# The SVF-Lite Configuration in the End To End Avionics System Test Bench Concept: Results and Lessons Learned

*Quirien Wijnands[(1)], Cristiano Leorato[(1)], Dubravka Ilić[(2)], Tuomas Räsänen[(2)]*

*ESA/ESTEC[(1)]*
*E-mail: quirien.wijnands@esa.int, cristiano.leorato@esa.int*

*Space Systems Finland[(2)]*
*E-mail: dubravka.ilic@ssf.fi, tuomas.rasanen@ssf.fi*

## ABSTRACT

The Avionics Test Bench (ATB) is an ESA/ESTEC development aiming to support the demonstration and validation of upcoming space avionics related standards and technologies in a representative environment. Furthermore it supports projects in their needs of assessing particular technology related issues and it provides hands-on experience for ESA staff. Under the ATB denominator currently 4 configurations can be distinguished: a Functional Engineering Simulator (FES), a Functional Validation Testbench (FVT), a Software Validation Facility (SVF) and the Real-Time Bench (RTB). Currently the successor of this ATB is being developed; the End-To-End Avionics System Test Bench (E2E ATB). This is done on the results of a complete requirements and architectural consolidation of the ATB and its use-cases.

Next to this, on the basis of cross-fertilisations of the ATB FES, SVF and RTB configurations, the concept of the SVF-Lite has been identified. This configuration looks very promising in terms of added value next to the already existing ATB configurations. The SVF-Lite is able to run the Onboard Software (OBSW), or at least the Application Software (ASW) part of it, however connected to a FES-type of simulator. By cross-compilation of the Basic Software of the OBSW, the system is capable of running faster than real time at the expense of accuracy. The End-to-End philosophy is still supported by direct reuse of the TM/TC test scripts and the Monitoring and Control system.

This paper briefly describes the ATB and its various configurations and it briefly depicts the E2E-ATB activity. The main part of the paper focuses on the implementation details of the SVF-Lite, in particular as being instantiated for the ESA SGEO project team. Regarding this SVF-Lite the first results and lessons-learned are described as well.

## INTRODUCTION

Functional System Simulation has become a key activity supporting the specification, design, verification and operations of space systems. In developing these facilities, experience has shown that there is much commonality across simulation and test facilities. This experience has been captured in the ECSS ETM-10-21 “System Modelling and Simulation” Technical Memorandum” [1].

Figure 1 shows the different Avionics Test Bench (ATB) configurations throughout the project lifecycle and the possible model reuse. As indication, the red ellipse shows the support that the ATB aims to provide during a project lifecycle.

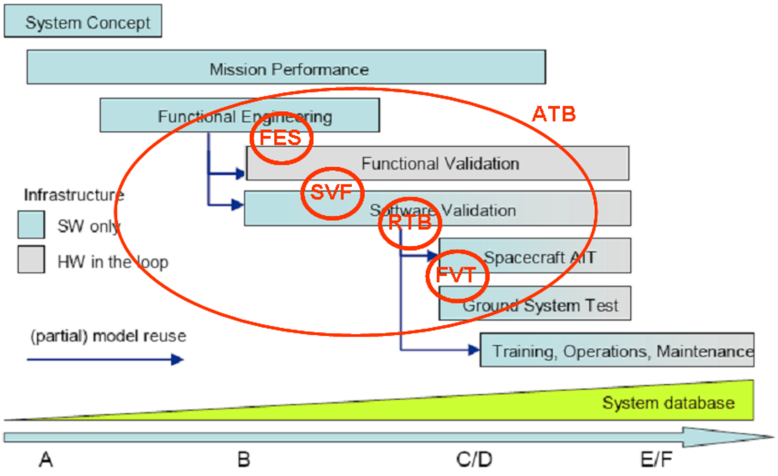


Figure 1: System Simulation Facilities and the ATB configurations throughout the project lifecycle.

The ATB concept has been proven and is now operational in the Avionics System Laboratory of ESTEC (Figure 2). The main purpose of this infrastructure is to support the demonstration and validation of upcoming space avionics related standards and technologies in a representative environment, as well as supporting projects in their need of assessing particular technology related issues. Within the context of the ATB, space avionics encompasses data handling (processing and storage), TM/TC processing, Attitude and Orbit Control System (AOCS) and mission management. Both development process and application related standards and technologies are within the scope of the ATB.

