Applications of Model-Based Systems Engineering for JAXA's Engineering Test Satellite-9 Project

Yuta Nakajima[1*]; *Tsutomu Fukatsu*[2]

[1]*Japan Aerospace Exploration Agency, Tsukuba, Ibaraki, Japan,* nakajima.yuta@jaxa.jp
[2]*Japan Aerospace Exploration Agency, Tsukuba, Ibaraki, Japan*

1. Introduction

JAXA's Engineering Test Satellite-9(ETS-9) project team is applying Model-Based Systems Engineering approach to the interface management of flight system development. Launching in the early 2020s, ETS-9 demonstrates the all-electric spacecraft technologies for the next generation communication satellite including the newly developed Hall Effect Thruster System as shown in Figure 1. The Hall Effect Thruster System consists of the three main components: the thruster, the power processing unit (PPU), and the propellant flow control module. The power processing unit controls and monitors thruster system performance. The comprehensive understanding of a complex Hall Effect Thruster system is a challenging issue for project systems engineers due to the complex interaction between components developed by different providers.

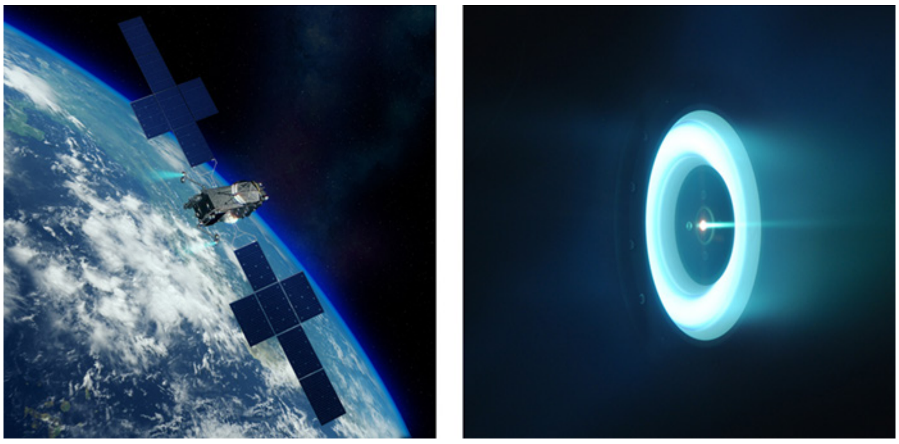


Figure 1: Artist concept for ETS-9 Mission (Left) and performance test of Hall Effect Thruster (Right)

2. How do we manage the complexity?

The main idea of our approach is a comprehensive system analysis supported by a system model and interactive digital artifacts that visualize system analysis results extracted from a SysML system model. We manage the system complexity by using the formalized descriptions of system requirements, behaviors and behavior allocations to system elements with SysML. We describe the system behaviors as sets of interactions between components by using the object flow of activity models. The object flow includes the electric energy, the xenon gas, information and the force generated by the thruster. This behavior model is expanded to the failure mode analysis by customised descriptions of the failure modes. We extract systems engineering products as html based interactive digital artifacts from the SysML model by using Python graph theory packages and data visualization packages. These digitalized systems engineering products support systems engineers to understand and analyse the complex system. The flow of extracting systems engineering products as an interactive digital artifact is shown in Figure 2. The prototype of interactive digital artifacts that support the system analysis and the failure mode analysis are shown in Figure 3 and Figure 4.

3. Conclusion

We present the practice of Model-Based Systems Engineering approach to the actual flight project focusing on the interface management of the Hall Effect Thruster system. The proposed system model and the interactive digital artifacts guide the system level analysis of the Hall Effect Thruster System supported by model queries and visualization of the hierarchical structure of system architecture. We find the effective use of MBSE application to the failure mode analysis in the implementation phase of the ETS-9 flight project.

