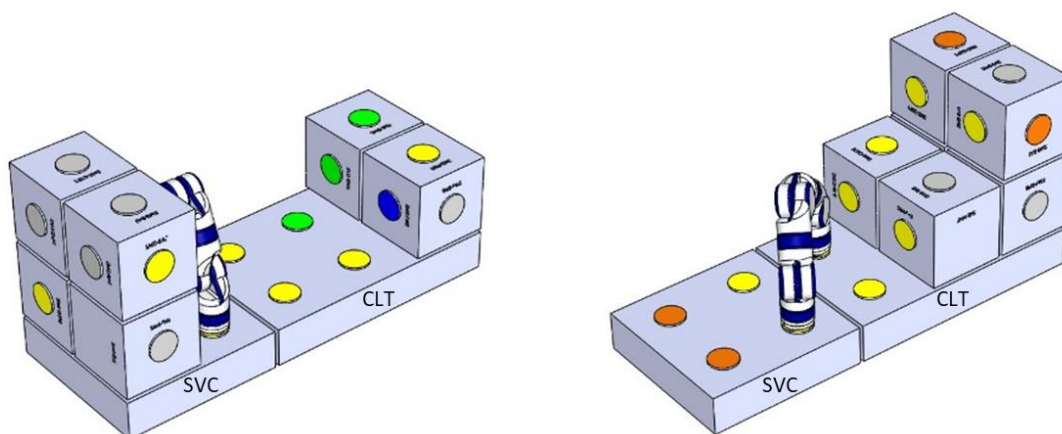


Figure 5-2: MOSAR scenario 1 initial and final configuration

5.3.2 Scenario 2: Replacement of a failed SM

Objective: demonstrating the detection and replacement of a failing module by an equivalent working module. This will be illustrated in the current scenario by the replacement of one of the optical payload modules by the second one.

Initial Conditions:

- the WM is stowed in parking position
- the CLT is assembled with the 6 SMs (final configuration of scenario 1) and operational (Figure 5-3-Left)
- the maintenance operation is ready to be carried out

Success Conditions:

- the faulty module should be brought back in the cargo area of the SVC



Preliminary Design Document

- the new optical SM has been mounted onto the same location of the failed SM on the CLT (Figure 5-3-Right)
- the newly replaced SMs should be powered and operational, i.e. recovery of functionality

The operations include the detection of the faulty module, the iteration with the MCC to define the execution plan for the re-configuration operation and successive manipulation operations of the SM by the WM, following the execution plan. This scenario also highlights rerouting of power and data to isolate the faulty module and to take into account the new configuration of the spacecraft.

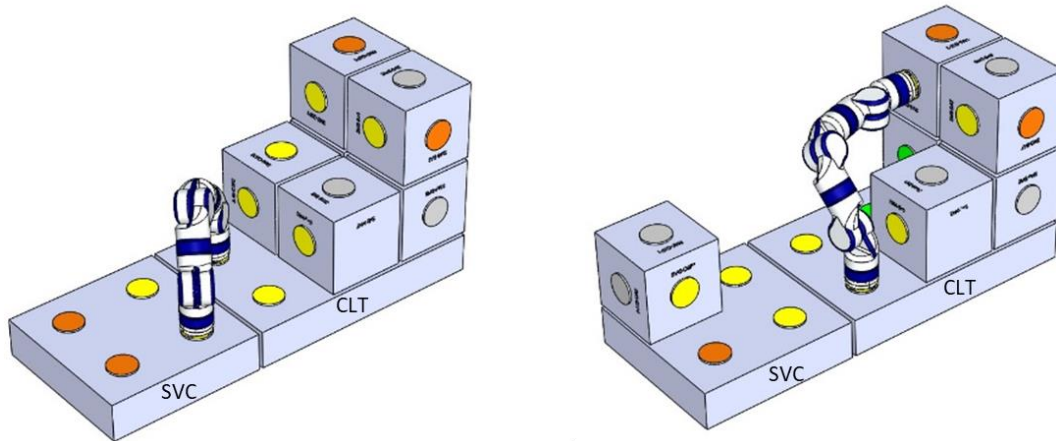


Figure 5-3: MOSAR scenario 2 initial and final configuration

5.3.3 Scenario 3: Thermal transfer between two SMs

Objective: demonstrating the active cooling of a SM producing heat (SM2-PWS) by a dedicated thermal handling module (SM4-THS)

Initial Conditions:

- the THS and the PWS modules are mechanically coupled and operational (final configuration of scenario 1)

Success Conditions:

- a heat transfer should be observed between the 2 modules (through telemetry reading with heat probes on the 2 sides).
- No leaks should have been observed.

The operations include the commanding of thermal sub-system pump by the CLT OBC, as well as the telemetry of the SM temperatures. The heat transferred to the SM4-THS from the SM2-PWS is evacuated through the passive radiator, located on one of the face of the SM4-THS.



5.3.4 Scenario 4 (S4): Automatic CLT Network Reconfiguration

5.3.4.1 Scenario Description

Objective: demonstrating the ability of the SpaceWire network to automatically detect and adapt to faulty interfaces without the need of the SVC-OBC to be attached to reconfigure the network.

Initial Conditions: the configuration of the SMs represents a constructed spacecraft. The SVC is disconnected from the CLT.

- The optical SM is located at least one SM away from the OBC.
- The optical SM is part of a grid of SMs of a size of at least 2x2 (to provide two distinct network paths from the OBC to the optical SM).
- The network is configured such that the OBC can communicate with any SM in the network using logical addressing.
- All network links are active and running
- The OBC-CLT is running the application, constantly fetching image frames from the optical SM.

Success Conditions:

- The faulty link is detected by the CLT-OBC
- The network topology is rediscovered with the faulty link excluded
- A new valid network mapping is found
- The network is reconfigured successfully, traffic starts to resume to and from the CLT-OBC and the optical SM.

5.3.4.2 Sequence of Operations

This scenario starts from the spacecraft constructed and operating nominally. The process is started by failing a link via the R-ICU debug interface.

1. Ensure that spacecraft is assembled and running the optical payload operation.
2. Fail one of the active links carrying the payload data between the CLT-OBC and the optical SM by commanding the SpaceWire link to DISABLED using the R-ICU debug interface.
3. CLT-OBC detects error due to receiving EEP or RMAP reply timeout.
4. CLT-OBC automatically initiates SpW-PnP network discovery sequence and rediscovers the network layout.
5. CLT-OBC runs network mapping algorithm on results of SpW PnP discovery to find shortest routing paths between nodes and CLT-OBC.
6. CLT-OBC uses RMAP to update the routing tables of all SpW routers in the network
7. CLT-OBC switches back to nominal mode
8. Optical payload operation continues using the discovered redundant network path.

Telemetry from the R-ICU debug port (USB UART) can be used to track this sequence, however when ran autonomously it is anticipated to be too fast to for an operator to interact with.



5.4 Demonstration Configuration Analysis

Figure 5-2 and Figure 5-3 illustrate the initial and final configurations of the test setup for the two envisaged scenarios. To make the transition, the WM will perform successive operations of spacecraft module manipulation and self-relocation on the structure.

When defining the demonstration setup, it is needed to find an optimum between the setup capabilities (related to the range of validation tests) and the constraints imposed by the testing environment (under gravity) and the practical limits of the project (system complexity, number of components, standard interfaces...). This has been achieved by initial hypotheses and through a trade-off analysis of possible demonstration configurations (number and type of SI) and sequences of operations.

The following initial hypotheses have been considered:

- The rendezvous and docking phases between the two satellites buses are not in the scope of this project. They are mechanically connected during all the demonstration.
- The CLT satellite bus is originally equipped with an OBC and a power generator module, that are in this case, embedded respectively in the SM1-DMS and the SM2-PWS. Both spacecraft modules are rigidly attached on the CLT bus and also together, with the possibility to transmit data and power between them.
- The four other SM are initially stowed on the SVC in power-off mode (representing the launch and travel conditions)

The main constraint of the demonstration is the weight of the movable spacecraft modules, as it has a direct impact on the design of the walking manipulator. In order to reduce the weight, two approaches have been considered. The first is the reduction of the number of SI and mainly the active version of them. The second path is the optimization of the box rigidity, by avoiding the need of the walking manipulator to move on the modules.

Based on the envisaged SM and WM size and kinematic, several analyses have been performed to find valid sequences of operations, with an optimized number and type of SI.



Preliminary Design Document

Table 5-2 provides the list and type of the current final selection of standard interfaces for each component of the MOSAR demonstrator (see Table 6-7 for version and colour references). It can be noted that each mobile SM owns only one active SI, while the WM has two actives, for each end-effector. This selection offers the following possibilities:

- Scenario 1 assembly
- Scenario 2 payload update
- Scenario 3 thermal transfer
- Data (re)-configuration / re-routing
- Power (re)-configuration / re-routing
- Simultaneous double and triple connections of SM

Moreover, the proposed configuration doesn't require the WM to attach and walk along the mobile SM. It is however envisageable to demonstrate this feature through interactions with the fixed SM on the CLT (that can be more rigid, as less limited by the weight).

Section 7.2 illustrates a valid sequence for scenarios 1 and 2 based on the selection of standard interfaces. It has to be noted that other sequences are also possible to reach the final configuration. This is not the minimum set of SI required to make the basic validation operations. However, it has been selected as it offers some versatility for the demonstration of mechanical, data and power re-configuration. If additional reduction of weight is necessary, additional optimization could be envisaged during the detailed design activities. Dummy and some mechanical versions of the SI are not mandatory and could be removed to gain some weight in each SM (see "optional" in the table).



Preliminary Design Document

Table 5-2: Standard interface selection for MOSAR components (O=optinal)

SVC	W	X	Y	X		
	Passive	Passive	Mech	Mech		
CLT	A	B	C	D	E	F
	Passive	Passive	Active	Passive	Fix	Fix
SM1-DMS	A	B	C	D	E	F
	Active	Fix	Active	Dummy (O)	Dummy (O)	Fix
SM2-PWS	A	B	C	D	E	F
	Thermal	Dummy (O)	Passive	Fix	Dummy (O)	Fix
SM3-BAT	A	B	C	D	E	F
	Dummy (O)	Passive	Dummy (O)	Mech (O)	Passive	Active
SM4-THS	A	B	C	D	E	F
		Dummy (O)	Passive	Mech	Passive	Thermal
SM5-OSP1	A	B	C	D	E	F
	Dummy(O)	Passive	Dummy(O)	Dummy (O)	Passive	Active
SM6-OSP2	A	B	C	D	E	F
	Passive	Passive	Mech	Dummy (O)	Passive	Active
WM	A	B				
	Active	Active				

The final amount of SI of each version is:

Active	8
Thermal	2
Passive	15
Mechanical	5 (4)
Dummy	11 (0)



5.5 Ground Segment Operations

The robotic reconfiguration of the satellite envisaged in the MOSAR project is based on an operation plan previously computed on ground and verified by simulation. The scenario to be demonstrated is thus divided in two independent phases:

- Design and verification on ground: the desired configuration of the satellite is defined and validated, and a plan for the robotic reconfiguration is developed and verified in the simulator.
- On-orbit reconfiguration: the verified reconfiguration plan is uploaded to the on-board system and executed using the Walking Manipulator and the platform (or demonstrator) capabilities.

This section focuses on the first stage, while the second is addressed below in section 0.

The starting point for the planning of a reconfiguration mission is a set of available SMs. The definition of these SMs is previous to the start of the scenario demonstrated in MOSAR. In a future scenario where a reconfigurable spacecraft is in orbit and a servicing mission is being planned, it can be envisaged that the payload of the servicer spacecraft will be selected among a set of available APMs from satellite subsystem manufacturers, and APMs from specific clients. The characteristics and capabilities of the available APMs and ASMs must be known by the mission designer to perform the mission analysis and plan and validate the entire operation.

In the context of the MOSAR demonstrator, the definition of the SM characteristics and capabilities is a manual stage of configuring the different tools that support the demonstration. In order to ensure consistency among the different components of the system, the configuration is defined in a centralized database / configuration file, from which each software tool will extract its required information.

In addition to the knowledge about the SMs, some information on the characteristics of the SVC and CLT spacecraft, the WM and the environment must be also provided.

An initial list of the aspects that need to be considered is provided in Table 5-3 below. The table indicates the components that need each piece of information and the purpose served. It must be noted that the information is used not only by the components at the design and simulation stage, but also during the demonstrator operations.

Table 5-3: System characteristics relevant for configuration

Information	Element	Information users	Purpose
SM identifier	SM	All SW components	Unique identification of SMs
Capabilities and requirements: power, thermal, data routing and processing, etc.	SM	<ul style="list-style-type: none">- Design Tool- Simulator	<ul style="list-style-type: none">- Validation of the configuration. The Design Tool may check simple constraints, while the Simulator may be used to check the system behaviour in time- Simulation of the system behaviour in the steady state and during reconfiguration
Geometry of the SM (with different levels of fidelity)	SM	<ul style="list-style-type: none">- Simulator- Robotic Arm SW (trajectory planning)- Vision Subsystem	<ul style="list-style-type: none">- Dynamic simulation- Visualization by the simulator- Computation of collision-free manipulator trajectories- Model matching



Preliminary Design Document

Information	Element	Information users	Purpose
Physical parameters: mass, CoG, etc.	SM	<ul style="list-style-type: none">- Simulator	<ul style="list-style-type: none">- Dynamic simulation
Location and characteristics of the SIs: active, passive, mechanical, thermal, dummy	SM	<ul style="list-style-type: none">- Simulator- Planner- Autonomy Agent	<ul style="list-style-type: none">- Simulate the SI behaviour- Generate plans according to the constraints- Command the SIs according to their capabilities
SM placement restrictions	SM	<ul style="list-style-type: none">- Design Tool- Planner	<ul style="list-style-type: none">- Validate configuration- Plan according to constraints
CLT and SVC geometry (platform and SIs)	CLT/SVC	<ul style="list-style-type: none">- Design Tool- Simulator- Robotic Arm SW (trajectory planning)- Planner- Autonomy Agent	<ul style="list-style-type: none">- Available locations of the SMs- Simulation of the reconfiguration- Visualization- Computation of collision-free manipulator trajectories- Computation and execution of valid WM and SM relocation steps
CLT and SVC SI characteristics: active, passive, etc.	CLT/SVC	<ul style="list-style-type: none">- Simulator- Planner- Autonomy Agent	<ul style="list-style-type: none">- Simulate the SI behaviour- Generate plans according to the constraints- Command the SIs according to their capabilities
WM geometry	WM	<ul style="list-style-type: none">- Simulator- Robotic Arm SW	<ul style="list-style-type: none">- Simulation of the reconfiguration- Visualization- Computation of collision-free manipulator trajectories
WM dynamics	WM	<ul style="list-style-type: none">- Simulator0- WM Control SW	<ul style="list-style-type: none">- Dynamic simulation of the robot- Control of the robot
Manipulation heuristics: reachability, constraints for operation in 1-g, etc.	WM	<ul style="list-style-type: none">- Planner	<ul style="list-style-type: none">- Generation of a reconfiguration plan compatible with the physical constraints of the system
Environmental parameters: orbit, illumination, visibility, etc.	Environment	<ul style="list-style-type: none">- Simulator	<ul style="list-style-type: none">- Simulation and visualization

Based on the description of the system and its environment, the design stage of the demonstration can start. The Figure 5-4 below presents an outline of the process, and the tools used by the mission analyst at each stage.



Preliminary Design Document

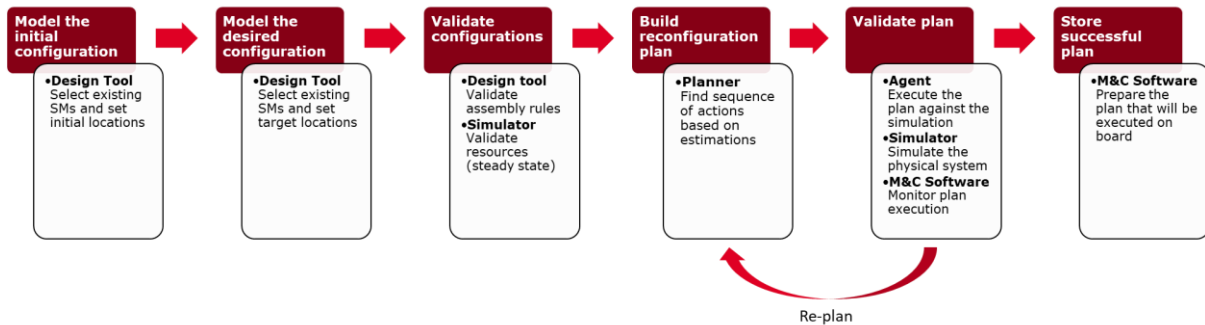


Figure 5-4: Design, simulation and planning stages

1. Model the initial configuration

The analyst uses the Design Tool (see section 6.1) to build a description of the initial configuration of the Client system (installed SMs, locations, connections, operational status, etc.), and the Servicer spacecraft (position of the new SMs and the WM at launch).

The chosen configuration is an input to the Simulator (see section 6.1) and the Planner (see section 6.2) tools.

2. Model the desired configuration

The analyst defined the desired configuration at the end of the scenario using the same process and tool. The result is an input to the Planner tool.

3. Validate configurations

The analyst checks that both initial and desired configurations are valid according to the system constraints.

The Design Tool can do an initial check of the configuration regarding the placement restrictions for each SM (if applicable) and the valid coupling of each pair of SIs (according to their type active, passive, etc.). Additional constraints may be implemented in the system, for instance regarding resource utilization (power, thermal, mass, data, etc.).

A second validation stage is performed using the Simulator. The operation of the system in the steady state (i.e., nominal operation before and after the reconfiguration of the client), using resource models of each SM, in order to assess if the target configuration performs as expected.

4. Build reconfiguration plan

Once the validity of the initial and desired configuration has been verified, the analyst uses the Planner to compute an initial robotic reconfiguration plan. The plan consists of a sequence of SM and WM relocation operations that lead from the initial to the desired configuration.

The Planner has the knowledge of the constraints for placing the SMs and for moving the WM (in particular, for demonstration in 1-g conditions). However, the Planner relies on estimation functions to calculate the cost in terms of duration or other aspects (e.g., energy usage). This means that the computed plan needs still to be validated using simulation.

5. Validate plan

The initial plan computed by the Planner is executed in closed loop with the Simulator. The Autonomy Agent (see section 6.2), identical to the one deployed on board, decomposes the plan in elementary steps that are executed by the Simulator.

The analyst uses the Monitoring and Control Software together with the visualization capability of the Simulator, to monitor the execution of the plan and investigate any issues that are detected.



Preliminary Design Document

It is possible that the execution of the plan fails, as it has been obtained using a simplified model of the problem and does not reflect all the physical details modelled by the Simulator. In that case, the Planner can be invoked in closed loop with the simulation in order to obtain an alternative plan. The process continues until a successful plan is found.

6. Store successful plan

The plan successfully tested is stored by the M&C Software for uploading to the on-board system during the actual demonstration.

The final aspect that needs to be considered at the design stage is the design and implementation of the OBSW for the nominal operation of the system. The OBSW runs on the OBC of the Client spacecraft (or on several OBCs distributed in different SMs). In order to manage the complexity of the software for a reconfigurable system, a model-based approach is followed using the TASTE framework, in line with the ESROCOS project.

The OBSW is modelled in TASTE, with a set of dedicated TASTE functions (components) that implement the telemetry and telecommand services needed to manage the modules during nominal operation. In addition, the OBSW must be able to transition to a different configuration when the set of available modules changes.

The Figure 5-5 below shows a sample TASTE model for the CLT software. The software contains a PUS Services component that manages the communication with the MCC (represented as the PUS Console component), a set of components specific to each SM in the current configuration, and a component to manage the reconfiguration of the system.

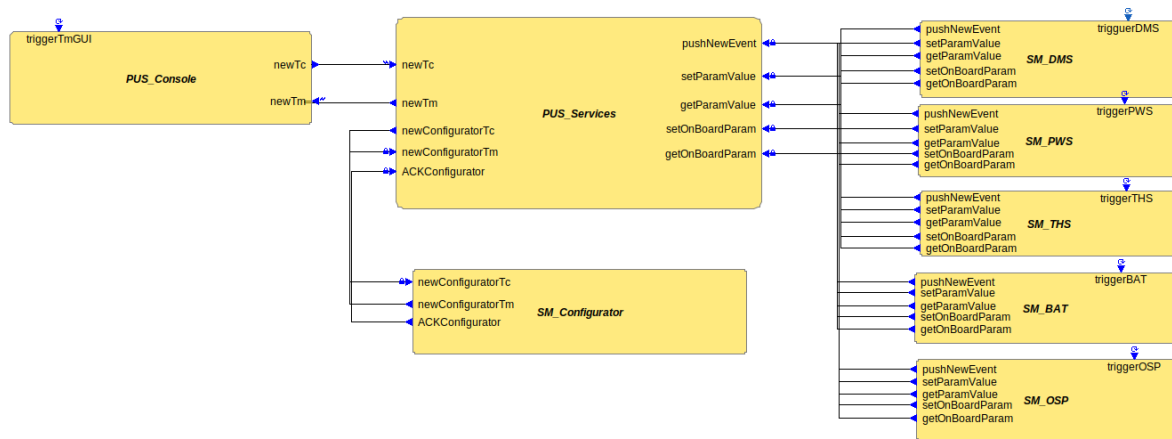


Figure 5-5: Sample TASTE model for the CLT software

When the system is reconfigured with modules added and removed, the software will reconfigure, deactivating the functions of the removed modules, activating the new ones, and updating the interfaces and other properties. Each of these will reflect in changes in the TASTE model.

The TASTE framework currently has no capability for software reconfiguration. Section 0 presented the envisaged extension of the framework to add this capability and allow for the modelling and analysis of reconfigurable systems.

In preparation for a reconfiguration mission, the software engineer will update the OBSW models to account for the initial and desired configurations of the system, implementing the functionality needed for the nominal operation of the satellite in the new configuration. The model may also account for alternative configurations (nominal or failure modes).



Preliminary Design Document

It can be envisaged that each SM is provided with a set of reusable software components for management and control. These can be provided as reusable TASTE functions, which will be imported into the TASTE model by the software engineer.

The new version of the CLT software will be developed by building on the model and the reusable functions, and will be uploaded to the OBC. The upload of the software image is not part of the demonstrator.

The TASTE model for the demonstrator includes not only the CLT software but also the SVC and the MCC. It can therefore support the nominal operations of the CLT before and after the reconfiguration, and the robotic activities by the SVC and the WM.



5.6 Space Segment Operations

The Space Segment presents two main mode of operations, the CLT/SVC reconfiguration and the CLT nominal operations.

The CLT/SVC reconfiguration operations consist of sequence of actions to go from an existing spacecraft configuration to a desired one, ready for operations. In the MOSAR demonstrator, the Servicer Spacecraft and its on-board computer manage the re-configuration operations. The Autonomy Agent running on the OBC-S executes the operation plan prepared by the MCC. As illustrated in the Figure 5-6, it will trigger successive actions on the components at different level of the system architecture, with the Component Management in the OBC, the Component Control in the R-ICU/WM and the Component Controller at the lowest level. The specific role and scope of each level can be chosen according to the level of abstraction in the OBC and the autonomy of the components or SM (R-ICU). For instance, having generic command from the OBC component managers allows replacing the lowest level elements with different internal configuration or control.

