A Digital Environment for Early Design, Analytics, and Verification

Workshop on Simulation and EGSE for Space Programmes (SESP)
26 - 28 March 2019

ESA-ESTEC, Noordwijk, The Netherlands

Christian Hennig[(1)]

[(1)]Airbus Defence and Space GmbH
Claude-Dornier-Straße
88090 Immenstaad
Germany
Email: christian.hennig@airbus.com

INTRODUCTION

While a significant number of system design studies is still conducted using traditional, loosely connected, modeling and data exchange means, there is a strong tendency towards a fully-fledged digital exchange of data between involved disciplines and their tools.
At Airbus' space division, the System Design Environment (SDE) is being used to digitally integrate system design efforts within early project phases (0, A, B0, B1), and to ensure the continuity of data towards later phases. This environment involves commercial engineering tools, custom tools, and a variety of data interfaces for facilitating the data integration as needed by the early phase system design process.
The SDE was successfully used in studies such as e.Deorbit, Galileo 2nd Generation Phase B0, as well as numerous internal activities, and is continuing to evolve.

CHALLENGES WITHIN MANUALLY MANAGED ENGINEERING PROCESSES

Classically, over the last years, a pragmatic approach to designing a system in its study phases is being pursued. This approach involves having numerous models, often realized via spreadsheet software, which may or may not be linked with each other. These models contain the system design, the subsystem design, and any kind of analysis. Results of analyses and updates on the system's design are then propagated over a multitude of channels that may involve e-mail, directly telling a colleague, mentioning changes in meetings, or performing an update of the spreadsheet that may be on in the cloud or stored locally.
While this pragmatic modelling approach is easily set up in the beginning of a project and easily accessible to most of the project team, it poses a number of challenges:

- Manual data transfer between models and documents: Data is often being taken from documents, entered into models, processed, and the results are then manually transferred to other models or documents. This requires a significant data refactoring overhead and has a risk of producing inconsistencies.
- Updates on data not being propagated: Sometimes, updates on the system are not being propagated throughout all disciplines, resulting in analysis iterations using outdated designs, resulting in RIDs and unplanned reworks.
- System engineering overhead: The system engineering discipline has a significant overhead due to manually ensuring consistency and manually propagating necessary updates throughout all parties involved in the project.
- Isolated knowledge cells: Knowledge about data used across several projects is often not documented, and if documented, not accessible throughout the company. This makes re-using a product, and integrating its lessons learnt, difficult.

The main consequence of these challenges is that the most powerful system that can be produced is somewhat limited by its overall complexity. If the data is to be managed manually, the effort will increase with growing system complexity. This means that the system is able to be built with exponentially increased cost, or not at all, assuming a cost-competitive environment.