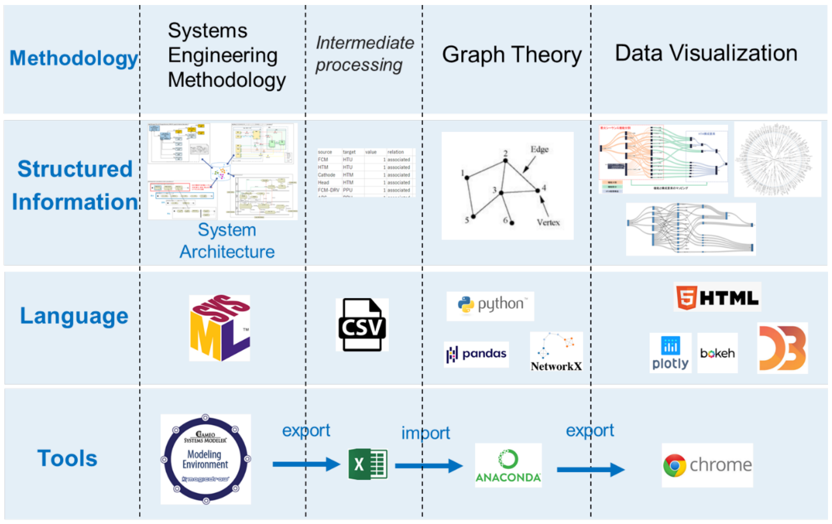


Figure 2: Flow of extracting interactive digital artifacts from a SysML model

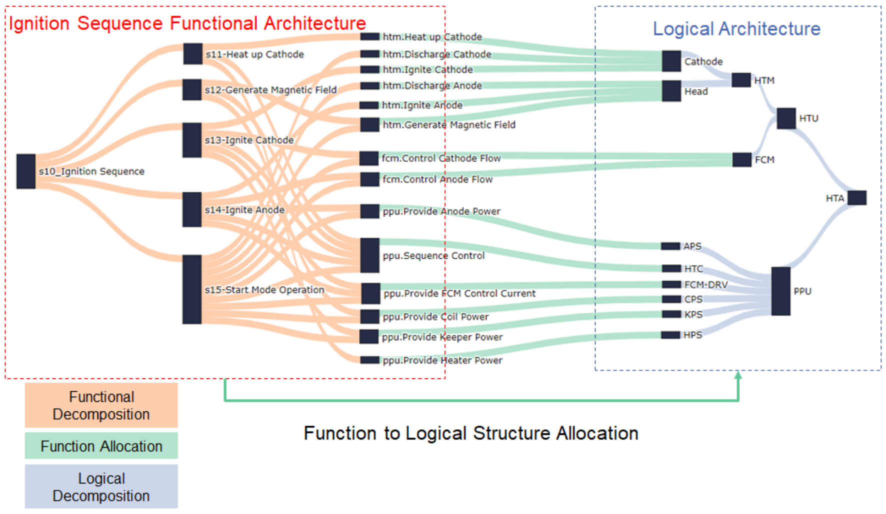


Figure 3: Interactive Sankey Diagram supports understanding a system of interest

Figure 4: Interactive radial failure mode hierarchy maps created by Python and D3.js

Modelling Avionics Interfaces and Generating Interface Control Documents for the Propulsion Subsystem of the MPCV European Service Module

Delia Cellarier, Antonio Preden
MPCV-ESM Engineering Team, European Space Agency, Noordwijk, the Netherlands
delia.cellarier@esa.int

***Abstract* — In complex systems such as the MPCV European Service Module (ESM), Interface Control Documents (ICDs) are key system engineering artefacts that are used to specify and control interfaces. In current practice, ICDs are largely created, maintained and verified manually, leading to tedious and error-prone activities. A model-based approach can be implemented to use a model as "single source of truth". It thus enforces consistency and can be a basis for generating ICDs. This paper explains why and how this approach was applied with Capella to the avionics interfaces of the ESM Propulsion Subsystem.**

Keywords — Model-Based System Engineering, Interface Control Documents, Capella

1. Context and Motivations

Interface management is a crucial system engineering activity for space projects. As described in the dedicated ECSS standard, its objective is "to achieve functional and physical compatibility amongst all interrelated items in the product tree" [1], ensuring that the different components will be integrated into a working system. It is particularly challenging for complex systems such as spacecrafts, involving many parties (space agencies, main contractors and suppliers) and disciplines.