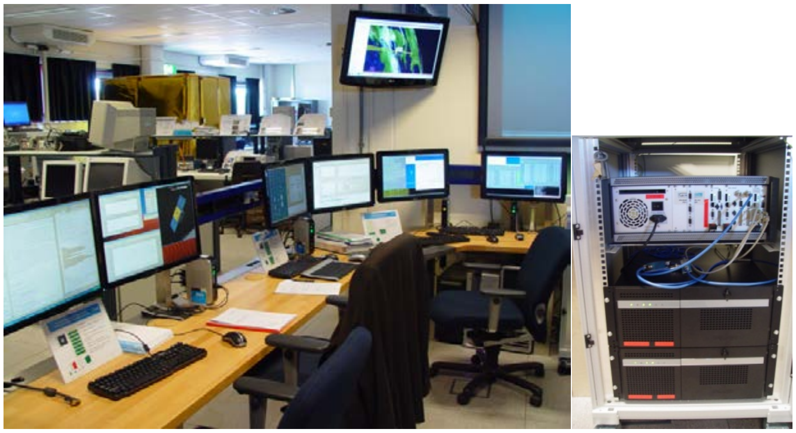


Figure 2: ATB console desk and Rasta assembly

The current ATB consists of a number of so-called configurations of the ATB. Following the naming convention in the ECSS ETM-10-21 [1] the following configurations can be distinguished: a Functional Engineering Simulator (FES), a Functional Verification Testbench (FVT), a Software Validation Facility (SVF) “Software in the Loop” configuration and a SVF “Hardware in the Loop” configuration. The latter is often referred to as the Real Time Bench (RTB). For more information on the different configurations, their architectures and the use-cases the reader is referred to [2].

Based on the experience from the developments of the ATB configurations and based on the results of a thorough Requirements and Architectural Consolidation [4], currently an activity is ongoing to prepare the ATB for the future in terms of upcoming avionics standards and technologies and also to ensure a sustainable and maintainable system. This is leading to a new development called the End-to-End Avionics System Test Bench (E2E-ATB) which represents the evolution of the ATB. In particular in the technical domain improvements are foreseen in the area of the:

- **Overall Architecture** taking into account the Simulation Model Portability (SMP) [5] and the Space Simulation Reference Architecture (SSRA) [6],
- **Deployment approaches** based on a Conceptual Data Model (CDM) of the ATB, corresponding Simulation System Database and a Configurator Tool,
- **Implementation** and **Regression Testing**,
- **Documentation**

One of the main goals is to obtain an E2E-ATB system in which, by means of a Configurator Tool based on a knowledge-base at any moment during the project's lifecycle an up to date configuration can be retrieved and instantiated. This requires both data as well as simulation models to be stored in a consistent way in the Simulation System Database. This is schematically depicted in Figure 3.

