SWARMSIM – The first fully SMP2 based Simulator for ESOC

11th Int. WS on Simulation & EGSE facilities for Space Programmes
SESP 2010

28-30 September
at ESTEC, Noordwijk, the Netherlands

Peter Fritzen[(1)], Daniele Segneri[(1)], Max Pignède[(2)]

[(1)]*VEGA*
Europaplatz 5, D-64293 Darmstadt, Germany
Email: Peter.Fritzen@VEGA.de, Daniele.Segneri@VEGA.de

[(2)]*ESOC/ESA*
Robert-Bosch-Straße 5, D-64293 Darmstadt, Germany
Email: Max.Pignede@esa.int

INTRODUCTION

ESOC has started the deployment of a new generation of operational simulators in the context of the Swarm mission, a constellation of three satellites planned to be launched in 2012. Compared to previous simulators developed by ESA, it is unique in a number of ways:

- It natively implements the Simulation Model Portability 2 (**SMP2**) Standard;
- It is the first simulator based on the ESOC Spacecraft Simulator Reference Architecture (**REFA**);
- It has been produced using a model driven development process using the EGOS Modelling Framework (**EGOS-MF**).

This paper illustrates the three above aspects and demonstrates that SMP2 can be used successfully in the context of operational simulators for ESOC. Further, the paper clearly proves how the Reference Architecture applied in simulators reduces development effort & cost. The paper is concluded with lessons learned and some proposals for future improvements.

OVERVIEW

In October 2005, Issue 1.2 of SMP2 was released. For an application in ESOC simulators, the SIMSAT simulation infrastructure was enhanced to support SMP2 (in addition to SMI, also called SMP1). Then, the ESOC Generic Models were ported from SMI to SMP2. These models are reused across operational simulators for thermal and electrical modelling, environment and dynamics, and for TM/TC encoding and decoding. Before starting with the development of an operational simulator, an SMP2 based reference architecture for spacecraft simulators was designed (making use of the Generic Models). Finally, the Swarm Simulator was developed using all these technologies provided by ESOC.

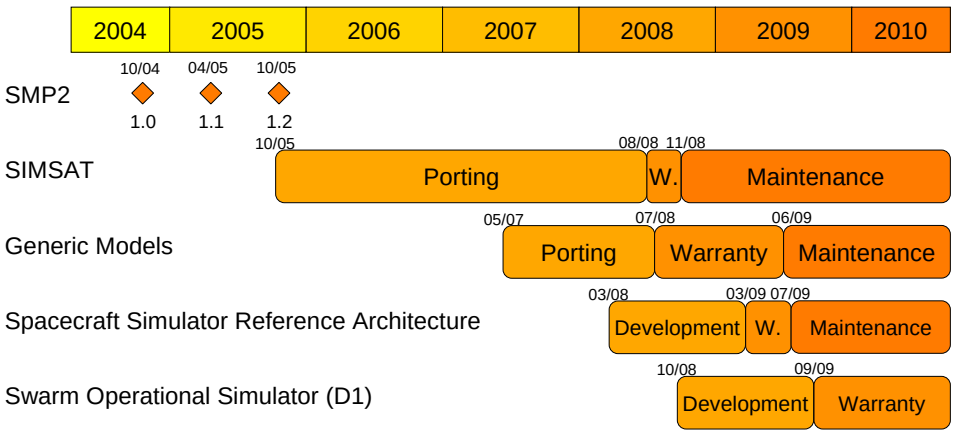

Figure 1. Migration of the ESOC Infrastructure from SMP1 (SMI) to SMP2

THE SIMULATION MODEL PORTABILITY 2 (SMP2) STANDARD

The SMP2 standard has been developed to enforce a component based, object oriented development approach, and to facilitate the use of modern programming language (namely C++). While the Simulation Model Interface (**SMI**) of SMP1 is a C API (which mainly focused on the interfaces between the "Simulation Environment" and the "Simulation", see Figure 2 below), SMP2 strictly requires object orientation, and only provides a C++ mapping. The three types of components that make up an SMP2 simulation are the simulation environment itself, the simulation services and the models within the Simulation. These components shall only talk to each other via well-defined SMP2 interfaces.

