INTEGRATION AND USE OF SIMLEON3 EMULATOR IN SPACECRAFT OPERATIONAL SIMULATORS BASED ON SIMULUS

Juan-Rafael García-Blanco[(1)], Joan Jorquera-Grau[(1)], Carmelo Carrascosa-Sanz[(1)], Vemund Reggestad[(2)]

GMV[(1)]
Calle Isaac Newton 11,Tres Cantos (Madrid), 28760 Spain
E-mail: jrblanco@gmv.com, jjgrau@gmv.com, ccarrascosa@gmv.com

ESA / ESOC[(2)]
Robert-Bosch-Str. 5, D-64293 Darmstadt, Germany
E-mail: Vemund.Reggestad@esa.int

ABSTRACT

Spacecraft operations simulators are software facilities devoted to the validation of the operational flight procedures, the training of the operation team and the validation of the mission control system. Spacecraft operational simulators execute the same on-board software (OBSW) that is boarded in the actual spacecraft. For this reason they incorporate a processor emulator that allows the OBSW to interact with the external modules that compose the different spacecraft subsystems.

The LEON family of processors is of common use in space systems. Spanish SEOSAT and ESA Sentinel-5P missions use the AS250 platform for the Data Handling System (DHS), which incorporates two SCOC3 computers, including one LEON3 processor per computer. In the frame of SEOSAT and Sentinel-5P missions, GMV has developed the spacecraft operational simulators, which are based on SIMULUS, ESOC's spacecraft simulation infrastructure.

At the epoch of starting the development of both SEOSAT and Sentinel-5P operational simulators, SIMULUS did not include any emulator of LEON3 processor. GMV decided to integrate the AirBus SimLEON3 emulator into SIMULUS infrastructure for its usage in both SEOSAT and Sentinel-5P operational simulators.

In general, in software applications, the integration of software components or products developed by separate companies for different purposes is a challenge. The performed integration of AirBus SimLEON3 product into ESOC SIMULUS infrastructure is a good practical case of integration of different software modules coming from different companies and purposes. Main technical challenges of this integration are explained; emulator wrapper, access to implemented peripherals, memory mapped I/O units and interfaces with the rest of components of the SCOC3 computer models developed by GMV. The coexistence of two OBSW mounted on top of two computers with LEON3 processors simultaneously working in a master-slave configuration and interchanging data through a space-wire interface is explained. Evidence of the proper working of the OBSW in the spacecraft operational simulator in the different AOCS modes is provided.

INTRODUCTION

The main purposes of operational simulators in space missions are: Mission Control System testing, preparation and validation of the operational procedures, training of operations staff and support to troubleshooting and maintenance during operations. A relevant characteristic of the operational simulator is that it allows running the flight OBSW as it were included in the actual spacecraft. This has made the operational simulators also essential for: operational validation of the satellite database, support to Spacecraft Validation Tests (SVTs), operational validation of the OBSW and finally the validation of OBSW patches/images before uploading to the spacecraft. Fig. 1 shows the location of the operational simulator inside a generic ground control centre and how the operational simulator is able to interact with the Mission Control System (MCS) reproducing all the TM/TC of the spacecraft.

