System Concept Simulation for Concurrent Engineering

Workshop on Simulation for European Space Programmes (SESP)
24-26 March 2015

ESA-ESTEC, Noordwijk, The Netherlands

Stephan Kranz(1), Borja Garcia Gutierrez (2), Arne Matthyssen (3), Martin Fijneman (3)

(1) Telespazio VEGA Deutschland GmbH
Europaplatz 5, 64293 Darmstadt, Germany
Email: Stephan.kranz@telespazio-vega-de

(2) ESA/ESTEC
Keplerlaan 1, 2200 AG Noordwijk, The Netherlands
***Email:* Borja.Garcia.Gutierrez@esa.int**

(3) RHEA System S.A.,
Schuttersveld 2, 2316ZA Leiden, The Netherlands
Email: a.matthyssen@rheagroup.com, m.fijneman@rheagroup.com

INTRODUCTION

The main aim of the System Concept Simulator (SCS) is to support the development of a space system in its early mission phases (0/A/B) by providing means to rapidly setup and execute system level simulations. While each engineering domain involved in system development uses their own bespoke tools to simulate the system under their viewpoint, the SCS targets a complete simulation of the entire space system based on inputs from the engineering domains. With focus on multidisciplinary, the use cases of the SCS comprise design feasibility studies, iterative analyses, system requirements validation, and shadow engineering of industrial design. Especially in the context of Concurrent Engineering (CE) activities in phase 0 of a mission, a major constraint is the limited time available from setup to delivery of simulation results. The typical SCS user is an advanced space system engineer with experience in different engineering disciplines, but not necessarily comprehensive programming skills.
Over the last years several system level simulation tools with different design approaches have been developed at ESTEC each showing its strengths but also its weaknesses. Yet, none of them managed to fulfil all requirements on satisfactory level especially in terms of usability within the context of CE. The architectural design of the SCS picks up these lessons learned and combines a model based, formalized simulation approach with an intuitive user interface. The aim is to reduce the complexity of the system simulation to a level that allows the user to make the right decisions in short time and to deliver results quickly, yet keeping the flexibility needed to promptly react on design changes of the system under development. The SCS needs to fit within the process steps, turn around times and modelling philosophy (e.g. required level of detail) in early mission phases on one side, while serving as a basis for more elaborate and extensive simulators used in later phases on the other side.
This paper elaborates on the general concept of a SCS developed in ESTECs System Concept Simulation activity and presents the prototype implementation of a SCS Workbench providing the means for simulation model development, simulation setup and simulation execution.

REQUIREMENTS AND USE CASES

At the start of the SCS activity, a list of user requirements, which gathered the experience from previous simulation activities, was available. Since the foreseen usage of the SCS during phase 0 within the Concurrent Design Facility (CDF) gives the highest constraints on most criteria such as timeliness, flexibility, and adequate level of detail, this has been taking as a focus in the use cases. As users, the ESA CDF Domain Engineer (DE) and System Engineer (SE) were identified with the latter including the ESA CDF team leader and system engineer assistant. The primary user of the SCS however is the Simulation Engineer, i.e. the DE of the simulation domain. SCS Stakeholders are the SE, DE and the ESA CDF customers.

From interviews with stakeholders a set of specific use cases has been identified in which the SCS would bring a high added value to the existing design process. The values expressed by the stakeholders providing added value wrt. what the CD activity team currently can give, were the flexibility to adapt to the (often) changing design and possible lack of data and performing simulations in the timeframe of the CD activity phases. The values expressed by the users are the ease of use, the flexibility to adapt to the (often) changing design and possible lack of data, performing simulations in the timeframe of the CD activity phases, usage/creation/re-use of specific models and simulators and extendibility to next space system design and development phases