The following section provides a preliminary list of the expected elementary actions. These actions can be associated to form routines that will be often implemented during a plan. The full scenario executions consists then in a sequence of routines and elementary actions.

During re-configuration or spacecraft operations, the MCC can communicate with the OBC-S Telemetry and Telecommand service, to respectively monitor the plan operations and components parameters as well as control the execution of the Agent and the components.

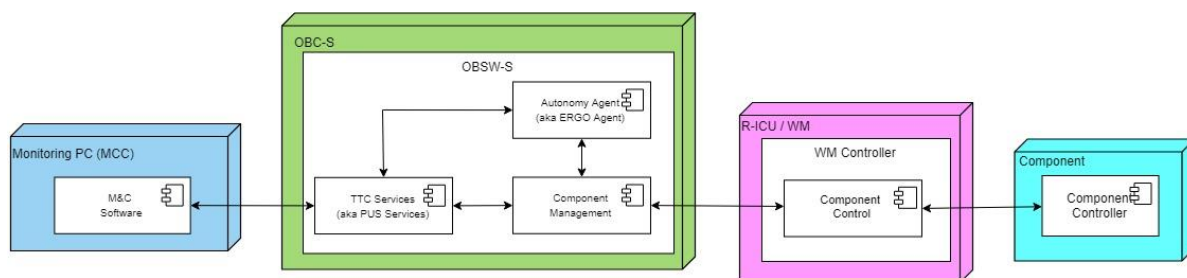


Figure 5-6: Re-configuration operation architecture

The CLT nominal operations consists mainly in the operations of the payloads of the spacecraft with TM/TC and data feedback (e.g. video images) to the OBC-C (on the CLT). Some re-configuration management can also be considered in the case of failure detection for the isolation of the faulty node and re-routing of power and data, without the need of intervention of the servicer spacecraft that could be present anymore. As for above the TTC service allows to communicate with the MCC, for TM/TC of components and payloads, and data visualization / recording.

The transition between the two modes needs also to be managed (Client Management in Figure 3-2). Before starting the robotic reconfiguration, the control/authority needs to be switched from the OBC-C to the OBC-S. After re-configuration, the OBSW-S should request the OBSW-C to reconfigure for the new architecture of the system and handle back control to the OBC-C.

Figure 5-6 provides a generic description of the software architecture, associated with a specific component of the system (subsystem, payload). The Component Control software starts automatically when the R-ICU is powered, initializing the corresponding Component Controller, initializing the SpW communication with the OBC (discovery, table routing) and going in *Idle* mode. In the *Idle* state, the payload is not operated and it allows to safely power off the SM. It periodically check if a TC is received from the Management component of the OBC, by polling the corresponding memory address (RMAP protocol). It periodically writes TM parameters and status to memory mapped to RMAP.



Preliminary Design Document

The Visualization software on the MCC provides a GUI interface with user command to control the physical components, and to display the payload/component parameters. The TTC service running on the OBC-C will trigger the Management software to relay the received commands from the MCC (PolyORB-HI) to the Control software (R-ICU).

When the Control software receives TC for updating the subsystem, it will communicate with the local Controller (e.g. through CAN) that will manage the physical changes on the setup. The Management component polls the TM data of the Control software via RMAP. They are then published to the Visualization component on the MMC through PolyORB-HI protocol, in order to be displayed on the GUI.

5.6.1 Planner Elementary Actions

This section provides the preliminary list of the elementary actions (TC) of the plan produced on the MCC and executed on the OBC-S. They are the basic functions from the OBC to the system components through the R-ICU / WM controller. The information in bracket correspond to the parameters of the function. This list doesn't cover the TM feedback, that could be used to control the sequence of the operations

Name:	Move Walking Manipulator Cartesian (Cartesian trajectory)	Element:	WM
"Coarse" motion of the free WM extremity to follow the trajectory in the Cartesian space. The WM position sensors validate the success.			
Name:	Move Walking Manipulator Joint (Joint trajectory)	Element:	WM
Motion of the WM joints along the trajectory in Joint space. The WM position sensors validate the success.			
Name:	Approach Walking Manipulator (Cartesian trajectory)	Element:	WM
Fine motion of the WM extremity to follow the trajectory in Cartesian space with impedance control. Used to align the WM extremity SI or a SM SI with another SI. The WM/SM SI proximity sensors (and optionally the WM torque sensors) validate the success.			
Name:	Power Switch (#CPDU, cPDU channel, OFF)	Element:	cPDU
Enable/disable of one cPDU power channel			
Name:	Configure R-ICU (#R-ICU)	Element:	R-ICU
Configuration of the SpW router, perform some levels of plug and play to expose SM services and allocate logical addresses for these services.			
Name:	Configure Data Routing (#R-ICU, route table)	Element:	R-ICU
Configuration of the data routing table of a R-ICU			
Name:	SI state update (#HOTDOCK, state)	Element:	HOTDOCK
Configure SI mating state to enable mating configuration transition (e.g. for HOTDOCK disconnected, latched, connected). The SI latching sensors validate the success.			

Table 5-4: Preliminary list of planner elementary actions



5.6.2 Routines

The two main scenarios that will be demonstrated in MOSAR (sections 5.3.1 and 5.3.2) can be performed by successive applications of the routines described below, and elementary actions.

5.6.2.1 Routine 1: WM Re-Localization

This routine allows to re-localize the WM extremity between SI of the spacecraft or the SM, as illustrated in Figure 5-7. In the context of MOSAR, the WM is fully managed by the SVC and the embedded OBSW-S.

The following initial conditions need to be full-filled to start the sequence:

- The WM SI-A is connected to the Initial SI, with power and data transmission
- The WM is powered on, is able to communicate with the OBC-S by SpW and is ready for operations
- The Target SI is able to provide power and data communication with the OBC-S
- The Target SI can be reached by the WM SI-B (following the required trajectory)

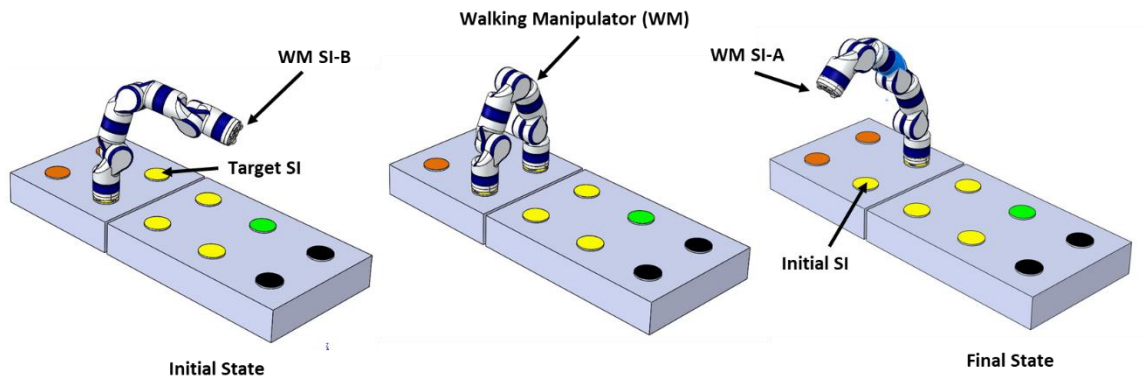


Figure 5-7: Routine 1 – WM re-localization between initial SI and Target SI

The following table provides the planner TC sequence to perform Routine 1, based on the elementary actions. The sequence can be repeated several times to make the WM walking along the structure.

Planner TC Sequence:

Step #	Action
1	OBCW-S sends successive TC “Move Walking Manipulator Cartesian (trajectory)” or “Move Walking Manipulator Joint (trajectory)” to the WM Controller. → The free WM extremity is moved to the SI target by successive free collision trajectories (ensured by the planner) to reach the position to initiate the final approach.
2	OBCW-S sends TC “Approach Walking Manipulator (trajectory)” to the WM Controller. → The WM SI is guided, in impedance control, in the latching position
3	OBCW-S sends TC “SI state update (SI-B, connected)” to the WM Controller → the WM SI-B is connected to the target SI. The WM shall allow end-effector motion (impedance control) to support final alignment of the SI
4	[Optional] (if target SI is an active SI) OBCW-S sends TC “SI state update (target SI, latched)” to the R-ICU controller managing the target SI.



Preliminary Design Document

	→ The two SI, WM SI-B and target SI, are mechanically connected on both sides. The “latched” state is used as the data plates are already connected. The WM shall allow end-effector motion (impedance control) to support final alignment of the SI
5	OBCW-S sends TC “ Power Switch (#cPDU, cPDU channel, ON) ” to the R-ICU Controller connected to the cPDU of the target SI. → The WM is able to get power from SI-B (in prediction of the disconnection of SI-A)
6	OBCW-S sends TC “ Configure Data Routing (#R-ICU, route table) ” successively to the different R-ICU Controller, to configure the routing of the SpW data between the OBC-S and the WM through the WM SI-B (TBC if relevant in the context of WM motion). → The WM is able to communicate with the OBCW-S by SpW through the SI-B (in prediction to the disconnection of SI-A)
7	OBCW-S sends TC “ Power Switch (#cPDU, cPDU channel, OFF) ” to the R-ICU Controller connected to the cPDU of the initial SI. → There is no more power transition passing from the initial SI
8	[Optional] (if element SI is an active SI) OBCW-S sends TC “ SI state update (initial SI, disconnected) ” to the R-ICU controller managing the initial SI. → The initial SI is disconnected from the WM SI-A
9	OBCW-S sends TC “ SI state update (SI-A, disconnected) ” to the WM Controller → the WM SI-A is disconnected from the initial SI
10	OBCW-S sends successive TC “ Move Walking Manipulator Cartesian (trajectory) ” or “ Move Walking Manipulator Joint (trajectory) ” to the WM Controller → The free disconnected WM extremity is moved to the final goal position

Note: Steps 1 to 3 could be considered as another routine or also as an elementary action in regards to the OBC/planner, as they can be fully managed locally by the WM controller.

5.6.2.2 Routine 2: SM Relocation

This routine allows to re-localize a SM between two different positions/orientations on the SVC or CLT, as illustrated in Figure 5-8. In the context of MOSAR, the operations are fully managed by the SVC and the embedded OBSW-S.

The following initial conditions need to be full-filled to start the sequence:

- SM powered-off (need potentially previous operations if the SM is powering other SM) and is latched through its initial SI (SI initially connected to another SM or structure)
- The WM SI-A is connected to another SI (spacecraft or module), with power and data transmission
- The WM is powered on, is able to communicate with the OBC-S by SpW and is ready for operations
- The position of the WM and SM allow performing the desired trajectory between the initial and final position of the SM.



Preliminary Design Document

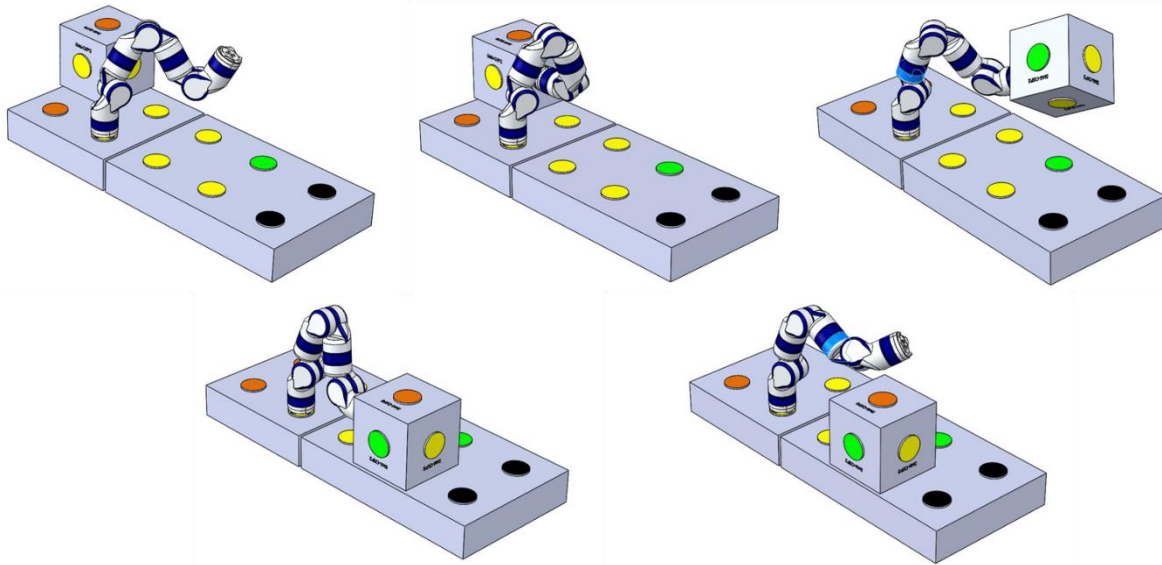


Figure 5-8: Routine 2 – SM re-localization

Planner TC Sequence:

The following sequence is based on the assumption that the SM is initially connected to only one SI and that the SM SI are active. Notes in the sequence highlight possible optional actions in the case of multiple connections or other active/passive configurations. Also this sequence doesn't include the required configurations of the SM, after the relocalization, and other components to make it ready for nominal operations.

Step #	Action
1	OBCW-S sends successive TC “Move Walking Manipulator Cartesian (trajectory)” or “Move Walking Manipulator Joint (trajectory)” to the WM Controller. → The free WM extremity is moved to the SI target of the SM by successive free collision trajectories (ensured by the planner) to reach the position to initiate the final approach.
2	OBCW-S sends TC “Approach Walking Manipulator (trajectory)” to the WM Controller. → The WM SI is guided, in impedance control, in the latching position
3	OBCW-S sends TC “SI state update (SI-B, connected)” to the WM Controller → the WM SI-B is connected to the target SI of the SM. The WM shall allow end-effector motion (impedance control) to support final alignment of the SI
4	OBCW-S sends TC “Power Switch (#cPDU, cPDU channel, ON)” to the WM Controller → The WM SI-B provides power to the SM (switching on the SM R-ICU and cPDU)
5	OBCW-S sends TC “Configure Data Routing (#R-ICU, route table)” successively to the different R-ICU Controller, to configure the routing of the SpW data between the OBC-S and the SM through the WM SI-B. → The SM is able to communicate with the OBCW-S by SpW through the WM SI-B
6	[Optional] (if SM target SI is an active SI) OBCW-S sends TC “SI state update (SM target SI, latched)” to the R-ICU controller managing the SM.



Preliminary Design Document

	→ The two SI, WM SI-B and SM target SI, are mechanically connected, on both sides. The “latched” state is used as the data plates are already connected. The WM shall allow end-effector motion (impedance control) to support final alignment of the SI
7	OBCW-S sends TC “ SI state update (SM initial SI, disconnect) ” to the R-ICU controller managing the SM. → The SM is disconnected from its initial attachment. Note: this action could need to be repeated several times for all the existing SI connections to the SM.
8	OBCW-S sends successive TC “ Move Walking Manipulator Cartesian (trajectory) ” or “ Move Walking Manipulator Joint (trajectory) ” to the WM Controller. → The WM extremity attached to the manipulated SM is moved to the final SM position by successive free collision trajectories (ensured by the planner) to reach the position to initiate the final approach. Note: in the first and last steps, the trajectories need to take into account the requirement of approach angles in case of multiple simultaneous connection.
9	OBCW-S sends TC “ Approach Walking Manipulator (trajectory) ” to the WM Controller. → The SM SI is guided, in impedance control, in the latching position
10	OBCW-S sends TC “ SI state update (SM final SI, connect) ” to the R-ICU controller managing the SM. → The SM is connected to its final position. The WM shall allow end-effector motion (impedance control) to support final alignment of the SI Note: this action could need to be repeated several times for all the existing SI connections to the SM
11	OBCW-S sends TC “ Power Switch (#cPDU, cPDU channel, OFF) ” to the WM Controller → The SM is powered off Note: it could be envisaged to first switch on power through the SM final SI to allow to keep the SM alive
12	OBCW-S sends TC “ SI state update (SI-B, disconnect) ” to the WM Controller → the WM SI-B is disconnected from the target SI of the SM
13	OBCW-S sends successive TC “ Move Walking Manipulator Cartesian (trajectory) ” or “ Move Walking Manipulator Joint (trajectory) ” to the WM Controller → The free disconnected WM extremity is moved to the final goal position

5.6.2.3 Routine 3: SM Power-On and Configuration

This routine describes the sequence of operations between the OBC-S and the SMs when a SM is powered on (e.g. after a SM re-location or when the spacecraft is fully restarted) that will configure the SM for data and power configuration.

The following initial conditions need to be full-filled to start the sequence:

- The SM1-DMS is powered and running (including the OBC-C and its R-ICU).
- The OBC-S is connected to the SVC R-ICU
- The SVC R-ICU is connected to the SM1-PWS R-ICU, such that the OBC-S can communicate and control it (TM/TC). This can require a a-priori software re-configuration to move R-ICU control authority from the OBC-C (client) to the OBC-S (servicer)
- Another SM (e.g. SM2) is connected to the SM1 and powered-off



Preliminary Design Document

Planner TC Sequence:

Step #	Action
1	OBCW-S sends TC " Power Switch (cPDU SM1, cPDU channel to SM2, ON) " to the SM1 cPDU Controller. → The electrical power connection is established between the SM1 and SM2. The SM2 R-ICU is automatically started and it established a SpaceWire link with SM1-R-ICU.
2	When the OBC-S polls SM1 R-ICU for TM status, it will be informed of the list of connected SpW ports and the availability of the expected SM2
3	OBCW-S sends TC " Configure R-ICU (SM2) " → Initial plug and play configuration are performed to expose SM2 services and allocate logical address for these services
3	OBCW-S sends TC " Configure Data Routing (SM1 R-ICU, route table) " to configure the SpW Router of SM1 → This enable the routing of packets to the services provided by SM2 → The OBC-S can poll TM both from SM1 and SM2

This sequence can be also applied between other SM and could require multiple R-ICU routing re-configuration. Once the spacecraft is assembled, other mechanisms of interactions (e.g. publisher/subscriber) could be considered.



6 Components Preliminary Design

6.1 Design and Simulation Tool

In the context of the MOSAR project, a simulation environment has been built that covers multiple multi-physics simulation aspect of the scenario. The simulator is implemented as a functional engineering simulator that provides a functional representation of the space system and in particular of the MOSAR system demonstrator. It allows for analysis of relevant aspects of the system reconfiguration procedures.

6.1.1 Design Tool

In MOSAR, the system configurations before and after reconfiguration are defined by the mission control team. This includes the definitions

- which SMs are involved in the given scenario,
- how the SM stack is assembled at the beginning of the reconfiguration and how at termination of the reconfiguration,
- where the WM is attached to the system, either with one or with two end-effectors

In order to guarantee for consistency of initial and final configurations a Matlab script-based tool is developed that is utilized to perform the following pre-processing steps.

- Selection of the components which are part of the scenario: The inclusion of components into the scenario takes place by selection of a HOTDOCK link. If a HOTDOCK link is set the components which the selected HOTDOCKs are attached to are part of the scenario.
- Description of the system configuration: The system configuration is implicitly given by the selection of HOTDOCK links and the relative orientation of the linked HOTDOCKs.
- Setting system parameters: Most of the system parameters will be constants, which do not change anymore once the spacecraft, SM, WM, and mission designs but also the demonstrator design are finished. The remaining parameters specify initial system states like the initial locking state of a single HOTDOCK, the initial temperature of a SM or the orbital pose of the platform.
- Check for system consistency: These checks avoid that HOTDOCKs are multiple times connected. Moreover, it is proven that all selected HOTDOCK links are kinematically realizable. This applies to the HOTDOCKs attached to the SMs and the SVC-CLT-platform but also to those attached to the WM end-effector.

The key element of this tool is a collection of text based data structures, which describe the components HOTDOCK, SM, WM, and SVC-CLT platform in a unified manner in terms of mechanical, electrical, thermal and data link properties as well as system assembly aspects.

These data structures are used as database for parametrization of the simulator model and the planner components. Individual parsers are used to translate the generic system information of this database into specific descriptions for simulator models and planner but also for synthesis of the WM controller.

6.1.2 Purpose of Simulator

The purpose of the simulator is the support of the system design and the replacement of the space system for operations planning on ground. The two application cases are shown within the overall system development scheme in Figure 6-1. The two iteration circles also show the components involved in the respective development stages.



Preliminary Design Document

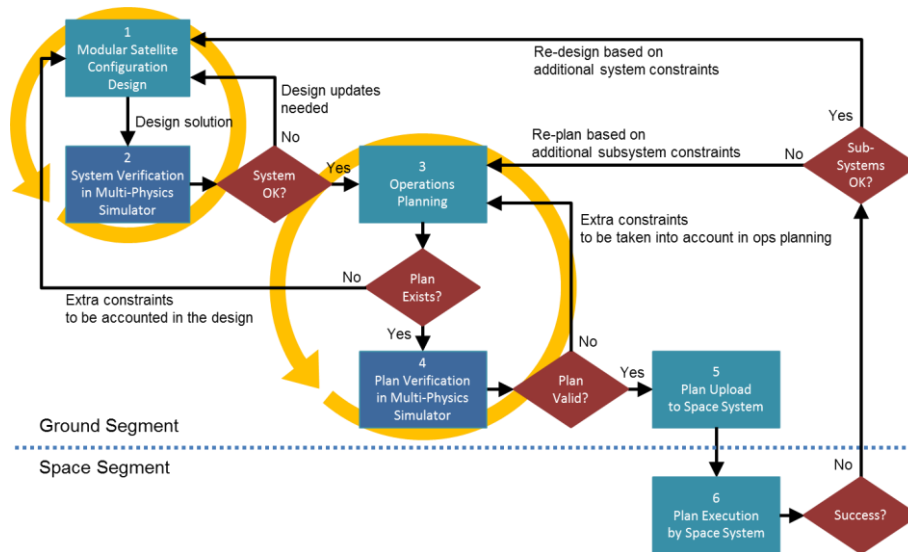


Figure 6-1: Simulator application cases within the overall system development scheme in MOSAR

6.1.2.1 System Design Support

For system design support, the simulator works as a stand-alone system and is operated by a simulation expert. This expert is setting the system parameters and initial conditions and selects the test procedures. With the simulation tool, various aspects of the system design can be verified:

- The assessment of the spacecraft modules (SM) dimensions with respect to the length of the walking manipulator (WM)
- The assessment of the walking manipulator's dexterity regarding reachability of the SMs at various HOTDOCK interface positions
- The analysis of elementary manipulator motions regarding potential self-collision and collision with SMs or the spacecraft platform
- The functional assessment of initial and final spacecraft and SM stack configurations regarding mechanical, electrical and thermal components
- The AOCS and the controllability of the overall spacecraft assembly in orbit

6.1.2.2 Operations Planning

For operations planning the simulator works as an independent executable in cooperation with the planner. In this application case, the simulator replaces the real space system in a way that is fully transparent for the planner. In this configuration the simulator has two purposes: In the pre-mission phase, it is a verification tool for the planner performance. During the mission, the planner develops the system operation plan on ground using the simulator for later upload and execution at the space system.

Control inputs to the simulator and sensor data feedback from the simulator are exchanged with the planner via Ethernet using the TCP/IP network communication protocol. The system parameters and the initial conditions of the scenario are taken from a common database, which is also used by the planner in order to guarantee for consistency of the underlying scenarios. The scenarios to be supported are introduced in Section 5.