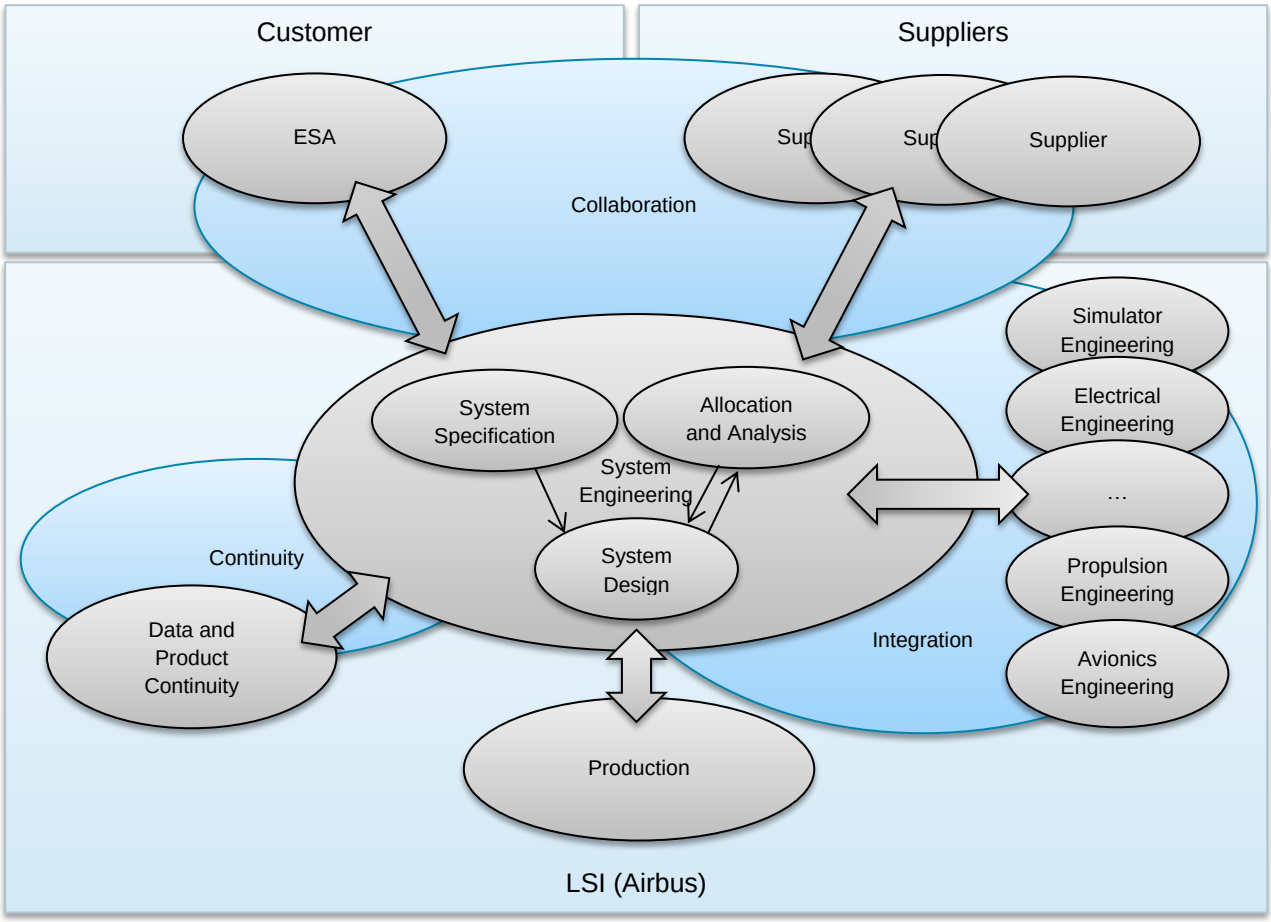

Fig. 1: Abstracted view on system engineering process

CENTRAL MOTIVES AND PRINCIPLES

The SDE is based on the principles of facilitating organization-internal engineering, co-engineering with suppliers, and the collaboration with the customer, using three principles (Fig. 1).

Integration of system engineering and discipline engineering tools

Within the organization, an integration of employed engineering tools within the early system design processes is performed. This includes an integration of tools as they are already used, refraining from forcing change on tool usage.

Continuity of data and product

The data produced in phase 0 is able to be reused in phase A, and can be used up to utilization and decommissioning of the system, if desired. There is no need to transform or extract the data, enabling the back-tracing of changes throughout all lifecycle phases. Furthermore, an integration of product data into any new project can be performed, allowing usage as-is, or with adaptions, while also allowing backflow of product improvements.

Facilitation of collaboration

Between engineering disciplines, but also with ESA and suppliers, means for collaboration facilitation are provided.

TOOLS, INTERFACES, AND ARCHITECTURE

The SDE architecture consists of a number of layers that each serve a distinct purpose within the system design process (Fig. 2). The most important layers are the Semantic Data Continuity layer (dark blue), and the authoring layer (yellow). Engineering data usually originates within an engineering process using one of the authoring tools. Should the data be required to be transferred across the discipline border, either for consumption within another domain and tool, or simply for versioning, it is passed to the Semantic Data Continuity layer, where it is persisted, versioned, checked regarding

