MASCOT OBSW Development and Verification Facility – A cost effective approach

Johan Marx[(1)], Eduard Baumstark[(1)], Federico Cordero[(1)]

[(1)] Telespazio VEGA Deutschland GmbH
Europaplatz 5
D-64293 Darmstadt
Email: info@telespazio-vega.de

INTRODUCTION

This paper will summarize the solutions used to develop the Mobile Asteroid Surface Scout (MASCOT) Software Development and Verification Facility (SDVF). The SDVF was developed based on a cost effective approach in order to fit within the tight cost and schedule constraints of the MASCOT mission. This approach included the reuse of ESA infrastructure (SIMULUS, SCOS2000), and COTS elements (TSIM emulator, hardware interface cards, etc.).
The experience gathered throughout the lifecycle of the MASCOT project, where the SDVF had different use cases, enables us to present the advantages and disadvantages of the chosen approach.

MISSION

MASCOT was a small asteroid landing spacecraft developed by DLR in collaboration with CNES as contribution to JAXA's Hayabusa-2 mission. The lander contained four scientific instruments and had the capability to relocate itself on the asteroid surface. The first MASCOT concept was established around 2008 during the European Marco Polo mission assessment. After the discontinuation of the original study, MASCOT received an invitation from JAXA to join the Hayabusa-2 mission, the follow-up of the first asteroid sampler Hayabusa. However, MASCOT was selected at a time (mid 2011) when its design had not been fully defined, with the carrier spacecraft already in its critical design phase, its interfaces fixed and only 3 years left until a final delivery.
The tight schedule, the tightly defined physical envelope and mass, and strict margins policy were challenges during its development at all levels. Nevertheless, Hayabusa2 and MASCOT were launched on December 3rd, 2014, and arrived at their destined target asteroid (162173) Ryugu on June 27, 2018.
The successful landing on the asteroid and the science operations took place on 3rd of October 2018 [1][2][3].
Telespazio VEGA Deutschland's roles were the technical management of the OBC HW procurement, the development of the OBSW, as well as the SDVF procurement and development.

A SDVF DESIGN BASED ON REUSE

The ESA SIMULUS infrastructure was adopted for the implementation of the SDVF to support the development and validation of the MASCOT On-Board Computer (OBC) flight software. The SIMULUS components used in the SDVF are the run-time framework, Simulation Infrastructure for the Modeling of SATellites (SIMSAT) Kernel and Man Machine Interface (MMI), and a set of reusable generic models (GENM).
The ESA SCOS-2000, providing a generic mission control system software, was also adopted to have a Central Checkout System (CCS) to support the testing/validation during the MASCOT hardware integration and testing activities, enabling the SDVF to be used as an EGSE. On top of the reuse of ESA infrastructure, several COTS elements were used to lower the need for new development.

To support the On-Board Software (OBSW) development, the SDVF allowed the replacement of the OBC hardware with a processor instruction emulator (TSIM) and the replacement of the equipment/science instruments hardware with simulated models. This allowed to proceed with the OBSW development, debugging and preliminary testing in parallel with the actual MASCOT hardware development, including the OBC, de-risking the tight schedule driven development.
In summary, the SDVF main off the shelf components were the following:

- I/O interface cards (bricks and cards for Hardware-In-The-Loop testing)
- A virtual simulator (based on ESA SIMULUS infrastructure)
- A processor emulator compatible with the selected LEON chip (Gaisler Research TSIM)

- A Central Check-out System (ESA SCOS2000)
- An Integrated Development Environment to be used during the development and debugging of the OBSW

In the end, only the equipment/instrument model needed to be developed from scratch (based on GENM). All other components were either COTS or reused and only required specialized interfacing, configuration and integration. This allowed the SDVF development to keep up with the demanding schedule of the mission.

SDVF OVERALL SYSTEM ARCHITECTURE

In order to connect all hardware and software components, a star architecture around a central "Model Switch" was put in place. This means that any possible software/hardware configuration could be achieved by modifying the configuration files of the switch. The model data interfaces are based on packets and the Model Switch routes the packets from their source model to their destination model based on a routing table, defining, as a matter of fact, the overall SDVF configuration and use case. The connection points can be either internal functional interfaces for the virtual models or TCP/IP ports for the real hardware interfaces. The modular design allows for hot swapping of any software models with their equivalent hardware. This design also enables the integration testing and validation even through remote connectivity, which was used in a few occasions during the HW/SW integration campaign. The use of TCP/IP for connecting the hardware interfaces was possible due to the maximum latency allowed by the OBC communication protocol with the equipment/instruments (200ms for most interfaces, 100ms for a couple of them).
The virtual simulator and the Model Switch are fully contained in the SIMSAT simulation environment which allows the usage of its inherent functionalities as the state saving, the script engine, the service scheduler and the MMI.
The Fig. 1 shows the SIMSAT based SDVF functional block diagram as implemented for the MASCOT project.

