First Application of the Generic Emulated Test Software, GETS, in the LISA Pathfinder Operational Simulator

10th International Workshop on Simulation for European Space Programmes

SESP 2008

7-9 October 2008 at ESTEC, Noordwijk, the Netherlands

Joachim Ochs[(1)], Michael Irvine[(1)], Mehran Sarkarati[(2)],, Mariella Spada[(2)]

[(1)] Vega IT GmbH
Robert-Bosch-Straße 7, D-64293 Darmstadt, Germany
Tel.: +49 (0)6151-8257 0
WWW.VEGA-GROUP.COM
joachim.ochs@vega.de, michael.irvine@vega.de

[(2)] European Space Agency, ESOC
Robert Bosch Str. , 64293 Darmstadt, Germany
mehran.Sarkratir@esa.int, mariella.spada@esa.int

ABSTRACT

The development of operational simulators is in many ways strongly dependent on the specification and the design of the actual spacecraft and its subsystems. One of the main challenges in developing operational simulators is accordingly the management of the external inputs, such as the on-board software and the mission database.

ESA operational simulators are based on the use of processor emulators at the core of the Data Handling and Attitude Control subsystems running the mission OBSW. The use of the emulator and the mission OBSW achieves a high level of modelling fidelity.

The operational simulator is an important element of the ground segment used for the validation of other ground data systems, such as the mission control system and the ground station facilities. The simulator is also primarily used for training the flight control team and for the preparation and validation of the flight operation procedures. The development of the operational simulator is therefore driven by the overall ground segment development schedule. It starts often before a consistent set of spacecraft design documentation, the OBSW and the database are available.

To address these challenges and in order to mitigate the impact of the frequent changes of the simulator development baseline, a number of measures have been introduced over the years in the domain of operational simulator development at ESOC, including the introduction of the incremental delivery model, the development of the Ground Systems Test and Validation Infrastructure, GSTVI, which allows the independent testing of various ground system elements and hence to some extent the decoupling of their development schedule; as well as the provision of the Generic Emulation Test System, GETS.

The GETS tools provide in this context the means to produce a simple but generic and configurable version of the OBSW, which can be used in the early phases of the simulator development to reduce the dependency on the delivery schedule and continuous changes of the spacecraft OBSW. The GETS OBSW can be integrated and run within the emulator in the Operational Simulator as an early stub replacement for the actual OBSW. Using GETS supports the integration within the simulator of the emulator and the models of all other CDMU subsystems such as the I/O boards, the communication buses and the various ASICs in the very first delivery of the simulator, hence implementing and validating all the interfaces as early as possible.

The LISA Pathfinder operational simulator is the first project which has adopted GETS. This paper will provide an overview of three different approaches that can be adopted when applying GETS to an Operational Simulator. It will discuss the advantages and shortcomings of each approach and present the lessons learnt from the first real-world application of GETS on the LISA Pathfinder operational simulator.

OVERVIEW and BACKGROUND

The Generic Emulator Test System (GETS) was a study for the European Space Agency (ESA), European Space Operational Centre (EOSC) performed by VEGA and Dataspazio in 2004/2005, and was presented at the SESP 2006 conference. The study investigated the possibilities of developing a simple Generic Emulator System (GES), which can be used during the development of the operational simulators to replace the real onboard software in early phases of the development, before the real OBSW becomes available.

Almost every operational simulator of an ESA mission comprises an emulator, on which the flight version of the OBSW shall be executed. The experience of previous simulator developments has however shown that delays in OBSW delivery schedule are not unusual and can accordingly impact strongly the development schedule of the operational simulator. The dependency of the simulator development schedule on that of the spacecraft development is illustrated in the figure 1.

The aim of the GETS study has been to specify and develop a set of tools used to generate a stub OBSW image, which can support integration of OBSW in the.operational simulator and validation of the HW/SW interfaces in early phases of the development. It is important to note that the objective of GETS is **not** to completely replace the mission OBSW.