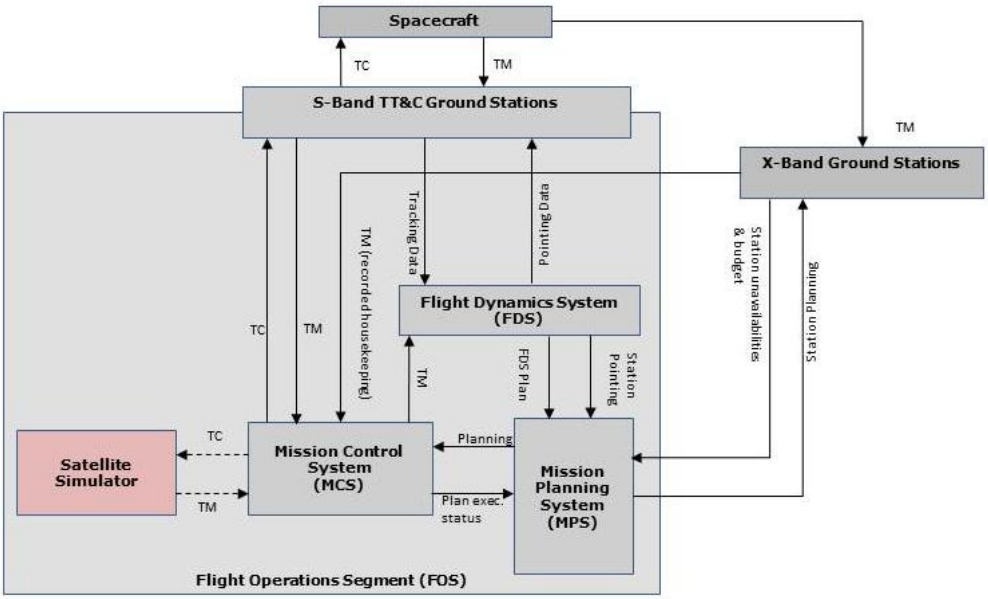

Fig. 1. Location of the operational simulator in a generic ground control system for Earth observation mission

COMPONENTS OF THE OPERATIONAL SIMULATOR

As shown in Fig. 2, a high level decomposition of the operational simulator consists on specific spacecraft models for all the subsystems and a simulation infrastructure.

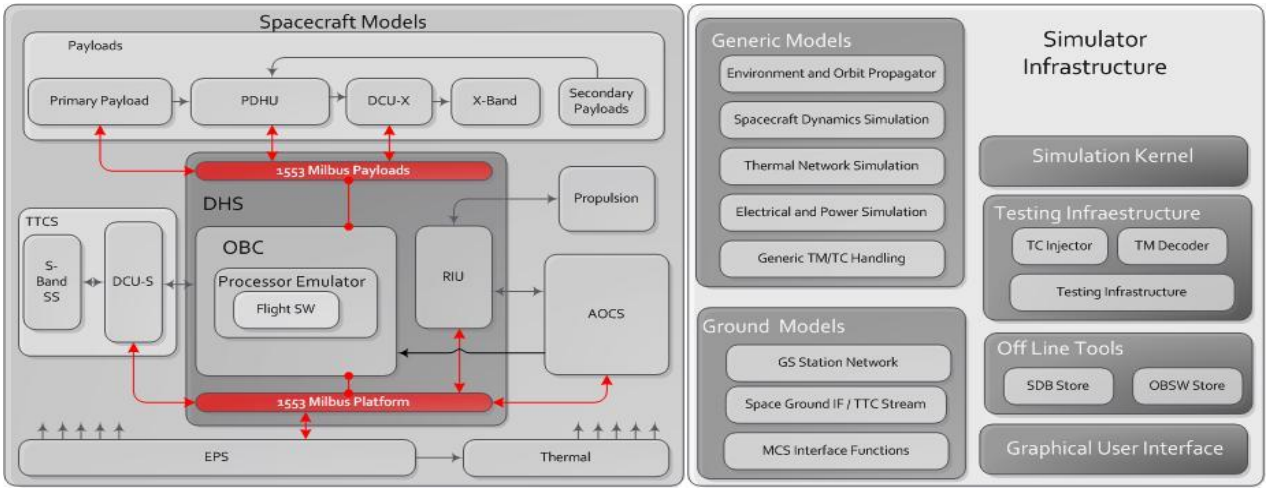

Fig. 2. High level decomposition of the operational simulator

The operational simulator is therefore characterised by the following aspects:

- Models for AOCS, Propulsion, Data Handling, Thermal, Communications and Power subsystems.
- The level of representativeness is primary driven by the need to allow the execution of the actual flight OBSW as in the real spacecraft. As example this makes necessary:
  - Emulation of the main computers of the command and control subsystem of the platform, including modelling of mass memory.
  - Reproduce all the TM/TC included in the Spacecraft Database (SDB) containing all the required flight configuration data.
  - Models of each on-board unit. All redundant and commutable items are represented.
  - Payload simulation limited to the OBSW bus interface and operational needs, (i.e. payload SW functionally modelled).