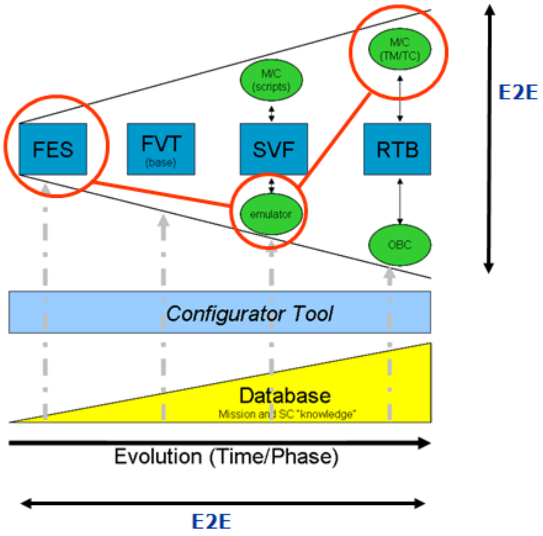


Figure 3: E2E ATB Deployment Approach

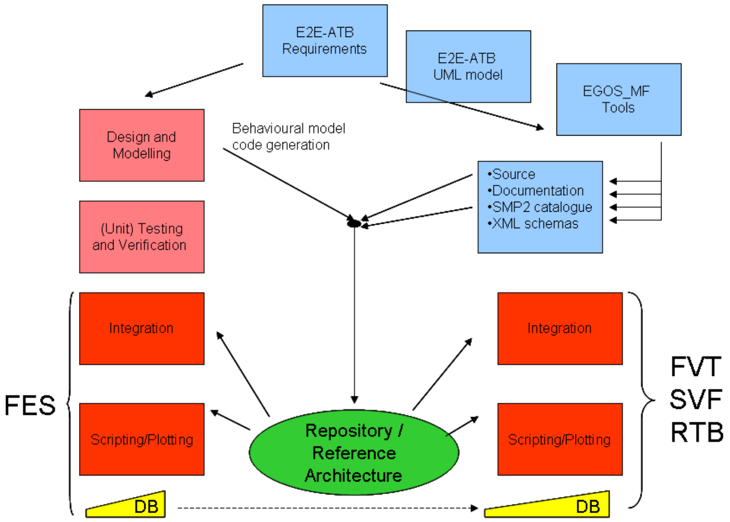


Figure 4: E2E ATB Simulation Model Design and Implementation approach

In this way consistency amongst the different configurations is supported with the advantage that e.g. also in later project-phases an up-to-date FES can be instantiated.

Another main goal in the E2E ATB is the usage of UML diagrams for capturing the design (based on software specifications) of for example the simulation models, but at the same time taking advantage of the modelling capabilities of the Mathworks® product family. This leads to the need to merge two conceptually different models (depicted in Figure 4). On top of this, the SMP2 "standard" and SSRA are used to ensure portability between different simulation environments and platforms.

Especially during the testing and verification activities of the different ATB configurations, in many cases "intermediate" configurations were established that turned out to be very useful in order to bridge and explain the (differences in) results between the different "formal" configurations. One of these "intermediate" versions is the so-called SVF-Lite, schematically depicted by the red ellipse in Figure 3.

In case of the SVF-Lite it was believed that, next to contributing to the ATB verification process, this configuration has substantial use cases in the areas of:

- Functional verification of the Onboard Software (OBSW),
- Performing re-assessment of engineering margins,
- Performing re-assessment of feasibility and performance parameters as part of shadow engineering in specific cases.

Specifically for FDIR, which is defined on system level and detailed on subsystem or equipment level, and therefore spread over many areas, it is believed that the SVF-Lite can contribute in providing a tool to probe, define and assess failure scenarios.

The SVF-Lite is able to run the OBSW Application Software part, connected to a FES-type of simulator. Next to this, a Monitoring and Control System is used based on the corresponding TM/TC definitions for the OBSW (e.g. direct reuse of the SCOS2000 MIB files). This allows to (re-)run the Mission Control test scripts. The direct reuse of the FES-type simulator removes the porting efforts. E.g. the need to comply to modelling rules in order to convert the models is eased as well as interfacing and scheduling issues after porting. By cross-compilation of the Basic Software (BSW) part of the OBSW, the system is capable of running faster than real time at the expense of timing-accuracy. It should be noted that the focus of the SVF-Lite is on the functional verification/validation of system level critical issues, meaning that hardware/real-time aspects are not of prime interest. Formal validation of the complete OBSW is not foreseen. In the next section the implementation of the SVF-Lite, as instantiated in support of the ESA SGEO team will be described.

## ESA SGEO SVF-LITE

The aim of the ESA SGEO SVF-Lite configuration is to provide **complementary** and **internal** support to **the SGEO ESA project team** for the purpose of **flight OBSW functional verification and validation**, especially in the area of **FDIR** on **System** and on **AOCS** level. The ESA SGEO SVF-Lite will **not** replace the SVF as built by Industry (OHB). Instead however the implementation efforts on the ESA SGEO SVF-Lite give **unmatched insight** and **review capabilities** on the SVF as built by Industry.

The SVF-Lite is built to be able to fulfil the following goals:

- to run the unaltered ASW.
- to (re-)run the original test cases from industry without any manual modifications to the test-scripts.
- to generate / develop new test cases and test scripts that can be run at the industry facilities.
- to execute in faster than real time mode
- to directly re-use the FES, i.e. both Simulation Models as well as infrastructure (scripts, plotting).

Based on the architectural concept from the ETM-10-21 [1] the SGEO SVF-Lite architecture can be further detailed as shown in Figure 5. The architecture comprises of the following 5 components: **Monitoring and Control System (MCS), OBC Simulator Component, Simulator (DEES)** and **3D Visualisation** and the **SVF-Lite infrastructure (SYNTID)**. These are further detailed below.