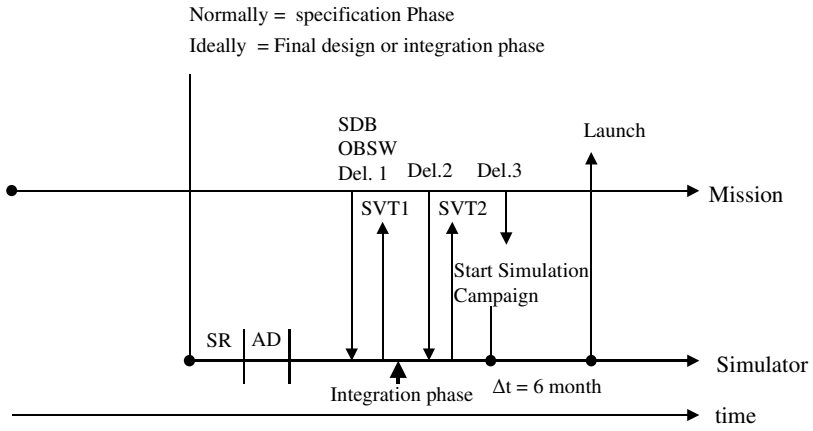

Fig. 1: Development timeline of a satellite and the related simulator

The following section provides an overview of the GETS system and its components.

THE GETS SYSTEM

The GETS tool enables a user to enter mission-specific parameters, and then to auto-generate a set of configuration files and the stub OBSW image, the Generic Emulator Software (GES). These files can then be loaded into the mission Operational Simulator. Such a solution takes advantage of the fact that ESA missions, and the corresponding OBSW sets, are typically based on the use of ECSS and PUS standards for Telecommands and Telemetry. In addition, at a high level the OBSW for the different missions perform the same core tasks, including the handling and distribution of Telecommands, and the acquisition and transmission of Telemetry. These cores OBSW activities also correspond to the OBSW functionality required to support the integration of subsystem models within the simulator.

Fig 2 below shows the GETS tools and the interfaces between them. The components themselves are described below.

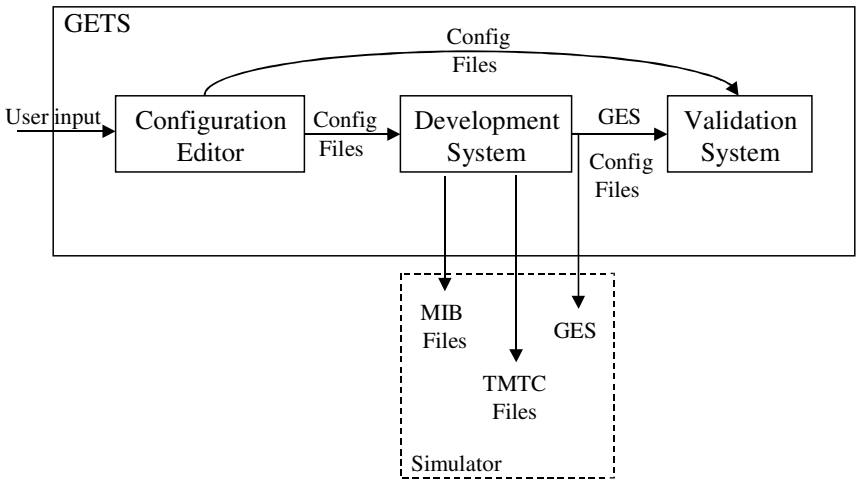

Fig. 2. Overall design of GETS

The Configuration Editor is an off-line tool used to enter the mission-specific information including:

- TM and TC packet and parameter definitions.
- The relationship of TM packets to virtual channels.
- Data bus interrogations.
- Bus addresses.
- The relationship of packet Application Id to data bus addresses.
- Import of existing mission SCOS-2000 MIB files (containing TM and TC definitions).