- Environmental and dynamics simulation. This allows closed-loop testing of the AOCS flight SW in the emulated computer with AOCS units and dynamics models.
- Simulate spacecraft TM/TC interface with the ground stations.

Regarding the simulation infrastructure, both SEOSAT and Sentinel-5P operational simulators use SIMULUS/SIMSAT, ESOC's spacecraft simulation infrastructure. This infrastructure provides a development and execution framework. Among others, the most relevant elements of SIMULUS/SIMSAT are:

- A Simulation Kernel, which provides SMP2 services: Scheduler, Time Keeper, Logger, Event Manager.
- A man-machine interface for managing the simulation context, visualisation of model data, user control of the simulation via commands and scripts, saving and restoring simulation state (i.e., *breakpointing*).
- A testing infrastructure and means to update the OBSW images and SDB handling.
- A standard interface and libraries to simulate the ground stations.

In addition, SIMULUS/SIMSAT favours re-use of existing models by means of design, catalogue and package files.

Fig. 3 provides a logical overview of the different components of SIMSAT. At the epoch of starting the development of both SEOSAT and Sentinel-5P operational simulators, SIMULUS did not include any emulator of LEON3 processor. GMV decided to integrate the AirBus SimLEON3 emulator into SIMULUS infrastructure for its usage in both SEOSAT and Sentinel-5P operational simulators.

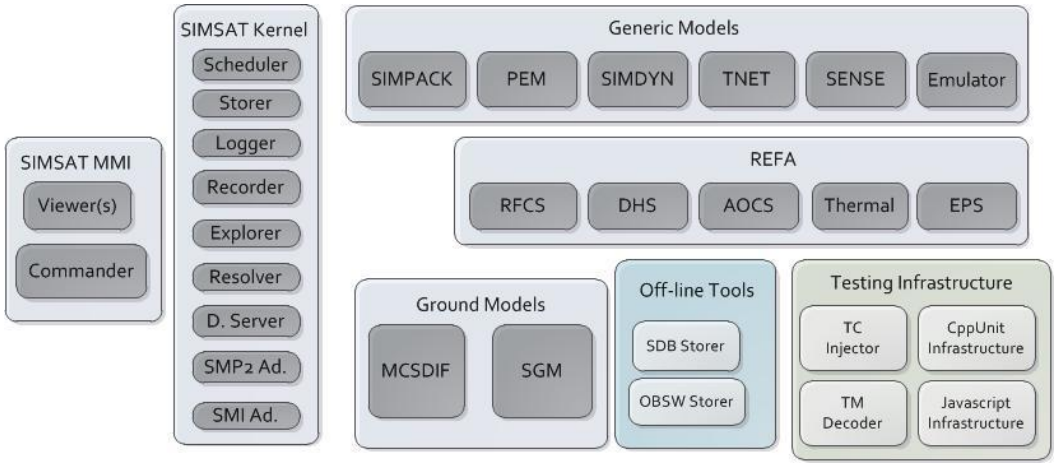

Fig. 3. Schematic overview of SIMULUS/SIMSAT environment

SIMLEON3 PRODUCT DESCRIPTION AND ON-BOARD COMPUTER MODELS

SEOSAT and Sentinel-5P OBCs are based on the SCOC3 ASIC, and extended with a Power Module and a Reconfiguration Unit. SimLEON3 is an AirBus product capable of emulating a LEON3 processor specifically tailored for the SCOC3 architecture. This emulator covers several modules in the SCOC3 processor, as shown in Fig. 4, such as the Timers unit or the Memory Controller unit. SCOC3 units not covered by the emulator are developed as part of the operational simulator. Tight integration between all the processor modules is of outmost importance for the simulator to work properly.

For redundancy purposes two identical OBCs are boarded into SEOSAT and Sentinel-5P. They operate in a master-slave configuration (any of the two can play either role) and they run the same flight OBSW. They communicate through a devoted Space-Wire Inter-Processor Link; a different Space-Wire link is used to access resources on the slave OBC in order to maintain hot redundancy. Each OBC model contains an instance of the SimLEON3 emulator, allowing the simulator to execute the master and slave parts of the flight OBSW.

