# DLR'S VIRTUAL SATELLITE APPROACH

**Schumann, H.[1]; Berres, A.[2]; Maibaum, O.[3]; Röhnsch, A.[4]**

[1] ***German Aerospace Center (DLR)***
***Simulation and Software Technology***
***Lilienthalplatz 7***
***38108 Braunschweig***
***Germany***
***Email: holger.schumann@dlr.de***

[2] ***German Aerospace Center (DLR)***
***Simulation and Software Technology***
***Rutherfordstr. 2***
***12489 Berlin***
***Germany***
***Email: axel.berres@dlr.de***

[3] ***German Aerospace Center (DLR)***
***Simulation and Software Technology***
***Lilienthalplatz 7***
***38108 Braunschweig***
***Germany***
***Email: olaf.maibaum@dlr.de***

[4] ***German Aerospace Center (DLR)***
***Simulation and Software Technology***
***Rutherfordstr. 2***
***12489 Berlin***
***Germany***
***Email: alexander.roehnsch@dlr.de***

## ABSTRACT

With the constitution of a new institute for space systems located in Bremen, Germany, the DLR has taken another important step towards developing complete space systems. To support the system engineering process, the DLR installed a Concurrent Engineering Facility (CEF) which is accompanied by the Virtual Satellite project. This project defines tasks, tools, and technologies for a simulation-based space systems engineering process. The major challenge is the automated generation of an SMP2 simulation, which is supported by following the model based development approach and by using a system component repository containing ready to use SMP2 models. This paper discusses the current status of the project and the gained experiences with modern simulation technologies like the SMP2 standard and related tools (SIMSAT, MOSAIC, etc).

## 1 INTRODUCTION

In October 2008, the German Aerospace Center (DLR) inaugurated the new Institute for Space Systems located in Bremen, Germany. This concentrates the competences in space engineering enabling the DLR to build space systems in-house. Furthermore, a Concurrent Engineering Facility (CEF) was established according to ESA's Concurrent Design Facility (CDF) [1] to offer the very effective approach of concurrent systems engineering. Additionally, the need for a tool supported process for the simulation-based space system development based on a modern and flexible software infrastructure was identified. The Virtual Satellite project aims at the definition of this process and the implementation of the needed infrastructure.

## 2 THE VIRTUAL SATELLITE

The project started in 2007 at the DLR facility Simulation and Software Technology and will be finished end of 2009. The main objectives are:

- Evaluation of system description languages for assisting the simulation-based development process of satellites.
- Definition of a system design model (data model) considering existing ESA approaches and ECSS standards.
- Creation of a central system component repository for the reuse of simulation models in further development phases or projects.
- Support of a rapid system simulation generation considering phase- and discipline-specific views on the system simulation.
- Application of the process in the compact satellite program [2] and integration of the process in the new CEF.

Three areas of work have been identified to achieve these objectives as shown in figure 1: Concurrent Engineering, Simulation Engineering and Automation Environment. The light dark boxes reflect the applied tools, the thick panes are data stores, and the white boxes represent tasks. The arrows roughly illustrate the data flow in the Virtual Satellite process.

In the Concurrent Engineering subprocess, first at all, mission requirements are collected, operation scenarios are defined, and needed system components are identified (1). The System Component Repository is filled with missing components using the Repository Explorer tool, the identified components are assembled to a system model in an integration task using the System Editor tool (2), and the system model is parameterised using a central Data Exchange store of an Integrated Design Model (IDM).

Afterwards, the System Simulator executes the dynamic simulations (3) representing the operation scenarios loaded to the simulator by a Simulation Control tool (4). This tool also collects the simulation results and initiates the evaluation and report generation process (5) considering the expected outputs from the operation scenarios (1). In a following Concurrent Engineering session, the reports can be reviewed (6) and optimisations with respect to System Components or parameters can be carried out (7). With several of these iterations (2-7) the system will be refined through early as well as late development phases.