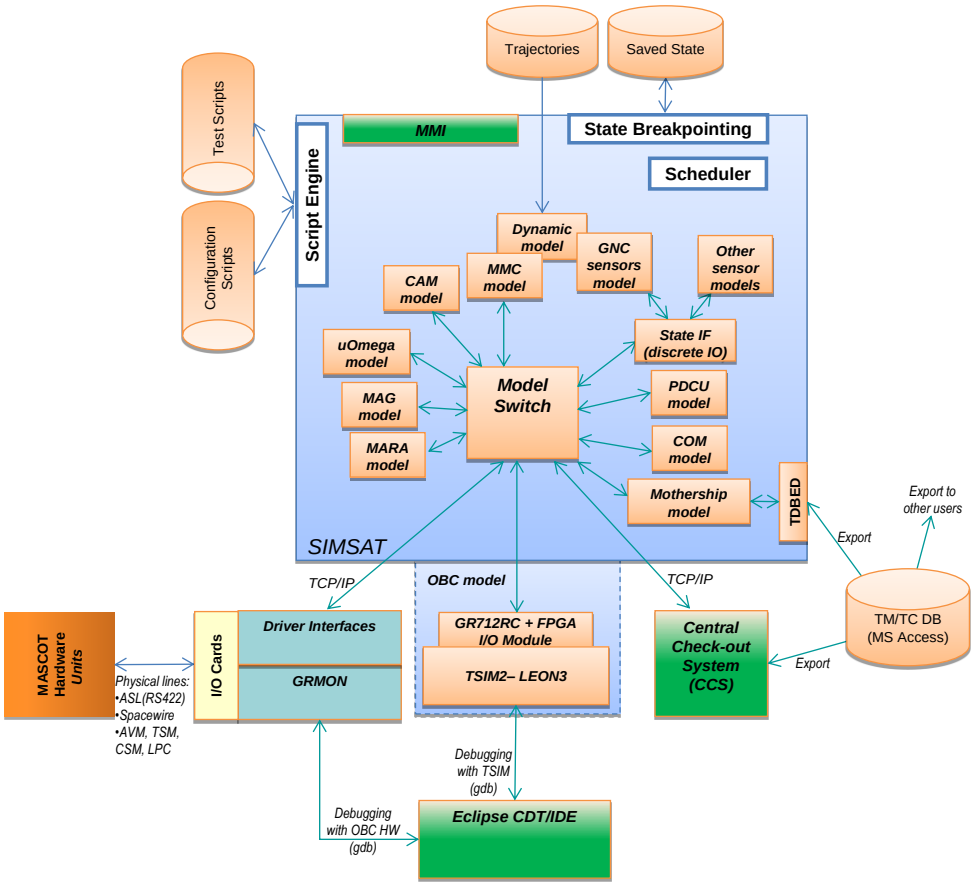

Fig. 1. SIMSAT based SDVF

The SDVF equipment/instrument virtual models are:

- LEON3-FT processor core emulator (TSIM)
- Specific MASCOT OBC IO module (Main and Redundant)
- Power Control and Distribution Unit (PCDU)
- Communication Module (COM)
- Guidance, Navigation and Control (GNC) Sensors
- Mobility Mechanism (MMC)
- Magnetometer (MAG)
- Radiometer (MARA)
- Wide Angle Camera (CAM)
- Infrared Microscope (uOmega)

In addition the SDVF includes a model of the Hayabusa-2 spacecraft to simulate the Communication Subsystem (COM) parent functionality, as well as a State Acquisition model responsible for generating the periodic Analogue Voltage Monitor (AVM), Temperature Sensor Monitor (TSM) and Contact Status Monitor (CSM) readings. A lander Dynamics model is driving the input/output for the GNC sensors and the mobility mechanism.