Preliminary Design Document

6.1.3 Components of Simulator

The simulator is based on the Modelica/Dymola technology. This technology provides a platform for object-oriented, acausal modelling of multi-physics systems which is required to solve the simulation objectives in MOSAR, in particular the system reconfiguration objectives. The simulator is based on Modelica libraries from various domains which are relevant for MOSAR:

- DLR SpaceSystems library
- DLR Robots library
- DLR RobotDynamics library
- DLR Visualization library

Using components of these libraries, a specifically tailored library for MOSAR was prepared that includes all models and sub-models which build up the MOSAR simulator. The library tree is shown in Figure 6-2 and an overview of the key components of the simulator can be found in Figure 6-3.

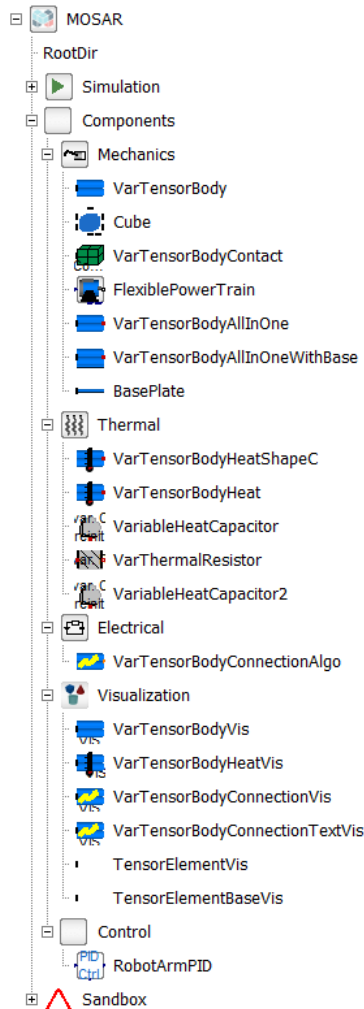


Figure 6-2: Modelica library prepared for MOSAR



6.1.3.1 Mechanical System & Multi-Body Dynamics

The multi-body simulation part includes models of all bodies and mechanical assemblies that characterize the MOSAR system:

- The servicer spacecraft (SVC) rigidly docked to the client satellite bus (CLT). Both bodies build the orbital platform where the reconfiguration activities take place.
 - For general system verification and planning verification, the platform is simulated in an orbital environment and AOCS capabilities in order to keep the platform stable in case of reconfiguration activities.
 - In the case where the simulator replaces the MOSAR demonstrator for planning purposes, the platform is rigidly connected to the ground without any AOCS functionalities.
- The walking manipulator (WM): According to the design, the manipulator consists of eight links and seven driven joints. Both manipulator ends are equipped with HOTDOCK interfaces to enable walking and SM grasping capabilities. Moreover, the WM model includes models of the joint motor and gearbox dynamics as well as corresponding joint controllers. For the Cartesian WM control option, the WM model includes also a 7-axis inverse kinematics algorithm that is applicable to kinematically redundant robotic systems.
- The spacecraft modules (SM): The spacecraft modules are modelled as a so-called grid body. It is generally a full SM stack with individually and dynamically activatable and deactivatable SM for implementing the modular reconfiguration, depending in the HOTDOCKs' respective lock status (see below). The SMs have individual inertia properties, depending on their respective internal equipment.
- Strictly speaking, the HOTDOCK interfaces are just sub-components of the SVC-CLT platform, of the walking manipulator's two end-effectors, and of the spacecraft modules. However, the HOTDOCKS are mentioned here since they provide the mechanism to lock and unlock body pairs and implicitly, to redefine the multi-body system of the overall scenario. Their respective functionality, in particular with respect to simulating the MOSAR demonstrator, are given



- Table 5-2.

6.1.3.2 Electrical System

The models of the electric domain are integrated in the SMs models. The electric models build up an electric network depending on the topology of the SM stack. They allow for analysis of voltage and current as well as charging status. The SMs may have the following electrical properties:

- All SMs are electric network nodes with 6 power lines connecting the node and the docking interfaces, respectively. The lines have a defined electrical resistance at closed HOTDOCK interfaces. If the HOTDOCK interface is open the electrical resistance is supposed to be infinitely high.
- Selected SMs may act as electrical energy buffer (rechargeable battery) inside the electrical network.
- Selected SMs may act as electrical energy consumer (sinks).
- Selected SMs may act as electrical power source, e.g. SMs with solar arrays attached to it.

6.1.3.3 Thermal System

The implementation of the thermal domain models is very similar to the electrical domain. The thermal models are also integrated in the SMs models and build up the thermal network. The vales to be analysed are the temperature of each SM and the heat flow between the SMs. The SMs may have the following thermal properties:

- All SMs are thermal network nodes with 6 thermal conductors linking the thermal node and the HOTDOCK interface. The conductors have a well-defined thermal resistance at closed HOTDOCK interfaces. If the HOTDOCK interface is open the conductors behave like ideal isolators. Further details for simulating the MOSAR demonstrator scenario can be found in
- Selected SMs may act as thermal buffers with given heat capacity to balance the overall temperature of the SM stack.
- Selected SMs may act as active heaters to warm up the thermal network.
- Selected SMs may act as radiators to cool down the thermal network.

6.1.3.4 Data System

The simulator does not provide any data processing capabilities. However, it provides models which are integrated in the SMs and which can build up a data network via the HOTDOCK interfaces. During simulation one can identify the individual SM data connectivity and verify if the processing units inside the SMs are able to exchange data.



Preliminary Design Document

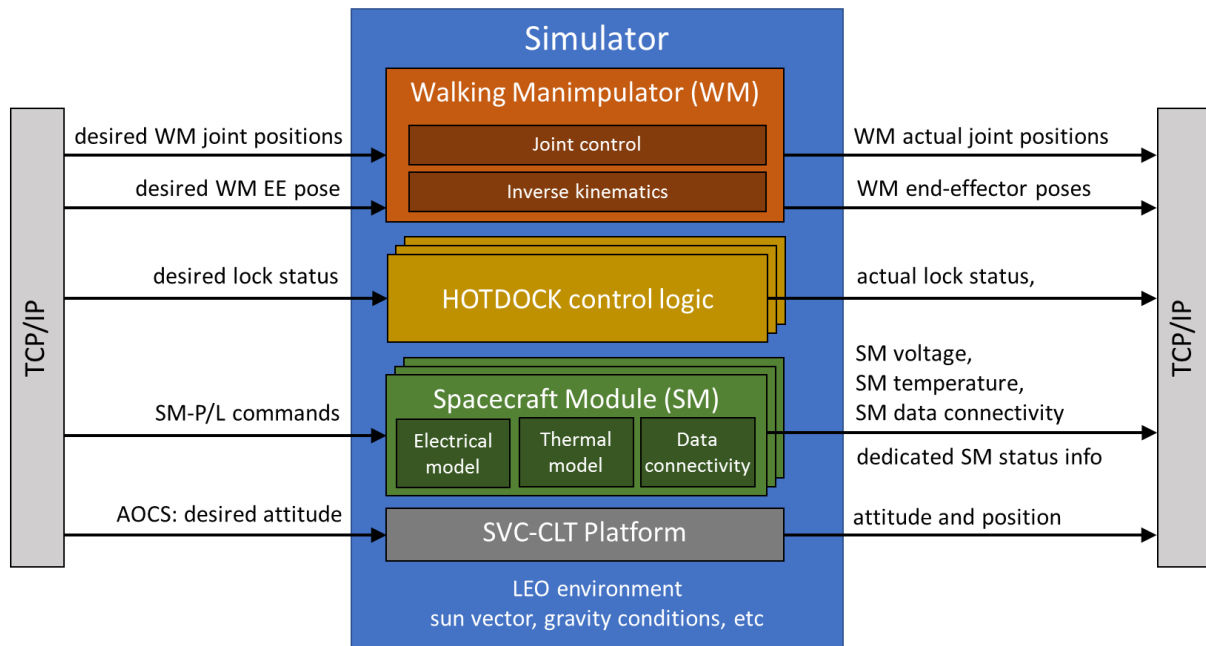


Figure 6-3: Simulator components and interfaces

6.1.4 Integration of the Simulator

In addition to the simulator components, Figure 6-3 gives an overview of the ingoing and outgoing signals. These signals can be split in more planner-specific signals and more mission-specific signals as shown in Table 6-1. While the planner-specific signals are more related to system reconfiguration and verification of the reconfiguration effects, the mission-specific signals are more related to payload operations described in section 5.6.

Table 6-1: Input and Output Signals of the Simulator

Signals	Input	Output
Planner-specific	<ul style="list-style-type: none">Desired joint positions of WMDesired end-effector poses of WMDesired lock status of HOTDOCKS	<ul style="list-style-type: none">Actual joint positions of WMActual end-effector poses of WMActual lock status of HOTDOCKSSM status<ul style="list-style-type: none">SM voltageSM temperatureSM data connectivity
Mission-specific	<ul style="list-style-type: none">Payload-specific commands to be sent to dedicated SMs carrying the payload (TBD)Desired SVC-CLT platform attitude	<ul style="list-style-type: none">Payload-specific telemetry of dedicated SMs carrying the payload (TBD)Actual SVC-CLT platform position and attitude

The signals are exchanged with the planner and the mission control center via Ethernet using the TCP/IP protocol.



Preliminary Design Document

6.1.5 Simulator Verification

The MOSAR simulator has been verified by a number of demonstration simulations of on ground scenarios as used for the MOSAR demonstrator (Figure 6-4) as well as orbital scenarios (Figure 6-5). In particular, the correct multi-domain (mechanical, electrical thermal) modelling and the free SM reconfigurability were verified. For better understanding, the simulation results are visualized in Figure 6-4 and Figure 6-5 in different ways:

- The respective electrical connectivity of the SM is expressed by green (connected) and red (disconnected) markers
- The respective heat capacity of the SMs is expressed by the size of spheres located in the center of the SMs.
- The respective temperature of the SMs is expressed by the darkness of the red sphere surface.

In the current status, the simulator is well prepared for the system verification and operations planning activities. In particular, verification work according to 6.1.2.1 has already been performed and results were shared during system design discussions within the MOSAR team.

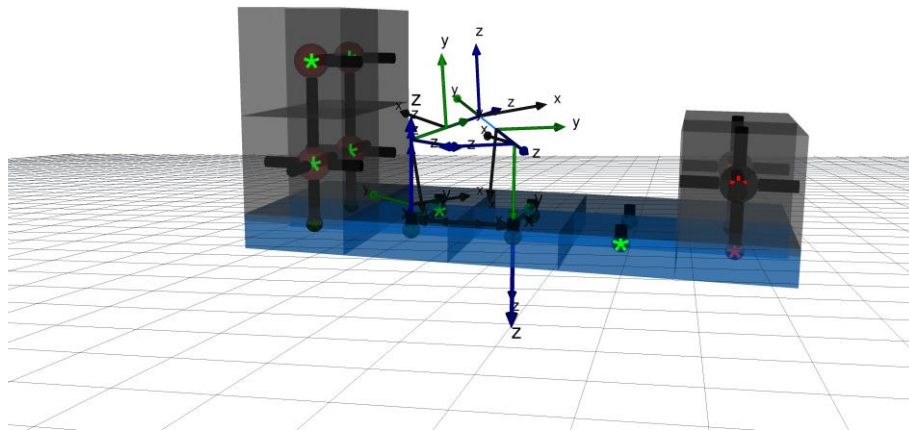


Figure 6-4: Simulator verification taking into account models from the mechanical, electrical, and thermal domain

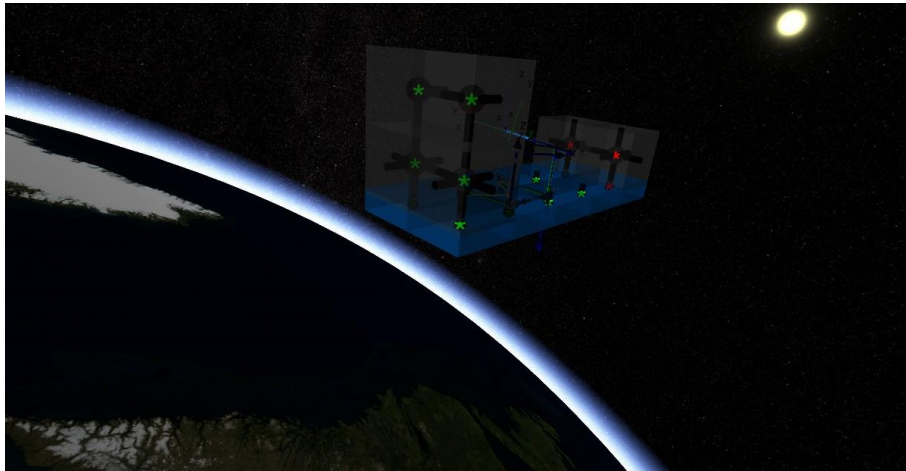


Figure 6-5: Simulator verification inside an orbital scenario



6.2 Autonomy Agent and Planning

The ERGO framework is used in MOSAR to develop the planning and autonomy functions that will support the reconfiguration of the spacecraft. The framework developed in the first PERASPERA call needs to be adapted and extended to fulfil the needs of the MOSAR project.

The core component of ERGO that needs to be adapted and extended for the MOSAR demonstrator is the Autonomy Agent. This Agent contains the hierarchy of reactors that manage the execution of the robotic operations in time, including the deliberative components that compute the plan. It needs to be tailored to the robotic operations envisaged in MOSAR, and extended to support the verification of the plan off-line in the MOSAR simulator, and the monitoring of the final execution in the demonstrator system.

The planning capabilities provided by the ERGO framework and applicable to MOSAR are of two types:

- **Mission Planning:** in the case of MOSAR, this corresponds to the planning of the robotic reconfiguration in terms of high-level operations, e.g., move a module from one position to another, or relocate the WM. This capability is provided by the **Mission Planner Reactor**, which is part of the **Autonomy Agent**. In the operations concept chosen for MOSAR, the planning is done on ground (see section 5.5), and the flight system only executes a precomputed plan that has been validated in simulation.
- **Robotic Arm Motion Planning:** this is the computation of collision-free trajectories for the motion of the manipulator to a desired position of the end effector, either for grasping a module or for relocating the manipulator. This capability is provided by the **Robotic Arm** component, which is part of the **Functional Layer** that interfaces the Autonomy Agent with the low-level software components that interface with the hardware.

6.2.1 Autonomy Agent

The Figure 6-6 below presents an overview of the tailoring of the ERGO Agent for the MOSAR demonstrator. The Agent is deployed on the Servicer OBC and manages the robotic reconfiguration.



Preliminary Design Document

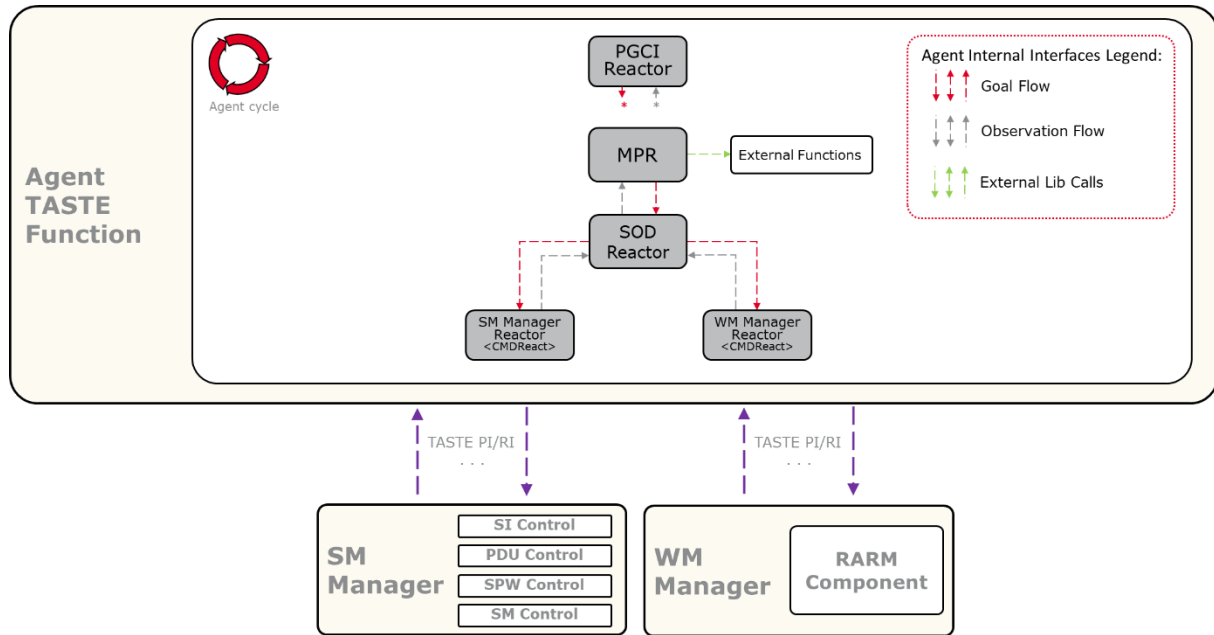


Figure 6-6: Overview of the ERGO Agent tailoring for MOSAR

The Agent and its interfaces are modelled as TASTE functions (which should be understood as components). The figure shows the Agent function and the functions in the Functional Layer that makes the capabilities of the system accessible to the Agent. In this case, two Functional Layer functions are defined: the SM Manager and the WM Manager, which respectively give access to the capabilities of the Spacecraft Modules (including SI, power, data and SM configuration) and the Walking Manipulator.

The Agent is the component that plans and controls the actions of the system. This component will also handle all telecommands related to goals sent to robotic system from ground and telemetry generated on board and received from ground.

The architecture of the Agent consists of a controller, in charge of coordinating the different execution loops, and a set of different reactors, in charge of executing a specific control loop. The Agent is a generic component that has to be configured based on the needs of the application to be developed. This involves:

- Defining the reactors that will be part of the agent, and
- Defining the variables (or timelines) that each reactor owns and shares with other reactors.

For the MOSAR, the following set of reactor types are used:

- PGC I Reactor (PUS Ground Control Interface Reactor): reactive reactor that manages the communication with ground; it does not own any timelines, but it is subscribed to all the timelines in the system in order to report all the observations to ground and to enable direct command from ground of all the reactors
- MPR Reactor (Mission Planner Reactor): deliberative reactor that computes the plan (on the plan calculation and verification stage on ground) and manages its execution (both for verification on the simulator and actual execution in the demonstrator)
- SOD Reactor (SM Operations Decomposer Reactor): reactive reactor that decomposes the plan goals into low-level ones
- CMD Reactor (Command Dispatcher Reactor): reactive reactor responsible for interfacing with the functional layer and performing elementary goals; two CMD reactors are defined:



Preliminary Design Document

- SM Manager Reactor: controls the execution of operations on the SMs (mechanical, power and data connections of the SIs, and operation of each SM)
- WM Manager Reactor: controls the operation of the WM (relocation and grasping)

The PGCi Reactor is an adaptation of the GCI Reactor existing in OG2, prepared to use PUS packets instead of files for receiving telecommands and sending telemetry. This new reactor is completely integrated with the PUS Services (see section 6.3).

The PGCi is partly implemented in a TASTE function with the following interfaces:

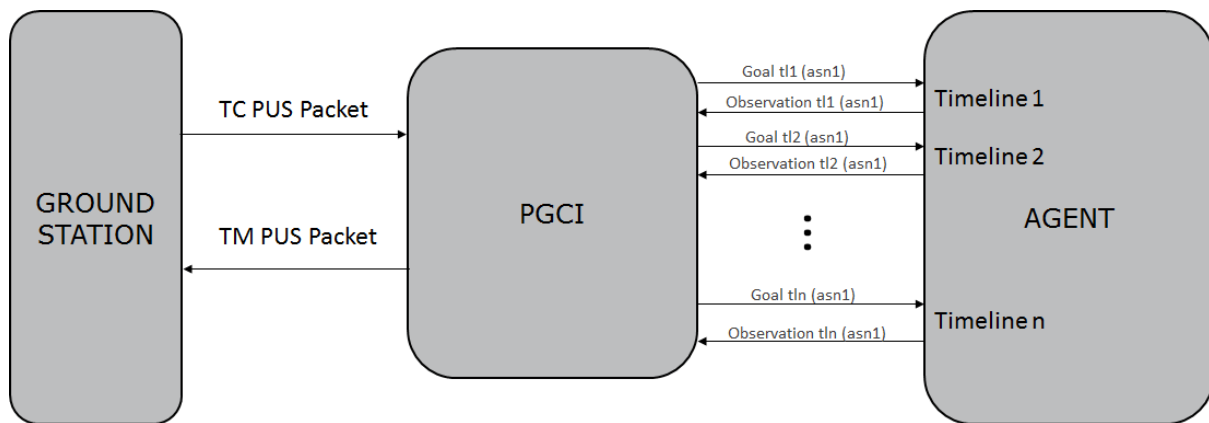


Figure 6-7: OG2/ERGO Ground Control Interface (GCI) Adaptation

The PGCi receives PUS packets with the commands and send packets with the telemetry. It also unpacks the data inside the packet and sends the goal to the corresponding timeline in the agent.

For the development of this new component, there is a strong synergy between different OG projects (e.g., PRO-ACT), being the structure and general development of this new reactor the same for all that just needs to be instantiated for each particular scenario.

6.2.2 Mission Planning

The Mission Planner Reactor (MPR) is the reactor in charge of interfacing with the PDDL planning software Stellar. PDDL is a standardized language to express planning problems with a temporal domain. In order to calculate a plan for the robotic reconfiguration, the domain and the problem must be modelled in PDDL. Stellar uses this representation to compute a plan that achieves the goals of the reconfiguration.

The MPR uses Stellar to generate a new plan and executes it. It is also able to execute a plan already computed on ground. To implement this reactor in the MOSAR scenario, several adaptations and extensions have to be developed.

The first step to instantiate the Mission Planner Reactor is to define the problem that has to be solved using the planner. In the MOSAR scenario this means reconfiguring the spaces modules (SM) from the initial state to the desired one, which includes taking into account the modules structure and the walking manipulator.

The computation of a valid plan is a search problem. The way that the problem is modelled determines the efficiency of the search, and the feasibility of obtaining a solution with the temporal and spatial (memory) constraints of the application.



Preliminary Design Document

The reconfiguration problem in MOSAR is combinatorial, in the sense that the number of possible moves increases quickly with the number of modules in the system. Initial tests have been performed with PDDL to assess the right abstraction level for modelling the MOSAR scenario.

Different strategies, such as dividing the problem in simpler ones, adding assumptions and heuristics, and pre-calculating part of the plan with alternative means have been explored. These activities will continue in the detailed design phase, in order to optimize the representation of the problem and the capabilities of the Mission Planner Reactor.

6.2.3 Robotic Arm Motion Planning

ERGO provides a Robotic Arm component (RARM) that implements the functionalities needed to plan and execute the movements of a robotic arm. It plans trajectories and paths between points avoiding any collision.

The main particularity of the MOSAR scenario is that the WM is capable of walking over the structure, relocating and alternating the roles of its base and end effector. The ERGO software must be extended to support the MOSAR WM and Simulator, and to implement the new capabilities such as walking.