Simulation Setup

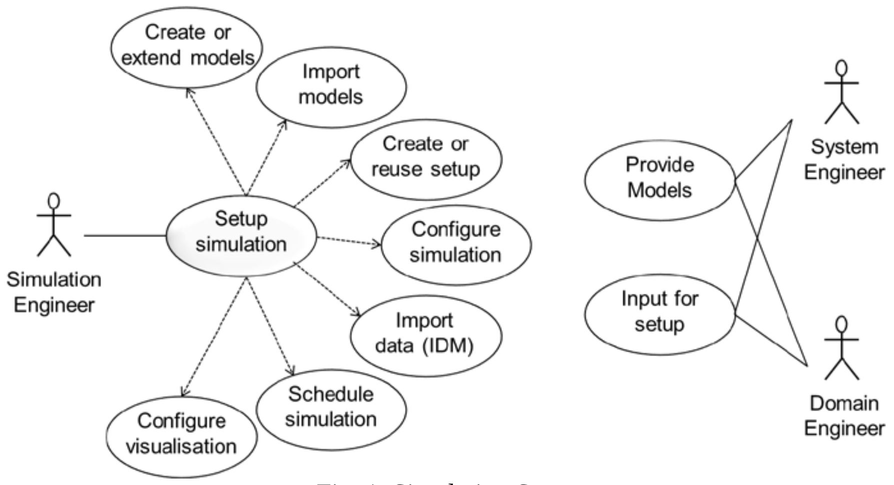


Fig. 1. Simulation Setup

Needs expressed in the interviews related to the simulation setup use case shown in Figure 1 were:

- Easy creation and adaptation of simulation models without extensive programming skills
- Flexible import of data from system domains
- Pragmatic usage of models from system domains
    - Import of models to use them as-is. Process to update models when modified by system domains.
    - Flexible import of results or outputs from system domain models, e.g. through an ASCII file.
- Simulation model reuse across sessions and missions.
- Persistence and reuse of simulation setup and ability for easy adaptation
    - across sessions
    - across missions, requiring a clear capturing of the context, preconditions, assumptions etc.
- Easy selection of parameters in the system database as simulation input according to needed level of detail

Simulation Execution

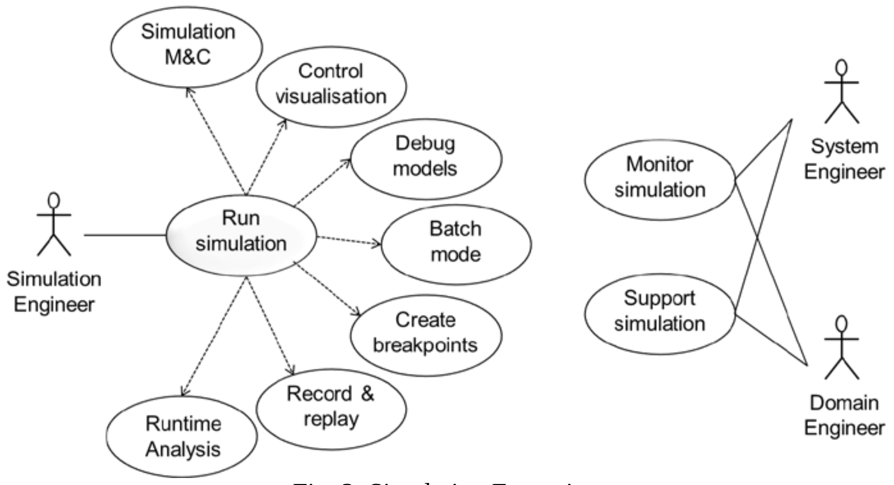


Fig. 2. Simulation Execution

Needs expressed in the interviews related to the simulation execution use case shown in Figure 2 were:

- Monitoring the simulation during runtime.
- Support for iterative simulation runs, i.e. stop, make adaptations, and restart
- Flexibility and adaptability to perform a flow based simulation, fitting into the iterative design process in the CDF

Capturing and sharing of Results

Needs expressed in the interviews related to the capturing and sharing of results use cases shown in Figure 3 are:

- For usage of the results, the information should be stored, with additional information on the simulation assumptions and conditions being available within the file with the simulation results
- Maximisation of reuse of information for later phases in the CDF study
- Maximisation of reuse of information in other projects
- For usage in the CDF environment, the results have to be shared and updated in an easy and flexible way
- The information should be available should fit in the CD process
  - at the correct level of detail
  - within the appropriate timing in the design iterations
- Within a CDF study, in most cases the availability of simulation results itself when needed is more important than the accuracy of the results, to allow the design process to continue. Only for the specific analysis more time is justified, and the simulation may still provide a high added value even when the results are only available at the very end of the CDF study.