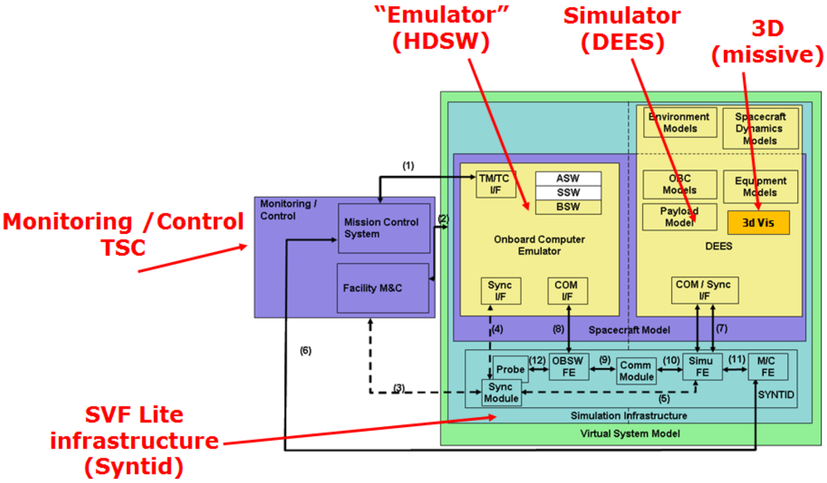


Figure 5: ESA SGEO SVF-Lite architecture and main components

The **MCS** is the main interface for the user to monitor and control the SVF-Lite. The COTS product CMDVS Test Sequence Controller (TSC) is used for this purpose. It has been developed by TERMA and Satellite Services BV, in order to provide a lightweight MCS. The TSC allows the user to connect, observe and interact with a system under test either in real time or for data analysis and replay. Its main goal is to allow automated testing using a test sequence scripting language (e.g. uTope). TSC uses the same database layout (the "MIB" files) to define the TM and TC as used in European spacecraft missions for mission operations and system level checkout. Not only the Ground/Space TM/TC are observed and inserted, also the SCOE commanding (e.g. Simulator Failure Injection) is done from the TSC. Next to this, in order to support the SVF-Lite concept, TERMA has introduced a new feature to TSC, **allowing to support faster than real-time simulations**. This means that TSC is able to operate not only on the basis of the host-clock time, but as an alternative, on the basis of an external time source (e.g. OBT), provided as a UDP packet. Figure 6 shows a screenshot of the TSC as used for the ESA SGEO SVF-Lite.

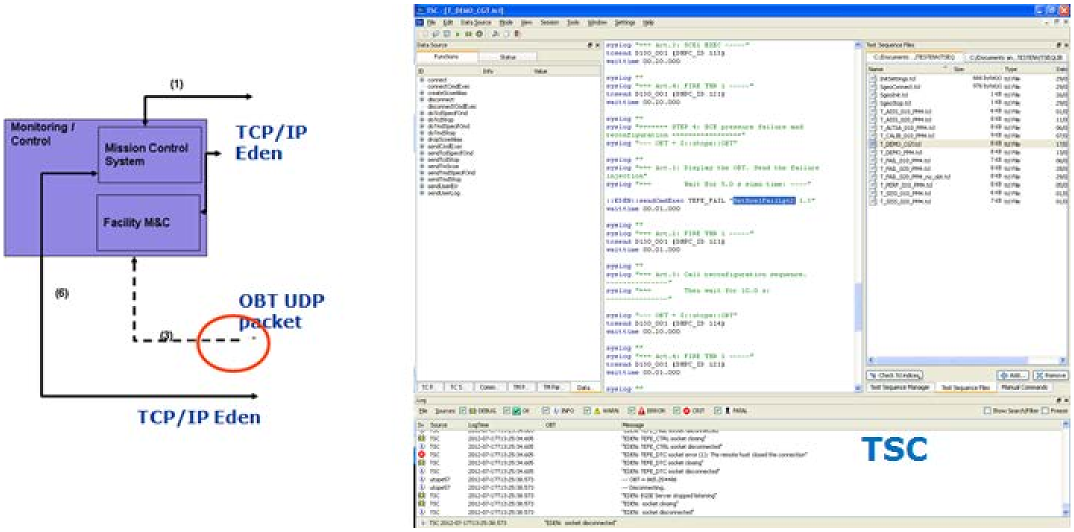


Figure 6: ESA SGEO SVF-Lite TSC usage

The **OBC Simulator Component** is the component responsible for hosting the OBSW, executing the OBSW, and providing interfaces to the rest of the SVF. Figure 7 gives the overview of this component.