This is the case for the European Service Module (ESM), ESA's contribution to NASA's Orion spacecraft (MPCV). Built by main contractor Airbus Defence and Space, with many other companies supplying components, it provides propulsion, power, thermal control, and water and air for astronauts. Several spacecrafts will be provided to support the Artemis missions , and the third one is currently in design phase.

In this context, Interface Control Documents (ICDs) are used throughout the lifecycle to specify and control interfaces of subsystems. Their role in the ESM development makes them difficult to manage, though. Indeed, because of both their technical and contractual aspect, ICDs are subject to a standardized change process that can create inconsistencies between documents. Moreover, information is sometimes redundant between ICDs of different levels (e.g. Propulsion subsystem and equipment ICDs) or separated in data ICDs specific to some equipment. Maintenance and verification of those ICDs are for now largely done manually, making it difficult to keep consistency.

The purpose of this work is to implement a model-based approach, using MBSE technologies to effectively manage information and generate ICDs from a model [2]. This approach is evaluated through a case study, the avionics interfaces of the ESM Propulsion Subsystem, with the objective of being applicable in the project's future.

This paper will first give an overview of the process which led to the selection of Capella as MBSE solution. It will then describe the first results and the methodology applied to the whole project.

2. Trade-off Between MBSE Solutions

The project started with a study and a trade-off to choose the most appropriate MBSE tool. First of all, requirements have been defined according to project's needs (e.g. document generation features) and ECSS standards, and then refined through interviews of projects experts in various disciplines.

A state of the art revealed similar approaches in space projects, all based on SysML. However, the authors either extended SysML with a profile [3], the language being too generic for their needs, or focused on software interfaces [4]. Some initiatives at ESA were also explored:

- *Electronic Data Sheets*, for Data Handling and Electrical interfaces of spacecraft avionics;
- *ESA SysML Profile*, developed by ESA MBSE Core team for System Engineering of space projects.

Technology readiness, coverage of multi-disciplinary interfaces and understanding for newcomers to MBSE narrowed the choice of an MBSE tool to Capella and Enterprise Architect extended with ESA SysML Profile. A trade-off based on a mock-up has been made between those two solutions to evaluate them against our requirements. Both tools offer most of the desired features, but Capella was preferred, among other things, for its accessibility and its customization.

3. Methodology to Manage Interfaces with Capella

The mock-up under Capella included two avionics boxes and a few electrical interfaces. Thanks to M2Doc, an open-source add-on by Obeo to generate MS-Word documents from Capella models, parts of an equipment ICD could successfully be generated. M2Doc uses Word templates written in a language built on top of Acceleo Query Language (AQL) for querying the model. This enables flexible and custom document generation.

As a consistent implementation is necessary to efficiently generate ICDs, scaling-up to the whole Propulsion Subsystem implies a more systematic approach. A mapping has thus been made between types of avionics interfaces (part of Mechanical, Thermal, Electrical or Numerical ICDs) and model elements (e.g. Physical Link, Component Exchange, etc.). On another hand, M2Doc templates can become complicated for non-practitioners when the model grows in complexity. To make them easily modifiable by end-users, M2Doc template patterns and services (i.e. Java functions) will be developed.

Our case study involves some specificities compared to a generic approach in Capella. Indeed, the model and generated ICDs are realized by shadow engineering, showing how Capella can be introduced in a project where the design is already well advanced and documents remain key deliverables. In this case, the model focuses on an effective management of information and interfaces with existing artefacts, such as the Harness Database, rather than trying to replace everything. Concerning the ARCADIA method, considering that a design was already existing, the usual steps were not followed and we started directly with the modelling of the Physical Architecture.

An interesting feature of Capella is the possibility to extend it thanks to the viewpoint technology provided by Capella Studio. It allows to adapt or add definition of new data, diagrams, user interfaces or validation rules. In our project, it can be used to help experts from each discipline focusing on their interfaces and controlling them.

Eventually, as traceability throughout the life cycle is a major concern, existing Viewpoints will be used to demonstrate its feasibility in Capella: the *Requirement Viewpoint* can specify links with data extracted from Interface Requirement Documents, while the *V&V/TestMeans Viewpoint* (a commercial add-on by Artal) can refer to test campaigns.