In addition, the scope and responsibilities of the manipulator control differ between ERGO and MOSAR. While in ERGO the RARM software interpolates the robot trajectory and commands the arm controller at fine trajectory level, in MOSAR the RARM software will generate a coarse trajectory and it will be the WM Controller that will interpolate it. The aim is to minimize latency.

In the MOSAR system, the WM is connected to the OBC-S through a chain of SpaceWire links. This can introduce an unpredictable latency that is not compatible with the impedance control of the arm. Impedance control is needed to push the standard interfaces together, compensating the possible misalignments to enable a successful locking of the mechanism. This is handled locally by the WM Controller, which reads the torque measurements and commands the joint motors at a very high rate.

The architecture of the Robotic Arm software component is shown in Figure 6-8 below.

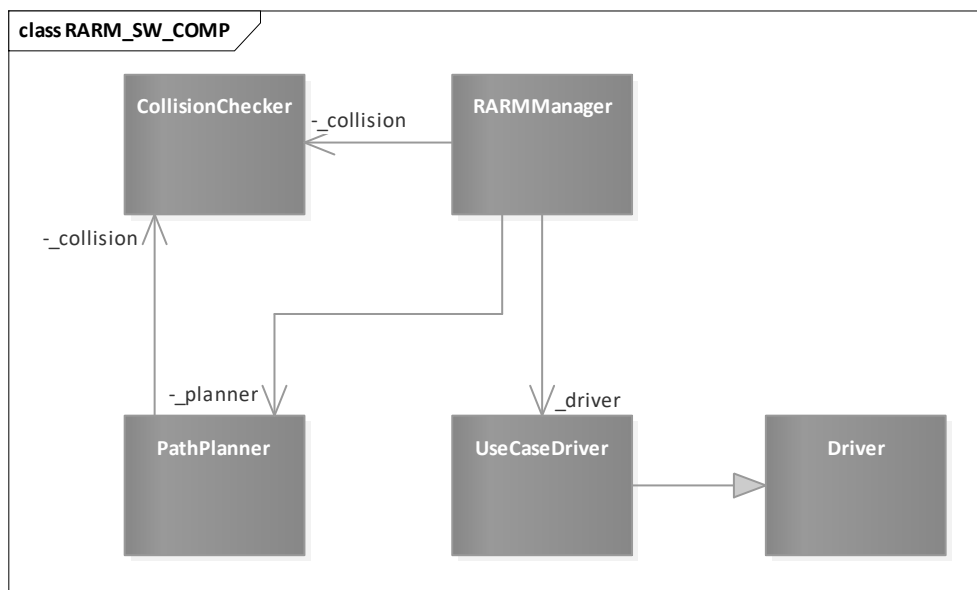


Figure 6-8: Robotic Arm Main Components



Preliminary Design Document

The ERGO Robotic Arm package is composed of a set of classes, mainly:

- **PathPlanner:** The PathPlanner class uses the library OMPL (Open Motion Planning Library) which implements various path planning algorithms in a generic and efficient way. It only needs to receive the description of the state space and an external function to check the validity of the sampled points.
- **CollisionChecker:** The collision checker provides functionality to load the robot model from URDF and keep track of all the objects in the scenario. Collisions are checked for a certain pose moving the arm virtually to the requested configuration and checking collisions with itself and the environment. The forward and inverse kinematics are computed purely based on the robot model. The forward kinematics moves the arm virtually to the requested joint configuration and retrieves the pose of the selected end effector. The inverse kinematics uses the default algorithm of the library, the returning position has to get checked for collisions afterwards.
- **Driver:** This package contains the classes associated to the movement of the robotic arm. The Driver class is an interface functionality that declares the desired methods based on the required functionality of the arm. To actually implement this functionality for each specific hardware, a class must inherit from Driver and implement the required functionality with the hardware API.
- **UseCaseDriver:** Implementation of for the specific use case (e.g., specific manipulator or simulator implementation).
- **RARMManager:** Class that handles the execution of the rest of the components.

This framework has as inputs from the rest of components the commands to be processed by the RARM manager, and which depend on the application being developed. In the case of MOSAR, in addition to the commands to move the end effector to a desired position and orientation, additional commands are needed to switch the end effector (allowing the arm to walk on the structure), and to indicate when the WM is carrying a SM.

The valid positions for the WM are constrained by the location of the SMs and the HOTDOCK interfaces that serve as attachment points. The trajectory to approach a given interface, however, varies according to the configuration of the system, in particular when a SM must be docked simultaneously with several standard interfaces.

The outputs of the RARM component are the requests sent to the WM or to the Simulator by the corresponding Driver class. An important change with respect to the original ERGO component is that the communication with the manipulator controller is interfaced at a different level. In ERGO, the RARM Driver commanded the arm controller at high rate, while in MOSAR the RARM component provides a coarse, collision-free trajectory to the WM Controller, and it is the latter the one that interpolates the trajectory and commands the actuators at a high rate. This avoids latency issues that might be introduced by the communication across several SpaceWire links.

The Driver components must be developed from scratch to interface with the MOSAR WM and its Simulator. As explained above, the interface with the WM Controller is done at the level coarse trajectory (instead of at the level of interpolated trajectory in ERGO). In addition, the MOSAR system does not rely on vision for the alignment of the end effector, so ERGO RARM Camera component will not be used.



6.3 Telemetry and Telecommand Service

The Telemetry and Telecommand (TTC) Service enables the operator to command and monitor the demonstrator from the MCC, both during the robotic reconfiguration of the system and in the nominal operation of the CLT platform before and after the reconfiguration scenario.

Two instances of the TTC Service are therefore defined:

- The CLT instance is used to monitor and command the SMs during nominal operations. In the demonstration scenario, it will enable the operator to determine when the SMs are operational, to receive telemetry and to command the functions provided by each module.
- The SVC instance is used to monitor and command the Autonomy Agent during the robotic reconfiguration.

The TTC Service will be based on the ESROCOS PUS Services library, which implements a subset of the services defined in the PUS standard [RD1]. The MOSAR demonstrator makes use of some of these services, and in addition it defines a mission-specific service 200, described below, for interacting with the ERGO Agent.

The MOSAR system contains three PUS nodes identified by a unique APID:

- SVC-OBC: running the ERGO Agent and the Functional Layer software during the robotic reconfiguration
- CLT-OBC: running the SM management software during the nominal system operations
- MCC: running the ground segment software

TCs are sent from the MCC to the two OBCs, and TM generated by the OBCs is sent back to the MCC.

The standard services that are selected, on a preliminary basis, for implementation in the MOSAR demonstrator are shown in the table below.

Table 6-2: Standard PUS services

Service	Name	Purpose and role in the demonstrator
1	TC verification	Telecommand acknowledgement and success/error reporting, for all the services implemented in the SVC and CLT OBCs
3	Housekeeping	Telemetry reporting of housekeeping parameters: <ul style="list-style-type: none">• From the Functional Layer software• And from the SM operational software
5	Event reporting	Telemetry reporting of on-board events: <ul style="list-style-type: none">• From the Functional Layer software• And from the SM operational software
8	Function management	Execution of on-board software functions, in support of testing and OBCP execution
18	On-board control procedure	Loading and execution of procedures defined in a high-level programming language (Micropython), for automation during testing and operations
20	On-board parameter management	Setting and reading of on-board software parameters: <ul style="list-style-type: none">• From the Functional Layer software• And from the SM operational software



Preliminary Design Document

Service	Name	Purpose and role in the demonstrator
23	File management	File transfer service, useful to upload configuration data to the on-board system at runtime

The standard services need to be missionized to fulfil the needs of the MOSAR demonstrator by dimensioning the different services and specifying the relevant housekeeping parameters, runtime events, software functions, etc. The configuration of the services is done using JSON files. The parameters for the MOSAR demonstrator are detailed at a later design stage.

The standard services will be implemented both in the CLT and the SVC components. While not all the services are required for the nominal execution of the scenario, they may be useful to support the development and investigate anomalies during the execution of the demonstrator.

In addition to the standard PUS services, a new private service will be developed to interact directly with the agent. This new **PUS service 200** will be used to send goals and receive telemetry from the agent interacting with the PUS Ground Control Interface (PGCI).

In OG2 the commanding and monitoring of the system was done via files. The approach to use files in ERGO was developed in the frame of a Martian operation with large latencies to send any data, but in an orbital mission like the one in MOSAR, a more direct communication approach can be used. The extension of the OG2-GCI needed to integrate this new PUS service is explained in the section 6.2.1.

To implement this new service, several new functions and definitions within the mission definition in the PUS library need to be created:

- Definition of the new PUS packets using the ASN.1 goals and observations defined in the agent.
- Definition of the new service.
- Definition of the functions in charge of the creation of the PUS packets, from a goal/observation and retrieving a goal/observation from a PUS packet.



6.4 Spacecraft Modules and Payloads

Modularity is expressed on the satellite structure as the capability to build the platform through standard thermo-mechanical modules assembled and interconnected together through HOTDOCKS.

6 modules with different functionalities are designed in order to perform MOSAR demonstration, as listed below:

- SM1-DMS, containing the Data Management Subsystem
- SM2-PWS, containing the Power Subsystem
- SM3-BAT, containing the Battery Assembly
- SM4-THS, containing the Thermal Subsystem
- SM5-OSP1, containing Optical Payload
- SM6-OSP2, containing Optical Payload

In this section we present the preliminary design of the 7 ASM/APM modules that will be demonstrated in MOSAR (including the backup one). The design will be worked out so that to satisfy WP1 requirements, taking into account the targeted ground test scenarios. The target implementation foresees the capability to integrate a HOTDOCK on each face of the SM. The exact number and positioning of the different HOTDOCKs variations, for each APM/ASM and test bench, has been analyzed in section 0 and are reflect in the proposed preliminary design. It followed an iterative analysis process to optimize the system design according with the ground demonstration constraints (SM weights, modularity demonstration).

6.4.1 Structural concept of a generic Spacecraft module

A standard cubic shape for the SMs was chosen in MOSAR preliminary design phase with a baseline edge of 350 [mm], based on a compromise between components integration, module weight and kinematic compatibility with the proposed WM size.

Figure 6-9 shows the initial concept of the module structure with slots for HOTDOCK interfaces.

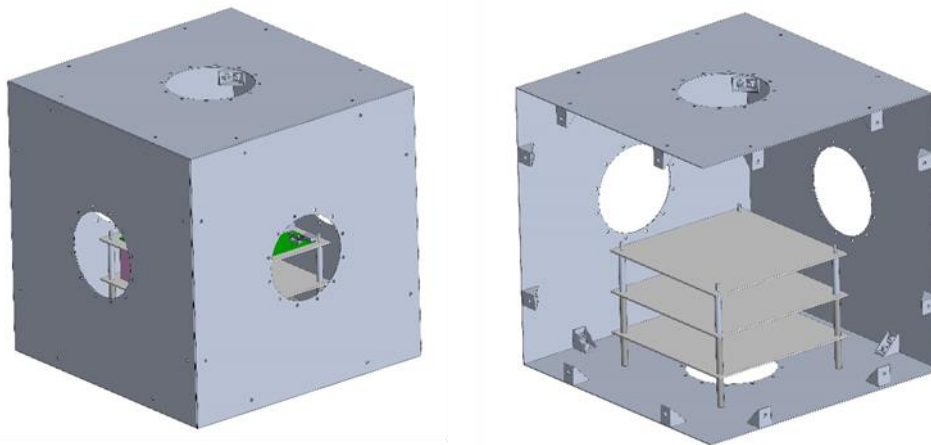


Figure 6-9: Initial concept of spacecraft Module structure

The baseline design of the SM features a customized panel with a central tray structure able to accommodate the module payload boards as well as the service boards. A second customized panel will be used as radiative panel in the Thermal Subsystem (THS) Module. For the OSP1 and OSP2 Modules one of the lateral panels will be adapted in order to accommodate the payload cameras.



Preliminary Design Document

In order to decrease the mass impact of the structure on the overall mass of the SM a panel-based concept has been adopted for the design of the structure. The module external structure will be composed by 6 aluminium panels giving the cubical shape to the Module. The structural panels are connected by means of two structural cleats for each edge of the panel.

The panel-based structure can guarantee high standardization during design and interface definition, manufacturing and integration. Exploiting this modular approach simplifies customizations for each subsystem integrated inside the module. The external structure manufactured for ground demonstration will be made of 6 lateral panels, 24 cleats, with, one customized face used to mount the central tray structure, and other lateral panel customizable due to the needs of each Module. The support structure is one of the few module components that will be specified and designed according to the units and devices that will be integrated inside. Next figure shows a preliminary concept of the internal structure that supports a generic avionic board. This approach is used into and the specific unit of the Module. This internal tray structure has been designed in order to provide a standard approach to the integration of internal boards inside the modules and it was applied to each module (R-ICU and cPDU). Due to the mechanical characteristics of the boards of each module the internal structure has been divided in 3 trays with the following functions:

- Top tray: has been used to accommodate the R-ICU
- Intermediate tray: has been used to accommodate the cPDU
- Bottom tray: has been used to accommodate the specific unit of the module

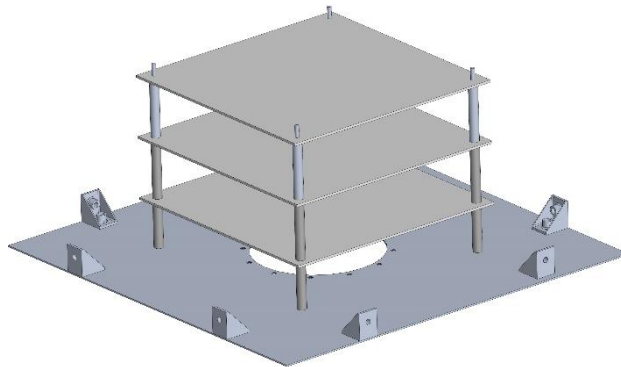


Figure 6-10: Customized face hosting the internal tray structure

In fact, R-ICU and cPDU are common boards to any standard module and they constitute the main module electronics for data and power management.

Following an early analysis, and to allow performing required demonstrations under Earth gravity, the total mass of the external and internal structure of each non-walkable spacecraft Module has been fixed to a maximum of 4 [kg], while the estimated mass of walkable spacecraft is less than 6 [kg].



Preliminary Design Document

6.4.2 SM1-DMS

The SM1-DMS is a service module, with data management system functionalities. The physical layout of the SM1 is shown in Figure 6-11. The internal equipment of the SM1 is composed by:

- The R-ICU for the local SM control and data routing, as well as the high level control of the SM2-PWS thermal payload (the SM2-PWS doesn't feature an R-ICU for accommodation optimization)
- The OBC-C, mainly used for the management of the CLT and the communication with the MCC, during nominal operations (e.g. payload interactions).
- A SpW bridge (USB Brick) for the interface between the R-ICU and the OBC-C
- The cPDU for the power distribution and local conversion (including the SM2 thermal payload)
- 2 Active, 2 fixed and 2 dummy HOTDOCK

The module is permanently fixed on the CLT structure. The two fixed HOTDOCK allow to pass cables respectively to the CLT or the attached SM2-PWS.

The current pre-selection for the OBC is the Intel NUC board, running Ubuntu 18.04 LTS 64-bit. Figure 3-2 provides the list of software components running on the OBC-C.

The SM1 top plate will be specifically designed to allow the walking of the WM on top of it.

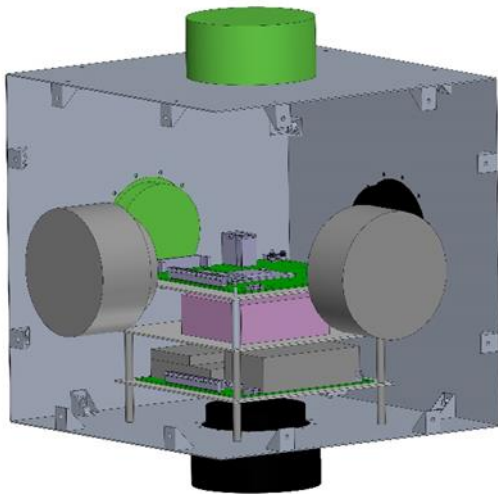


Figure 6-11: SM1-DMS physical layout and Intel NUC board



Preliminary Design Document

6.4.3 SM2-PWS

The SM2-PWS is the powering module of the spacecraft. In the context of MOSAR it is playing two roles:

- The power generator / input for the other SM / spacecraft components, through the 28V main power bus. In the demonstration, the power supply will be external to the module and directly connected to the SM2-PWS power input.
- One of the two parts of the thermal payload (used to demonstrate thermal transfer between SM through HOTDOCK). Being always fixed to the structure, this SM will include the heat generator as well as the heavy components of the thermal payload subsystem (reservoir, pumps...).

The physical layout of the SM2 is illustrated in Figure 6-12. It is currently envisaged that the R-ICU and cPDU of the SM1-DMS are also managing the control and power transmission of the SM2. The internal equipment of the SM2 is composed by:

- The thermal payload (pump, flowmeter, valves, reservoir)
- The thermal payload heater (based on OG5 design)
- The low-level thermal controller that is communicating with the SM1-DMS R-ICU to get TC and provides sensors or status telemetry
- 1 thermal active HOTDOCK that can support fluid transfer to another SM, 1 passive, 2 fixed and 2 dummy HOTDOCKs

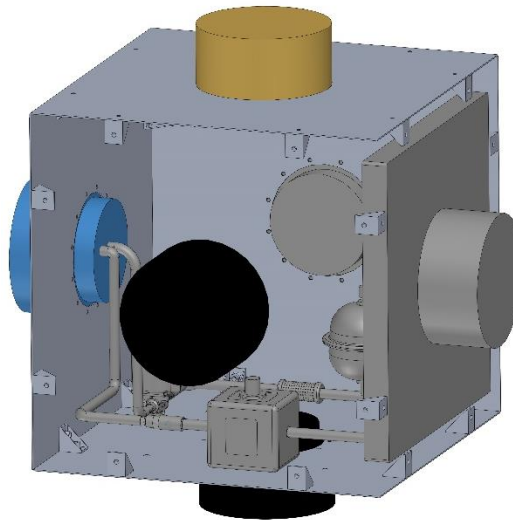


Figure 6-12: SM2-PWS layout

The electrical power transmission through the active/passive HOTDOCK are managed by the (SM1) cPDU, as explained in section 4.1.3.

The purpose of the thermal payload is to control thermal exchange between SM with the goal to dissipate heat generated by the SM2-PWS in the SM4-THS (see below). Table 6-3 provides the software components and TM/TC related to the Thermal Payload operations (split between the SM2-PWS and SM4-THS). The software components are defined according to the generic components used in Figure 3-2. Section 5.6 describes the software architecture and relation between them.



Preliminary Design Document

Table 6-3: Thermal Payload Parameters

Thermal Payload Operations	Context: Spacecraft nominal operations
Software Components	
<ul style="list-style-type: none">• <u>Payload Data Visualization (MCC)</u>: Thermal Visualization• <u>Payload Management (OBC-C)</u>: Thermal Management• <u>Payload Control (R-ICU)</u>: Thermal Control• <u>Payload Controller (Component)</u>: Thermal Controller	
TM	
<ul style="list-style-type: none">• Thermal Control / Mode Status (Idle, Operational, Error)• Thermal Control / Liquid Flow• Thermal Control / SM2-PWS Temperature [x]• Thermal Control / SM4-THS Temperature [x]	
TC	
<ul style="list-style-type: none">• Thermal Control / Mode Command (Idle, Operational)• Thermal Control / Pump Rate Command• Thermal Control / Solenoid Valve Command [x]• Thermal Control / SM2 Heat generator Command (On/Off)• Thermal Control / SM4 Fan Command (On/Off)	
Data	
<ul style="list-style-type: none">• N/A	

When the Thermal Control software receives TC for updating the subsystem, it will communicate with the local Thermal Controller (through CAN) that will manage the physical changes on the setup. That will allow to control the fluid transfer between the two SM demonstrating the thermal management, through monitoring of the SM temperatures.

The Thermal Management component polls the TM data of the Thermal Control software via RMAP. They are then published to the thermal Visualization component on the MMC through PolyORB-HI protocol, in order to be displayed on the GUI.



Preliminary Design Document

6.4.4 SM3-BAT

The SM3-BAT is a battery payload. The purpose of the Battery Payload is to enable the storage of electrical power in the spacecraft, and to be able to deliver this power to other SM (e.g. in isolation or in absence enough energy from the SM power supply). The physical layout of the SM3 is illustrated in Figure 6-13. The internal equipment of the SM3 is composed by:

- The R-ICU for the local SM control, data routing and interface with the Battery subsystem
- The Battery subsystem that enables the storage and control of power direction.
- The cPDU for the power distribution and local conversion
- 1 active, 2 passive, 1 mechanical and two dummy HOTDOCKs

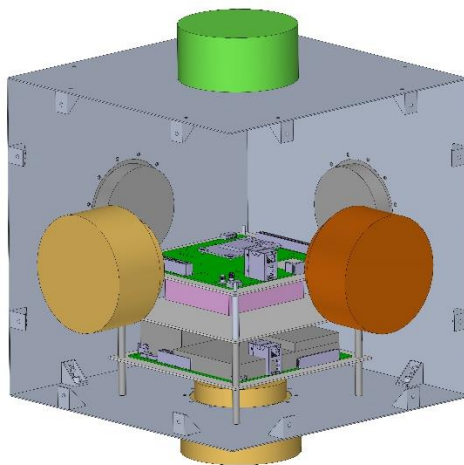


Figure 6-13 SM3-BAT layout

The battery sub-system is based on the design of the OG5 APM battery payloads (see OG5 architecture in Figure 6-14). It is composed of:

- The battery pack, made of 12 individual IR18650 cells
- The battery charger and balancing circuitry
- The battery controller that includes the charging/discharging circuitry (DC/DC converters, power transfer switches) and a microcontroller to control the board components and the communication with the high-level controller (R-ICU). In MOSAR, the board design will be updated to be compatible with the 28V main power bus, enable multi-channel power transfer control and provide CAN communication.

Table 6-4 provides the software components and TM/TC related to the battery payload operations. The software components are defined according to the generic components used in Figure 3-2. Section 5.6 describes the software architecture and relation between them.

All TM/TC and data transfer, between the OBC and R-ICU, are implemented with the RMAP protocol. The current TM/TC definition is based on the assumption that each SI power line can be used independently and simultaneously for charging or discharging.

When the Battery Control software is commanded to charge/discharge, it will communicate with the local battery controller to configure the battery control board and the required electrical switches. That will allow power transmission between the power lines and the battery according to the desired configuration. The Battery Management component polls the TM data of the Battery Control software via RMAP. They are then published to the Battery Visualization component on the MMC through PolyORB-HI protocol, in order to be displayed on the GUI.