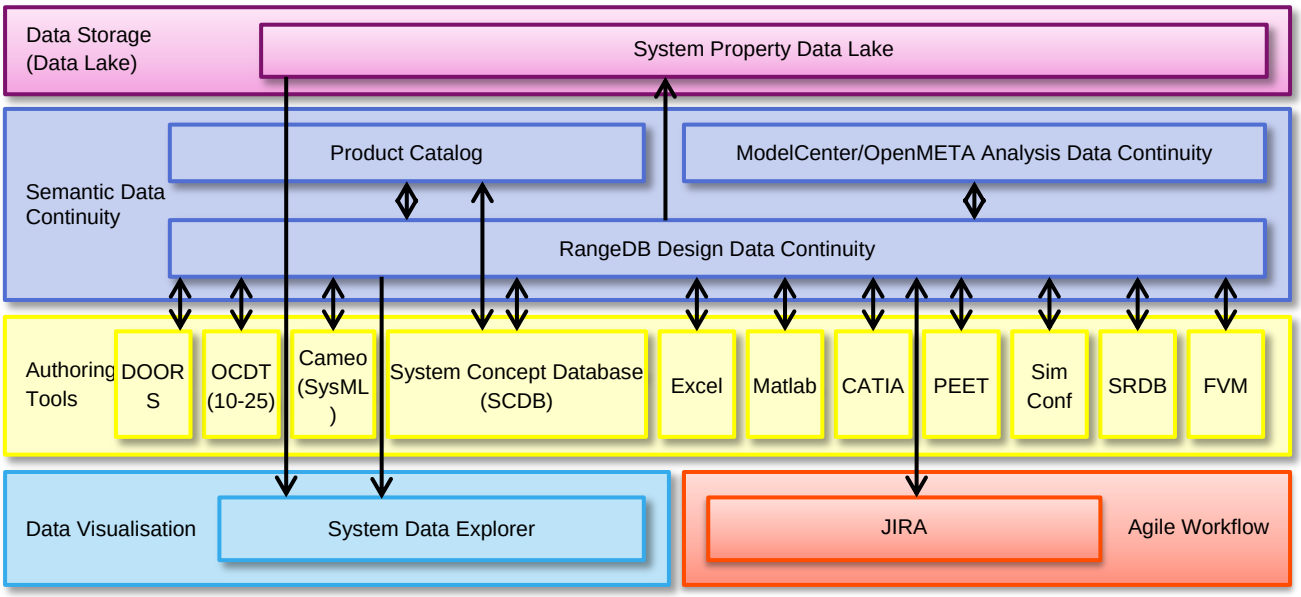

Fig. 2: System Design Environment Tools and Interfaces

consistency. All authoring tools shown in the process have interfaces to the continuity layer. Another layer is focused on Data Storage. This layer contains the capability to handle what is commonly called Big Data, and takes care of storing the data acquired during system production and testing. The Data Visualization layer visualizes data stored within the Semantic Continuity and Data Storage layers. This includes graphical representations of data, or document-based and web-based representations.The Agile Workflow layer manages the workflow aspect of system design, handling RIDs, tickets, actions, etc. their relation to engineering data, and process dependencies.

RangeDB Data Continuity Layer for Versioning and Exchange

The Data Continuity Layer is the central pillar of the SDE. It is realized through the RangeDB technology platform, which provides the capability for data representation, data organization, and persistence, as well as common data management functionality such as versioning, baselining, querying, report generation, branching and merging, etc. Through this layer, a continuity of data across all system decomposition levels, all phases of the system lifecycle, and across all disciplines, is provided.
The RangeDB architecture follows closely the concepts from the ECSS-E-TM-10-23 technical memorandum [1]. One of the key principles is the integration of different engineering applications by integrating local conceptual data models of the individual tools via a global conceptual data model. The central data model of RangeDB is derived from the global conceptual data model as defined by Annex B of [1].

Product Catalog for Reuse

The Product Catalog building block in Fig. 2 also directly interfaces with the RangeDB continuity layer, providing access to the data of engineering products that are already available and ready for use by a project. It provides mechanisms to (re-) use these products as they are, to adapt and evolve these products, and to make available new products to other projects. The compatibility with RangeDB allows effective data exchange and versioning in line with established principles.

System Authoring Tools

The various authoring tools are integrating with the RangeDB continuity layer via dedicated interfaces specifically developed for their use case.

One key building block in this context is the System Concept Database (SCDB) application. This application assumes the role of the system engineering and system modelling application, as it provides the capabilities to

- manage the system's product tree
- manage different system options/variants
- access and integrate product data
- manage key parameters
- manage domain-specific properties
- perform operational design
- manage budgets such as mass, and mode-dependent budgets such as power and data rate

- generate design reports
- trace the history and changes of the system's design
- perform baselining of the system
- perform activities such as data querying, viewing, extraction, etc.
- control data exchange with other tools

For functional design in terms of mission activities, operational aspects, and functional electrical architecture, an integration with SysML [2] is available. This integration allows a separation of the functional and logical architecture (SysML) from the concrete physical architecture (RangeDB/SCDB. The SysML-RangeDB integration is bi-directional, providing a flexible approach to exchange data within design iterations between both tools.

Discipline-Specific Authoring Tool Integration

As an important integration of an established engineering tool, a CATIA interface is provided. This interface allows directly integrating mechanical design data such as component masses, mass assumptions, dimensions, moments of inertia, etc. of applicable mechanical components.