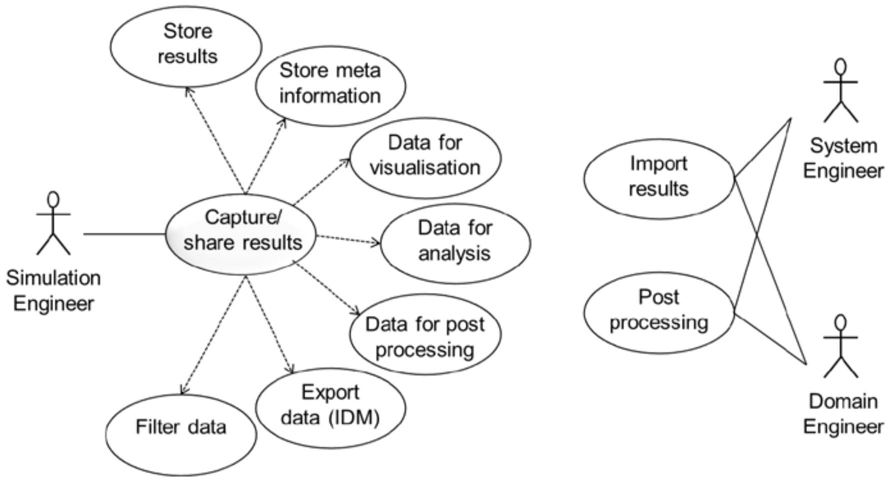


Fig. 3. Capturing and sharing of results

MODELLING & SIMULATION IN THE MISSION LIFETIME

Modeling and simulation is a key activity supporting the specification, design, verification and operations of space systems. In developing these facilities, experiences has shown that there is much commonality across simulation and test facilities. This experience has been captured in the ECSS ETM-10-21 [1] “System Modelling and Simulation” technical memorandum. Figure 4 shows the different tasks of the system engineering process over the mission lifecycle. With progression of the mission lifetime the system database providing the specification of the mission is growing in size and complexity. Along with that, the frequency of changes in the space system specification is decreasing until the final specification and implementation is reached within Phase D/E. These changing characteristics are driving the requirements for simulation tools supporting the system engineering tasks.

Operational spacecraft simulators are supporting training, operations and maintenance tasks in later phases of the mission lifetime. Alike the system database, operational simulators are very complex and typically developed in time-ranges of month up to years. The maturity of the system database makes model development, simulation composition and scheduling mainly a software engineering task against a given space system specification, typically done by IT companies. The simulation user deals with initialisation, execution, recording and analysis of the results.

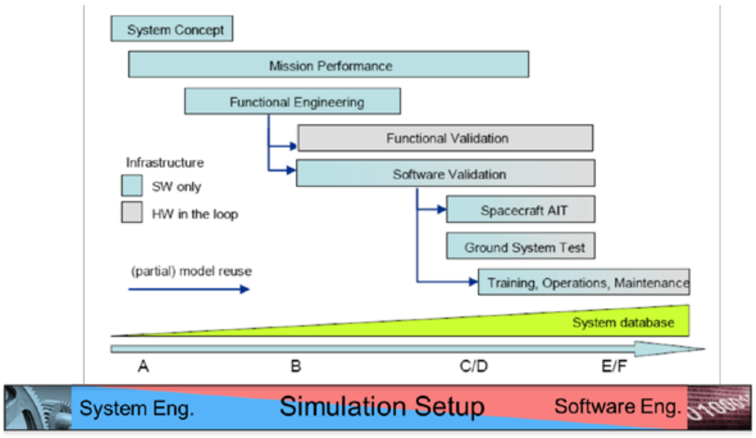


Fig. 4. ECSS-E-TM-10-21A: Space System Engineering Tasks and Facilities

Going towards earlier mission phases, simulation becomes less complex on one side, but is required to be available faster and to be much more flexible to rapidly adapt to changes of the system design. The available development and adaptation time of a simulator is reduced to days eventually targeting a couple of hours in the CE context. While the simulation infrastructure may still be developed by IT companies, the simulator itself is developed by simulation facility staff, in the CE context even including model development. To adapt to that different user profile, the simulator development process needs to become system oriented, while software engineering issues need be taken over by the supporting infrastructure and hidden from the user as far as possible. Typical engineering tools used by system engineers like Matlab/Simulink give an indication of how the UI of a SCS needs to be designed.