Preliminary Design Document

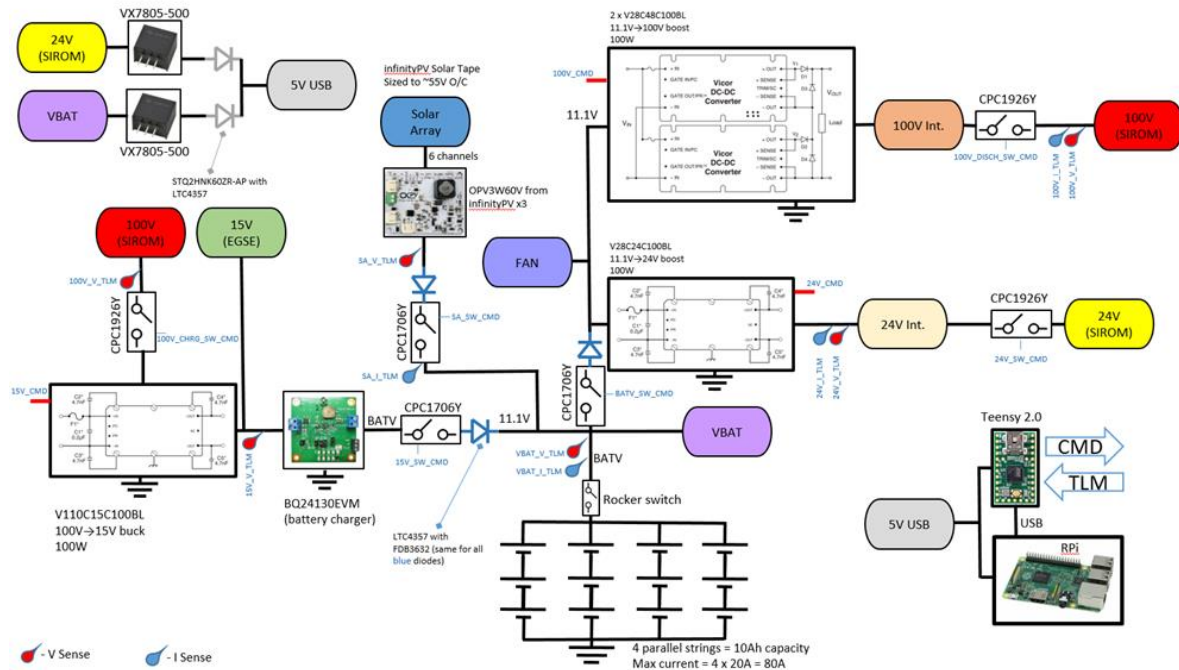


Figure 6-14: OG5 battery controller

Table 6-4: Battery payload parameters

Battery Payload Operations	Context: Spacecraft nominal operations
Software Components	
<ul style="list-style-type: none">• <u>Payload Data Visualization (MCC)</u>: Battery Visualization• <u>Payload Management (OBC-C)</u>: Battery Management• <u>Payload Control (R-ICU)</u>: Battery Control• <u>Payload Controller (Component)</u>: Battery Controller	
TM	
<ul style="list-style-type: none">• Battery Control / Mode Status (Idle, Charging [6], Discharging [6], Error)• Battery Control / Battery Voltage• Battery Control / Battery Current• Battery Control / Battery State of Charge• Battery Control / Bus Voltage [6]• Battery Current / Bus Current [6] (positive when charging)• Battery Control / Battery Temperature	
TC	
<ul style="list-style-type: none">• Battery Control / Mode Command (Idle, Charging [6], Discharging [6])	
Data	
<ul style="list-style-type: none">• N/A	



Preliminary Design Document

6.4.5 SM4-THS

The SM4-THS is the thermal module of the spacecraft. In association with the SM2-PWS, it allows to manage power transfer along the spacecraft structure. In MOSAR, it is used to demonstrate heat transfer with the SM2-PWS. The physical layout of the SM4-THS is illustrated in Figure 6-15. In order to minimize the weight of the module, most of the thermal sub-system equipment has been located in the SM2-PWS. The internal equipment of the SM includes:

- The R-ICU for the local SM control, data routing and interface with the thermal controller
- The cPDU for the power distribution and local conversion
- The thermal controller to manage the thermal equipment inside the SM4-THS
- Valves for the isolation of the thermal circuitry, once the module is disconnected
- A radiator and fan to catch the heat transferred by the pipes and dissipate it in the surrounding of the spacecraft
- 1 thermal active HOTDOCK that can support fluid transfer to another SM, 2 passive, 1 mechanical and 1 dummy HOTDOCKs

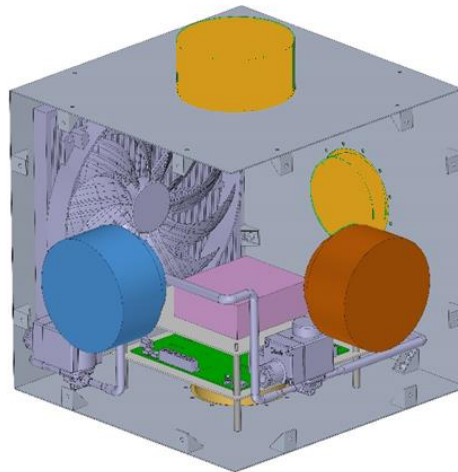


Figure 6-15: SM4-THS layout

The thermal control parameters are provided with the SM2-PWS description in Table 6-3. When the Thermal Control software receives TC for updating the subsystem, it will communicate with the local Thermal Controller (through CAN) that will manage the physical changes on the setup. That will allow to control the opening of the valves, as well as the action of the fan.

The Thermal Management component polls the TM data of the Thermal Control software via RMAP. They are then published to the thermal Visualization component on the MMC through PolyORB-HI protocol, in order to be displayed on the GUI.



Preliminary Design Document

6.4.6 SM5/6-OSP

The SM5/6-OSP are optical payload modules. The purpose of the Camera Payload is to provides images of the surroundings of the spacecraft up to the MCC. In MOSAR, it is used for demonstration purpose. The physical layouts of the SM5/6 are shown in Figure 6-16. The internal equipment of the modules is composed of:

- The R-ICU for the local SM control, data routing and interface with the camera (see section Figure 6-7)
- The Zed Stereo Camera to support space observation or for proprioceptive visualisation of the spacecraft.
- The cPDU for the power distribution and local conversion
- 1 active HOTDOCK and other passive/mechanical versions (accommodated to support the foreseen demonstration scenarios)

The module can be manipulated by the WM. It is initially attached on the SVC at the beginning of the scenario.

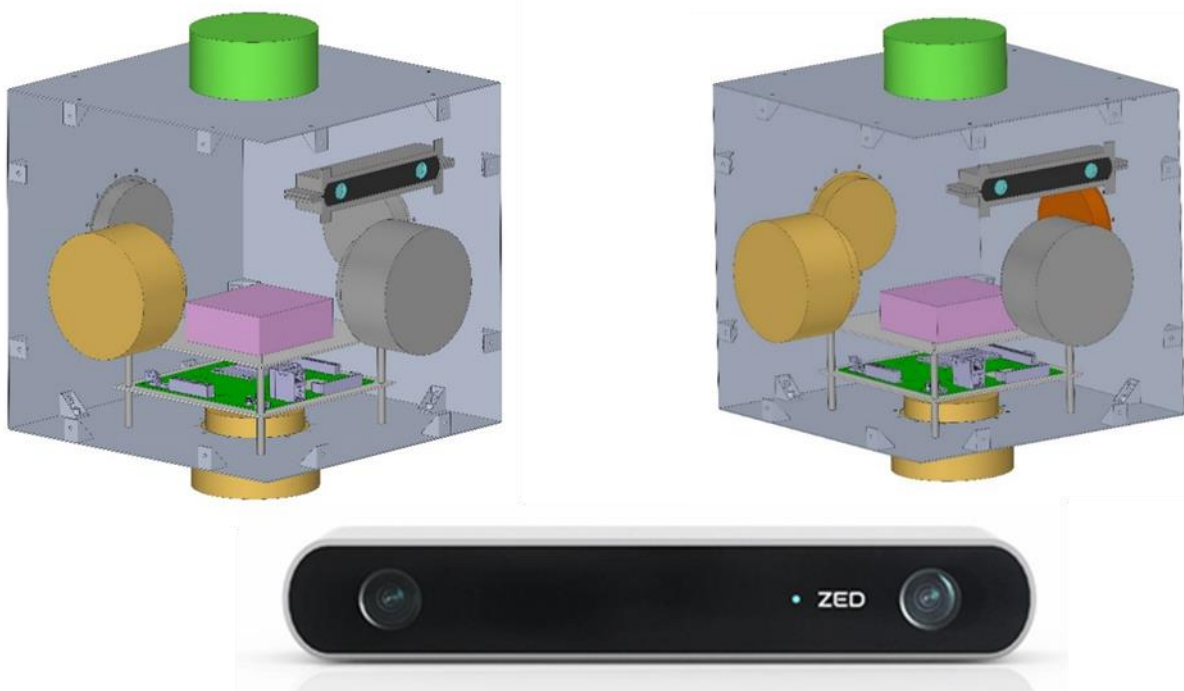


Figure 6-16: SM5/6 optical payload and the stereo ZED camera

Table 6-5 provides the software components, TM/TC and data related to the Camera Payload operations. The software components are defined according to the generic components used in Figure 3-2. Section 5.6 describes the software architecture and relation between them.

All TM/TC and data transfer, between the OBC and R-ICU, are implemented with the RMAP protocol. When the OSP Control software is commanded to *Operational* mode, it starts acquiring images from the camera using the driver interface of the Camera. The images are written to memory accessible through RMAP.

The OSP Management component polls the *Frame Available* status of the OSP Control software via RMAP. When a frame is available, it commands the OSP Relay to read the image through RMAP and



Preliminary Design Document

publishes it using the zeroMQ protocol to the OSP Visualization Component. This last receives each frame (ZeroMQ subscriber) and displays it in the GUI of the Visualization software on the MCC.

Table 6-5: Camera payload parameters

Camera Payload Operations	Context: Spacecraft nominal operations
Software Components	
<ul style="list-style-type: none">• <u>Payload Data Visualization (MCC)</u>: OSP Visualization• <u>Payload Management (OBC-C)</u>: OSP Management• <u>Payload Relay (OBC-C)</u>: OSP Relay• <u>Payload Control (R-ICU)</u>: OSP Control• <u>Payload Controller (Component)</u>: Internal Camera Controller (COTS)	
TM	
<ul style="list-style-type: none">• OSP Control / Mode Status (Idle, Operational, Error)• OSP Control / Frame available	
TC	
<ul style="list-style-type: none">• OSP Control / Mode Command (Idle, Operational)	
Data	
<ul style="list-style-type: none">• Image Frame	



6.5 Walking Manipulator

6.5.1 Preliminary Design

The MOSAR's walking manipulator aims at installing/releasing satellite modules and relocating itself over the satellite structures. The walking manipulator makes use of its end-effectors, equipped with HOTDOCK standard interfaces, to achieve its actions. Figure 6-17 illustrates an overview of the walking manipulator when moving a satellite module, as well as a representation of the preliminary design and components integration.

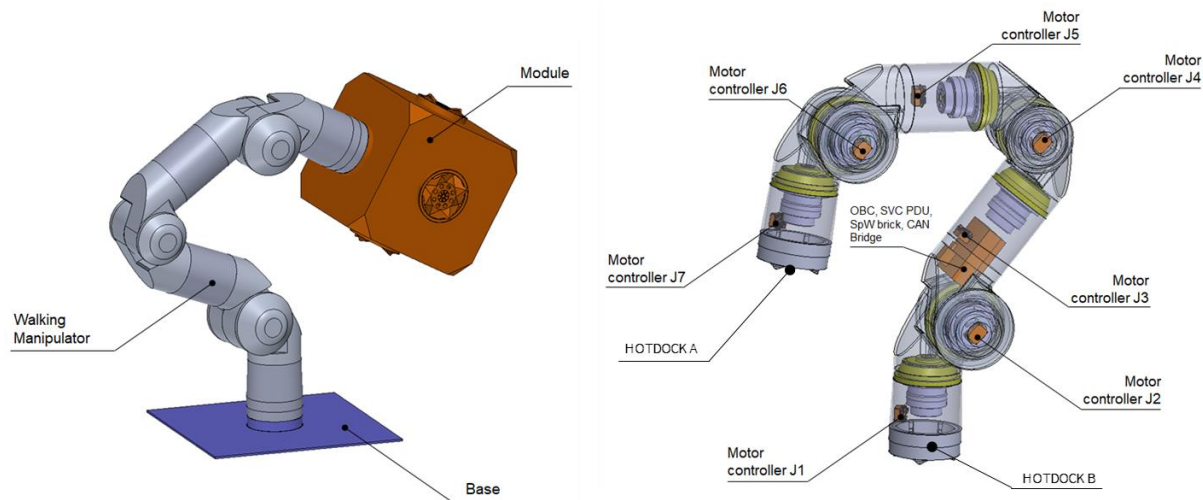


Figure 6-17: MOSAR Walking Manipulator

The walking manipulator kinematic is based on a human like structure with asymmetric joints, in order to optimize dexterity, range of motion and compactness. It presents a symmetric design between the base (side A) and the end-effector (side B) to allow self-relocation and walking, independently of the connected side.

The arm features seven active revolute joints based on brushless DC actuator and harmonic drive reducer. Each joint is equipped with position (incremental and absolute) and torque sensors to support the different control mode of operations, as well as an active brake.

Figure 6-8 illustrates the WM avionics architecture. The WM OBC Controller is the central component, playing the equivalent role as the R-ICU in the spacecraft modules (see Figure 3-2). It manages the control of the arm, of the two end-effectors HOTDOCK and the internal cPDU. As for the SM, the WM OBC interfaces the SpW bus (through USB/SpW brick) for TM/TC with the spacecraft OBC-S, through the RMAP protocol.

Each joint of the WM is equipped with a local controller for the closed loop position/current control of the actuator and the measurements of the joint sensors. All joint controllers are interfaced through an EtherCAT bus, managed by the WM OBC, which ensures the real-time exchange of information required for the control algorithm of the arm at 1 kHz.

The cPDU enables the control of the power transmission between the two HOTDOCK end-effectors, while providing the required DC/DC conversions for the powering of the internal components of the arm. The HOTDOCK and the cPDU are managed by the WM OBC, through a CAN control bus.



Preliminary Design Document

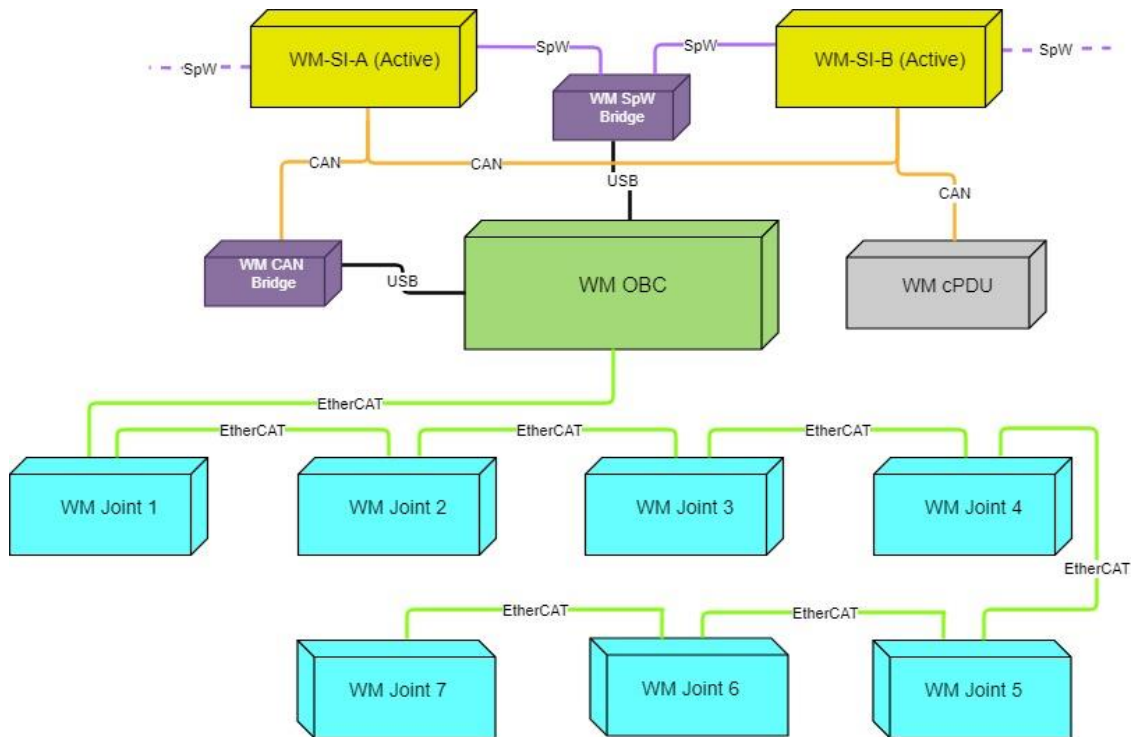


Figure 6-18: Walking Manipulator avionics / data architecture

6.5.2 WM Operations in MOSAR

The first two steps of Routine 1 (see 5.6.2.1) and Routine 2 (see 5.6.2.2) are related to pure robot motion commands. The WM should move from an initial pose to a new pose, which is in close proximity and alignment to a desired SI of an SM, SVC or CLT, so that the interface mechanism can be activated for mating. These two elementary robot actions are fully managed locally by the WM controller, based on a high level command sent by the OBC-S planner agent:

1. move into the alignment pose
 - a. goal pose in Cartesian coordinates relative to the target object, i.e. objects not in contact with each other, but in an appropriate alignment w.r.t. the connection pose
 - b. This action phase can be supported by a vision sensor, if the accuracy of the WM positioning is not sufficient for mounting, or if a form fit guidance of the SM interfaces is not available.
 - c. collision-free path from the current pose to this alignment pose
2. move into connection pose
 - a. guarded Cartesian motion, e.g. move in z direction if alignment with xy-plane, until contact detected
 - b. sensor-based insertion phase, like “peg-in-hole”, motion dependent on the SM interface (guidance available or built-in compliance of the SM)
3. keep position
 - a. keep current position from last command goal with sensor based adaptation to support remaining motion during the SI connection process (impedance control mode).



Preliminary Design Document

- b. If no adaptation is required the manipulator can be configured in position control mode, with force sensing monitoring.

Steps 8 and 9 of Routine 2 are also relying on these two elementary robot actions. Even though a SM is connected to the WM for assembly, the robot routines are following the same strategy. The SI of the “grasped” SM is moved to the SI target of the SM or SVC.

An important change with respect to the original ERGO component from OG2 is that the communication with the manipulator controller is interfaced at a different level. In OG2, the equivalent OBC-S manager commanded the arm controller at high rate, while in MOSAR the WM management component provides a coarse, collision-free trajectory to the WM Controller, and it is the latter the one that interpolates the trajectory and commands the actuators at a high rate. This avoids latency that might be introduced by the communication across several SpaceWire links.

Table 6-6 provides the software components and TM/TC related to the WM operations (at the WM Controller level). The software components are defined according to the components used in Figure 3-2. Section 5.6 describes the software architecture and relation between them.

Table 6-6: WM operations parameters

WM Operations	Context: Spacecraft reconfigurations
Software Components	
<ul style="list-style-type: none">• <u>MCC</u>: M&C Software (including GUI interface of the WM)• <u>OBC-C</u>: WM Management• <u>WM Controller</u>: WM Control• <u>Local Controller (Component)</u>: Joints Controller	
TM	
<ul style="list-style-type: none">• WM Control / Mode Status (Idle, Operational, Error)• WM Control / Base Reference (connected extremity reference, A or B)• WM Control / Control Mode Status• WM Control / Joint Position [7]• EM Control / Joint Velocity [7]• WM Control / Joint Motor Current [7]• WM Control / Joint Torque [7]• WM Control / Joint Temperature [7]• WM Control / Joint Brake Status [7]• WM Control / Extremity A Cartesian Position• WM Control / Extremity B Cartesian Position• WM Control / Extremity A Cartesian Force (computed)• WM Control / Extremity B Cartesian Force (computed)	
TC	
<ul style="list-style-type: none">• WM Control / Mode Command (Idle, Operational)• WM Control / Control Mode Command (position, impedance) – including control paramters• WM Control / Move Cartesian (trajectory) – (coarse collision free motion)• WM Control / Move Joint (trajectory) – (coarse collision free motion)• WM Control / Approach (trajectory) –(fine approach with impedance / sensor based control)• WM Control / Brake Command [7]• WM Control / Emergency Stop	
Data	
<ul style="list-style-type: none">• N/A	



6.6 HOTDOCK Standard Robotic Interface

6.6.1 Status

The HOTDOCK standard robotic interface is used in MOSAR to interconnect the different components of the MOSAR demonstrator that are the spacecraft modules, the walking manipulator and the spacecraft buses.

Figure 6-19 shows the current state of the HOTDOCK prototype. It provides the following interfaces:

- The mechanical interface that provides the alignment, connection and load transfer capabilities. It is composed of a fixed structure and movable elements on its circumference to perform the mating.
- The power interface, for the transfer of electrical power, through the central connector plate and POGO connectors
- The data interface, for the transfer of CAN and/or SpW data, through the central connector plate and POGO connectors

The interface features also an internal PCB controller for local management (actuators, sensors, TM/TC communication) and external harnessing to access the power/data interface pins and the internal controller/powering of the device.

In MOSAR, HOTDOCK is also feature a thermal interface for heat transfer management between two SM. This feature, based on the connectors implemented in OG5, are not implemented in the current prototype.



Figure 6-19: HOTDOCK prototype

The table below defines the features of the different envisaged declinations of HOTDOCKs in the context of their deployment in MOSAR:

- **Active**: full feature interface with active locking mechanism, and supporting mechanical, data and power transmission,
- **Active Thermal**: active version including thermal interface for fluid transfer,



Preliminary Design Document

- **Passive**: interface without actuation but providing capabilities to transfer mechanical, data and power. An Active interface is required to lock to a passive one,
- **Mechanical**: interface without actuation and without data/power transfer (connector plate). It supports only mechanical transmission. An active interface is required to lock to a mechanical one,
- **Dummy/Visual**: visual representation of a HOTDOCK (e.g. 3D printed of the external mechanical structure), not supporting mechanical transmission.
- **Fixed**: pass-through connection between two static components

Table 6-7: Supported feature per HOTDOCK declination

	Name	Visual	Mating	Mechanical Transmission	Data Transmission	Power Transmission	Thermal Transmission
●	Active	✓	✓	✓	✓	✓	
●	Active Thermal	✓	✓	✓	✓	✓	✓
●	Passive	✓		✓	✓	✓	
●	Mechanical	✓		✓			
●	Dummy/Visual	✓					
●	Fixed						

More information about the HOTDOCK design and interface definitions are provided in the HOTDOCK Preliminary Design Definition File [RD2].

6.6.2 HOTDOCK Control in MOSAR

Table 6-8 provides the software components and TM/TC related to the HOTDOCK operations. The software components are defined according to the components used in Figure 3-2. HOTDOCK control is specifically used during the reconfiguration phase.

The control operations of HOTDOCK mainly consists to request state transition in order to perform the sequence to mate and connect two SI. Figure 6-20 provides the sequence and definition of the control states. In MOSAR, this will be triggered typically by the R-ICU HOTDOCK control components, based on the feedback of the telemetry, through CAN commands to the local HOTDOCK controller. The R-ICU receives the commands from the OBC-S and the planner, using the RMAP protocol on the SpW bus (see section 5.6).

The RMAP protocol is also used for TM/TC with the HOTDOCK Management component and TTC service, enabling monitoring and control from the MCC through the PolyORB-HI protocol.