The SimLEON3 product is distributed as a product package containing both development and run-time parts. It provides a C++ public API to access simulation state and device units; header files describing these services are provided for development. The run-time part is composed of a set of libraries compiled for the simulator target platform.

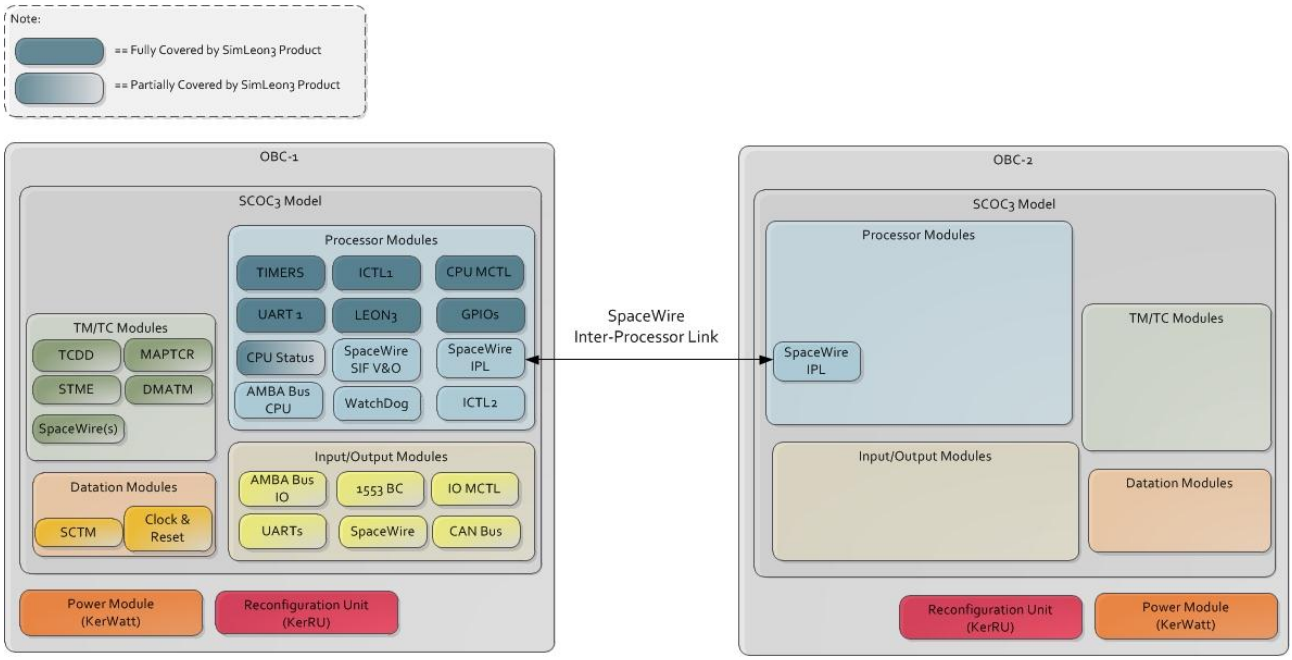

Fig. 4. Schematic view of OBC architecture and coverage of the Processor models by SimLEON3 product

Besides mere emulation of LEON3 processor and selected device peripherals, SimLEON3 allows for loading binary memory images, debugging of running software, inspecting memory contents, scheduling events, etc. Of these, it is of particular interest the access to the emulator scheduler, which provides a higher resolution than that of SIMSAT.
SimLEON3 exported API is not SMP2 compliant, which makes integration in a SMP2 simulation environment a not direct process; the API being written in the C++ language contributes to ease the process, though.

INTEGRATION OF SIMLEON3 PRODUCT WITH THE OPERATIONAL SIMULATOR

The process of integrating an external emulator into a SIMSAT-based simulator is split into two main tasks. One of these tasks consists in effectively configuring the SIMSAT environment for loading the emulator run-time components before starting the simulation, so that emulator symbols can be found when the simulator is running. The other task corresponds to actually adapt models to make use of emulator provided services; this includes providing a means for the emulator to actually emulate the loaded software.