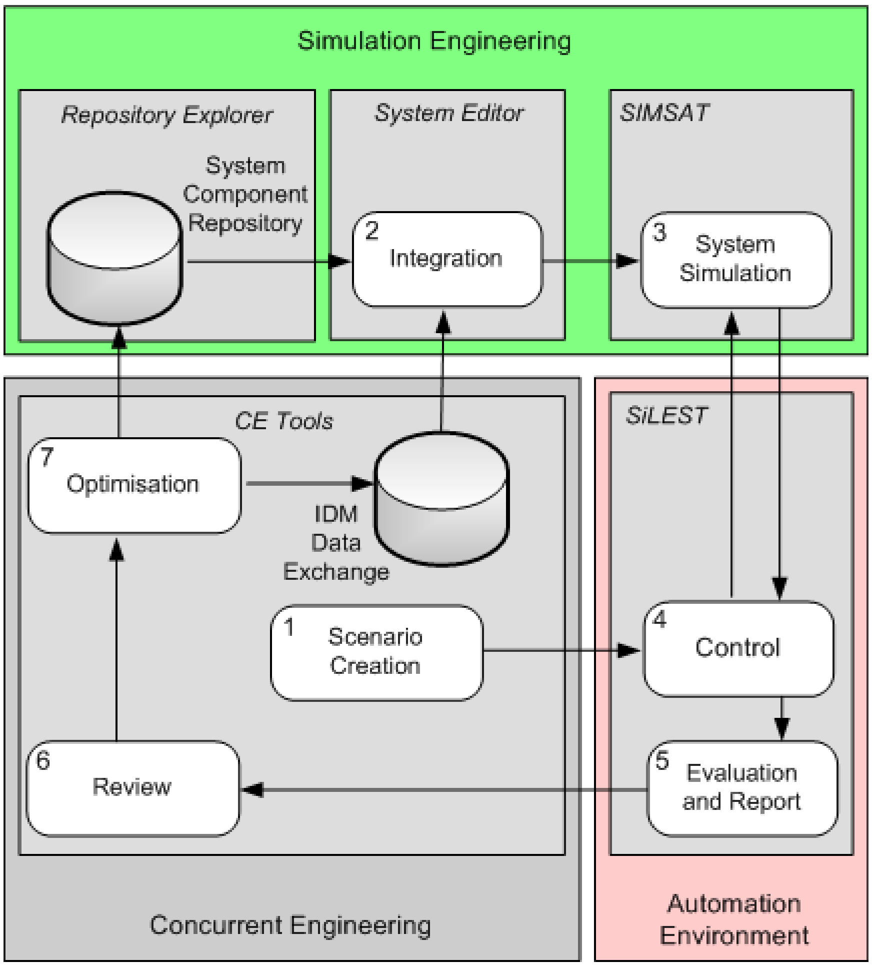

Fig. 1: Overview of the Virtual Satellite process

## 3 IMPLEMENTATION

### 3.1 System Simulator

To increase the rapid simulation generation, it was agreed to use one single simulator for simulating the complete space system instead of coupling several specific simulation tools, which is rather complex. The decision which simulator to choose is difficult, because on the one hand, a future change of this simulator could create much migration costs and on the other, there could occur an unwanted dependency from the supplier of the simulator due to changing simulator versions. That’s why there was the need for a standardised interface between simulation models and the simulation environment and that’s why it was decided to use SMP2.

Beside the SMP2 simulators developed by a few space agencies on its own, there exist two SMP2 simulators for public acquisition: SIMSAT and EuroSim. While EuroSim is a commercial product distributed by Dutch Space (NL), SIMSAT is licensed by ESA and therefore free of charge for DLR to use it within the Virtual Satellite project.

#### *3.1.1 Experiences with SMP2 and SIMSAT*

Using SIMSAT’s Model Integration Environment is a helpful way to understand the SMP2 concepts, because the integrated editors provide only the allowed elements for insertion and there is a good validation function integrated, though the validator messages are not always easy to understand.

Beside some small problems with the user interface, it was possible to define several SMP2 simulations using the very helpful code and Makefile generator. So, a few legacy C++ models using external data files were embedded into an SMP2 wrapper and this was successfully run using SIMSAT.

A further very helpful feature is the code merging tool, because the design files were often changed after implementing the model code. Here is a potential for small improvements seen. The earlier defined folders related to merging should be remembered and it should be allowed to use the same folder for the existing as well as the merged folder (recommended only, if a version controlled folder is used).

For running an SMP2 simulation, SIMSAT's Run-time Environment needs at least a binary containing the implementation (e.g. shared object), an SMP2 assembly and an SMP2 schedule file. Because our System Component Repository stores a binary for every component or subsystem, the ability to use several binaries, several assembly files and several schedule files for running one single system simulation were evaluated. SMP2 provides this ability and SIMSAT was able to realise this successfully.