4. Conclusion and Future Work

Through a study of related work and a detailed trade-off, MBSE and Capella in particular have proven to be a promising solution for specifying and controlling interfaces as well as automatically generating ICDs.

At the time of writing this abstract, ongoing work is carried out to extend the model to all avionics interfaces of the ESM Propulsion Subsystem, by applying a tailored methodology and by implementing artefacts for ICDs generation in Capella. A Viewpoint dedicated to interfaces will potentially be developed. Results will be compared with the ones obtained by industry with the standard approach. The expected outcome is that, providing initial efforts to endorse MBSE, management of interfaces will be facilitated, even for an advanced document-based project.

Future work would involve applying this concept to a real project, so lessons learned from this experience will be exploited to provide recommendations for the integration of modelling activities in current processes.

References

[1] ECSS, ECSS-E-ST-10-24C – Interface Management, June 2015.

[2] D. Cellarier, ESA-MPCV-RP-0060 YGT Report: Modelization of Avionics Interfaces.

[3] E. Fosse and C. L. Delp, “Systems Engineering Interfaces: A Model Based Approach,” in *IEEE Aerospace Conference Proceedings*, 2013.

[4] S. J. I. Herzig, R. Karban, G. Brack, S. B. Michaels, F. Dekens and M. Troy, “Verifying Interfaces and Generating Interface Control Documents for the Alignment and Phasing Subsystem of the Thirty Meter Telescope from a System Model in SysML,” *Proc. 8th Modeling Syst. Eng., Project Manage. Astron.,* vol. 10705, no. Art. no. 107050V, 2018.

Applying MBSE across Flight and Ground on Small and Nano-Satellite Missions

Peter Mendham
Bright Ascension Ltd, Scotland, UK
peter@brightascension.com

The so-called "New Space" industry represents a particularly challenging environment for software. This paper presents the commercial development and application of an Model-Based Software Engineering (MBSE) solution to a diverse range of small and nano-satellite missions. Particular attention is paid to the successful use of this solution across both flight and ground software, and the benefits this has brought.

The paper concludes by summarising lessons learned and looks to future developments which will expand the scope and capabilities of the solution.

Commercial Context

There is currently a global boom in the development and application of small satellites, usually characterised as those with a mass between 1kg and 50kg. In many cases, the approach taken to the development and deployment of these satellites is one that accepts a higher risk in product assurance for significantly lower development costs and faster development times.

Characteristics of missions in this category typically include:

- **rapid development times**, commonly 6-12 months per satellite and 1-2 years from conception to being able to offer an initial commercial service;
- l**arge numbers of satellites**, with constellations of 10-50 satellites being common targets although there are high-profile cases of organisations aiming to deploy hundreds of satellites;
- a **high degree of heterogeneity** between satellites in a constellation;
- **complex payloads** which embed much of their functionality in software;
- use of **Commercial Off The Shelf** (COTS) hardware from a wide range of vendors;
- the need for **highly automated operations**, typically aiming for unattended operations for the duration of a week at a time; and
- use of commercial **Ground Station Network** (GSN) providers.

This places consequent demands on the mission software, across both flight and ground.

User Needs

Space-based service developers utilising small and nano-satellites are facing a range of challenges which give rise to demands on the mission software:

- **availability**, software must be available early (at least partially) to support development and test;
- **flexibility**, requirements during development change as design is iterated;
- **rapidity**, overall schedule is short and software must be ready quickly;
- **capability**, many spacecraft functions are implemented in software;
- **operability**, software must make it easy to achieve mission and service delivery;
- **reliability**, software is mission-critical and must be robust;

- **scalability**, flight and ground software must integrate to form part of a complete system which may include multiple, complex spacecraft and ground segments.

Addressing these challenges places great demands on software, resulting in complexity both in the software itself and in the process of applying it to the mission or service.

The GenerationOne Approach

In response to these needs Bright Ascension have developed a technology called GenerationOne which combines:

- **model-based** software engineering, permitting machine comprehension of software architecture and the use of tools to assist with software development and product/quality assurance;
- **component-based** software engineering, enabling reuse of software across a wide range of scenarios and applications, combining software with its documentation and tests within libraries; and a
- **service-oriented** architecture, providing consistent and well-defined semantics for component interactions at all levels, enabling low-level aspects of the system to be expressed as components whilst improving operability.