Integration with Simulation Infrastructure

A SIMSAT-based simulator can be executed as either a standalone process or integrated with the SIMSAT Kernel. In order to run the simulator as a stand-alone process all simulator models are compiled and linked together to produce a unique binary executable file. In this case, SimLEON3 libraries are directly linked into this executable file. The simulator based on SIMSAT Kernel is defined by means of a Kernel Architecture File (KAF) that describes, among other things, the libraries that need to be loaded and initialised at simulation start-up. SimLEON3 libraries are specified in this KAF so that they are readily available for SIMSAT to load them. In this case, the simulator symbols are dynamically linked at run-time. Regardless of the simulation mode, SimLEON3 being available as a set of pre-compiled libraries makes the process of integrating it with simulation infrastructure a straightforward process.

Integration with Simulator Models

The processor emulator is a key component in an operational simulator and is managed in a different way than the rest. As previously said, SimLEON3 API is exported as a set of C++ classes and interfaces. This fact eases integration into a C++ SMP2 environment.

In order to abstract the data types and conventions used by SimLEON3, an adaptation layer is implemented and placed in between the emulator and the rest of simulator models. This adaptation layer comprises services of three categories:

- Simulation services: comprise the family of *execute* methods and utilities for binary image loading and software debugging.
- Unit access services: comprise methods for directly accessing registers of emulated units, as well as specific functionality of these units.
- MMIO services: comprise a homogeneous set of methods for accessing units external to the emulator, being these emulated of not.

While Simulation and Unit access services are provided by the emulator, MMIO services need to be provided by the surrounding simulator.

Fig. 5 shows schematically where the adaptation layer lies within the simulator and how it is split. The Processor Emulator Proxy model is mainly concerned with Simulation services. This model complies with the proposed architecture in REFA.

The I/O Proxy models are concerned with MMIO services. Since SimLEON3 is specifically tailored for SCOC3 architecture, it routes every MMIO operation through a different interface representing a region in the memory map (i.e. a specific bus or device unit); an I/O Proxy model is instantiated for each one of these interfaces.

Unit access services are illustrated with ICTL1 Proxy and GPIO Proxy models. As previously stated, ICTL1 and GPIO are device units emulated in SimLEON3; they both are accessed from different simulator models. In order to maintain clean interfaces and isolate the different parts of the adaptation layer, a different adaptor model providing a specific interface is developed for every emulated unit.

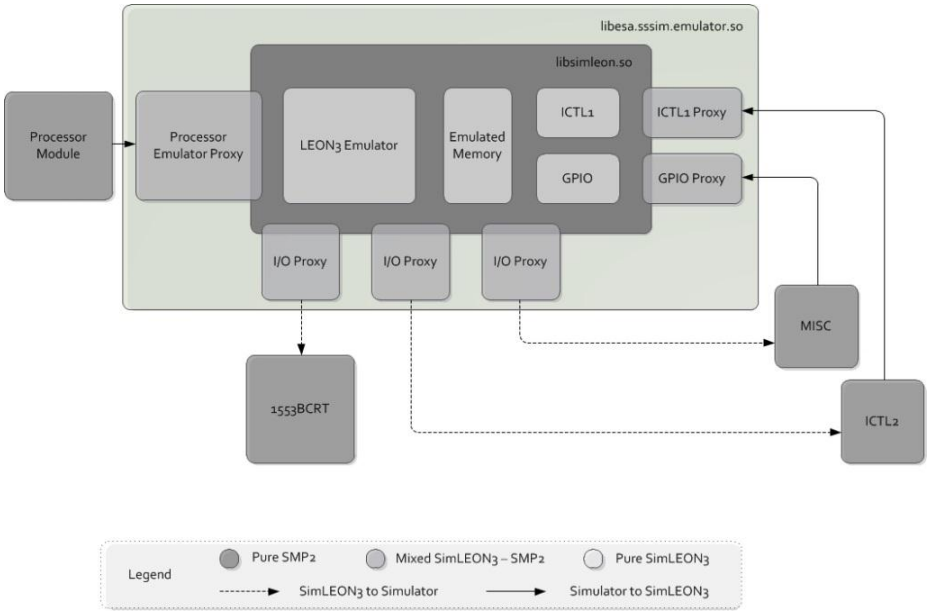

Fig. 5. Depiction of SimLEON3 integration in a SMP2 environment

Integration with Simulator Scheduler

A particularly useful feature of using an external emulator and specifically of SimLEON3 is that they grant the simulator access to fine grained timed events. The SIMSAT scheduler provides resolution of nanoseconds and is used for triggering one-shot events of any period and cyclic events with period in the order of microseconds. Some device units require greater accuracy for working properly with the running OBSW. Fig. 6 shows an example of combined use of SIMSAT and SimLEON3 schedulers.