Table 6-8: HOTDOCK parameters

HOTDOCK Operations	Context: Spacecraft reconfiguration
Software Components	
<ul style="list-style-type: none">• <u>MCC</u>: M&C Software (including GUI interface of the SM)• <u>OBC-C</u>: HOTDOCK Management	



Preliminary Design Document

<ul style="list-style-type: none">• <u>R-ICU / WM Controller</u>: HOTDOCK Control• <u>Payload Controller (Component)</u>: HOTDOCK Controller
TM
<ul style="list-style-type: none">• HOTDOCK Control / Temperature motor• HOTDOCK Control / Temperature MCU• HOTDOCK Control / Temperature connector plate• HOTDOCK Control / Proximity sensor [4] (on/off)• HOTDOCK Control / Locking ring status (open/closed)• HOTDOCK Control / absolute position sensor• HOTDOCK Control / motor current• HOTDOCK Control / motor speed• HOTDOCK Control / HOTDOCK state• HOTDOCK Control / absolute position sensor• HOTDOCK Control / Error state
TC
<ul style="list-style-type: none">• HOTDOCK Control / SI State Update• HOTDOCK Control / Emergency stop• HOTDOCK Control / Led pattern (TBC)
Data
<ul style="list-style-type: none">• N/A

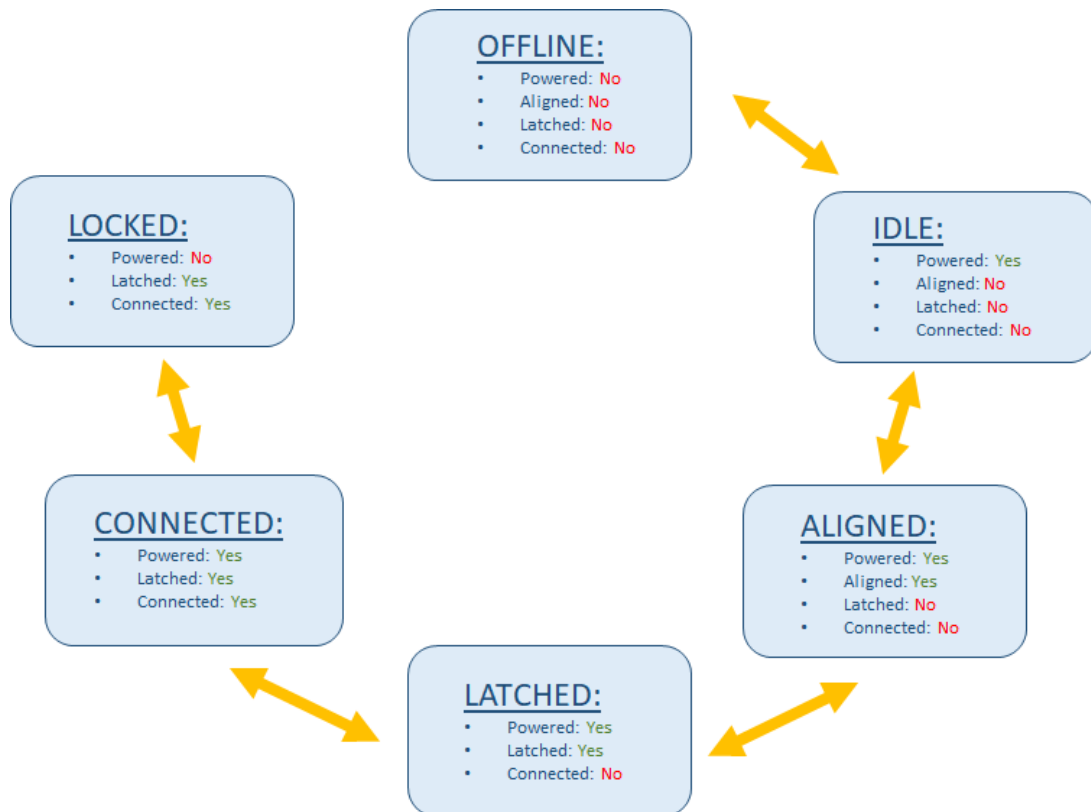


Figure 6-20: HOTDOCK control states



6.7 R-ICU

6.7.1 Status

The evolution of the I3DS system has started through procurement of four of the Trenz TE0808-04-9BE21-AS development starter kits. These provide a development board for the UltraScale+ MPSoC SOM containing the Xilinx ZU9EG and a FM connector for our bespoke interface card to be fitted (arrangement shown in Figure 6-21). The boards are later removed from the starter kits and integrated into the SMs.

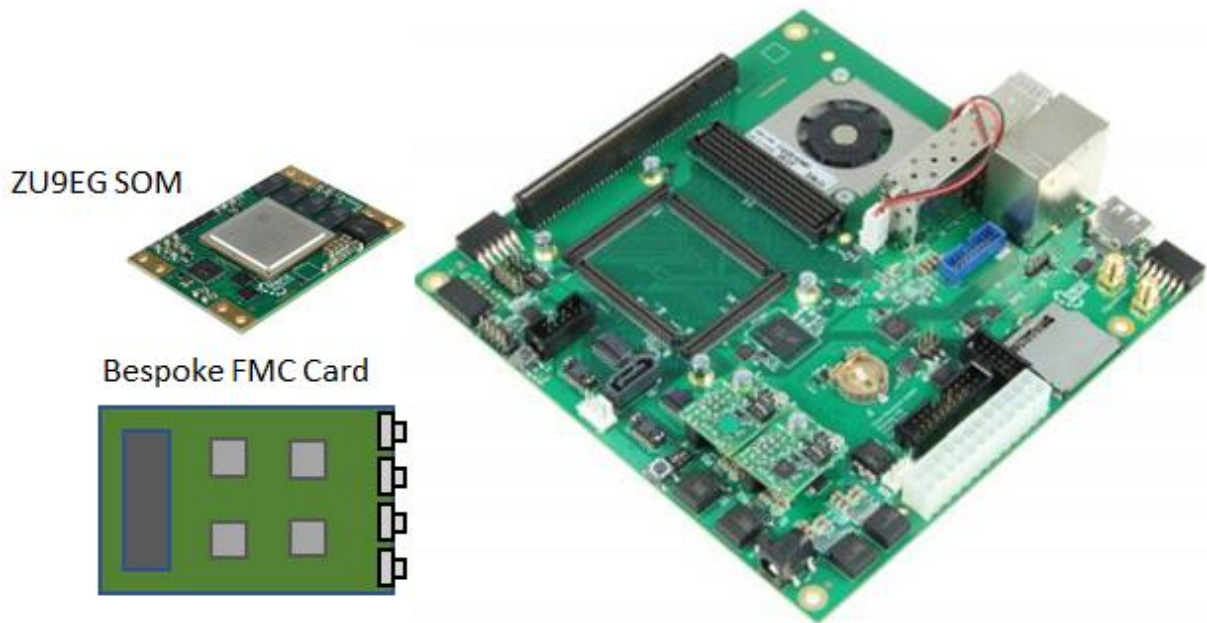


Figure 6-21 Components of the R-ICU

The development board contains the Ethernet and USB interfaces required for connection to the identified cameras (Zed Stereo Camera, F) for the optical payload. Support for the CAN and SpaceWire interfaces required for MOSAR will be provided by the FMC daughter board which will be connected directly to the development board. The hardware for the SpaceWire interfaces are to be implemented within the programmable logic of the ZU9EG.



Figure 6-22 Zed Stereo Camera for the optical payload

6.7.2 FMC Board Design

The design of the FMC board has been completed subject to final approval of the IO interfaces. The board provides 8 SpaceWire interfaces, 2 CAN interfaces and 16 IO connections consisting of: 8x 3.3V Open Collector Outputs and 8x 3.3V LVTTTL Inputs. All interfaces are exposed via pins for connection to the mechanical interfaces at SM integration. The interface is described in the ICD document [AD5].



The SpaceWire throughput requirement for the FMC board is at least 50Mbps (as identified in [AD5] and specified at SRR). During pre-integration testing the data rates achieved between R-ICUs via the Hotdocks will be tested at 50, 100 and 200 Mbps. A decision at what rate to run the network at will then be made as a result of these tests. None of the planned demonstrations will be functionally impacted by the SpW data rate. Reliable operation by using a comfortable margin is preferred for the demonstrators.

6.7.3 SpaceWire Router IP

The SpaceWire router will be implemented in software to allow a faster route to implementation and ensure that the deadline for the R-ICU delivery is met. A software router will also allow a more flexible implementation should supporting the dynamically reconfigurable network require functionality beyond the SpaceWire standard (e.g. network discovery, knowledge of connected device capabilities).

The ZU9EG has 6 processing cores, four of these are high power application cores (ARM Cortex-A53) that will be used to run the I3DS framework. The other two cores (ARM Cortex-R5) are of lower performance and capability but have access to the same memory as the application cores and also have interfaces to the programmable logic. The SpaceWire router will be implemented using one of the R5 processors, connecting to the 8 interfaces in the programmable logic via the AXI interconnect, the general design is shown in Figure 6-23.

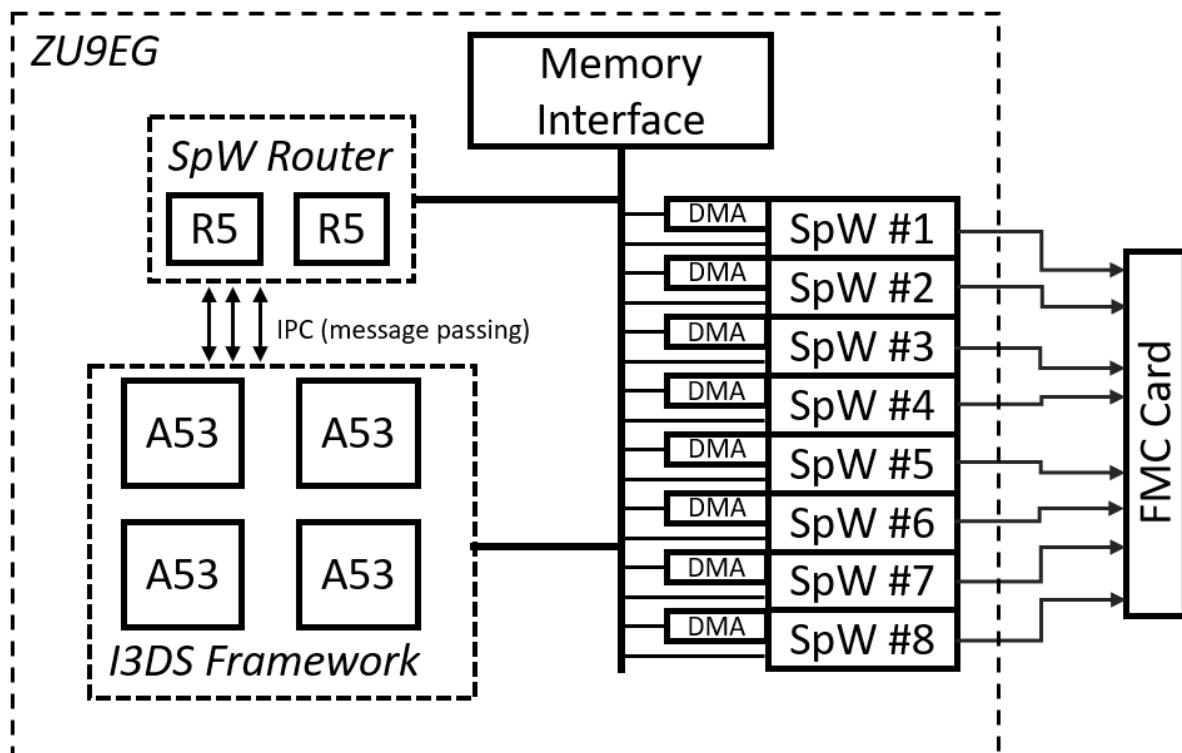


Figure 6-23 Block design of the SpaceWire router hardware

The R5 will configure the SpaceWire interfaces (start up, shut down, set speed) and set up the SpaceWire DMA descriptors. When data arrives at a SpaceWire interface, these DMA descriptors are used to determine where to put the data. The router functionality will be implemented through software management of these descriptors. This puts a restriction on the maximum size of a single SpaceWire packet which will need to be defined. The I3DS framework will need an extension for SpaceWire support, this will handle splitting of large packets via a protocol on top of SpaceWire or natively through segmentation of the I3DS data.



Preliminary Design Document

A simulation model of the SpaceWire router (4 interfaces connected to a MicroBlaze soft-processor) has been created, this will allow the software router to be developed and tested whilst waiting for the delivery of the Trenz starter kits.

6.7.4 R-ICU Operations in MOSAR

Table 6-9 provides the software components and TM/TC related to the specific R-ICU operations (this is not including the components control, described in the other respective sections). The software components are defined according to the generic components used in Figure 3-2. Section 5.6 describes the software architecture and relation between them

Table 6-9: R-ICU operations parameters

R-ICU Operations	Context: Spacecraft reconfigurations Spacecraft nominal operations
Software Components	
<ul style="list-style-type: none">• <u>MCC</u>: M&C Software (including GUI interface of the R-ICU)• <u>OBC-C</u>: Data Management• <u>R-ICU Controller</u>: Data Control / SpW Router• <u>Local Controller (Component)</u>: N/A	
TM	
<ul style="list-style-type: none">• Data Control / R-ICU status (reference, status of operations, connected nodes, GPIO, Error)	
TC	
<ul style="list-style-type: none">• Data Control / Configure R-ICU (configuration parameters)• Data Control / Set GPIOs (GPIO configuration)• SpW Router / Configure Data Routing (routing table)	
Data	
<ul style="list-style-type: none">• N/A	



6.8 SVC/CLT Satellites Mockups

Two satellite buses are simulated in the frame of MOSAR:

- The servicer satellite bus (SVC)
- The client satellite bus (CLT).

The structures of the two buses will be simulated by means of two test benches provided with the required HOTDOCK interfaces and service electronics. The exact number and positioning of the different HOTDOCKs variations, for each test bench, will be further analyzed and then reflected in the proposed preliminary design.

Table 6-7 recalls the list of the different versions of HOTDOCK used in MOSAR with the associated features. This same colour scheme will be used in this section to describe the interfaces implemented on the busses.

6.8.1 Test benches general design concept

Two separate benches have been designed in MOSAR in order to simulate the SVC and CLT buses (Figure 6-24 Spacecraft Bus Overview). The benches will provide the required interfaces with the SM as well as with the WM and they will be featured with the specific functionalities in order to support operational scenarios.

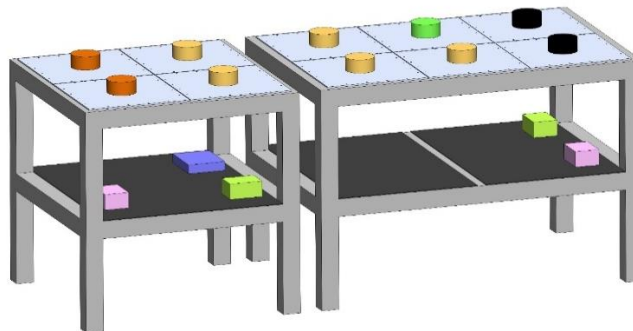


Figure 6-24 Spacecraft Bus Overview

Parameters used as drivers for preliminary design of the benches are simplicity of the parts for manufacturing, flexibility during integration of the benches and versatility with respect to recovering actions, due to any issue on a bench item during validation campaign.

The mechanical parts composing the bench assembly are:

- frame structure,
- interface panels,
- shelves for internal electronics
- structural cleats,

as shown in Figure 6-25.

Each interface plate is independently integrated on the bus frame so that they can be dismounted in case of problems with minor impact on the entire system. Similarly, the shelves for the internal electronics can be dismounted from the frame structure during the integration phase or the validation campaign, in case of inspection and repairing is required. The envelope of the bench assembly containing the SVC and the CLT has the following dimensions: 2600mm (L) x 1000mm (W) x 1000mm (H).

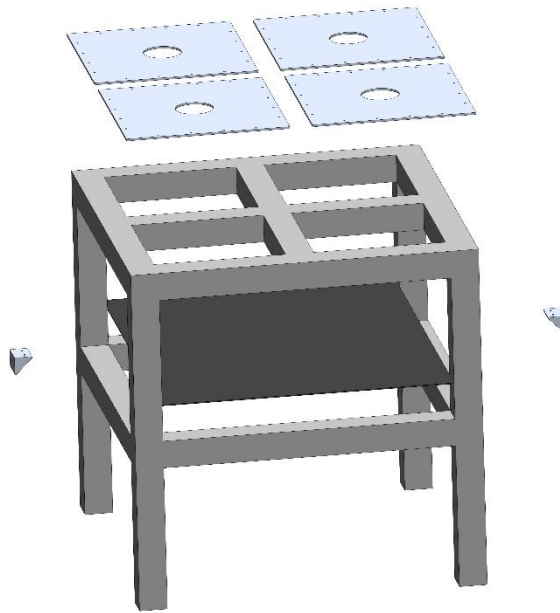


Figure 6-25: SVC structure exploded view

6.8.2 SVC/CLT Architecture

Figure 6-26 represents the SVC/CLT data architecture, as a more specific view from the Figure 3-2.

The Servicer Spacecraft (SVC) has the role to bring the new spacecraft modules and the WM up to the client satellite, and, after the docking operation, to manage the reconfiguration operations, based on an operation plan prepared and sent by the MCC. For this purpose, the SVC bus includes the following elements:

- The OBC-C that manages the reconfiguration operations both on the SVC and CLT, by controlling the WM motion, HOTDOCK interfaces and the power and data routing components.
- An R-ICU to control the local HOTDOCK standard interfaces.
- A set of HOTDOCK standard interfaces to carry the SM during the travel of the SVC, as well as for connecting the WM to manipulate them.
- A local cPDU to provide power to the SVC components and control the power transfer of the SVC bus HOTDOCK interfaces. The cPDU is directly connected to the EGSE to represent the own power of the servicer (or potentially the power provided by the CLT, through the docking interface)

The SVC has a connection with the MCC to represent the data link, mainly used during the reconfiguration operations.

The Client Satellite bus (CLT) is initially equipped with a Data Management Module (SM1-DMS) that includes the OBC-C, and a power module (SM2-PWS). On top of that, the CLT features:

- Its local R-ICU for the routing of the SpW bus to the SMs, as well as the control of the local HOTDOCK interfaces of the CLT bus.
- Its local cPDU for the management of the power transfer of HOTDOCK interfaces.



Preliminary Design Document

- A set of HOTDOCK to allow the connections of the SM as well as the walking of the WM along its structure (allowing to reach the SM)

As for the SVC, the CLT is directly connected with the MCC to represent the datalink, mainly used to monitor and control spacecraft operations (after the reconfiguration).

The SVC and CLT are directly connected by SpW to represent the data link provided by the docking interface.

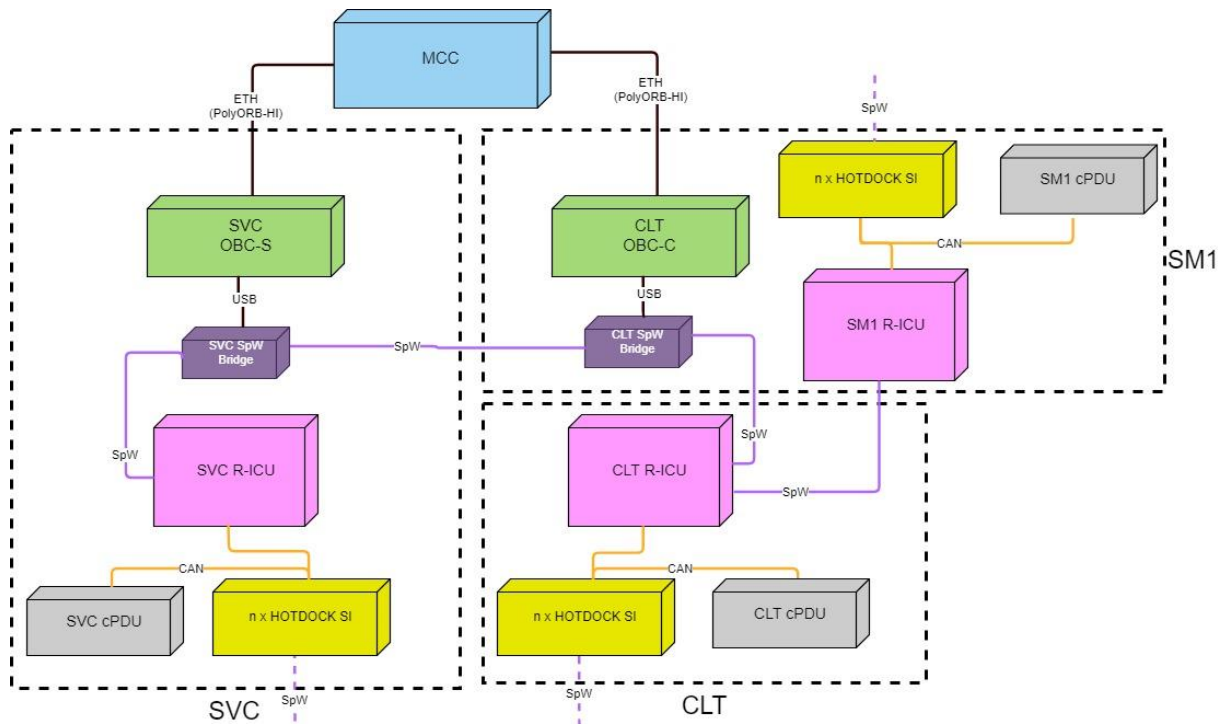


Figure 6-26: SVC/CLT data architecture



6.9 Visual Processing System

This section describes the high level architecture of the vision processing subsystem. Firstly, the system functional requirements are revisited according to new priorities emerged during the design of the demonstrator. Secondly, the physical organization of the subsystem is presented in relation to the other physical components of the demonstrator. Finally, an overview of the software architecture is given.

6.9.1 Update of the Functional Requirements

As documented in AD4, the original objective of the vision processing system is the surface reconstruction of the satellite and the pose estimation of the walking manipulator from images collected by a camera mounted on the walking manipulator itself (or on a spacecraft module).

This objective poses several engineering challenges:

- (i) a proper location needs to be designed on the walking manipulator to host the camera,
- (ii) close proximity to the surface would make pose estimation and reconstruction challenging and very likely to fail, as position reference may be easily lost,
- (iii) image quality may be poor unless constraints on the speed of the walking manipulator are defined
- (iv) multiple cameras might be needed and synchronized as a single camera might be occluded at times,
- (v) the walking manipulator would need an image-transfer interface to the vision processing hardware.

The initial purpose of the vision processing subsystem in the context of the MOSAR demonstrator was to use it as part of the operation control for supporting alignment of the components and mating of the SI. However, from initial analysis during the preliminary design activities and internal discussions in the Consortium and EC/PSA, it has been decided to not exploit this approach in the closed loop operations, taking into account:

- That the current demonstrator design and operations (misalignment tolerances of the SI, SM structural/weight optimization, sequence of operations) would allow a successful demonstration without the need of the external sub-system
- The added complexity in terms of hardware and software to include this feature in the closed-loop operations

We think however that this technology has a high interest for the future mission based on spacecraft modularity, regarding initial inspection, operations tracking or assembly validation, on top of the re-use, demonstration and further improvement of the activities of OG3-Infuse from the first call. In the support of autonomous operations, this is very relevant for detection of unexpected configuration, spacecraft deterioration and discrepancies from the anticipated models. External sensing of the MOSAR demonstrator is useful for analysis of the demonstration scenarios and also has mission value as a means by which an external spacecraft can inspect the structure of a modular satellite. It has been then decided to re-align the objective of this sub-system with a setup that is external to the main MOSAR demonstrator but allowing at the same time to demonstrate interesting feature for future missions.