GenerationOne technology comprises a meta-model definition, a language-independent set of service and protocol definitions, cross platform tools and framework implementations for target platforms. The GenerationOne approach permits almost the entirety of a software system to be expressed as components, from hardware drivers and communications protocols to applications. The underlying framework is lightweight and most components are portable across platforms and operating systems.

The component and service model is specifically designed to be applicable across both flight and ground environments with the model capturing both ground- and flight-based functionality. The encompassing nature of the model also extends to the life-cycle, with the model representing the system from early payload prototyping through development, Assembly Integration and Test (AIT) through to in-orbit operations.

Developing and Implementing GenerationOne

Bright Ascension’s primary focus for GenerationOne has always been to solve industry problems. As such, the technology was not fully designed up front, but instead has been developed iteratively, with improvements, additions and changes based on the experience gained from deployment in operational missions. Here the technology has benefited from the rapid launch cadence of the small and nano-satellite industry with twelve on-orbit spacecraft making use of GenerationOne, with many more in development and some slated for launch this year.

GenerationOne has been used by customers on all six continents across a wide variety of computing environments, including as flight software running on a number of COTS Onboard Computers (OBCs). The process of supporting so many target platforms has introduced a number of improvements and revisions to permit optimisation for low resources, and flexibility to different processor and operating system architectures.

Mission Case Studies

It is perhaps illustrative to highlight two missions which have benefited from the use of GenerationOne technology.

The **KIPP** and **CASE** spacecraft, intended to pilot Kepler Comminations’ Internet-of-Things (IoT) constellation are 3kg nano-satellites flying a single, highly-capable Software Defined Radio (SDR) payload. With a strong commercial case, these spacecraft were developed by AAC Clyde

Space within 8 months from concept to launch shipment. Bright Ascension used GenerationOne to develop and deliver flight and ground software in 5 months using 6 months of engineering time. The model developed from the flight software was ingested into the ground software for immediate use with minimal configuration effort necessary.

Faraday-1, the first of In-Space Space Missions’ hosted payload spacecraft offers different challenges in terms of complexity and operability. Despite having a mass of less than 10kg, Faraday-1 includes six payloads from different organisations, including four SDRs each of which hosts multiple software applications, effectively “massless payloads”. Faraday-1 is a highly distributed system, with a total of 6 OBCs of three different architectures and operating systems requiring 13 software images. The complete system is captured in a single model which can be viewed and used by both development tooling and the Mission Control Software.

Benefits of a Coordinated Flight-Ground Approach

The interface between the spacecraft and the ground segment sits within the software functions responsible for managing and delivering the mission and overall service. As such, although it is traditional, selecting the spacelink as the position for a significant division in system composition and even development responsibilities is a short-sighted decision leading to poor architecture and inefficient mission delivery. There are many reasons for this division spanning technical, organisational, commercial and political concerns. Within small and nano-satellite missions, most of these drivers do not exist, leaving technical arguments at the forefront.

There is no doubt that many of the challenges faced by flight and ground software are different, often generated by the distinct computing environments: from low-resource, real-time embedded systems, to enterprise systems with many simultaneous human operators handling large quantities of data. However, with GenerationOne, Bright Ascension has shown that a well designed architecture can accommodate these technical differences within one coherent system. This gives significant advantages for all aspects of system design, development, operation and maintenance.

Key to these cross-system efficiencies is the ability to capture the complete system in a single model which describes the functional architecture and its relationship with the physical system. The model acts as the basis for increased operability as well as the basis for a domain model which can be used by automation systems, including planning.

Conclusions and Future Work

The application of MBSE to Bright Ascension's GenerationOne technology has been instrumental in creating a technology which facilitates the rapid development and efficient operation of complex missions. Key to this has been the development of a single coherent architecture and meta-model which can be utilised across both flight and ground systems.