The Development System is the component of the GETS system dedicated to the building of the Generic Emulator Software (GES).
It performs the following main operations::

- Import the configuration files defining the TC/TM packet structure and the bus configuration (XML files produced by the Configuration Editor).
- Load its own configuration files defining the GES Memory Map I/O, the PUS constants definition and the list of telecommand APID codes reserved to high priority TC, GES and user code;
- Inserts the configuration data into the GES source code;
- Builds the GES onboard software image;
- Exports a set of SCOS-2000 format MIB files.

The GES is an OBSW image generated by the Development System. It provides a subset of typical mission OBSW functionality aimed at supporting simulator integration activities. The main GES OBSW features are:

- Implementation of many of the TM and TC Standard Services.
- GES is configured based on user input to CE and DS tools.
- Interfaces to external I/O hardware devices are via GES API software.
- Distribution of TCs using external bus interfaces (via GES API).
- Acquisition of TM using external bus interfaces (via GES API).

The Validation System (**VS**) is used to support validation of the GES software image generated by the Development Environment. It is a real-time SIMSAT-2003 based simulator that loads and executes the GES in an ERC32 emulator.

APPLICATION ON THE LISA PATHFINDER OPERATIONAL SIMULATOR

As described in the previous section, GETS has been developed in the context of a study. Lisa Pathfinder is the first mission to adopt GETS in its operational operational simulator, which provides a number of advantages:

- No need for a switching/functional model of the CDMU in the first delivery of the simulator
- Implementation of a high fidelity CDMU model already in the first delivery of the simulator
- Validating the I/O interfaces of all simulation models (HW/SW interface)

- On-going use of GETS for regression testing of I/O device model interfaces and mission API OBSW.
- Provision of the GES-based simulator to the flight control team for early validation, training and familiarisation purposes
- Streamlining the integration of the real OBSW, once it becomes available

By default, the GES software comes with its own API to interface to the external I/O devices. However, the GES image generated by the Development System for Lisa Pathfinder has to be loaded into the Lisa Pathfinder Simulator (LPFSIM). The LPFSIM contains I/O device models representing the real spacecraft units/devices. This means that the GES image cannot run in the LPFSIM without some level of modifications to the simulator design/code, or to the GES OBSW. Actually there are several possible approaches to enable the GES to be loaded and run within the LPFSIM, as presented below.

Approach 1 – Modify the GES to use the mission OBSW API replacing the GES API.