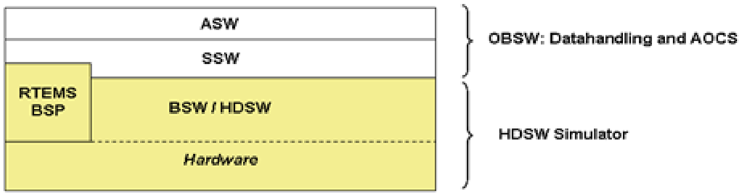
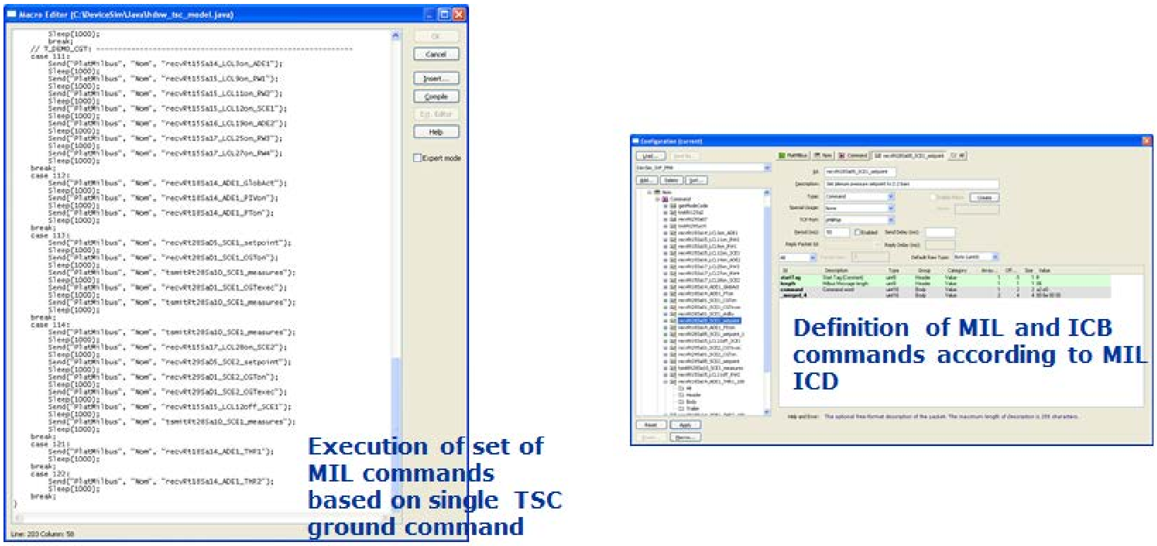


Figure 7: OBC Simulator Component (1/2): implementation based on HDSW Simulator and ASW.

The OBC Emulator Component is mainly composed of the Hardware Driver SoftWare Simulator (HDSW Simulator) from RUAG Space that replaces the BSW and the underlying hardware, including processor. The ASW and SSW layers are built together with the HDSW Simulator into a single executable. Currently this is done using a Sparc Sunblade 100 host platform.

The HDSW Simulator does not provide a cycle based simulation (e.g. as in classical TSIM-based emulators), but only a functional model of the BSW. Following this approach, the SVF-Lite is expected to support system level analysis with significant performance improvements. The HDSW Simulator is estimated to run 10-20x faster than real time.