The development of GenerationOne has been incremental and is far from complete. Current improvements are focussed on scalability, operability and automation, impacting all aspects of GenerationOne from the meta-model through to standard service definitions. Once this next phase of development is complete, it is expected that the opportunity will be taken to supplement the model with a more capable range of tooling and support infrastructure, offering benefits to a wide range of stakeholders involved in mission and service delivery.

First lessons with a Model-Based System Engineering approach for nanosatellites

B. Segret[1*]; *R. Jain[1], B. Mosser[1],*

[1]Paris Observatory – PSL University, Meudon, FRANCE,
**Primary author contact details: boris.segret@observatoiredeparis.psl.eu*

1. Abstract

Model-Based System Engineering (MBSE) is promising to support space instrumentation. CCERES [1] set up an MBSE approach for the early phases of scientific nanosatellite missions with free tools only. Our mission analysis covers more than only the trajectory of a space mission: it addresses the scientific coverage as the main driver, then also its coupling with the functional modes, the pointing requirements, the data volume and the power. The early requirements for these main functions are translated in tiny models, i.e. pieces of codes in python or GNU Octave, whose outputs are formatted to get displayed in the CNES' VTS display, a free tool available for Windows and Linux. An in-house software called DOCKS has been developed in python and made open-source. DOCKS simplifies some parts of modelling with the same philosophy of producing outputs to VTS display. We will present some of our scientific nanosatellite projects translated in MBSE terms with VTS displays, in Earth orbit or in deep space, and discuss about their advantages and limitations. We will also report on their application during Concurrent Engineering sessions. As a result, the traditional studies and tools for system analysis at platform level are certainly not deemed deprecated. Our MBSE approach was made possible and necessary in a CubeSat context: the CubeSat form factor simplifies many aspects and also provides the space laboratories with opportunities for entirely new measurement concepts and not only for payload development. The main lesson of this MBSE approach is to guide an instrumentation team to increase its Concept Maturity Level (CML) from CML 0 to CML 4 typically, and to install an efficient dialog among all actors, within a project and with its partners from the New Space or from the traditional space sector.

2. References

Acknowledgments: C²ERES ([1], pronounced "C-CERES", Figure 1) has been hosted by Paris Observatory [2] since 2014 and funded by the French Laboratory of Excellence ESEP [3] under the grant ANR #2011-LABX-030 in the "Investissements d'Avenir" and by PSL University Paris ([4]).

[1] C²ERES, Space pole of PSL University Paris (Figure 1): https://cceres.psl.eu/?lang=en

[2] Paris Observatory: https://www.observatoiredeparis.psl.eu/?lang=en

[3] Exploration Spatiale des Environnements Planétaires (ESEP): http://www.esep.pro/en/

[4] Paris Sciences Lettres (PSL Université): https://www.psl.eu/en

Figure 1: C²ERES is the Space Pole of PSL University Paris.

Stepwise adoption of model-based solution for a full MBSE transition, an Industrial perspective

D. Perillo[1], C. S. Malavenda[2]
[1] Eng David Perillo PhD, ESA/ESTEC, Noordwijk, Netherlands
[2] Eng. Claudio Santo Malavenda PhD MBA, ELT, Rome, Italy
Email: david.perillo@esa.int, claudosanto.malavenda@elt.it

Abstract — In this abstract we will discuss the need to provide European Space Companies and Research Institutions with a European framework for model-based engineering (MBE) and model-based system engineering (MBSE). Our assumptions will be motivated by the Lesson Learned from research programs performed by Elettronica SpA (ELT), one of the most referenced European players in the production of Electronic Warfare equipment (EW), in its transition toward model-driven engineering (MDE) and subsequently to MBSE as a mean to increase quality, to increase productivity and to reduce costs.

1 Introduction

According to the original definition given from INCOSE, Model Based System Engineering (SE) is the formalized application of modeling to support system requirements, design, analysis, verification and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases.
In an organization that adopt a mature model-based workflow, models at different levels of abstraction coexist in an interwoven structure held together by system-level architectures that act as a backbone for SE activities. For instance, it shall be possible to graphically navigate a system level architecture, traversing all the subsystems and visualizing inner details of software-firmware interfaces, as well as conducting Reliability, Availability, Maintainability, and Safety (RAMS) analysis in an almost automated manner by means of dedicated model-checking techniques (i.e. COMPASS toolset *https://essr.esa.int/project/compass*).