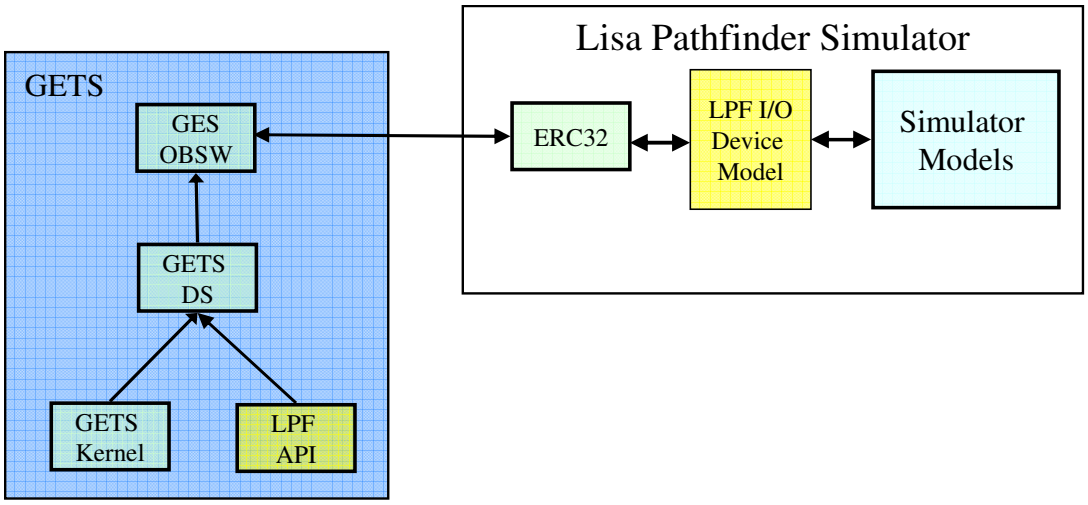

Fig. 3. GETS Approach 1

This approach allows the generated GES image to run in the LPFSIM with minimal changes to the I/O device models. It therefore represents the most representative system compared to the use of the actual mission OBSW. The GES image can effectively be loaded into the simulator as a replacement for the actual OBSW. However, the process of replacing the use of the GETS API with the mission OBSW API could be complex, depending on the level of similarities between the GES API and the mission API. If there are large differences then applying this approach can require a large part of the GES Kernel OBSW to be re-written to work correctly with the mission API.

Approach 2 – Use of intermediate adapter interface and LPF I/O Device Models.

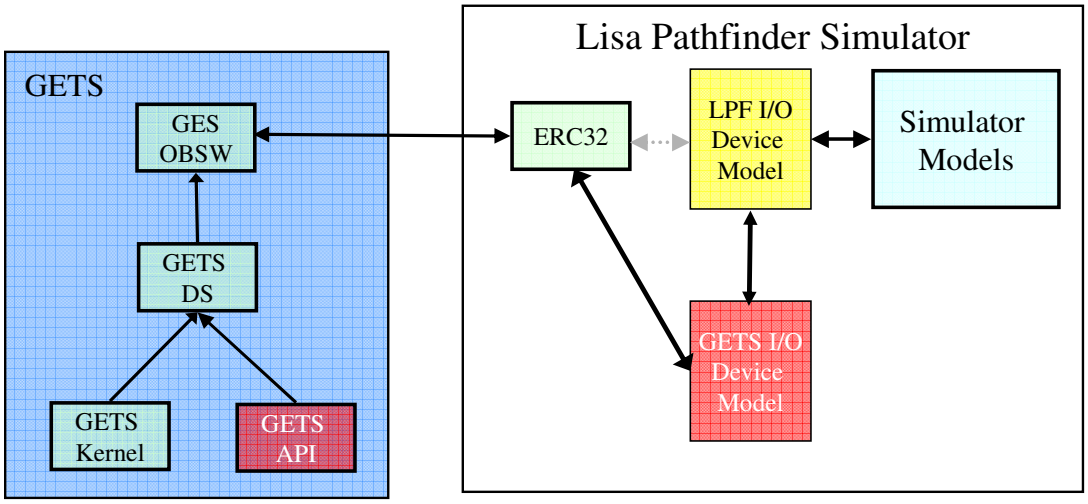

Fig. 4. GETS Approach 2

This approach involves the development of an adapter layer within the GETS I/O device models to enable the GETS Device Models to interface with the corresponding LPF Device models. This approach enables the standard GES API to be used within the GES – the GES communicates with the GETS Device models via the GES API. The GETS Device models then handle the I/O operations and translate these into the equivalent calls to the Lisa Pathfinder I/O models. This approach can be selected in cases where there are sufficient commonalities between the GETS and mission I/O devices. As described above for Approach 1, if such similarities do not exist then this approach can be complex and require considerable code to be developed in the adapter layer of the I/O device models.

Approach 3 – Use of only the GETS I/O Device models.