SYSTEM CONCEPT SIMULATOR DESIGN

Within the scope of the SCS activity an SCS Workbench has been designed providing support for all high-level simulation tasks closely linked to (CE) space system databases. Taking into account lessons learned from previous activities described above, the SCS UI is targeting the profiles of the SCS users. Furthermore the design of the SCS Workbench aims at providing an integrated UI presenting the underlying tool chain as a seamless and consistent front-end. Although not directly exposed to the user, the simulation related parts and artefacts follow the SMP2 [2] standard ensuring a maximum reuse potential of simulation models and simulation setups across different CE sessions and missions. For the sake of simplification and to adapt to the system engineers picture of a system, the SCS primarily features the data-flow flavour of the SMP2 standard, representing data exchange between models as transfer of values.

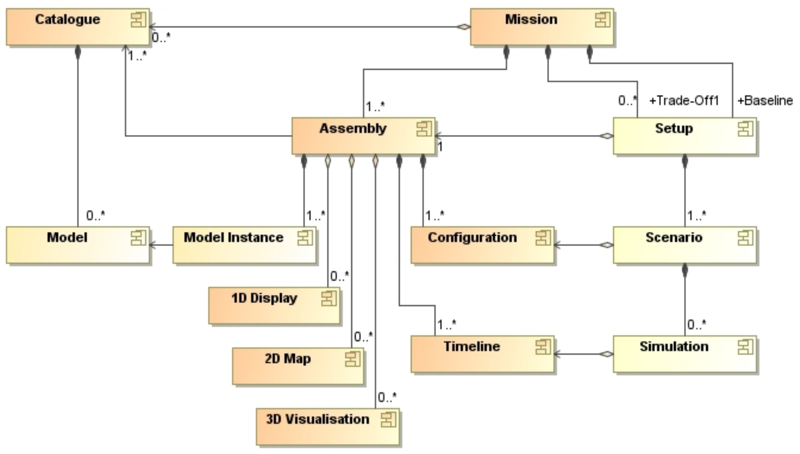


Fig. 5. SCS Workbench – Data Model

Figure 5 shows a simplified version of the SCS data model. Simulation Models are stored in SMP2 Catalogues, which can be reused across different Missions. A Mission contains a baseline Setup and a number of Trade Off Setups as specialisations of the baseline. A Setup uses an SMP2 Assembly that specifies the simulation composition, i.e. the instances of the simulation Models. A Setup may contain a number of Scenarios, each using a specific Configuration that specifies the initial values of the Model Instance parameters defined in the Assembly of the Setup. A Scenario my contain a number of Simulations, each using a Timeline that specifies the order of execution of the Model Instances defined in the Assembly of the Setup. Furthermore an Assembly can be assigned different monitoring displays, 2D maps and 3D Visualisations.

SCS WORKBENCH PROTOTYPE

The SCS Workbench has been prototyped following an iterative implementation and verification approach. Within each use case a number of test steps linked to the direct operation of the SCS Workbench have been performed, allowing for direct feedback on design and implementation, and verification of the required capabilities for the SCS. The SCS workbench prototype is a Java based Eclipse rich client application using state-of-the-art UI technology provided by the Sirius Eclipse project [3] based on the Eclipse Modeling Framework (EMF) and the Graphical Modeling Framework (GMF) allowing for the creation of diagram based user interfaces. Further, the SCS Workbench uses parts of the Eclipse C/C++ Development Kit (CDT), the GCC compiler and GDB debugger. The tool chain underlying the SCS Workbench makes reuse of existing ESA tools like Universal Modelling Framework (UMF) [4] and the SimSat simulator [5] as shown in Figure 6.

