A Software-based Environment for Development & Validation of Spacecraft On-board Software

Alessio Di Capua [(1)], Sante Candia [(1)], Tomas Di Cocco [(1)], Marco Anania [(2)]

(1) Thales Alenia Space Italia – Via Marcellina, 11 – 00131 Roma
(2) Thales Alenia Space Italia – Via Saccomuro, 24 – 00131 Roma
mail: name.surname@thalesaleniapace.com

INTRODUCTION

The usage of simulators in space systems engineering is highly advisable in order to provide a cost-effective validation facility during the project early phases, when typically the hardware equipment is not yet available, and during the formal on-board software validation and qualification phases.

This paper reports the Thales Alenia Space Italy (TASI) approach to this matter. The TASI Software Development and Validation Environment (SDVE) has been designed and developed during the last years and has been successfully deployed for Missions like COSMO SkyMed and Sentinel-1A. Moreover, it is the baseline environment for future Missions such as COSMO Second Generation and Sentinel-1B.

MODELLED SYSTEM

The SDVE runs on a commercial workstation and results from the integration of the SIMSAT-4 framework, one or more TSIM instruction-level simulators and a set of custom models for simulating/emulating the behaviour of the on-board computer, payloads, avionics subsystems/sensors/actuators, communication links, Spacecraft environment.

The SDVE scope is to simulate as closest as possible the On-Board software environment. This includes the simulation of the On-Board Computer, the RTU (remote terminal units) and the payload as well. A general schematic of the modelled environment is reported in Fig .1 (*Savoir* functional block of a platform and payload system).

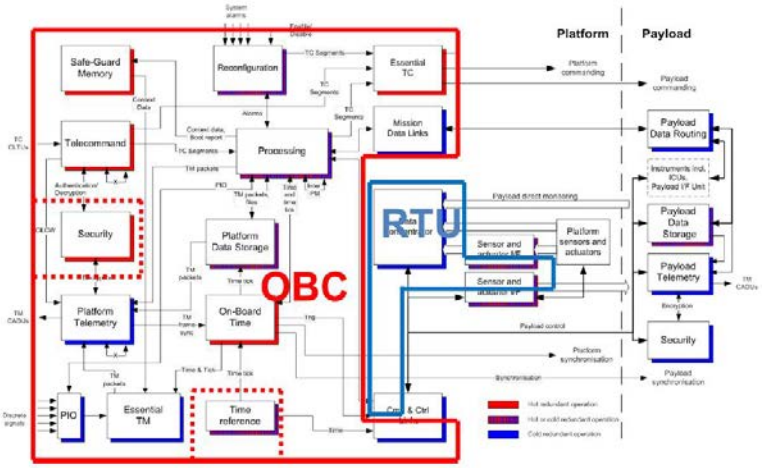

Fig. 1 The general Savoir Functional diagram of platform and payload

The current SDVE implementation follows the TASI Avionics design based on the flying ICS (Integrated Control System) avionics subsystem developed in the scope of the *PRIMA* project employed in the *Sentinel1-A* and *COSMO-SkyMed* missions. The PRIMA design is reported in Fig. 2

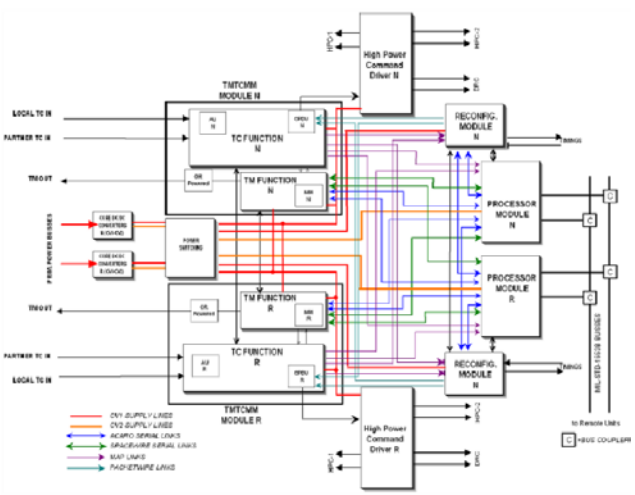

Fig. 2 The PRIMA Avionics design on which the current SDVE is based

The SDVE features the following main characteristics:

- The simulation is capable to model up to the single register the sysem is made of and not just from a functional point of view.