The SDVF was initially used in a pure virtual configuration as a development environment and for software unit integration and testing; subsequently, it is was used as a powerful tool for system test and validation of the flight software versions with the real hardware equipment.
The development cycle was such that an OBC flight software version was engineered, deployed on the processor emulator and debugged/preliminarily tested using the virtual models of the scientific instruments / equipment. The final validation step of the software version, before delivery to the customer, foresaw the processor emulator replaced by a real OBC hardware model (EM) and tested while still using the instrument/equipment virtual models. Once the flight software version was delivered, the customer could integrate and test it at system level on the lander under assembly using a SDVF instance configured as an EGSE, where all hardware virtual models could be replaced by the real instruments/equipment hardware units as they became available.
The development of the software could therefore start, continue and evolve in parallel to the development of the hardware until the final fully featured flight software product was delivered and integrated.

SDVF WITH HARDWARE-IN-THE-LOOP

Software components

As mentioned previously, the simulated/emulated components can be replaced by the corresponding hardware. However, additional interface drivers were needed for connecting the Model Switch with the hardware components. The interface programs are connecting to the Model Switch over TCP/IP and a corresponding configuration of the switch is necessary a priori to their start. The interfaces depend on the connection type and several of them were developed to provide all the connectivity needed by the OBC. The following interface programs were developed for the SDVF:

- Digital to Analog TCP/IP interface to allow communication with both analog and digital signals (DAC/DIO boxes)
- SpaceWire TCP/IP interface
- Asynchronous Serial Link (RS422 UART) TCP/IP interface

The Fig. 2 shows how the interfaces are integrated in the purely virtual SDVF to allow communication with the hardware.

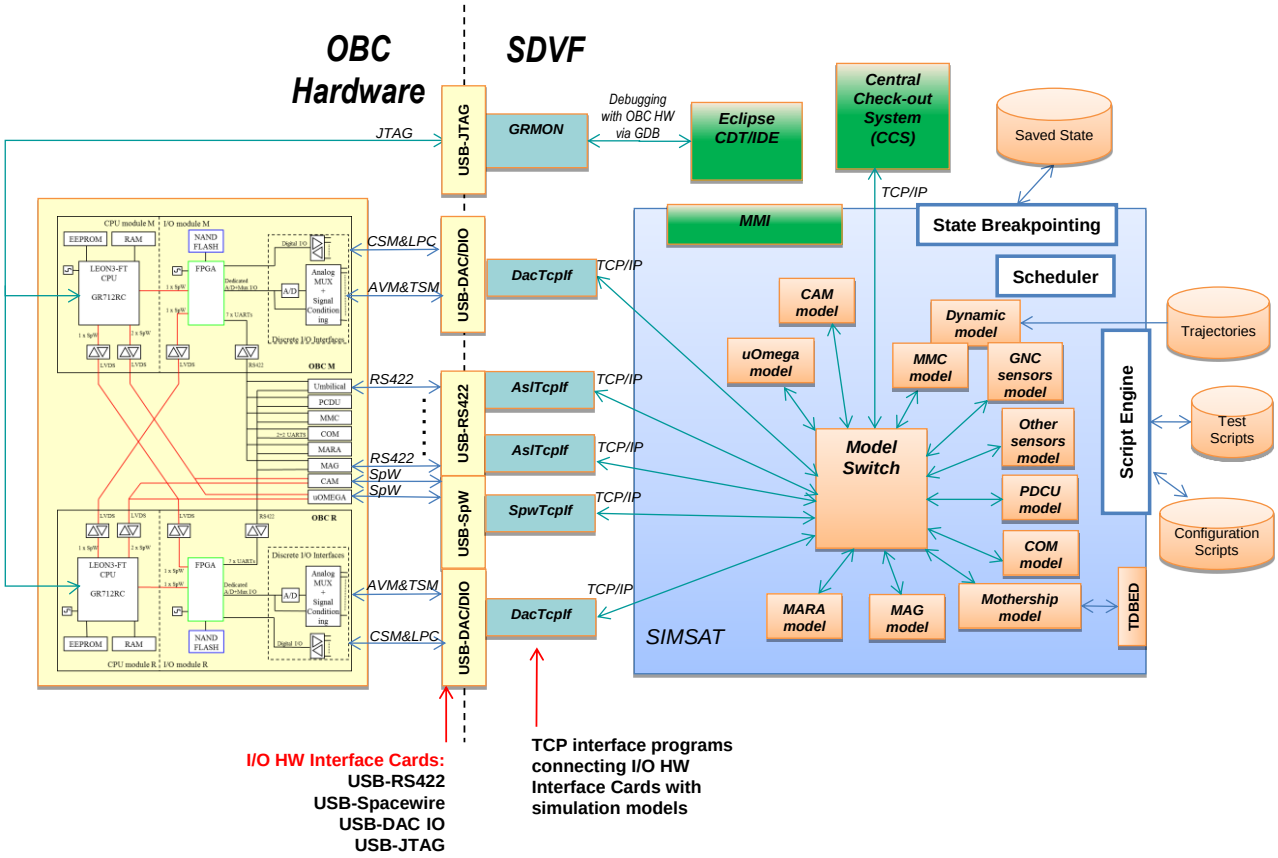

Fig. 2. SDVF connected with the Hardware OBC (Processor-In-the-Loop configuration)