In order to prepare the different components of the SVF-Lite before a full version of the ASW is available, the Device Simulator Software [8] from Space Systems Finland (SSF) was used. The Device Simulator replaces the HDSW simulator and the ASW in terms of reception and handling of the interfaces. Screenshots of the Device Simulator usage for ESA SGEO SVF-Lite are shown in Figure 8.



```
        Sleep(1000);
        break;
    // T_DEMO_CGT: ----------------------------------------------------------------
    case 111:
        Send("PlatMilbus", "Nom", "recvRt15Sa14_LCL3on_ADE1");
        Sleep(1000);
        Send("PlatMilbus", "Nom", "recvRt15Sa15_LCL9on_RW1");
        Sleep(1000);
        Send("PlatMilbus", "Nom", "recvRt15Sa15_LCL11on_RW2");
        Sleep(1000);
        Send("PlatMilbus", "Nom", "recvRt15Sa15_LCL12on_SCE1");
        Sleep(1000);
        Send("PlatMilbus", "Nom", "recvRt15Sa16_LCL19on_ADE2");
        Sleep(1000);
        Send("PlatMilbus", "Nom", "recvRt15Sa17_LCL25on_RW3");
        Sleep(1000);
        Send("PlatMilbus", "Nom", "recvRt15Sa17_LCL27on_RW4");
        Sleep(1000);
    break;
    case 112:
        Send("PlatMilbus", "Nom", "recvRt18Sa14_ADE1_GlobAct");
        Sleep(1000);
        Send("PlatMilbus", "Nom", "recvRt18Sa14_ADE1_PIVon");
        Sleep(1000);
        Send("PlatMilbus", "Nom", "recvRt18Sa14_ADE1_PTon");
        Sleep(1000);
    break;
    case 113:
        Send("PlatMilbus", "Nom", "recvRt28Sa05_SCE1_setpoint");
        Sleep(1000);
        Send("PlatMilbus", "Nom", "recvRt28Sa01_SCE1_OGTon");
        Sleep(1000);
        Send("PlatMilbus", "Nom", "tsmitRt28Sa10_SCE1_measures");
        Sleep(1000);
        Send("PlatMilbus", "Nom", "recvRt28Sa01_SCE1_OGTexec");
        Sleep(1000);
        Send("PlatMilbus", "Nom", "tsmitRt28Sa10_SCE1_measures");
        Sleep(1000);
    break;
    case 114:
        Send("PlatMilbus", "Nom", "tsmitRt28Sa10_SCE1_measures");
        Sleep(1000);
        Send("PlatMilbus", "Nom", "recvRt15Sa17_LCL28on_SCE2");
        Sleep(1000);
        Send("PlatMilbus", "Nom", "recvRt29Sa05_SCE2_setpoint");
        Sleep(1000);
        Send("PlatMilbus", "Nom", "recvRt29Sa01_SCE2_OGTon");
        Sleep(1000);
        Send("PlatMilbus", "Nom", "recvRt29Sa01_SCE2_OGTexec");
        Sleep(1000);
        Send("PlatMilbus", "Nom", "recvRt15Sa15_LCL12off_SCE1");
        Sleep(1000);
        Send("PlatMilbus", "Nom", "tsmitRt28Sa10_SCE1_measures");
        Sleep(1000);
    break;
    case 121:
        Send("PlatMilbus", "Nom", "recvRt18Sa14_ADE1_THR1");
        Sleep(1000);
    break;
    case 122:
        Send("PlatMilbus", "Nom", "recvRt18Sa14_ADE1_THR2");
        Sleep(1000);
    break;
}
```

Execution of set of MIL commands based on single TSC ground command

Definition of MIL and ICB commands according to MIL ICD

Figure 8: OBC Simulator Component (2/2): implementation based on Device Simulator

The advantage of this two-step approach is that also test scripts can already be developed and executed obtaining results that can be used as reference when executing the real ASW (the unit under test).

The **Dynamics, Environment, and Equipment Simulator (DEES)** component is a Matlab/Simulink-based full representative closed-loop dynamic spacecraft simulator including all models of the environment, of the spacecraft dynamics and the relevant spacecraft equipment. Via a TCP/IP link it is connected to a 3D visualisation for intuitive comprehension of results. This is shown in the following figures. The DEES includes a DEES database (DEES DB), currently implemented in Excel. The DEES DB is used for parameterisation, interface definition (between models as well as the interface to the spacecraft-bus), failure-injection definitions and automated plotting definition.