- The On-board Computer (OBC) model runs the actual Avionics Software (ASW) executable image. This capability can be extended to the Payloads and 'intelligent' subsystems in case the SDVE is going to be also used to validate the on-board software running on them.

- Full integration of the Spacecraft Telemetry & Telecommand Database.

- The hierarchical Spacecraft model fully represents the avionics architecture and includes all the foreseen redundancies and the models provide fault injection capabilities. This allows the full validation of the Failure Detection, Isolation & Recovery (FDIR) designed logic.

- The validation/qualification test procedures development environment is the same used at avionics subsystem test bench level and at Spacecraft level. This guarantees the 100% portability across the various environments without any need for error-prone porting activities.

- The full validation of functional and performance requirements is possible. This implies that a very restricted set of test procedures (usually, 5%÷10%) need to be formally run on the final target.

- The freeze/save/restore operations of the complete simulator state, including the Spacecraft dynamics, is supported.

- The ASW debugging capabilities are provided: step through instructions, continue execution, read/write variables maintaining consistency between processor emulator and other models.

- The SDVE is designed using a modular approach, thus it can be used as a stand-alone software-based environment or as part of the avionics test bench and at AIT phase (with hardware-in-the-loop). Indeed, each model can be selected as 'simulated' or 'real' through an external configuration file used to initialise the full

environment. The configuration file also allows the modification of parameters such as dynamics parameters (orbit, external disturbances torques, ...), sensors/actuators mounting matrixes, communication links parameters.

SDVE ARCHITECTURE

The SDVE implementation can be split into the blocks reported in Fig. 3:

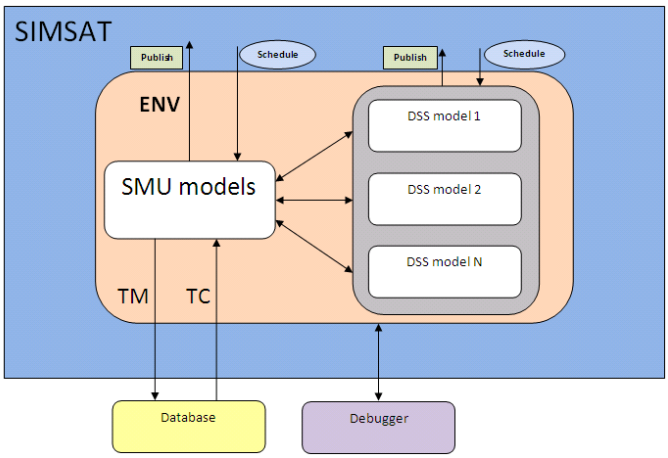

Fig. 3 – SDVE architecture

- **SIMSAT**: ESA simulation environment for models scheduling and logging.
  - It Publishes on its GUI the public models parameters and functions allowing the dynamic interaction between simulation and user.
  - Saves and restores the full simulation status.
  - Schedules the events registered by models.
- **ENV:** spacecraft dynamic model.
- **SMU:** boards models. These objects are a representative emulations of the On-Board computer HW as buses (1553, SpW, CAN), asic (EPICA, TMTCMM), memories (EEPROM, SGM). The core of SMU is the microprocessor model: the TSIM emulator is used as plug in model to realize it; however SDVE can integrate easily any microprocessor emulator.

- **DSS:** sensors and actuations dynamic models as thermal nodes, power nodes, GPS, gyro, reaction wheel, buses slaves. Each model is functionally representative of the real HW allowing simulating tests with the integrated spacecraft HW. The DSS has been projected with a stack architecture (of three layers) to be portable in more simulation environments. (See chapter 'The "Generic Object" approach')

The SDVE scheduling algorithm assures the completely consistency between

- Simulation time
- OBSW on board time
- HW simulated time
- Models dynamic propagation

OBSW and model dynamics advance at discrete time managing the SIMSAT events registered. Using this feature, SDVE can run at speed greater than real-time and can run stress test without performance degradation.

- **TMTC IF**: interface with external databases and on board event loggers (using sockets).
- **Debugger IF**: interface with gdb for debugging models and OBSW (using socket).