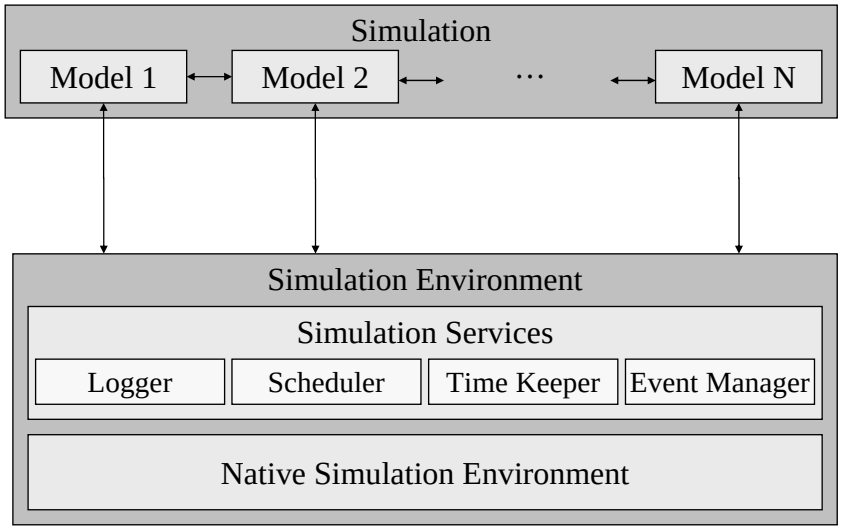

Figure 2. Communication of SMP2 Components via well-defined Interfaces

In the **Swarm Simulator**, most models are native SMP2 models, i.e. they have been designed and implemented starting from a complete design in an SMP2 Catalogue. This approach is different from a porting of existing models as e.g. done for the Generic Models. Some statistics about the SMP2 models used in the various subsystems are shown in Table 1.

Table 1. Number of SMP2 classes and models per subsystem, divided into different model types

	Native Models	Wrapper Classes		
		C++	F77	ADA
Generic Models	**110**	**18**	**34**	**0**
Common math library	23			
Environment			34	
Dynamics	11			
Encoding/Decoding	76			
Thermal Network		8		
Electrical Network		10		
Swarm Simulator	**144**	**3**	**0**	**1**
Generic	15			
Data Handling	36			1
Data Links	4			
Attitude Orbit Control	20			
Reaction Control	11			
Radio Frequency Control	8			
Electrical Power	26			
Thermal Control	5			
Space Ground Interface		2		
Payloads	12			
Utilities	7	1		

It can be seen that the **Generic Models** (which have been ported from SMP1 to SMP2 earlier) have a significant share of models which are wrappers of existing models written in another programming language (mainly C++) whereas for the **Swarm Simulator**, only the Emulator model of the Data Handling System has been developed as a wrapper around existing Ada code. Further, the Space Ground Interface talks to an existing, CORBA based interface to the ESOC Ground Models. Hence in total, the simulator can be considered truly SMP2 based.

THE SPACECRAFT SIMULATOR REFERENCE ARCHITECTURE (REFA)

The objective of the Spacecraft Simulator Reference Architecture (REFA) is to identify and integrate, using SMP2, a reference spacecraft simulator architecture which can be used as the basis for future operational simulators. This is essential to achieve shorter (and therefore more cost-efficient) spacecraft simulator development cycles by means of extensive reuse by relying on a common architecture. This relies on the specification of interfaces between the various spacecraft subsystems and on the identification of models which can be developed in a generic fashion, thus providing the start of the actual development of an operational simulator.