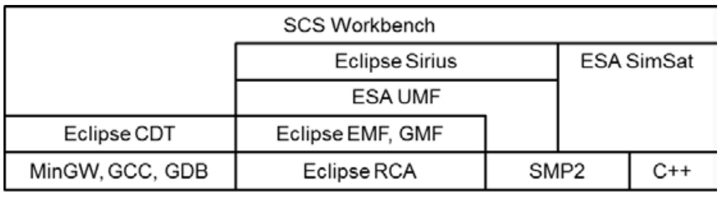


Fig. 6. SCS Workbench SW Components

The SCS Workbench UI layout follows the standard layout of modern UIs although the underlying Eclipse technology allows for a customisation by the user. On the left side of the SCS Workbench window a navigation tree provides an overview of the space system and simulation components, provides the means to navigate among them and to select those to be worked on. For an selected component, an editor is opened in the central part of the UI allowing the user to interact with it. Depending on the kind of component, different kinds of editors are provided, e.g. interactive diagrams, tables, source code listings, allowing the user to work with the system in the most suited way. Diagram based editors come along with a tool palette providing direct access to common functions and an outline window showing which part of a large diagram is currently visible in the editor. The lower part of the SCS Workbench window is reserved for additional views, like properties of the selected component, a console for feedback messages of the workbench and a problems view showing rule violations from the build-in system validation or compilation errors.

Model Development

Following the SMP 2 standard the SCS workbench organises simulation models within model catalogues and namespaces. Eventually the SCS will be deployed with a set of standard catalogues featuring the most common models allowing to compose a first simulation of a space system. In case the design of a space system requires components not yet covered by the standard models or requires extended, more detailed modelling, the user can create additional catalogues containing new models or extended versions of standard models. A model is characterized by different kind of fields. Input fields shown on the left side of a model block specify which values a model receives from other models. Configuration fields inside the model block allow session specific configuration/initialisation of a model. Output fields shown on the right side of a model block specify which values are calculated/provided to other models. For each output field an equation can be specified for calculating the output field value based on values of input and configuration fields as shown in Figure 7.

Based on the model specification the SCS workbench allows to auto-generate C++ code implementing the model. The generated C++ files can be opened in a CDT source code editor for inspection and further editing. Finally the SCS workbench allows for compiling a model catalogue in a binary executable library.

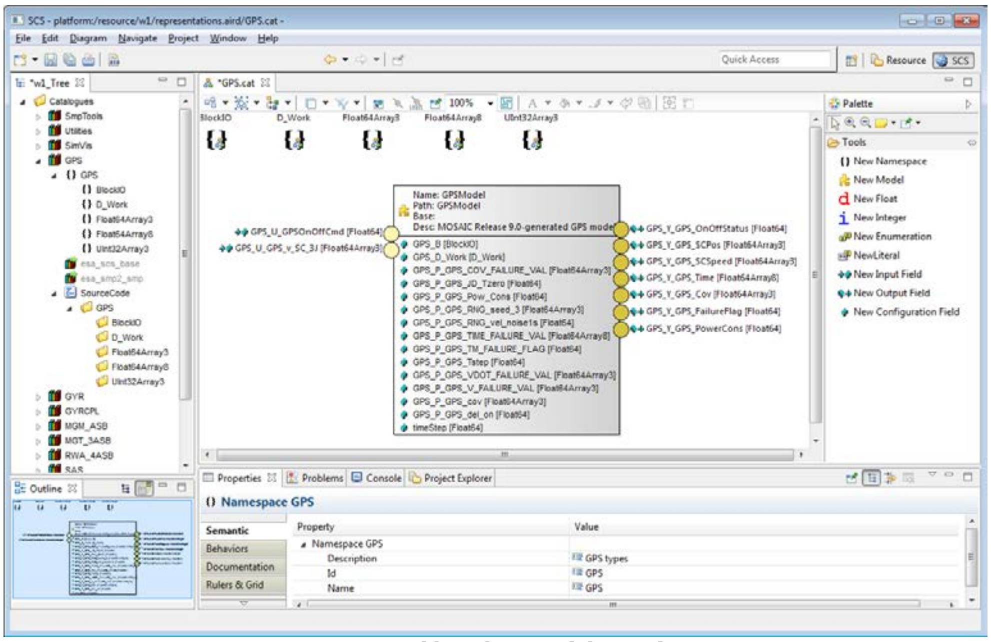


Fig. 7. SCS Workbench – Model Development

Next to models it is possible to specify user-defined float, integer and enumeration types to be used as field types. Importing of existing models from domain specific tools can be done using the MOSAIC tool which delivers a SMP 2 catalogue containing the imported model together with the corresponding C++ implementation.