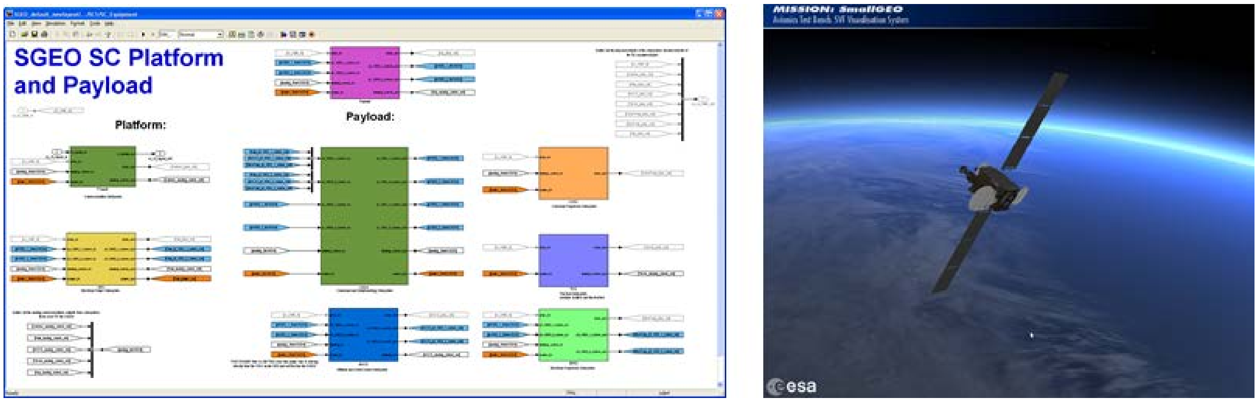

SGEO SC Platform and Payload
Platform:
Payload:
MISSION: SmallGEO
Avionics Test Bench SVF Visualisation System
esa

Figure 9: SGEO SVF Lite DEES architecture with layout similar to the SGEO platform layout and the 3D visualisation

The **Synchronize and Transmit Internal Data (SYNTID)** component is in charge of driving the execution of i.e. DEES and the HDSW Simulator in a synchronized manner, of exchanging the data between these components in a consistent way, and for debugging / logging of the state of different components. The SYNTID also includes a probe mechanism that is used as a sniffer of the bus traffic.

**Synchronisation** is achieved by means of specific messages sent by the SYNTID towards the HDSW-simulator and the DEES, triggering the next 10Hz step of both the HDSW Simulator and DEES. The SYNTID also extracts the 10 Hz broadcast Mil1553 message from the HDSW-simulator, in order to provide the required time source to the TSC via a UDP packet. The DEES acknowledge is required, because the (not auto-coded) Matlab/Simulink environment cannot guarantee real-time performances at all simulation steps. Significant jitters are expected, depending on the simulation scenario (especially during failure injections into the DEES). On the other hand, the HDSW acknowledge is a nice-to-have. Although it would definitely maximize the simulation acceleration factor, its absence does not prevent from achieving faster-than-real-time.

The other main functionality implemented in the SYNTID is **internal data exchange.** Data exchange (i.e. Mil1553 and ICB data) is achieved by means of a double buffer. The double buffer mechanism allows the SYNTID to answer immediately to data request messages coming from the HDSW Simulator. With that respect, the SYNTID acts as an On Board Software Front End, working continuously, as a separate thread, answering HDSW-simulator requests and filling the appropriate buffer with commands that the DEES will take into account in the following 10 Hz. cycle, i.e. after the buffer swapping.

## FIRST RESULTS, LESSONS LEARNED and CONCLUSIONS

The ESA SGEO SVF-Lite as currently deployed at ESTEC includes the full infrastructure (i.e. TSC, SYNTID, OBC Simulator component and DEES).The DEES contains the environment models and most of the SGEO platform Remote Terminal simulation models.

Because the currently available version of the SGEO ASW only provides limited functionality two SVF-Lite configurations are currently in use:

- A configuration running the HDSW Simulator (including the real ASW).
- A configuration where the HDSW Simulator is replaced by the SSF Device Simulator.

The first configuration is the target configuration and will be used when more complete versions of the ASW become available. The latter configuration allows to prepare and validate complex simulation scenarios and to obtain the needed reference results. For example, a rather complex de-tumbling scenario, including a pressurization failure and resulting simulated OBSW reconfiguration of the propulsion subsystem according to the OBSW specification, has been run successfully using the second configuration.

During the development of the ESA SGEO SVF-Lite a number of challenges have impacted the design and implementation of the SVF-Lite:

- Because the SGEO ASW is not endianity-neutral, it was decided to deploy the SVF-Lite facility on two different machines: a dedicated Big-Endian machine to execute the HDSW simulator and ASW executable, and a Little-endian host for the TSC, the SYNTID and the DEES. The use of the TCP/IP based interfaces enables a multi-host system. As an alternative for the Little-endian host. A Sparc emulation (e.g. QEMU) is still considered.
- The faster-than-real time requirement has led to the adaptation of the TSC timing to be driven by an external OBT UDP packet.
- On the DEES side, the direct usage of Simulink environment for execution (auto-coding is not currently foreseen) increases the openness of the platform, enabling lightweight script injections, direct investigation of the simulation models when required, etc. However, this choice introduces significant jitter on the DEES execution times. This implies that particular care is required when designing the interfaces of the DEES component with the rest of the facility, e.g. in order to ensure proper synchronization.
- Furthermore on the DEES side, it was intended to directly reuse a number of Simulink models which were conceived for FES simulators. It is well known [7] that Simulink models developed for FES simulators are not always one-to-one suitable for SVF simulators, and performance penalties must be considered.