In SEOSAT and Sentinel-5P simulators, correct implementation of 1553BCRT unit is essential to achieve an accurate and realistic simulation, as it controls data exchanges between the DHS subsystem and the rest of S/C subsystems, in particular the AOCS subsystem. This unit implements a WAIT instruction that halts instruction execution by a few

nanoseconds; at the end of the halt period, an interrupt is raised in the processor for the OBSW to know about this. Since emulator run cycle is several orders of magnitude greater than the possible arguments to WAIT, SIMSAT scheduler is not an option for simulating the WAIT delay; SimLEON3 scheduler is used for this purpose.
The synchronisation signal scheduled by SEOSAT and Sentinel-5P OBSW has a frequency of 16 Hz and is provided by the SCTM unit. This signal is used by the OBSW to perform cyclic operations; therefore, it is essential for an accurate simulation. Although SIMSAT scheduler could cope with such a frequency, the importance of this signal makes SimLEON3 scheduler better suited for triggering the associated interrupt. In contrast, the 1 Hz signal in charge of updating the internal coarse time register is of much less importance, and is assigned to SIMSAT scheduler.

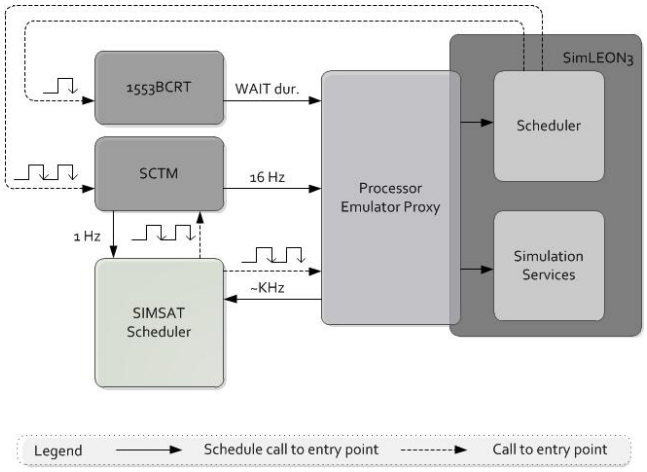

Fig. 6. Example of combined use of SimLEON3 and SIMSAT schedulers

On-Board Computers in Hot Redundancy

As previously stated, SEOSAT and Sentinel-5P Data Handling Subsystem includes two hot redundant OBCs in master-slave configuration. Communication between processors is achieved through a dedicated Space-Wire link. Slave processor idles until it receives a work package from master.
This feature of the DHS is simulated using two instances of the emulator, instead of modelling one of the computers only functionally. Loading two emulator instances increases the memory requirements of the simulator; executing both emulators concurrently greatly decreases overall performance. In order to avoid this performance penalty a technique is developed to detect when a processor is idling. This technique is similar to that typically offered by processor emulators. Dedicated behaviour is added to the Space-Wire terminal model to detect when a petition arrives to the slave processor, so that it is effectively awakened. Fig. 7 shows a timeline where slave processor is executed only when strictly required plus a small period to perform idling detection.