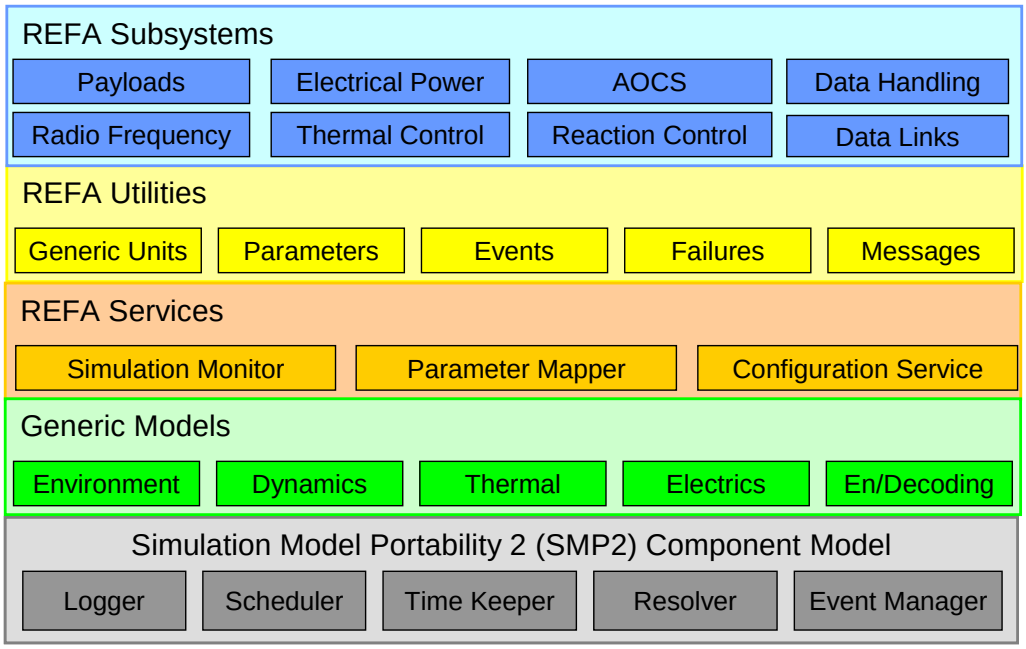

Figure 3. Logical Model of Reference Architecture based on SMP2 and Generic Models

In the Swarm Simulator, all models are derived from models defined by REFA. As a minimum, models reuse generic concepts for logging, tracing, failures, events and TM/TC parameters. This ensures consistency of such mechanisms across all subsystems of the simulator. Most models are derived from subsystem specific models and implement interfaces defined by the Reference Architecture. In a few cases, these REFA interfaces have been extended for mission specific needs. Only in few other cases, Swarm specific interfaces had to be defined.

Table 2. Number of SMP2 models and interfaces per subsystem, divided into their base model/interface

	Interfaces		Models	
	New	Derived	New	Derived
Swarm Simulator	**34**	**13**	**42**	**106**
Generic	2	4		15
Data Handling	24	3	18	19
Data Links				4
Attitude Orbit Control	3	1	7	13
Reaction Control				11
Radio Frequency Control	1			8
Electrical Power	3	5	8	18
Thermal Control			2	3
Space Ground Interface				2
Payloads	1			12
Utilities			7	1

It can be seen that for most subsystems the Swarm Simulator defines few interfaces for inter-model communication, because most of the communication is via existing REFA-defined interfaces. The only exception from this rule is the Data Handling System (this is due to the obvious specific nature of the on-board hardware, typically different across satellites). Equally, the majority of models used in the simulator are derived from an existing REFA base model. Swarm specific models are mainly in the Data Handling System and the Utilities.

THE EGOS MODELLING FRAMEWORK (EGOS-MF)

The SMP2 standard has been developed with a model driven development process in mind. The Simulation Model Definition Language (**SMDL**) allows implementation of a process where models are fully designed in UML and skeleton source code can be generated from this design. Such a process has been implemented by ESOC in the EGOS Modelling Framework and the SIMSAT Model Integration Environment (**MIE**), which are currently merged into the Universal Modelling Framework (**UMF**). This process not only generates skeleton source code for the model operations defined in the design, it as well auto-generates SMP2 compliant C++ code for e.g. publication and dynamic invocation. Further, from the same source, documentation can be generated, which is hence guaranteed to be fully consistent with the source code.