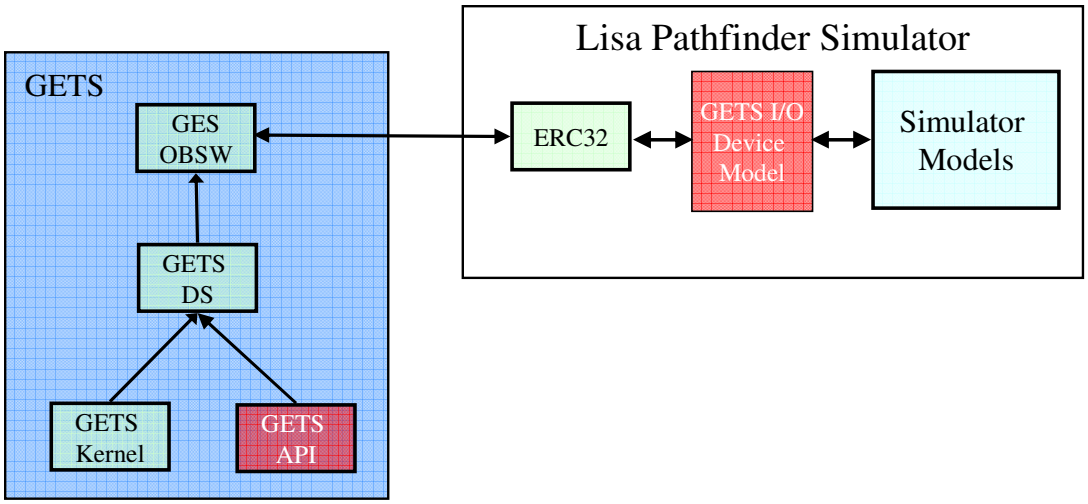

Fig. 5. GETS Approach 3

This approach bypasses the Lisa Pathfinder I/O models completely. In this approach no changes are required to the GES OBSW sources, and the GES I/O device models are used in the simulator. The GETS I/O models then interface directly with the other LPF models, e.g. to the data buses, to the X-band transmitters, etc. There is also the option of avoid the use of the data buses completely and interface the GETS I/O device models directly to the units connected to the bus(es), e.g. payloads. However, this effectively reduces the number of LPF models running with the GES OBSW. The decision as to how to implement this approach is driven by the design of the particular spacecraft units and corresponding models.

Approach Selection for the Lisa Pathfinder Operational Simulator

Where possible, Approach 1 described above is the preferred approach for using GETS with an ESOC operational simulator. It allows the GES OBSW to be loaded into the simulator with minimal changes to the simulator models. It represents therefore the most realistic simulation, in particular of the HW/SW interface and the I/O device models. However, it is also the most demanding approach, regarding the integration of the mission API with the GETS Kernel OBSW.

Approach 2 offers the same advantages as Approach 1. However, it involves the use of both the GETS and LPF I/O device models within the simulator, and requires the development of interface adapters. This means that the simulator configuration is different when using the GETS GES OBSW compared to when the mission OBSW is being executed in the emulator. This is not ideal and is a disadvantage compared to Approach 1.

Approach 3 is typically the simplest solution since it involves less development and integration effort. However, it is the least realistic setup, since it involves only a subset of the spacecraft models running with the GES OBSW, at least the mission I/O device models are not used, and perhaps also the data bus models.

For the Lisa Pathfinder Simulator Approach 1 for adopting GETS has been selected. In addition to the benefits compared to the other approaches (described above), this approach was also more consistent with the objectives behind applying GETS to the LPF simulator (as described at the start of this section). The main driver, however, is the ability to swap between the GETS OBSW and the real LPF OBSW while using the same unchanged LPF I/O device models in the simulator. This has the advantage when later in the project the real OBSW is available all I/O device interface are already tested against the same API.

LESSONS LEARNT

This section describes the lessons learnt from the first application of GETS to an operational simulator. In general, the conclusions can be split into two categories: General points, and those specific to the Generic Emulator Software (GES).