Simulation Composition

While model development happens on global level, a simulation itself requires the context of a mission. Thus, the first step is to create a new mission referencing the model catalogues to be used. From these catalogues simulation models can be picked for instantiation. These model instantiations are stored in an assembly which represents the composition of the simulation. The SCS workbench simplifies the instantiation process allowing for dragging models from the catalogue tree-view and dropping them onto an assembly diagram provided by the assembly editor. On the assembly diagram each model instance is represented as a box showing input fields on the right side and output fields on the left side as shown on Figure 8.

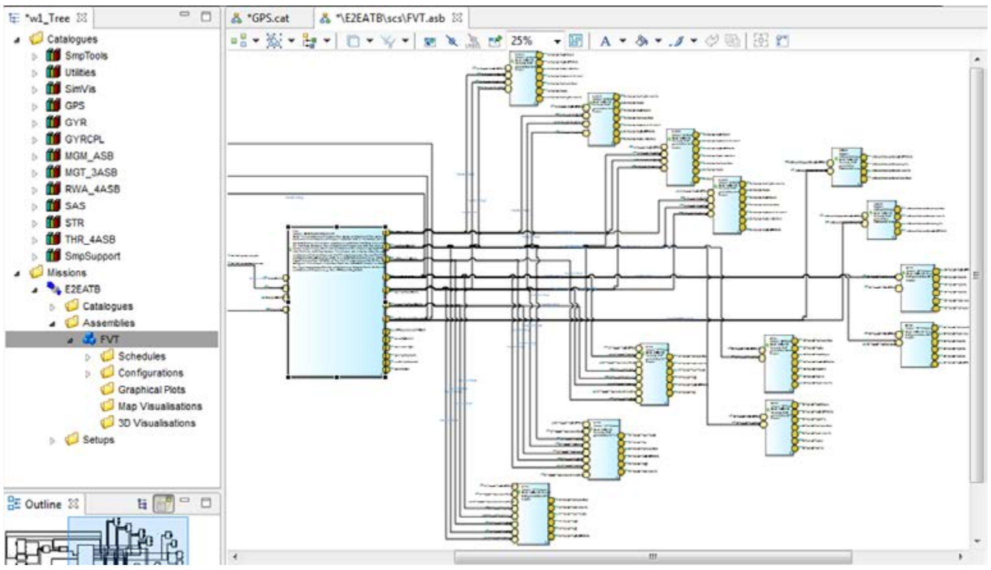


Fig. 8. SCS Workbench – Simulation Composition – Mechanical Viewpoint

Once the assembly has been populated with model instances, the assembly editor allows to connect model input fields with model output fields and thus establishing a data flow between the model instances. The definition of an assembly can be distributed over multiple diagrams, each representing a different viewpoint onto the simulated system. For example individual diagrams can represent the viewpoints of different disciplines, like mechanical, power, thermal or payload as shown on Figure 9 and 10. In addition, the assembly editor provides several filters to show/hide certain groups of elements, e.g. hide all connected input fields to only show the yet unconnected ones.