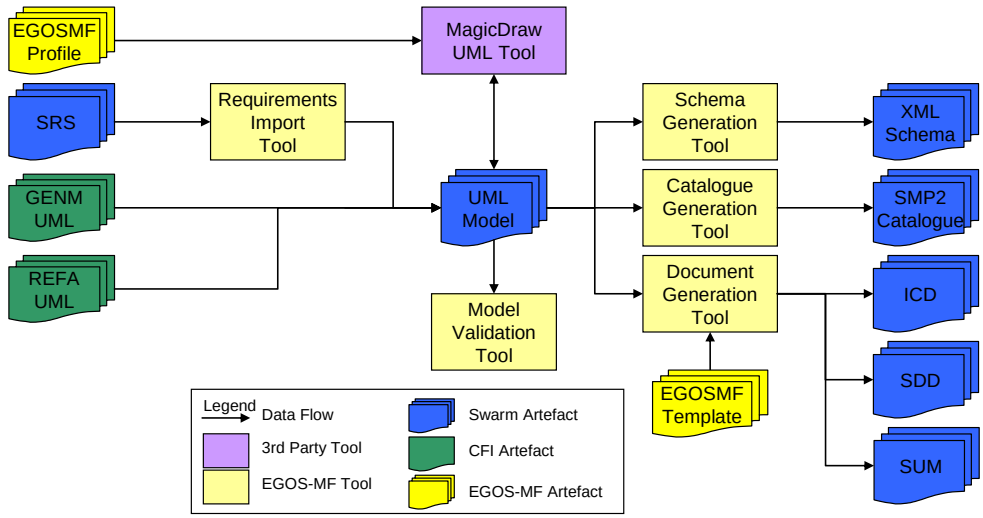

Figure 4. Model Driven Software Development with a centralised UML Model

In the Swarm Simulator, the existing process has been tailored to project specific needs: The C++ code templates have been updated to generate code compliant with the BSSC Coding Standards and the document templates have been updated to comply with the applicable ECSS standard. While the high-level summary of the design is written by hand, detailed design documentation, interface control documents and reference sections of the user manual are generated from the UML design. This has not only significantly reduced time and effort to write and update documentation, it as well improved the quality, completeness and consistency of the relevant documents.

EXAMPLE MODEL: GPS RECEIVER INTERFACE TO DATA HANDLING SYSTEM

As an example, we demonstrate the design and implementation of the GPS Receiver interface (from the Attitude and Orbit Control System) to the Data Handling System of Swarm.

Architectural Design

First, a Swarm specific interface for a UART connection has been defined as an extension to the existing Bus Transaction interface of the Reference Architecture.

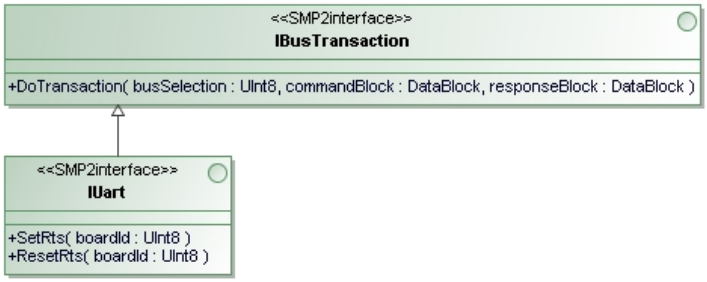

Figure 5. Definition of a Swarm specific UART interface extending an existing REFA interface

The GPS to DHS Interface model is then defined as a specific DHS Interface model (which is a REFA defined base model). As such, it inherits the interfaces to receive packets, and to transmit them to a packet decoder. For Swarm, specific interfaces to the UART have been added, as well as some fields for the internal model state (see in the "GpsDhsInterface" model below). For the implementation, all methods defined by the (REFA and Swarm) interface need to be implemented by this model, even those that are inherited from the REFA base model: the reason is that REFA provides a reference architecture, but no reference implementation.