The Matlab integration allows to access system design data in Matlab, utilize it for analyses or simulations, and to push back analysis results to the system domain.

The integration of the PEET tool is used to digitally exchange data between pointing error analysis and the other domains. As data such as focal lengths and distortions is provided by other disciplines, it can be accessed from the RangeDB continuity layer by the PEET tool.

An interface with DOORS is used to manage requirements, and to trace them to system design.

An important interface is given by the connection between RangeDB and Excel, as this provides the capability to generate reports, and to execute custom analyses. These analyses include budget calculations, parametrized sizing considerations, trades, or fully-fledge iterative calculations.

Analysis and Simulation Integration

Apart from Matlab and Excel for basic simulation, other bridges are also provided. In early phase design, analysis management tools such as ModelCenter [3] or OpenMETA [4] are increasingly being used [5]. Within these tools, RangeDB-based data can be loaded, utilized, and manipulated within defined simulation chains.

Logical Element	Physical Element	System Level	eDeorbit_ST_KP2	Description	Dataset
eDeorbit		System		Configuration for eDeorbit mission	e.Deorbit
SpaceSegment		Segment	1	edeorbit space segment definition	e.Deorbit
SpaceTug		Element	1	SpaceTug configuration for e.deorbit	e.Deorbit
Structure		Subsystem	1	-	e.Deorbit
Service_Module		SubsystemSet	1	-	e.Deorbit
Payload_Module		SubsystemSet	1	-	e.Deorbit
Top_Floor		SubsystemSet	1	-	e.Deorbit
Thermal		Subsystem	1	-	e.Deorbit
Heatpipes		SubsystemSet	1	-	e.Deorbit
Coatings		SubsystemSet	1	-	e.Deorbit
Power		Subsystem	1	-	e.Deorbit
Harness		Unit	1	-	e.Deorbit
Battery		Unit	1	-	Battery_TypeB
PSR		Unit	1	PCDU	PSR_TypeA
SADM		Unit	1	-	SADM_TypeA
SolarArray		Unit	1	-	SolarArray_TypeA
TCR		Subsystem	1	-	e.Deorbit
S-Band RxTx		Unit	2	-	e.Deorbit
S-Band LGA		Unit	2	-	e.Deorbit
Boom		Unit	2	-	e.Deorbit
Chemical_Propulsion		Subsystem	1	-	e.Deorbit
Tubing_and_valve		Unit	1	-	e.Deorbit
Thruster_22N		Unit	12	-	Thruster_Typ22N
Thruster_430N		Unit	1	-	Thruster_Typ430N
Propellant_Tank		Unit	2	-	Propellant_Tank_TypeA
Pressurant_Tank		Unit	2	-	Pressurant_Tank_TypeA
DHS		Subsystem	1	-	e.Deorbit
CDMU_TypA		Unit	1	-	CDMU_TypA
RemoteControlUnit_TypA		Unit	2	-	RCU_TypA
DataBus		Unit	1	-	e.Deorbit
Image Box		Unit	1	-	e.Deorbit
ArmControlUnit		Unit	2	-	ManipulatorArm_MDA
AOCS_GNC		Subsystem	1	-	e.Deorbit
STR_TypA		Unit	1	-	STR_TypA

Fig. 3: e.Deorbit Product Tree within SCDB

Another integration is given by the SimConf module, connecting RangeDB system data with simulators based on Airbus' SimTG simulator infrastructure. For early system design, the System Concept Simulator (SCS) and the Functional Engineering Simulator (FES) are the most applicable [6]. These simulators allow getting an early estimation on the system's performance and design correctness [7].

Compatibility to Later Phase Authoring Tools

While the SCDB forms the central system authoring tool in early design stages, other RangeDB-based authoring tools are being used further d the system design cycle. These tools are fully data-compatible.

The Functional Verification Manager (FVM) is used to engineer the verification part of the system design process, spanning verification definition, verification detailed design, verification execution, evaluation and closeout. It provides requirement traceability in terms of which requirements will be closed out by which verification activities, what the items under test are, etc. It stands in close relation to the system design tool and the system simulators based on the Airbus SimTG framework, and with AIT support tools used to execute tests, store test data, and to analyze the results.

The System Reference Database (SRDB) forms a central pillar of avionics design in this context. It is used to specify the definitions of operational aspects such as packets, parameters, their calibrations, etc. and provides interfaces to import and export dedicated formats, including MIB, EGS-CC, CCS5, etc.