Currently the SVF-Lite executes 5 times faster than real-time. This is quite a promising acceleration factor, taking into account that no particular optimization has been performed yet.

In the current configuration, where either the OBSW is stubbed or only implements limited functionality, the performance bottleneck is the DEES. First profiling activities on the DEES have already identified bottlenecks which can easily be removed, so that the acceleration factor can be increased to at least 10x. Next to these “easy-to-fix” DEES improvements, a number of possible modifications are identified in the field of simulation modelling, as mentioned above.

A number of lessons-learned are currently being collected. These pave the way not only for improvements of the SVF-Lite concept, but also for harmonization of simulation models and of model-based System Simulation Facilities in general. The lessons-learned emphasize the importance of the following aspects:

- Establishment of a consistent and (even more) centralized data management, e.g. TM/TC interface, simulation model parameters, Remote Terminal data, etc.
- The definition of the granularity of the simulation models as appropriate for the actual use cases: vectorisation vs. atomic models; benefits and limitations of Simulink bus objects.
- Benefits, limitations and performance of Simulink blocks, e.g. Embedded Matlab functions, different type of integrators.
- The choice of the Simulink solver (e.g. discrete solver vs. continuous) for the problem at hand (e.g. FES vs. SVF). This choice has important consequences, e.g. in terms of Simulink guidelines, synchronisation and performances.

It is believed that a full understanding of these topics is important when undertaking the desired strategic next steps activities, such as:

- The definition of library of models, and of their applicability in the project life-cycle.
- The improvement of simulator reference architectures.

In conclusion, the SVF-Lite configuration leverages from different concepts of the FES, the traditional SVF and the RTB configurations, in order to support system level analysis with the real OBSW in the loop. Also it allows for the usage of the real TM/TC commands and support direct re-use of the FES simulation models. Because hardware/real-time aspects are not the prime interest of the facility, the configuration can execute several times faster-than-real-time (the aim is an acceleration factor of 20).

On the DEES side, the direct usage of Simulink models increases the user-friendliness of the platform, i.e. all types of operations that are typically used in FES facilities. In case properly facilitated, direct use of the FES pre-serves the SVF developer from "autocoding", "interfacing" and "scheduling" issues.

As it is the case in all innovative approaches, the SVF-Lite development implies several challenges. Facing and solving such challenges have provided material to trigger important discussions, ranging from components reuse, to Model-based Design, and Monitoring & Control systems. Such research may pave the way for significant improvements of the core components of the System Simulation Facilities.

The results of this activity are clearly not limited to the SGEO project although the instantiation efforts at the same time resulted in very detailed insight and reviewing of the industrial SGEO SVF.

## REFERENCES

[1] ECSS-ETM-10-21: Space Engineering, System Modelling and Simulations

[2] Q. Wijnands, "The SVF-Lite Configuration in the End to End System Avionics Testbench Concept", DASIA 2011.

[3] J. Rebelo, Q. Wijnands, R. Blommestijn, F. Pace, "The Avionics System Test Bench, Functional Engineering Simulator: New Developments in Support of Mission and System Verification", 11th SESP (Simulation and EGSE for Space Programmes) Workshop, 2010.

[4] F. Pace, V. Barrera, N. Lindman, Q.Wijnands, "Evolving Infrastructure for Avionics Verification and Validation", 11th SESP (Simulation and EGSE for Space Programmes) Workshop, 2010.

[5] SMP 2.0 Handbook, EGOS-SIM-GEN-TN-0099, Issue 1, Revision 2, 2005/10/28, ESA/ESOC, Darmstadt, Germany

[6] A. Walsh, "The Spacecraft Simulator Reference Architecture", 11th SESP (Simulation and EGSE for Space Programmes) Workshop, 2010.

[7] C. Leorato, E. Guidolotti, D. Segneri, "Reusing Legacy Code in a Simulink Environment for Real-Time Industrial Simulations: Techniques and Prototype Support Tools", DASIA 2005.

[8] http://www.ssf.fi/pages/index.php?fid=16&pid