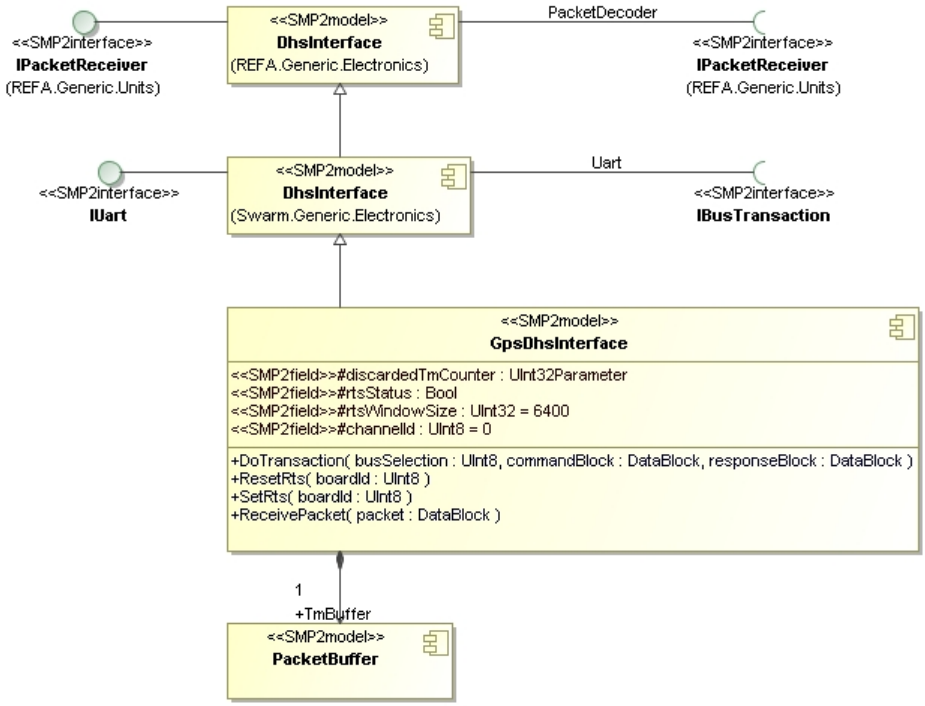

Figure 6. Design of the GPS Receiver interface to the Data Handling System

With the EGOS-MF Catalogue Generation Tool (see Figure 4 above), the information is translated into an SMP2 Catalogue. This is then translated into C++ code files `GpsDhsInterface.h` and (skeleton) `GpsDhsInterface.cpp` using the SIMSAT MIE Code Generator. The implementation of the operations has to be done manually.

```
// --OPENING ELEMENT--GpsDhsInterface::ResetRts--
        /// Resets the Request To Send.
        /// @param boardId the identifier of the board sending the signal.
        void GpsDhsInterface::ResetRts( const ::Smp::UInt8 boardId )
        {
            // MARKER: OPERATION BODY: START
            // INSERT HERE OPERATION BODY
            // MARKER: OPERATION BODY: END
        }
// --CLOSING ELEMENT--GpsDhsInterface::ResetRts--
```

Figure 7. Example of skeleton code generated from UML design

In a second implementation file, named `GpsDhsInterfaceSmp.cpp`, SMP2 specific code for state transitions, publication and dynamic invocation is generated, taking the description from the UML model into account. For most models (including the GpsDhsInterface model), no manual modifications to this code are required.

```
// --OPENING ELEMENT--::Swarm::Aocs::GpsDhsInterface/GpsDhsInterfaceSmp.cpp::Publish--
        /// Publish fields, operations and properties of the model.
        /// @param receiver Publication receiver.
        void GpsDhsInterface::Publish( ::Smp::IPublication* receiver ) throw (
::Smp::IModel::InvalidModelState )
        {
            // Call base class implementation first
            ::Swarm::Generic::Electronics::DhsInterface::Publish( receiver );

            // Publish field discardedTmCounter
            receiver->PublishField( "discardedTmCounter", "This counter is incremented
each time a TM packet is discarded due to a full TM buffer.", &discardedTmCounter,
::Generic::Types::Uuid_UInt32Parameter, true, true, false, false );

            // Publish field rtsStatus
            receiver->PublishField( "rtsStatus", "The status of the RTS.", &rtsStatus,
true, true, false, false );
...
```

Figure 8. Part of the auto-generated Publish() operation of the model

Using the EGOS-MF Document Generation Tool, the design can be turned into a WORD document including detailed design documentation. An example is shown in Figure 9 below. Of course, the level of detail of this document is only as good as the description entered into the UML design. Therefore, such an automated documentation process cannot guarantee a high quality – it can only ensure consistency between design, documentation in source code (which is generated from the same UML) and documentation in design documentation.