Related to this composed simulation, the inter model communication abilities are evaluated. Currently, there are problems with pointing to the communication target objects using interface-based and component-based designs. Using the still optional Resolver Service implemented at SIMSAT was easily possible instead.

At next, the ability to control the SIMSAT simulation host as well as to read the simulation results using our Automation Environment will be evaluated.

## 3.2 System Editor

The System Editor should be the central tool for the system engineer while assembling the model of a space system. Related to the functionality, it is the same as an SMP2 Assembly Editor. But it was found out, that the DLR engineers would not like to be directly confronted with the complex SMP2 technology. This is why this System Editor must get a graphical block-oriented editor comparable to Simulink. In the background, there is an SMP2 Assembly generated.

In addition, there is a requirement to have the System Editor as well as other supporting tools integrated into one design or development environment. The editor should be connected to a central project and data management. Also, it should provide different views for every discipline or subsystem as well as remote collaboration to support the Concurrent Design approach.

### *3.2.1 Reconfigurable Computing Environment (RCE)*

A flexible integration framework was developed by our facility called Reconfigurable Computing Environment (RCE) [3] which is well suited to meet most of these requirements. It was created during a project with the ship building industry to create a modern software environment for the early ship design.

The framework is based upon Eclipse and the Open Services Gateway initiative (OSGi) and provides often needed software components like Graphical User Interface, Data Management, Distributed Computing, and so on. Modern concepts like service orientation and bundles (plug-ins) allow adding or removing of functionalities of any application dynamically at run-time.

This framework was successfully approved by the ship building industry and is best prepared to meet the requirements of the Virtual Satellite tools and the main objectives of the Open Concurrent Design Server (OCDS) project [4].

### *3.2.2 Data Model*

At the DLR facility Simulation and Software Technology, a study has been performed to evaluate useful data models that support space craft development in early as well as later development phases [5][5].

Within ESA's Concurrent Design Facility (CDF), the Integrated Design Model (IDM) is used to be applied in early design phases and feasibility studies.

The Institute of Astronautics of the Technical University of Munich (TUM) has been researching many years on the field of Concurrent Design. The data model developed at this institute is object-oriented and very modern but not that much approved as the IDM is.

Within the study, the flexible data model of the TUM was extended to support dynamic simulations of later phases and further, the possibility to import ESA's IDM was ensured. Also the ongoing ECSS standardisation of the IDM was already considered. Finally, the new data model was implemented using the data management framework of RCE and approved using an Excel bundle running the IDM within RCE.

### *3.2.3 Model to Model Transformation*

Currently, an easy Domain Specific Language (DSL) is developed in cooperation with DLR engineers to achieve a system description language on design or assembly level. The Eclipse Modeling Framework (EMF) is used for the definition of the DSL and the Graphical Modeling Framework (GMF) is used to generate a block-oriented editor for this DSL. Using the openArchitectureWare (oAW) technology, the translation of the DSL into an SMP2 Assembly or to a platform specific language (e.g. to Simulink templates) is possible. This follows the Model Driven Architecture (MDA) approach.

## 3.3 System Component Repository

The System Component Repository is a very important part not only in the Virtual Satellite project. The common objective is to store developed simulation models and to collect the related knowledge acquired from development phases, missions or projects, departments and institutes in the DLR to support later reuse. The specific objective for the Virtual Satellite is that the repository has to provide model implementations for easy and fast assembling using the System Editor. A big System Component Repository would enable rapid system simulation generation also for the early design.

To support reuse, the models should be stored in a platform independent way to increase readability and to enable automated transformations to other target platforms. This is possible for the model architecture (model code skeletons) if the models are described in e.g. UML or SMP2 Catalogues. But there exists no pleasant way to store the implemented model code (model behaviour) also in a platform independent way.

For that purpose, it is allowed to store several model implementations or binaries for different platforms (e.g. C++, Simulink or Modelica) as far as they meet the constraints (e.g. interfaces) defined by their platform independent description which is verified by a validation process. To allow a manual transformation to other platforms, the correspondent source code should also be stored wherever possible.

But if the models are supposed to run on the Virtual Satellite's system simulator, they must be transformed to SMP2 compatible C++ code. This is a suitable approach because the simulation platforms typically used in DLR (like Simulink and Modelica) provide transformations to C/C++, and a further transformation to SMP2 compatible C++ is not that complicated. The advantage of this approach is twofold: First, the transformations allow all models stored in the System Component Repository to be used for the system simulation regardless of the tools used for their development, and second, our engineers can develop their models in their favourite environment. The feasibility of the transformations from C++, Simulink and Modelica to SMP2 was already approved by an internal study [6].