The preferred solution foresees that the vision processing system will be completely external to the walking manipulator and the satellite mock-up. A camera located at a fixed location will observe the entirety of the demonstrator, and will transmit images to a computing platform dedicated to vision processing. Results of the processing will be visible (ideally in real-time) on a dedicated monitor connected to the platform.

Table 6-10 lists the requirements in order of priority, it explains possible approaches and challenges.



Preliminary Design Document

Table 6-10: InFuse objectives in MOSAR

Requirement	Description	Approaches	Challenges
<p>Validation of spacecraft module shape</p> <p>Rationale: Shape might change due impacts or heating processes (during transport or space exposure)</p>	<p>Spacecraft modules will be detected within the scene.</p> <p>Size information will be extracted and compared to the expected shape.</p>	<p>It is possible to extract dimension directly from the image</p> <p>It is possible to extract information from the point cloud.</p> <p>Defect could be visually highlighted.</p>	<p>Illumination and reflectance as well as camera parameters will have an impact on the quality of the extracted information.</p> <p>We need to define a measure of acceptable defect / non-acceptable defect.</p>
<p>Validation of spacecraft module interface</p> <p>Rationale: Interface might be damaged by impacts or heating processes (during transport or space exposure)</p>	<p>Spacecraft modules will be detected.</p> <p>Interface profile is compared to the model profile.</p>	<p>It is very unlikely that the dimensions can be extracted directly from the image due to the small size and uniform colour.</p> <p>It should be possible to extract information from the point cloud.</p>	<p>Illumination and reflectance are still problems.</p> <p>Define the smallest detectable defect.</p> <p>Define the acceptable defect / non-acceptable defect threshold.</p>
<p>Validation of module re-configuration</p> <p>Rationale: Modules might still be in the correct location, but they might be slightly misaligned due to interface damage</p> <p>This might allow WM replanning.</p>	<p>Spacecraft modules and position will be detected.</p> <p>Module structure is compared with respect to the expected structure</p>	<p>Camera need to be placed in a good position in order to view as many modules as possible.</p> <p>We need a structure verification algorithm.</p> <p>Tracking is good for speed and real-time.</p>	<p>Position of the modules need to be detected in real time for online validation.</p> <p>We might want to use FPGA or GPU to speed up detection/tracking</p>
<p>Selective Zooming</p> <p>Rationale: This can offer increased accuracy in the detection of anomalies in small field of views</p>	<p>The camera might allow digital configuration of parameters such as aperture and focal length.</p>	<p>The camera intelligently adapts to the situation. For example, by focusing on an interface or reducing the illumination absorption rate in case of high illumination.</p>	<p>Control the camera.</p> <p>Definition of the model for selection of the optimal parameters.</p> <p>Different area of interests might required different 3D reconstruction parameters.</p>
<p>Fixed Adaptive Reconstruction</p>	<p>As the camera parameters change, we have different views of the scene.</p>	<p>A reconstruction algorithms for different camera views, like bundle adjustment, and fusion of 3D surfaces.</p>	<p>Speed is a concern, this process cannot be executed in real time if we plan to take too many views.</p>



Preliminary Design Document

Rationale: This can offer increased accuracy in the detection of anomalies in the available field of view.	<p>Different parts of the scene are reconstructed with different levels of details.</p> <p>These details should be combined in one unique scene with a requested amount of details and minimum amount of noise.</p> <p>It could be a building block for a reconstruction from cameras located on the walking manipulator.</p>		<p>What is the optimal thread-off between processing speed and number of views?</p> <p>Handling a dynamic scene is challenging, as different views will be taken at different point in time. Known algorithms for dynamic scenes will probably not work.</p>
<p>Validation of walking manipulator shape / validation of walking manipulator interfaces</p> <p>Rationale: The walking manipulator might be damaged (during transport or space exposure)</p>	<p>In case the walking manipulator remains in orbit for a long time, it may be subject to damage.</p> <p>Same specifications as the validation of spacecraft modules and interfaces.</p>	<p>Same approach as the validation of the spacecraft modules and interfaces.</p>	<p>Same challenges as the validation of the spacecraft modules and interfaces.</p> <p>The walking manipulator shape is more complex.</p>

6.9.2 Physical Organization of the Vision Subsystem

The vision processing subsystem is composed of a vision processing platform, and a fixed camera. Figure 6-27 displays the set-up of the vision processing system.

The processing platform is a standard computer with dedicated memory and processors, and input/output peripherals such a monitor, a keyboard, and a mouse. The processing platform will host a Ubuntu Operating System, and will execute the vision processing algorithms.

The camera will be put on a stand at fixed pre-determined location at the side of the satellite mock. The camera field of view will cover the upper surface of the satellite, and non-occluded walking manipulator and spacecraft modules will be visible. The camera will have its own power source and will be directly connected to the PC with an USB or Ethernet cable for data transfer.

Space illumination conditions will be simulated by setting the demonstrator in a dark room with a black background. Room lighting conditions will approximate those that could be experienced in space, in a similar fashion to DLR's OOS-SIM simulator. A dark backdrop will be placed behind the demonstrator to absorb light and minimize features detected by the vision system. Two adjustable servicer LED lamps will be located at the side of the camera and will illuminate the visible side of the demonstrator. In a first scenario, we will assume that there is no sunlight, and that the earth albedo is simulated by room light. In a second scenario a third lamp will be positioned on a movable stand and will simulate the sunlight. We will evaluate the vision processing results under different light intensities of the two servicer lamps and different directions of the sunlight lamp. This will allow the performance of the vision system to be evaluated in realistic re-configuration scenarios.

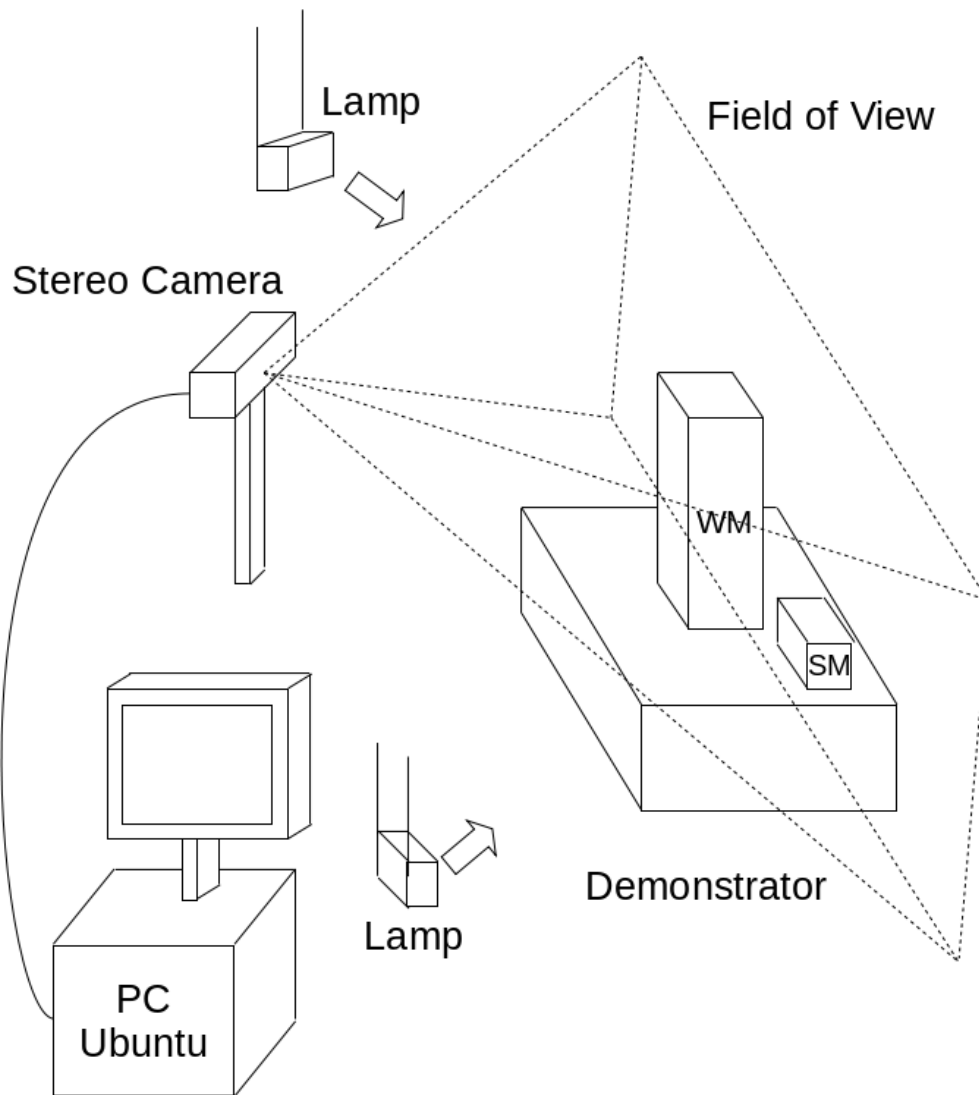


Figure 6-27: Hardware configuration of vision subsystem

6.9.3 Software Organization of the Vision Subsystem

The vision processing algorithms will run on the Ubuntu Operating system hosted on the processing platform.

The Operating System will contain all the third party software libraries necessary to the correct execution of the vision processing algorithms (such as OpenCV, and Point Cloud Library). Alongside these libraries, InFuse will be available and will contain the core vision processing algorithms.

The final processing software will be composed of three components: (i) an image acquisition system that will receive the images from the camera and convert them into a format useful to processing; (ii) the functional algorithms for the actual image processing and computation of poses and 3D structures; (iii) an output display system that will show results on the monitor. Figure 6-28 shows the software architecture.

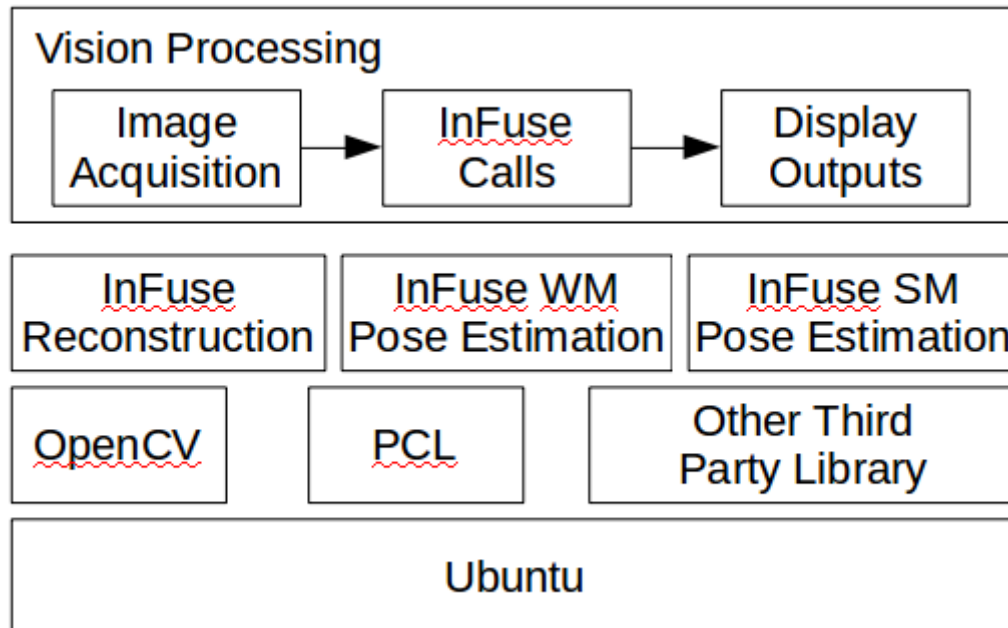


Figure 6-28: Software architecture of the vision subsystem

In addition to the software, a configuration file will be available for setting computational properties.



6.10 MCC and Demonstration Setup

This section presents the anticipated setup and layout for the MOSAR Monitoring and Control Center (MCC) and the space made available for the “flight segment” demonstrator.

Figure 6-29 presents a top view layout of the foreseen space for the MOSAR MCC + demo setup (not showing the Visual Processing subsystem, see section 6.9.2).

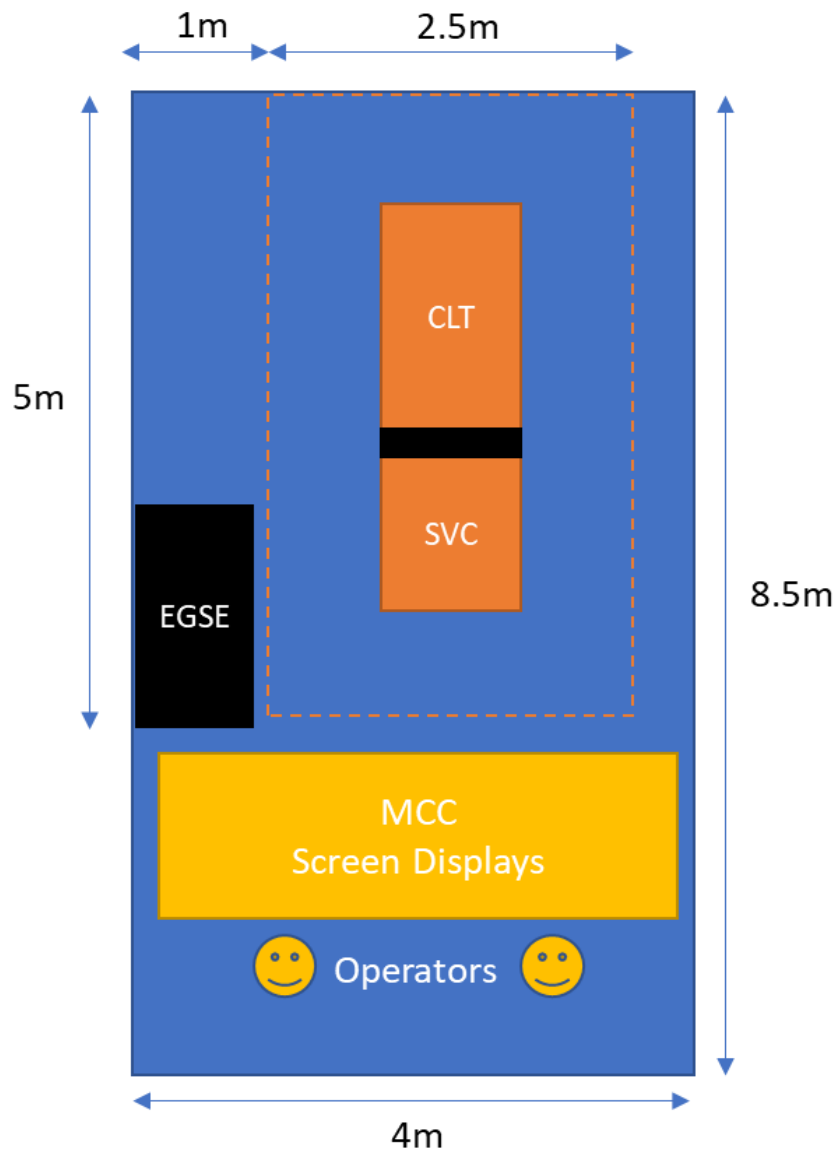


Figure 6-29: MCC and demonstration setup space - notional layout (numerical dimensions are correct but elements geometry and size are only indicative)

The MCC will rely on existing infrastructures including a screen wall (3 x 55" curved UHD Samsung), a large desk and a series of touch screen (that may be used as regular screens if touch is not required).

The picture below is the MCC as currently available.



Figure 6-30: Monitoring & Control Center (MCC) facing the demonstration setup space

Any regular computers may be connected in the MCC – however to make use of multiple screens, the computer(s) to be used should feature suitable graphic cards with multiple HDMI outputs. Alternatively, the MCC may be used with a subset of screens, if the available graphic card(s) do not allow using all the available screens (or if it is simply not deemed necessary).

The MCC setup will be composed of three computers:

- The Design and Simulation PC, able to support the preparation of the spacecraft design (text based + visualization tool) and the simulation of the proposed design, either to validate it or in collaboration with the Planner Agent to find and validate the reconfiguration plan
- The Planner PC that is running the Planner Agent to find valid sequence of operations to reconfigure the spacecraft
- The Monitoring, Control and visualization PC that is responsible to communicate with the two spacecraft in order to upload the validated plan, to enable telecommands to the systems components and payloads, and to get telemetry from them and display them to the operator .

The visual processing system could be interfaced with the visualization PC.

All computers and the spacecraft OBCs will be interconnected by a central Ethernet switch, to support the communication between the software components:

- Simulator / Planner interaction communication
- MCC to Service data link
- MCC to Client data link



Preliminary Design Document

The MCC setup provides also an EGSE to power the components of the system that includes:

- The ground segment equipment's (computer, screens, Ethernet switches)
- The spacecraft OBCs (SVC and CLT/SM1-DMS)
- The main CLT power bus (28V, as input in the CLT/SM2-PWS)
- The Visual Processing System components (lights, cameras,...)

In order to materialize a separation between the ground segment (MCC) and the space segment (demonstration setup) side, a mate black curtain will be installed in between. Additionally, a mate black curtain will be installed on the windows side, on the back side, and on the most exposed side (on the right). The ceiling will also be covered with mate black materials (panels or painting, tbc). A partially representative lighting system additionally will be installed (ARRI Daylight spotlight tbc. – trade-offs ongoing).

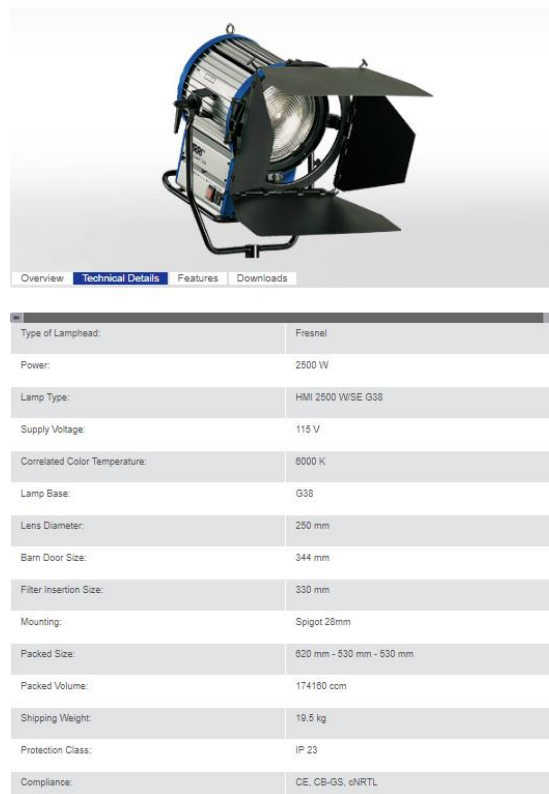


Figure 6-31: Technical sheet of an ARRI Fresnel Daylight Compact 2500 (possible lighting solution)

Figure 6-32 gives an impression of the space available for the flight segment demonstrator.

The MCC and flight segment demonstrator areas will be prepared so that to be ready from the CDR milestone, so that perception tests can start being carried out even in the absence of the demonstration setup.

For what concerns safety, a combination of different approaches will be implemented:

- Hard emergency stops – one on the MCC side, and one on the flight segment demonstrator side.
- Soft emergency stops (at SW level, possibly relying on force sensing or other sensors sources)



Preliminary Design Document

- Operation signal (light)
- Security bands with a sufficient security distance
- Depending on more detailed analysis on risks in case of collision (which may be limited due to anticipated low speed of the manipulator), rigid protection around the setup or a fence may be envisaged.

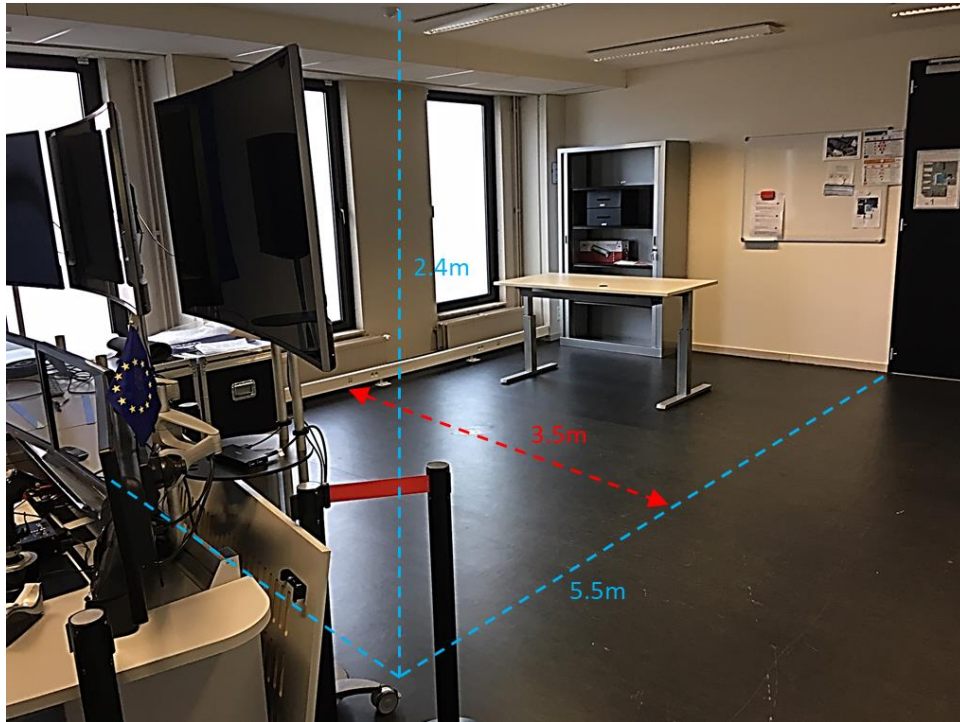


Figure 6-32: Space for demonstration setup



7 Annex

7.1 MOSAR Setup Position References

This following figures provides the position reference definition of the HOTDOCK standard interfaces for the different components of the MOSAR setup that are the satellites mockups, SVC and CLT, and the spacecraft modules. Spacecraft modules faces are defined relatively to the initial MOSAR setup configuration.