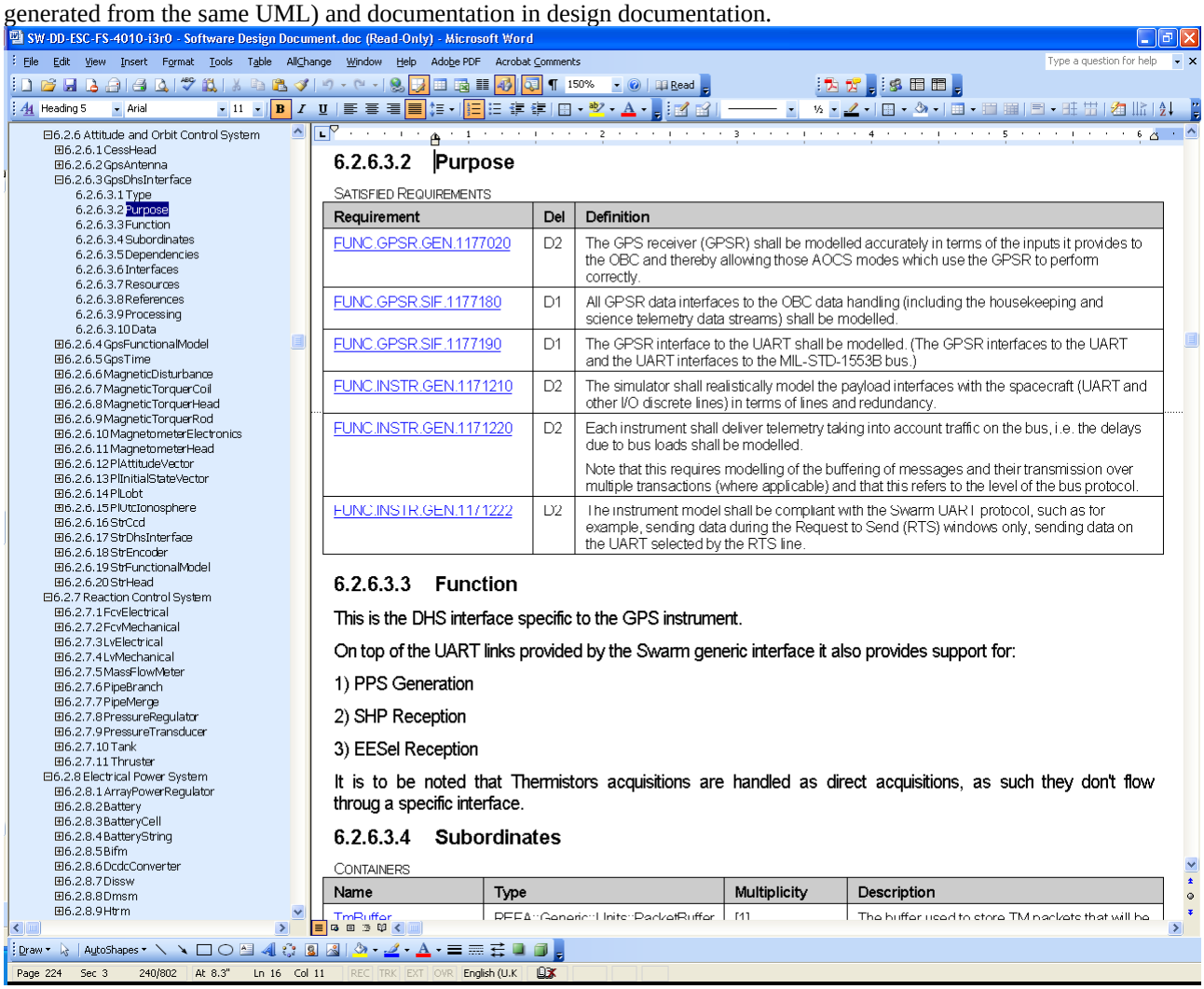

Figure 9. Detailed Design Documentation generated from UML using EGOS-MF

LESSONS LEARNED

Overall, it can be concluded that the application of the SMP2 standard did not have any negative side-effects.

Performance

During the development of SMP2, the use of C++, object orientation and interface based design was seen as a possible performance risk. The Swarm Simulator (which makes heavy use of SMP2 interfaces and events) has demonstrated that this is not the case: Although Swarm is the first mission using the SMP2 implementation of SIMSAT, simulator load time as well as the time it takes to store or restore a breakpoint is at least comparable to existing SMP1 based simulators. As for other operational simulators running the on-board software in a software emulator, run-time performance is dominated by the emulation of the processor, not by the individual simulator models.