In summary, a System Component consists of the following parts:

- Platform independent description of a simulation model (SMP2 Catalogue) or a simulation subsystem (SMP2 Assembly) referencing other System Components

- Dependencies to other System Components

- Optional: binaries of different platforms (C, C++, Simulink, Modelica, or SMP2) and related files (e.g. Matlab scripts or SMP2 Assemblies)

- Optional: related source code (to allow transformation, deriving, or improving)
- Additional information (e.g. usage history, responsible persons, …)

#### *3.3.1 Repository Explorer*

The Repository Explorer is a tool for managing System Components in the repository. It is implemented as a RCE bundle to ensure the integration in the complete development and design environment based on RCE. Beside adding, removing, and versioning of components, the tool provides a clear hierarchical view on all elements in the repository, which may be single System Components, derived components, subsystems or contained components. Additionally, it displays related information like description, inputs, outputs, dependencies, and so on. A drag and drop feature allows easy interaction with the System Editor where the System Components can then be assembled.

### 3.4 Automation Environment

The Automation Environment is represented by a system called SiLEST (Software-in-the-Loop for Embedded System Test), which was developed amongst others by our facility within the correspondent project [7]. Together with a partner from the automobile industry and other research organisations, a process and tools for the simulation-based test of embedded systems as well as the automated analysis of the simulation results were defined. Meanwhile, SiLEST was professionally applied for testing the Attitude Control System software of small satellites as well as for testing motor control devices of automobiles.

Up to now, the test bed layer of SiLEST was represented by a Simulink simulation. The flexible architecture of SiLEST allows the replacement of this Simulink plug-in by a SIMSAT plug-in whereby it is possible to reuse the test case management, version control, evaluation, and report generation plug-ins. For the Virtual Satellite, it is planned, to derive SiLEST test cases from the mission requirements and operation scenarios. On the one hand, these test cases contain test or operation specific inputs, which are loaded to the simulation by the SIMSAT plug-in. On the other, the test cases contain the expected simulation results, which can be compared with the corresponding SIMSAT output using the evaluator plug-in.

After that, the report generation plug-in generates PDF reports which could be reviewed in Concurrent Design or Engineering sessions.

## 4 CONCLUSION AND FUTURE WORK

With this paper the objectives and the current status of the Virtual Satellite project have been presented. The process and the intended supporting tools have been illustrated and the gained experiences with the related software technologies like SMP2 and available tools like SIMSAT, RCE framework, and SiLEST have been discussed.

The evaluation studies performed could show that the SMP2 standard as well as the mentioned tools are generally well suited to achieve the intended definition of a tool supported simulation-based development process for satellites.

After finishing the evaluation phase, the next steps are to define a software design for the tools, to organise cooperation meetings with our engineers to ensure a engineers-friendly application environment and to develop the tools until end of 2009.

## REFERENCES

[1] M. Bandecchi, B. Melton, B. Gardini, F. Ongaro, “The ESA/ESTEC Concurrent Design Facility”, Proceedings of 2nd European Systems Engineering Conference (EuSEC 2000), Munich, Sep. 2000

[2] S. Montenegro, L. Dittrich, “The Core Avionics System for the DLR Compact-Satellite Series”, Small Satellites Systems and Services (The 4S Symposium), Rhodes, Greece, May 2008

[3] Christiansen, K.R., “Integration of Grid Resources in the RCE Environment”, Vdm Verlag, ISBN 9783836475860, Apr. 2008

[4] Daysha Consulting, “ESA: Web Services and Data Integration for Collaborative Design”, ESA Case Study, 2008

[5] H. Wendel, “Development of a spreadsheet based concurrent engineering software infrastructure”, Diploma thesis, Berufsakademie Mannheim, Sep. 2008

[6] A. Röhnsch, A. Berres, O. Maibaum, H. Schumann, “Transformation from Graphical Model Representations into SMP2 Models”, 10th International Workshop on Simulation on European Space Programmes (SESP 2008), Noordwijk (NL), Oct. 2008

[7] H. Schumann, A. Berres, T. Liebezeit, O. Maibaum, „Simulation-Based Testing of Small Satellite Attitude Control Systems“, 6th IAA Symposium on Small Satellites for Earth Observation, Berlin, Apr. 2007