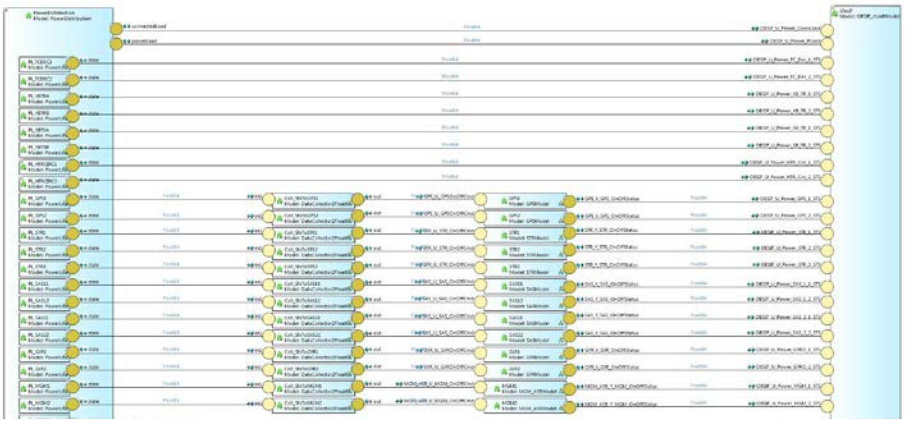


Fig. 9. SCS Workbench – Simulation Composition – Power Distribution Viewpoint

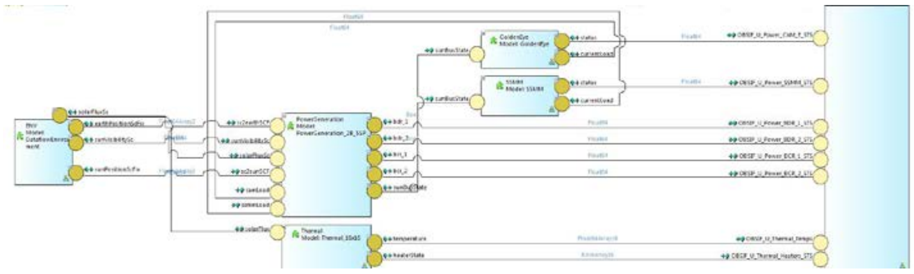
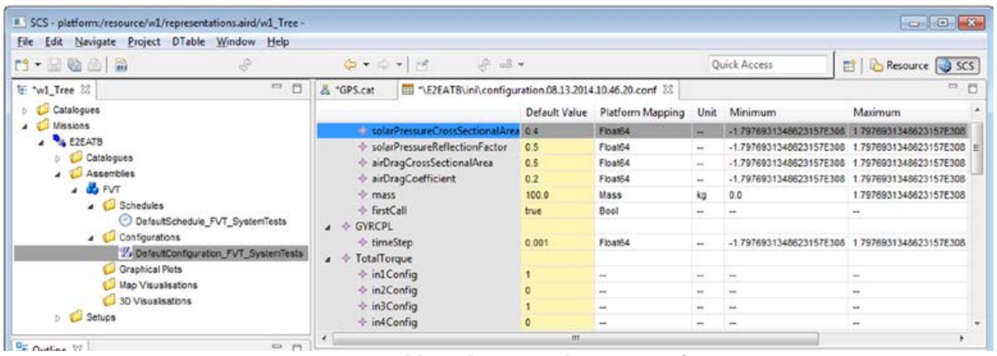


Fig. 10. SCS Workbench – Simulation Composition – Power Generation and Thermal Viewpoint

While a simulation can be composed from scratch it is also foreseen to reuse and adapt existing assemblies from previous or template missions to minimise simulation setup time. How far the simulation composition can even be automated using data from the system database is still under investigation.

Simulation Configuration


SCS - platform:/resource/w1/representations.aird/w1_Tree -
File Edit Navigate Project DTable Window Help
Quick Access
Resource
SCS
*w1_Tree
Catalogues
Missions
E2EATB
Catalogues
Assemblies
FVT
Schedules
DefaultSchedule_FVT_SystemTests
Configurations
DefaultConfiguration_FVT_SystemTests
Graphical Plots
Map Visualisations
3D Visualisations
Setups
Outline
*GPS.cat
*\E2EATB\ini\configuration.08.13.2014.10.46.20.conf

	Default Value	Platform Mapping	Unit	Minimum	Maximum
solarPressureCrossSectionalArea	0.4	Float64	--	-1.7976931348623157E308	1.7976931348623157E308
solarPressureReflectionFactor	0.5	Float64	--	-1.7976931348623157E308	1.7976931348623157E308
airDragCrossSectionalArea	0.5	Float64	--	-1.7976931348623157E308	1.7976931348623157E308
airDragCoefficient	0.2	Float64	--	-1.7976931348623157E308	1.7976931348623157E308
mass	100.0	Mass	kg	0.0	1.7976931348623157E308
firstCall	true	Bool	--	--	--
GYRCPL					
timeStep	0.001	Float64	--	-1.7976931348623157E308	1.7976931348623157E308
TotalTorque					
in1Config	1	--	--	--	--
in2Config	0	--	--	--	--
in3Config	1	--	--	--	--
in4Config	0	--	--	--	--

Fig. 11. SCS Workbench – Simulation Configuration

Once a simulation has been composed and stored within an assembly, an assembly specific configuration can be created containing initial values for each configuration field of every model instance. Within the prototype implementation the user can specify initial values in a configuration table editor showing the configuration fields of the model instances together with unit and minimum/maximum value defined by the type assigned to each field as shown on Figure 11.