Development

Due to the application of new technologies, development of the Swarm simulator started slowly. The effort needed to train the team members on the use of UML, EGOS-MF, SIMSAT MIE and SMP2 was considerably higher than initially assumed. However, during the lifetime of the project this was compensated by a more efficient development phase, and especially by a highly automated delivery process. Using an automated build & test environment together with auto-generation of large parts of the documentation, a delivery can be made almost immediately, and with little overhead. Therefore, such a process is especially well suited for an incremental development approach (as applied for the Swarm Simulator), or even for an Agile development.

FUTURE IMPROVEMENT

The application of SMP2, Generic Models, REFA and EGOS-MF for the development of an operational simulator has proven to be a cost-effective approach. Despite the positive experiences, some further improvements of the ESOC simulation infrastructure are proposed here, which would increase acceptance of the process both by developers and by end-users of the simulator.

SMP2 Improvements

Most of the shortcomings of the SMP2 standard will be resolved in the proposed ECSS SMP standard. During the Swarm Simulator development, the following limitations were noticed:

- The SMP2 Metamodel does not provide support for the modelling of Simulation Services. Therefore, no proper support from a Code Generator is available for the services included in Generic Models and Reference Architecture.
- The SMP2 Component Model does not allow controlling the visibility of operations (while this is possible for model fields). Therefore, the SIMSAT Code Generator generates code for publication and dynamic invocation for each public operation. It is not possible to limit the operations that shall be exposed to end-users (e.g. for use in JavaScript during run-time).

Generic Models Improvements

The latest Generic Models have been developed by porting the SMP1 based models to SMP2. This has been done before the development of the Reference Architecture has started. This comprises the following limitations:

- Rather than defining common mechanisms e.g. for logging and tracing, the existing (inconsistent) mechanisms of the old Generic Models were maintained and only ported to SMP2. As an example, the Generic Models now contain a Generic Tracing Service – but they do not use it.
- Various generic mechanisms introduced by REFA (especially forcing and failing) are not supported for the Generic Models (which provide custom implementations of forcing and failing). This jeopardises the objective to achieve consistency across all models of a simulator.

Reference Architecture Improvements

Given that Swarm was the first mission applying the Reference Architecture, the concept can be considered successful. However, further improvements are possible to further reduce development time, cost and effort of simulators:

- To date, REFA is an architecture, where only some base classes provide an implementation. Subsystem specific models come without any implementation. Therefore, where sensible, REFA could be enhanced by a reference implementation of models – at least as a starting point for future missions. As long-term perspective, a library of models could be built up from the individual missions based on REFA.
- While it is expected that each mission will have to define mission specific models and interfaces, it should be analysed which of the interfaces and models introduced by Swarm are actually specific to this mission. Where possible, these types should be brought into the Reference Architecture. Examples include a generic Switch interface and model, which was missed in the Reference Architecture, or some of the Utility models implemented for Swarm for testing of TC Injection and TM Reception.

Acknowledgements

Nuno Sebastiao and Vemund Reggestad, ESOC Engineering Department, European Space Agency, for the founding and definition of REFA.
Peter Ellsiepen, VEGA, for the creation and implementation of EGOS-MF for the European Space Agency.

References

[1] SMP 2.0 Handbook, EGOS-SIM-GEN-TN-0099, issue 1.2, 28/10/2005
[2] SMP 2.0 Metamodel, EGOS-SIM-GEN-TN-0100, issue 1.2, 28/10/2005
[3] SMP 2.0 Component Model, EGOS-SIM-GEN-TN-0101, issue 1.2, 28/10/2005
[4] SMP 2.0 C++ Mapping, EGOS-SIM-GEN-TN-0102, issue 1.2, 28/10/2005
[5] Generic Models, EGOS-SIM-GENM-SRN-1002, issue 1.4, 25/06/2009
[6] Spacecraft Simulator Reference Architecture, EGOS-SIM-REFA-SRN-1001, issue 1.3, 30/10/2009
[7] SIMSAT, EGOS-SIM-SIM-SRN-1001, issue 2.10, 28/11/2008