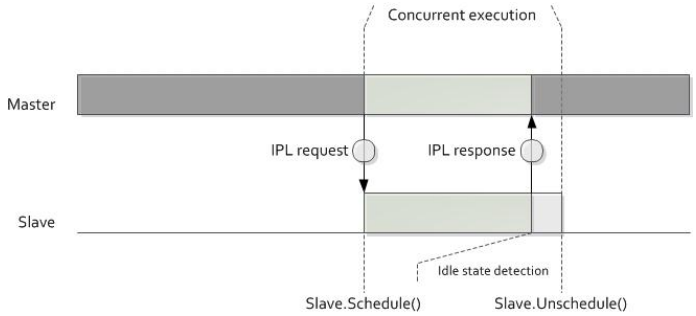

Fig. 7. Execution of slave OBC on request by master

OPERATIONAL SIMULATOR CAPABILITIES

Interaction with the Mission Control System

The environment composed of the Mission Control System (MCS) connected to the operational simulator is very useful for the testing of the MCS, the preparation of the operational procedures and the training of the operation staff. Fig. 8 presents this environment. In this configuration the MCS user can operate and monitor the operational simulator through TM/TC as they were operating the actual spacecraft. In this environment, the simulator can be also directly operated by the user to simulate failure injection in any spacecraft unit; the effects of this failure in TM can be monitored from the MCS team. In general, for the user of the operational simulator, there are two ways to interface with the spacecraft models: via a MMI and/or through scripts.

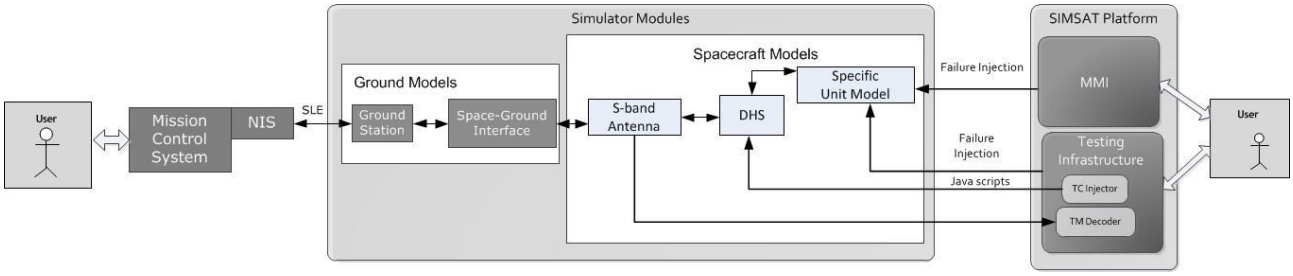

Fig. 8. Environment composed by the Mission Control System and the operational simulator

AOCS OBSW Closed-Loop Testing

All the models in the operational simulator are able to work in closed-loop (see Fig. 9); i.e. dynamic models, sensor and actuators models and real OBSW are working together, ensuring the proper working of the OBSW in all operative modes and its transitions.

For SEOSAT and Sentinel-5P missions, the AOCS SW mode of the spacecraft is IDLE on start-up, where it performs OBSW configuration, spacecraft separation from launcher and solar array deployment. Later, it moves to Acquisition Safe Hold mode, where the spacecraft initializes the AOCS modules to determine and stabilize and the spacecraft attitude. It then moves to Normal Mode, where the spacecraft is completely functional.

In Autonomous Operation, the spacecraft will transition between Sun Pointing (SUP) mode and Geocentric Accurate Pointing (GAP) mode when in eclipse. When commanded, the spacecraft performs data acquisitions over a specific geographic area, by means of a transition from SUP/GAP to Manoeuvre submode (MAN) and then to Custom Accurate Pointing (CAP). The spacecraft automatically performs the transition between SUP and GAP, when it comes in and out of the eclipse region of the orbit, respectively. A typical data acquisition manoeuvre will imply the transition: SUP → MAN → CAP → SUP.