The “Generic Object” Approach

A pure abstract class "GenericObject" implements the common general purposes models properties and methods, like data loading from the setup file, diagnostics and messaging, run-time instructions dispatching and parsing. Moreover it also implements the common base object properties (ON/OFF status, power consumption, etc.) and provides access to standard system model data (X, Xdot, U, Y) and to their handling. This model has been called “level_1” and is common to all test environments (simulator or test-bench with hardware-in-the-loop). A specific equipment model inherits from the GenericObject class (and from specific interface classes like the Bus-1553 RT when applicable) the general-purposes properties and methods; it has to implement at second inheritance level (equipment) the standard data structures definition and the pure virtual methods to handle them. This level of model is totally portable and can be shared among the different test environments by implementing in each of them a third inheritance level (environment dependent) implementing the interfaces with the specific simulation environment. This simulation models has been addressed as the “level_2”. A third set of models inherits from the “level_2” classes, and allow the different environments to interface the common equipment models. This level, called “level_3” is custom for each test environment. This approach has been adopted in the frame of several projects where three different simulation environments are used: Matlab AOCS Development Environment (M-ADE) , Software Development Verification Environment (SDVE) and Avionic Test Bench (ATB). Each simulator is designed to suit the needs of the different phases of the software development, like necessity of versatility in the design phase and need of a fully representative simulator in the test phase. This approach guarantees a great flexibility and assures, at the same time, the comparability between simulation’s results.

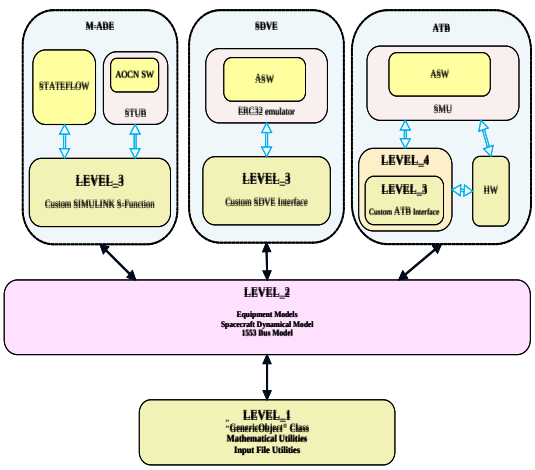

Fig. 4 – The “Generic Object” approach concept

SDVE MMI

The SIMSAT MMI provides a window interface that allows simulation users to interact with a running SIMSAT simulation. The environments allow to set-up and check in real-time alpha-numerical display (AND) and plots for each of the published objects. The MMI can be linked to an external MTP in order to run a test-script. The link with an external MTP (and its relevant Database) allows to run on in the SDVE environment the same test script that will be used during later test-phases (ie: Subsystem and Spacecraft test phases). The same MMI, on the other hand, allows the user to execute single commands or specific scripts (eg: processor emulator scripts). In Fig. 5 a snapshot of an MMI with a running test is reported

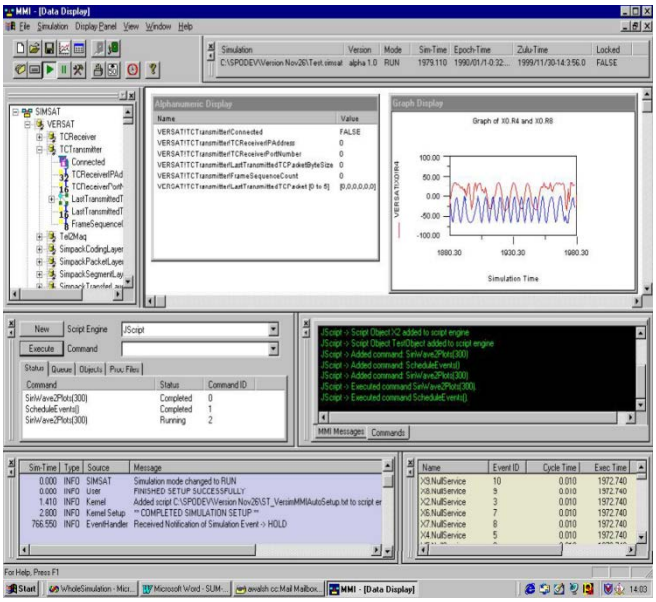

Fig. 5 – Simulator MMI

CONCLUSION

The SDVE has been successfully used throughout the on-board software test campaign of several projects proving its efficiency in terms of accuracy (test results VS flight data) and in terms of overall time saving. Nevertheless, its development is continuously ongoing: next enhancement foreseen multiple microprocessor emulators (feature that can be useful to simulate an OBSW and a payload software at the same time or to simulate multiple processor architectures) and standardization towards SMP-2.