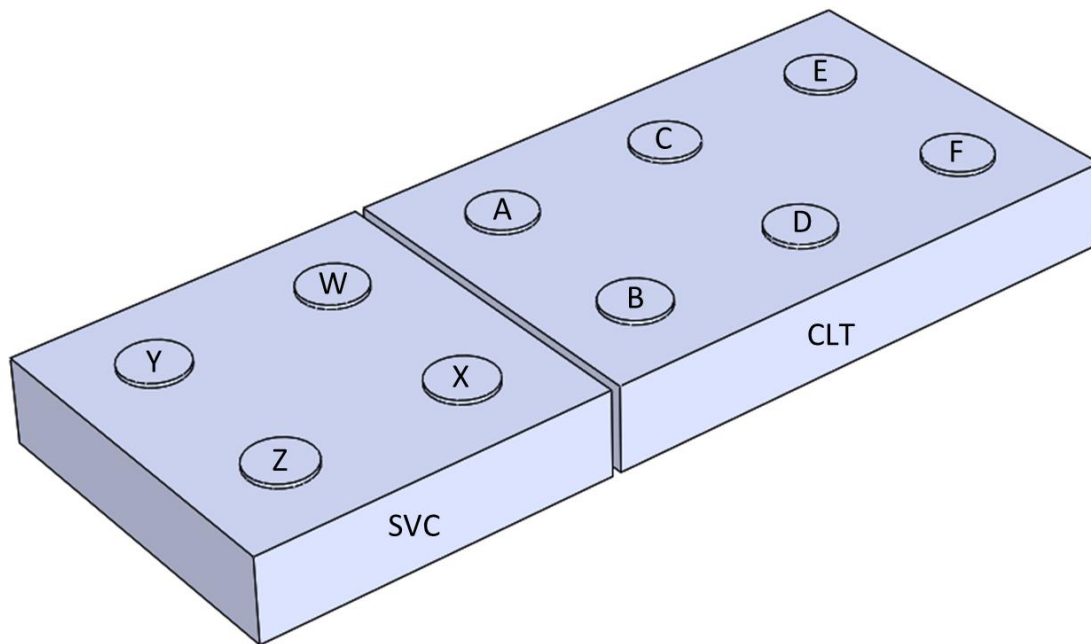


Figure 7-1: References of the HOTDOCK standard interfaces spot on SVC and CLT

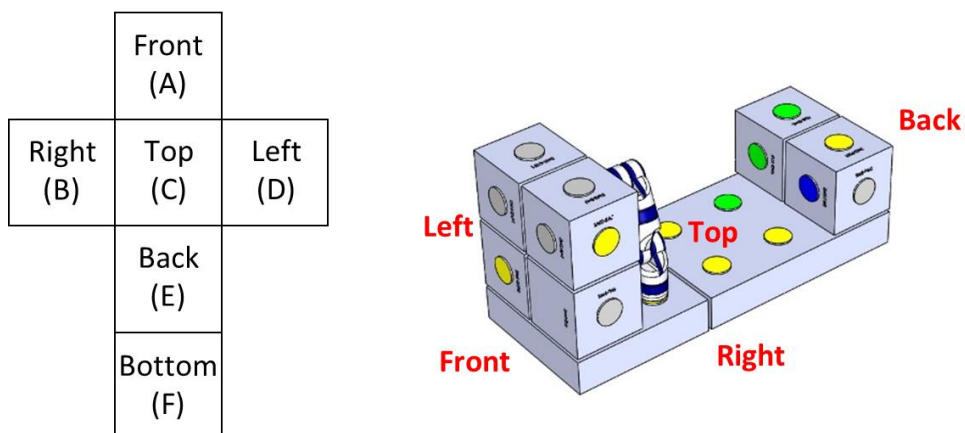


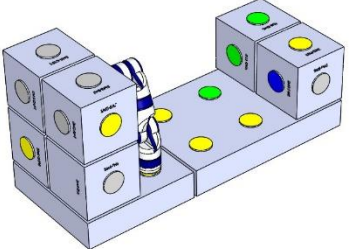
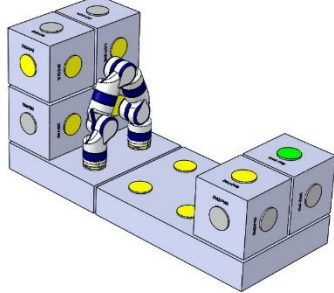
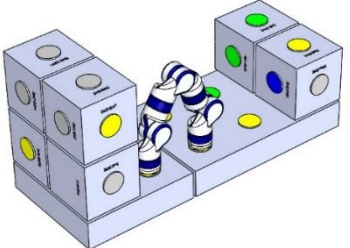
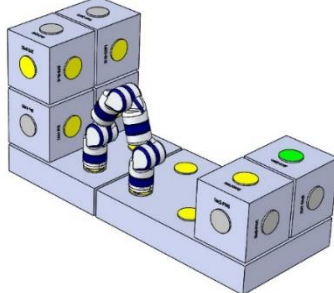
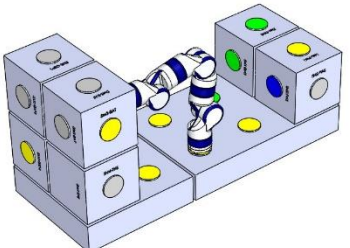
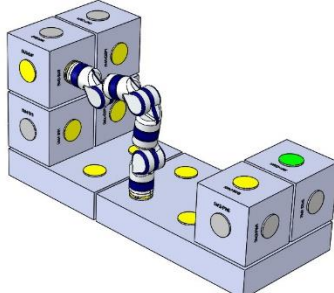
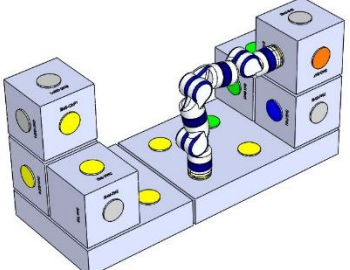
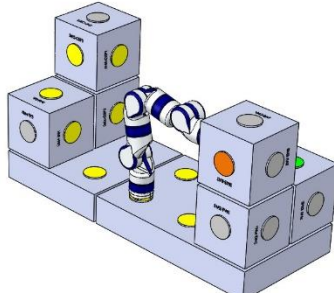
Figure 7-2: References of the spacecraft module faces (in the initial MOSAR setup configuration)



7.2 MOSAR Sequence of Manipulations Example

The following table provides a possible step-by-step sequence of operations covering the scenarios 1 and 2

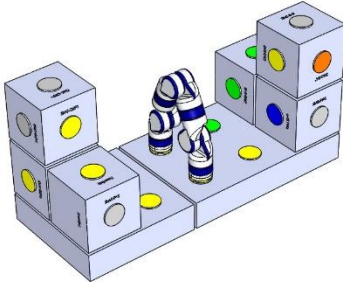
Table 7-1: Scenarios 1 and 2 step-by-step sequence of operations

<p>Step 0: Initial Position</p> 	<p>Step 0: Initial Position</p> 
<p>Step 1: WM-A to CLT-B</p> 	<p>Step 1: WM-A to CLT-B</p> 
<p>Step 2: WM-B to SM3-BAT</p> 	<p>Step 2: WM-B to SM3-BAT</p> 
<p>Step 3: SM3-BAT to SM2-PWS</p> 	<p>Step 3: SM3-BAT to SM2-PWS</p> 

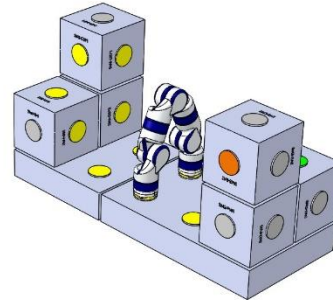


Preliminary Design Document

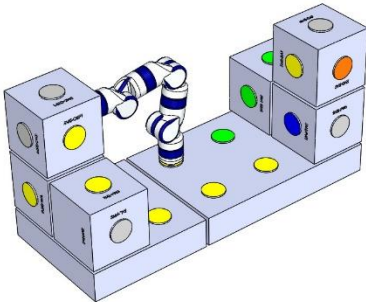
Step 4: WM-B to CLT-A



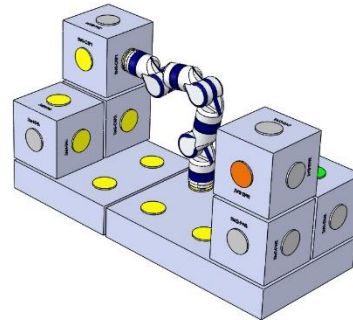
Step 4: WM-B to CLT-A



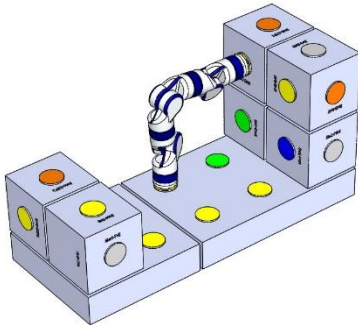
Step 5: WM-A to SM5-OSP1



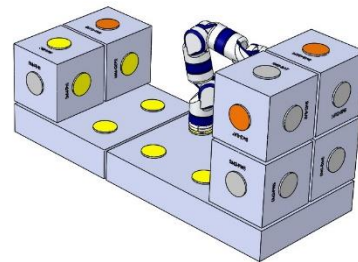
Step 5: WM-A to SM5-OSP1



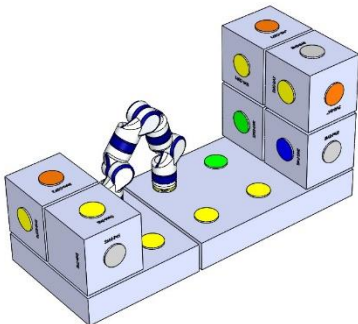
Step 6: SM5-OSP1 to SM1-DMS and SM3-BAT



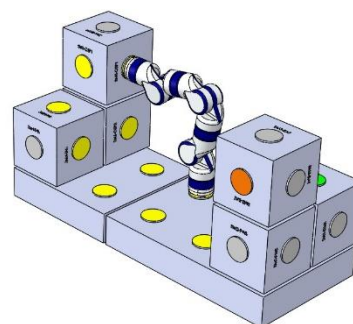
Step 6: SM5-OSP1 to SM1-DMS and SM3-BAT



Step 7: WM-A to SVC-W

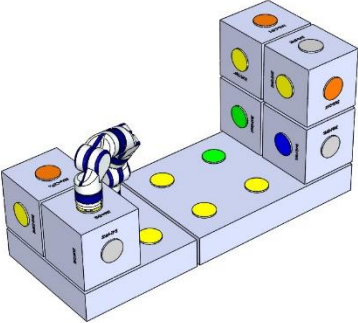
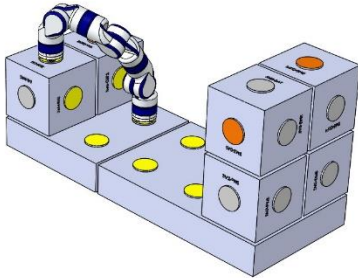
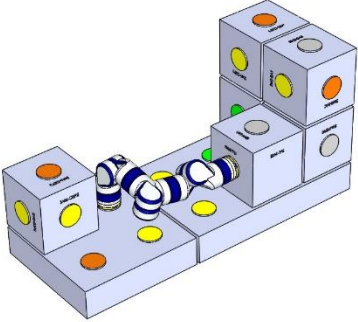
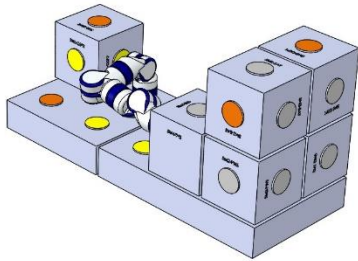
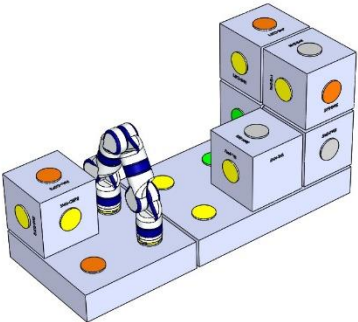
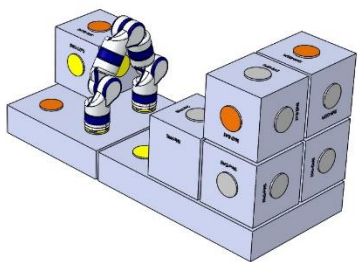
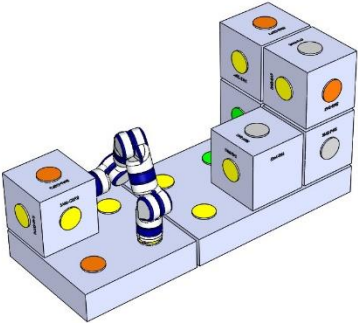
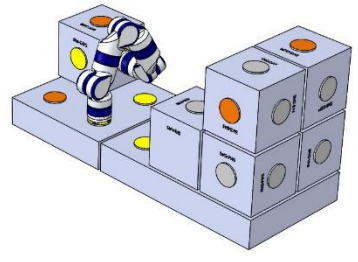


Step 7: WM-A to SVC-W



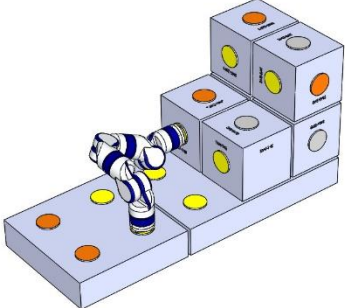
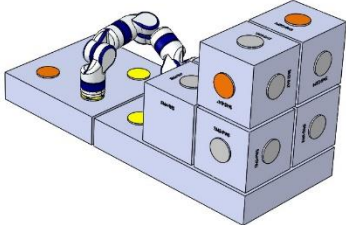
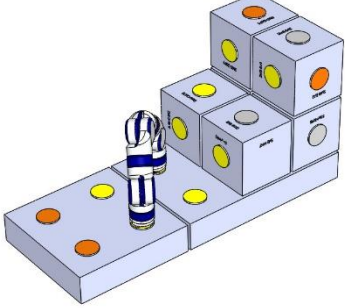
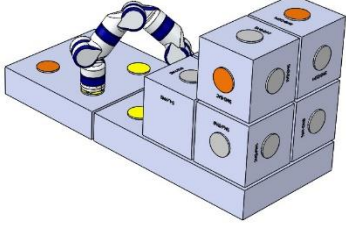
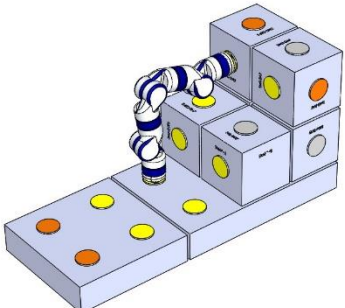
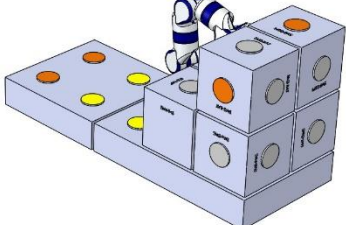
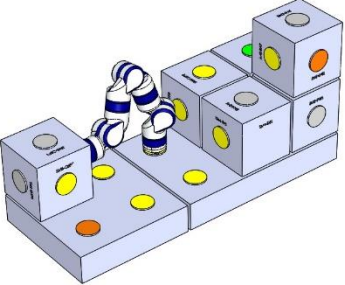
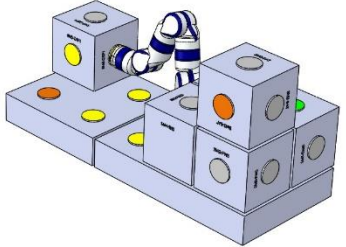


Preliminary Design Document

<p>Step 8: WM-B to SM4-THS</p> 	<p>Step 8: WM-B to SM4-THS</p> 
<p>Step 9: SM4-THS to SM2-PWS</p> 	<p>Step 9: SM4-THS to SM2-PWS</p> 
<p>Step 10: WM-B to SVC-X</p> 	<p>Step 10: WM-B to SVC-X</p> 
<p>Step 11: WM-A to SM6-OSP2</p> 	<p>Step 11: WM-A to SM6-OSP2</p> 



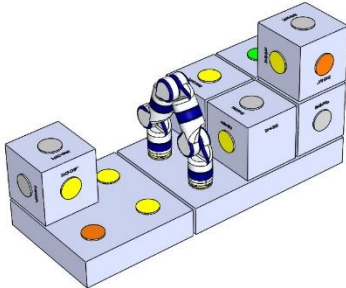
Preliminary Design Document

<p>Step 12: SM6-OSP2 to CLT-E, SM1-DMS, SM4-THS</p> 	<p>Step 12: SM6-OSP2 to CLT-E, SM1-DMS, SM4-THS</p> 
<p>Step 13: WM-A to CLT-A</p> 	<p>Step 13: WM-A to CLT-A</p> 
<p>Step 14: WM-B to SM5-OSP1</p> 	<p>Step 14: WM-B to SM5-OSP1</p> 
<p>Step 15: SM5-OSP1 to SVC-Y</p> 	<p>Step 15: SM5-OSP1 to SVC-Y</p> 

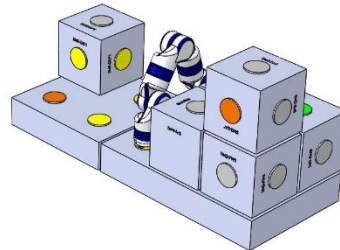


Preliminary Design Document

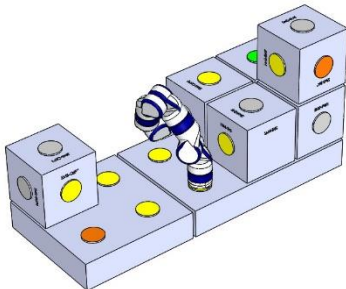
Step 16: WM-B to CLT-B



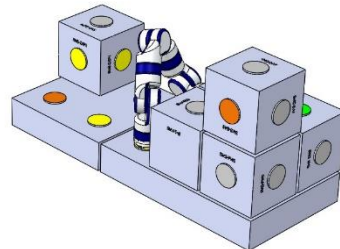
Step 16: WM-B to CLT-B



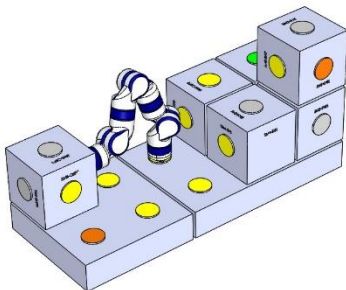
Step 17: WM-A to SM6-OSP2



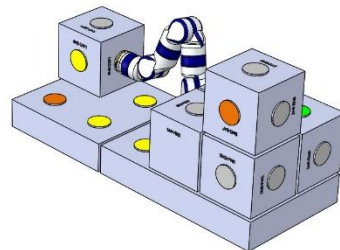
Step 17: WM-A to SM6-OSP2



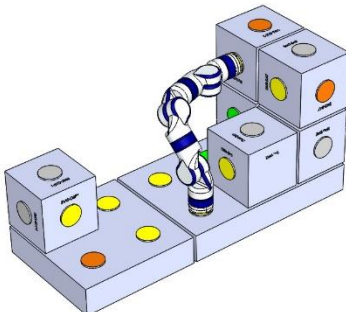
Step 14: WM-B to SM5-OSP1



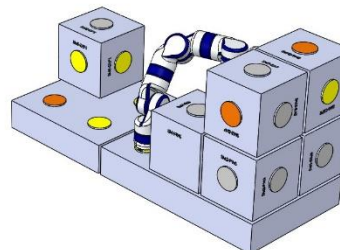
Step 14: WM-B to SM5-OSP1



Step 18: SM6-OSP2 to SM1-DMS and SM3-BAT



Step 18: SM6-OSP2 to SM1-DMS and SM3-BAT





7.3 Software Reconfiguration Documents References

- [1] Micro Python scripting language over SPWF04S, STMicroelectronics technical rapport, Nov 2017
- [2] VxWorks Kernel Programmer's Guide, 6.2, Wind River, Oct 2005
- [3] Rufino, J., Craveiro, J., Schoofs, T., Tatibana, C., & Windsor, J. (2009, May). AIR Technology: a step towards ARINC 653 in space. In *Proc. DASIA*.
- [4] Jorge Garrido, Juan Zamorano, and Juan A. de la Puente. 2015. ARINC-653 Inter-partition Communications and the Ravenscar Profile. *Ada Lett.* 35, 1 (December 2015), 38-45.
- [5] K. Zhang, J. Wu, C. Liu, S. S. Ali and J. Ren. Behavior Modeling on ARINC653 to Support the Temporal Verification of Conformed Application Design, in *IEEE Access*, vol. 7, pp. 23852-23863, 2019.
- [6] P. Hambarde, R. Varma and S. Jha. The Survey of Real Time Operating System: RTOS, 2014 International Conference on Electronic Systems, Signal Processing and Computing Technologies, Nagpur, 2014, pp. 34-39.
- [7] F. Nicodemos, O. Saotome and G. Lima. RTEMS Core Analysis for Space Applications, 2013 III Brazilian Symposium on Computing Systems Engineering, Niteroi, 2013, pp. 125-130.
- [8] Gallmeister, Bill. POSIX. 4 Programmers Guide: Programming for the real world. O'Reilly Media, Inc., 1995.
- [9] Stallings, W. Operating systems: internals and design principles. Boston: Prentice Hall, 2012.
- [10] Stankovic, J. A., & Rajkumar, R. Real-time operating systems. *Real-Time Systems*, 28(2-3), 237-253, 2004.
- [11] Parkes, S., & Mc Clements, C. Space wire remote memory access protocol. In *DASIA 2005-Data Systems in Aerospace* (Vol. 602), 2005.
- [12] Airlines Electronic Engineering Committee(AEEC), Avionics application software standard interface, ARINC Specification 653, Part 2 (ExtendedServices), Draft 5. ARINC, August 2006.
- [13] J. Penix, W. Visser, E. Engstrom, A. Larson and N. Weininger, "Verification of time partitioning in the DEOS scheduler kernel," *Proceedings of the 2000 International Conference on Software Engineering. ICSE 2000 the New Millennium*, Limerick, Ireland, 2000, pp. 488-497.
- [14] J. Sahoo, S. Mohapatra and R. Lath, "Virtualization: A Survey on Concepts, Taxonomy and Associated Security Issues," 2010 Second International Conference on Computer and Network Technology, Bangkok, 2010, pp. 222-226.
- [15] C. Huang and P. Hsiung, "Hardware Resource Virtualization for Dynamically Partially Reconfigurable Systems," in *IEEE Embedded Systems Letters*, vol. 1, no. 1, pp. 19-23, May 2009.
- [16] Abel Gordon, Nadav Amit, Nadav Har'El, Muli Ben-Yehuda, Alex Landau, Assaf Schuster, and Dan Tsafirir. 2012. ELI: bare-metal performance for I/O virtualization. In *Proceedings of the seventeenth international conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS XVII)*. ACM, New York, NY, USA, 411-422.
DOI=<http://dx.doi.org/10.1145/2150976.2151020>
- [17] K. Dang Pham, A. K. Jain, J. Cui, S. A. Fahmy and D. L. Maskell, "Microkernel hypervisor for a hybrid ARM-FPGA platform," 2013 IEEE 24th International Conference on Application-Specific Systems, Architectures and Processors, Washington, DC, 2013, pp. 219-226.
- [18] F Singhoff, J Legrand, L Nana, L Marcé. Cheddar: a flexible real time scheduling framework, *ACM SIGAda Ada Letters* 24 (4), pages 1-8
- [19] Silva, H., Sousa, J., Freitas, D., Faustino, S., Constantino, A., & Coutinho, M. (2009). RTEMS improvement-space qualification of RTEMS executive. 1st Simpósio de Informática-INFORUM, University of Lisbon.
- [20] Hildebrand, D. (1992, April). An Architectural Overview of QNX. In *USENIX Workshop on Microkernels and Other Kernel Architectures* (pp. 113-126).
- [21] Dietrich, S. T., & Walker, D. (2005, November). The evolution of real-time linux. In *7th RTL Workshop*.



End of Document