Data Exchange with ESA

The SDE provides an interface to the Open Concurrent Design Tool (OCDT) [8] developed and owned by ESA. OCDT is mainly used to exchange concurrent engineering models in order to facilitate concurrent design sessions. OCDT manages data such as product structure, system options, system and component parameters, etc. and is based on the data specification given in ECSS-E-TM-10-25 [9]. Besides concurrent engineering, this interface can also be used for digital data deliveries as review input, and to deliver post-review data packages.

Data Visualization

Apart from visualization of design and analysis data in the usual discipline-specific tools, the capability for cross-discipline data visualization is also available. The System Data Explorer is a Web application displaying system data in different formats useable for analysis. These analyses include visualization of budget trends over time, visualization of budget distribution across different subsystems, or just tables of properties common to all system elements.

Agile Workflow Integration

An agile workflow management system is used to collect tickets in the form of issues, actions, etc. within the project. These issues may originate directly within the system engineering process performed with SCDB, addressing a concrete technical issue, or have their origin more within the execution with any involved engineering activity. For addressing issues and tasks that have a relation to the system's design, an integration within the RangeDB continuity layer is provided, allowing the automated update or closing of the respective ticket, should the issue fix be published to the continuity layer.

USAGE AND LESSONS LEARNED

Project Use

Components within the System Design Environment are used as required and desired by the project. For the Galileo 2nd Generation Phase B0 activity, focus was put on performing the system design with SCDB, integrating numerous trades and analyses done in Excel, and integrating a Matlab model for discipline-specific analysis, while relying on the RangeDB layer for data continuity with following activities.

The e.Deorbit activity started with defining the functional system design, mission activities, and operational aspects in SysML. This data was then integrated via the RangeDB layer with SCDB, and distributed to Excel based analyses. Furthermore, the ModelCenter integration was used to execute analyses, and to perform design of experiments in order to find the system design optimum under the given boundary conditions. The resulting system design was then delivered to ESA via the OCDT interface.

Lessons Learned

Usage of the SDE as a vehicle to facilitate MBSE within early space system design brought several challenges, but has also improved several aspects of the engineering process:

- Having a centrally hosted, version controlled system model where disciplines can access the current system baseline and provide updates is helping improve overall system design consistency significantly. This resulted in considerably less inconsistency RIDs in several activities.
- Having an Excel integration that directly accesses system data, and can produce reports of current design data on demand, helped save a considerable amount of time in report production.
- The accessibility and browsability of product data for any given SDE-based project helps to disseminate knowledge about products, and streamlines product use within the organization.
- The automated transfer of design data to simulation tools enables new system analysis use cases that optimize the overall system design.

Challenges have risen especially in the following fields:

- A continuous trade is being performed between providing an expert tool that allows manipulating every bit of data within the system, and providing a simple tool that can easily be learned by new users.
- The data models of involved tools are not easily mapped, as they cover different aspects of a system's design. In many cases, data mappings have to be correctly interpreted in accordance with their user groups, and the mapping and algorithms thoroughly validated. This is especially true for the involved SysML interface.
- As with any digitalization activity, new tools and principles are emerging rapidly that have to be considered, and, if needed, integrated. Interfaces have to be evolved continuously.

REFERENCES

[1] ESA, 2011. ECSS-E-TM-10-23A: Space engineering - Space system data repository. Noordwijk, The Netherlands: ESA.

[2] OMG, 2017. OMG Systems Modeling Language (OMG SysML) Version 1.5. [Online]
Available at: http://www.omg.org/spec/SysML/1.5/

[3] Phoenix Integration, 2019. ModelCenter Integrate. [Online]
Available at: https://www.phoenix-int.com/product/modelcenter-integrate/

[4] MetaMorph, Inc., 2019. OpenMETA. [Online]
Available at: https://openmeta.metamorphsoftware.com/

[5] Estable, S., et al., 2016. Generation of chaser requirements and budgets for the e.Deorbit mission applying a MBSE process. 7th International Conference on Systems & Concurrent Engineering for Space Applications (SECESA 2016), 5-7 October.

[6] ESA, 2010. ECSS-E-TM-10-21A: Space engineering - System modelling and simulation. Noordwijk, The Netherlands: ESA.

[7] System Verification Through the Life-Cycle Final Report (SVT.RP.ASU.SY.00011)

[8] ESA, 2019. Open Concurrent Design Tool (OCDT) Community Portal. [Online]
Available at: https://ocdt.esa.int/

[9] ESA, 2010. ECSS-E-TM-10-25: Engineering design model data exchange (CDF). Noordwijk, The Netherlands: ESA.