The general lessons learnt can be summarized by the following:

- The focus of GETS study was to support Simulator Developers, i.e. the main focus was on the S/C onboard interfaces in terms of TC/TM, and not fully modelling the TM/TC over the Space/Ground interface
- The GETS-based LPF Operational Simulator is provided to the simulator End Users, the Flight Control Team (FCT). This lead to the following points:
  - What subset of the mission OBSW functionality should the GETS solution provide?
  - GETS should provide the same TM/TC interface to the users as the real OBSW
  - Manual adaptation of the GES was required to implement the discrepancies between OBSW, Mission DB and SW/HW ICDs: Packet definitions, parameter types, calibration curves, etc.
- GETS has been developed as a prototype in a study. It was able to prove that the concept works with a focus on the handling of TM/TC for external units, and not on OBSW-specific TM and TC. This means what:
  - GES can not and does not replace the need for the mission OBSW within an operational simulator
  - GETS was not developed with focus on extendibility.
  - GES does not contain a Data Pool (common concept within Mission OBSW).
  - GETS does not implement the functionality of the OBSW Application layer, e.g. FDIR, AOCS, etc.

The integration of the LPF API into the Generic Emulator Software to replace the GETS API has been impacted by the following:

- The development of GETS was based on the use of the XGC compiler, using a single thread and interrupt handlers. The LPF OBSW is based on RTEMS which uses multiple RTEMS threads and messages to inform the application software about incoming calls. These different approaches required a major redesign of the concept how the GES software handles interrupts and transfers data to the I/O decives. Beside this RTEMS is in general more complex than XGC.
- Complexity due to LPF OBSW “re-formatting” of acquired TM – format in the OBSW DataPool is different from the acquired TM format
- The LPF OBSW uses several interfaces to the SpaceWire bus. The SpaceWire bus and the capablity the have more then one data protocol were missing from the GES and had to be added.

- The implementation of the LPF API required extension to service 8 telecommands in order to implement configuration facilities for the LPF API.
- The GES Kernel OBSW does not contain a Data Pool. This means that the TM acquisition is coupled with the TM packet generation. Not only is this not representative of typical mission OBSW, it introduces the risk of performance spikes in the ERC32 when large packets are acquired/generated.

FUTURE CONSIDERATIONS FOR GETS

This section describes some suggested improvements to GETS, in order to improve the provided functionality to future simulators.

At the moment the Configuration Editor and the Development System have their own simple prototype GUIs. These are not very intuitive and require a significant amount of time to perform modifications due to the validation schemes of the XML files. In case of the prototype system these XML files where relatively small, but when applied to a real mission the XML files can become large.

It is recommended that the Configuration Editor and Development System GUIs should be combined into a common GUI which guides the user through the process and required inputs.

The design of Generic Emulator Software should be re-factored to take into account not only the needs of the Simulator developer, but also the use of the system by the Flight Control team in the earlier phase of the mission. This implies a stronger focus OBSW telemetry and telecommand handling and generation. This in particular applies to providing a semi-automated or automated means of decoding telecommands and encoding telemetry packets. In this context the provision of a Data Pool within the GES OBSW should be considered. Currently the GES supports only OBDH and 1553 data buses, and the GES can only be configured for one bus type at compile time. The application of GETS to Lisa Pathfinder has already extended GETS/GES to include the use of the SpaceWire bus, and enabling the use of the 1553 data bus in parallel with Spacewire. However, the buses supported and the bus selection/configuration process should be reviewed further.

The GES design should be reviewed/updated to provide more native support for the use of RTMES. This should simplify the process of adopting the GES to a particular mission and mission OBSW – the OBSW for the recent ESA missions have been developed based on the use of RTMES.

SUMMARY

The process of adopting GETS to the Lisa Pathfinder operational simulator demonstrated that the GETS concept is valid and can be applied to a real mission. The generated GES OBSW image can be used as a reduce-scope substitute for the actual mission OBSW. However, applying GETS to the LPFSIM was a complex task for various reasons: use of RTMES, discrepancies in the ICDs, and reformatting of telemetry and telecommands inside the OBSW. This led to certain limitations concerning the generated telemetry packets as well as forwarding of telecommands to specific units which required special data handling for each single parameter inside the packets.

GETS would strongly benefit from a standardisation of the I/O Devices used in the different spacecrafts. The adoption of a Reference Architecture for mission OBSW would allow the extension of the functionality of the Generic Emulated Software, GES.