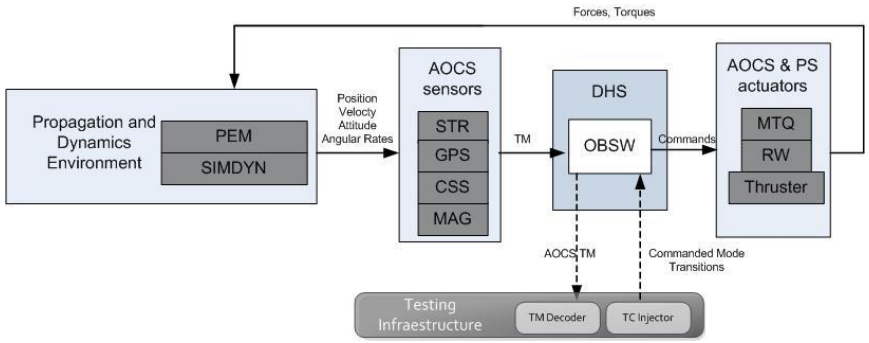

Fig. 9. OBSW working in closed-loop with the rest of simulator models

Therefore, a complete AOCS closed-loop validation of the operational simulator includes the transition between IDLE, Acquisition Safe Hold and Normal modes, and exercises all of the previously depicted submodes, including a data acquisition manoeuvre. The results of a closed-loop simulation, using the operational simulator are presented in the Fig. 10.

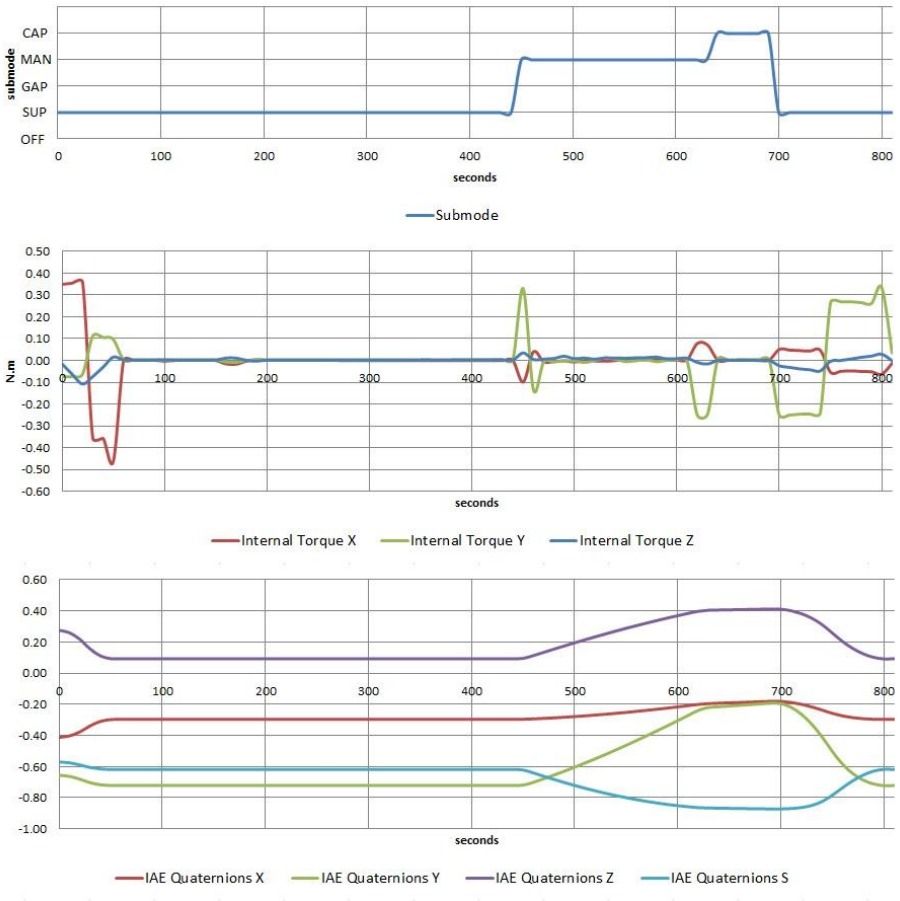

Fig. 10. Simulation results of an example of a complete AOCS closed-loop validation test

ACKNOWLEGEMENTS

SEOSAT Operational Simulator has been performed under a contract with ESA/ESTEC and funded by the Spanish Administration. Sentinel-5P Operational Simulator has been developed in the frame of ESA/ESOC GFC8 contract. Authors want to acknowledge Antonio Ceballos, Mónica Rollán, Miguel A. Iribarren, Javier Herrero, Rafael Portero, Eduardo Zornoza, Helena Iglesias and Ioannis Angelis for their invaluable contribution to the development and testing of the SEOSAT and Sentinel-5P operational simulators.