Eventually it is envisaged to retrieve the initial values from the system database and automatically populate the configuration, leaving the configuration table editor as a cross-check instrument.

Simulation scheduling

The SCS workbench provides a diagram based schedule editor allowing to inspect and modify the auto-generated model instance update sequence. Like on the assembly diagram each model instance is depicted as a box, while the arrow connections of the boxes define the order of update as shown on Figure 12. Updating of a model instance means the calculation of the output field values according to the defined equations. It is foreseen to fully automate the schedule generation with the help of model update dependencies specified already during model development.

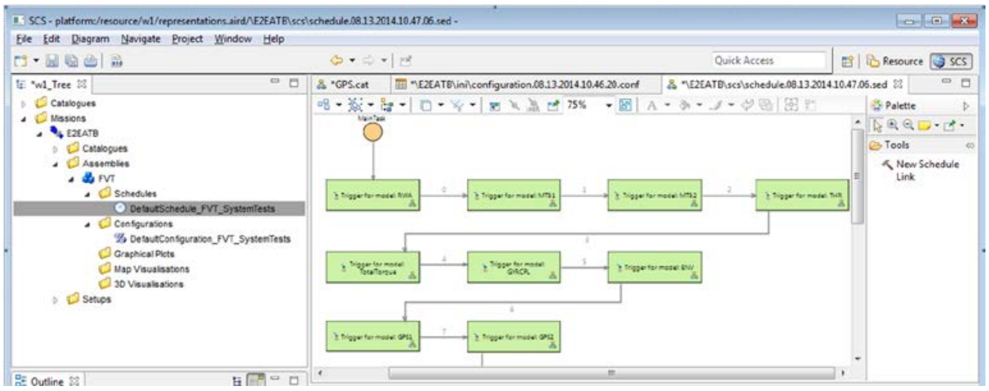


Fig. 12. SCS Workbench – Simulation Scheduling

Simulation Execution

The current implementation of the SCS workbench prototype allows creating all artefacts needed for executing the simulation. Since these artefacts follow the SMP2 standard, any SMP2 compliant simulation infrastructure (e.g. SimSat or EuroSim) could be used to run the simulation. It is envisaged to integrate the SimSat simulation kernel into the SCS workbench allowing for simulation execution and to reuse parts of the SimSat MMI together with other UI components (3d plots, 2d maps tracking, 3d animations) for monitoring.

CONCLUSION AND OUTLOOK

Within ESTECs “System Functional Simulations in the Concurrent Design Process” activity performed by Telespazio-VEGA and RHEA the architectural design of a SCS workbench has been created. The SCS workbench targets at rapidly setting up and executing a simulation of the space system under design. Key aspects are to reduce time from system design to simulation to a minimum, to provide a UI designed for system engineers, and to allow for reusing simulation models across missions and in other simulators within later mission phases. In an agile approach a prototype for the SCS Workbench has been implemented undergoing several cycles of design verification and validation. The prototype uses state-of-the-art modelling and UI technology freely available on the market under Eclipse public license.

As next steps the direct connection of the SCS Workbench to the Simulation Runtime Environment (SimSat Kernel) will be implemented in order to monitor and control the simulation process directly from the SCS Workbench UI. Furthermore it is envisaged to implement extended monitoring and visualisation elements starting from alphanumeric displays and plots, via 2d maps up to 3d animations.

REFERENCES

[1] ECSS-ETM-10-21: Space Engineering, System Modelling and Simulations.
[2] SMP2 Standard, portal.vega.de/smp
[3] Sirius Eclipse project http://www.eclipse.org/sirius/index.html
[4] P. Ellsiepen, P. Fritzen, V: Reggestad, A. Walsh, “UMF – A Productive SMP2 Modelling and Development Tool Chain”, SESP 2012
[5] SIMSAT as part of SIMULUS http://www.esa.int/Our_Activities/Operations/Ground_Systems_Engineering/SIMULUS