Hardware components

In order to integrate the required hardware subsystems, several types of I/O interface bricks had to be procured and connected to the workstation hosting the SDVF. The communication with these bricks was done via the TCP/IP driver interfaces as mentioned in the previous section. The I/O bricks were attached through USB and offered access to RS422 Serial, SpaceWire and Digital and Analog I/O interfaces of the OBC as follows:

- 2 SpaceWire-USB Bricks used to connect the OBC hardware to the CAM and uOmega models
- 14 RS422 interfaces for the connection of the OBC hardware (main and redundant links) to the COM, PCDU, MMC, MARA and MAG models
- 2 USB DAC bricks for connecting the OBC hardware digital and analog I/O interfaces with the relevant models (GNC sensors, thermal sensors, separation sensor, etc.)

The Fig. 3 shows the complete SDVF with the Processor-In-the-Loop configuration.

Fig. 3. SDVF with OBC Hardware in the Loop and Interface Cards

CONCLUSION

The chosen solutions led to an original all-in-one design for supporting the OBSW development and use as EGSE for MASCOT AIT/AIV at DLR. Furthermore the SDVF subsequently evolved into an Operational Simulator to support the DLR Flight Control Team.
The SDVF's modular design not only ensured that the whole MASCOT project could be realized within the tight development schedule, but also provided a substantial cost reduction for the customer.
Our experience with this SDVF design in terms of advantages and disadvantages can be summarized as follows:

Advantages

1. Basing the SDVF on SIMULUS allowed us to keep up with the tight schedule and cost demands of the mission. The SDVF could be quickly developed and deployed with a low cost due to the reuse of the generic models and the SIMULUS infrastructure.
2. The integration of a debugging interface with the emulator in the SDVF (via GRMON for OBC hardware) allowed the OBSW developers to debug directly from their integrated development environment (Eclipse). This reduced the troubleshooting effort during the OBSW development and maintenance considerably.
3. Using the SIMULUS infrastructure allowed a quick and seamless evolution of the SDVF into an Operational Simulator to support the DLR Flight Control Team. The existing instruments / equipment models already contained a fully-fledged data handling interface and needed only to be extended with the corresponding functional models.
4. The modular design based on the Model Switch allowed for hot swapping of any software models with their equivalent hardware.
5. Interfacing the software models with the hardware through TCP/IP allowed the integration, testing and validation even through remote connectivity.
6. The SDVF was designed to keep the timing between all models as consistent as possible among all the models, to ensure suspension and resumption of the overall simulation at any point in time. This allowed the OBSW to be easily debugged using the Full Virtual SDVF configuration, allowing breakpointing in order to resume the OBSW execution at any line of the code.

Disadvantages

1. Interfacing the software models with the hardware through TCP/IP led to high latency in the communication. This was however still tolerated by the OBSW due to an OBC communication protocol with the equipment/instruments allowing for such latency by design. Extra effort was however necessary in order to increase the robustness of the hardware in the loop OBSW testing.

2. SIMULUS is not a real time simulation infrastructure. Mainly the non-timely and inconsistent interaction between the Script Engine and Simulation Scheduler, both SIMSAT components, caused some issues in the hardware in the loop OBSW validation. This also required extra effort in order to increase the testing robustness.

REFERENCES

[1] C. Grimm, J.-T. Grundmann, J. Hendrikse, C. Lange, C. Ziach, T.-M. Ho, From idea to flight - A review of the Mobile Asteroid Surface Scout (MASCOT) development and a comparison to historical fast-paced space programs, *Progress in Aerospace Sciences*, Vol.104 (2019) 20–39

[2] F. Cordero et al, MASCOT lander operational concept and its autonomy, general services and resource optimisation implementation in the on-board software, *SpaceOps 2016 Conference AIAA*

[3] S. Habinc et al, MASCOT On-Board Computer Based on GR712RC, *DASIA 2013*.