2 Adopting MBSE

From an Industrial perspective, to become an effective MBSE practitioner requires a significant investment [1] and the time needed to return from the investment can be hard to calculate in advance. In addition, this transition entails to overcome a cultural resistance to change within the organization [2]. In some cases, the road to create a Company culture on model-based technologies can be rough and steep. That is partially due to the lack of experience with formal languages and object-oriented thinking for engineers that usually have different specialties and partially to the absence of a commercial general-purpose solution capable to support every engineering domain aspect. This means that companies are often left alone with the burden to select and tailor model-based tools and methodologies on their specific needs and value chain analysis. It is therefore evident that MBSE is producing a major transformation in the way of doing system engineering, which can be probably compared with the advent of personal computers in the workplace in the late 70s and 80s.

2.1 Facing Cultural Resistances

When it turns to overcoming the cultural resistance to the adoption of models as means to enclose system and subsystem details, it should be kept in mind that models are more than just drawing:

- Models can act as single source of truth, whereas natural languages and document-based approaches are subject to interpretation and misunderstanding.
- Model based toolchains can be extended incrementally, according to perceived benefit of users and stakeholders.
- Models can be a turnkey solution to manage complexity by means of Views and filters.
- Formal languages are a powerful mean to stimulate reasoning and evaluating alternatives.
- Models can be automatically processed to produce artefacts, such as code and documentation, as well as to verify integrity and overall consistency of the finalised architecture.

One possible approach to build a Company culture around model-based engineering is to initially leverage model-driven solutions to automate processes that have a direct return on investment. Software engineers can be a key element in this first stage of the transformation process as they are usually keen to exploit model-driven solutions to automate implementation and verification of software and firmware components.

2.1 The ELT case

As described in [3], a similar approach was followed by Elettronica Spa (ELT) in his transition toward model-driven engineering (MDE). Motivated by the need to increase quality, to increase productivity and to reduce costs, ELT has decided to evolve and update the design and development process with a model-driven approach. The transition has begun in 2010 with the implementation of a Company-internal model-based toolchain to automate coding and documentation of software interfaces, operative system drivers and verification facilities such as Wireshark-dissectors, simulators and emulators.

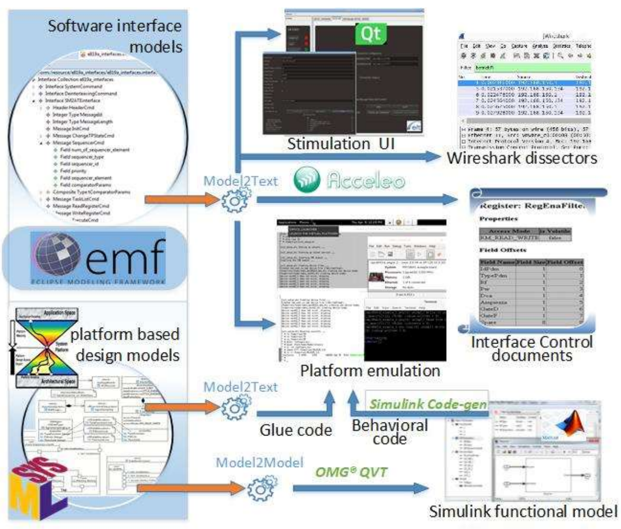


Figure 1- model-driven workflow implemented in ELT

The toolchain is also integrated with Simulink, for code-generation of behavioural code, and with IBM Rational Doors to trace architectural decisions against system and sub-system requirements. However, since Electronic Datasheets (CCSDS EDS) were not publicly available in 2010 and ELT needs were peculiar to EW systems, both the metamodel and the model-based toolchain have been built from scratch leveraging the Eclipse Modeling Framework (EMF) and related Ecore technologies (Acceleo, QVT) for model to text (M2T) and model to model (M2M) transformations. The SysML models representing the System were defined according to Platform-based design (PBD) principles. As such, design elements were decomposed into three model hierarchies: a Functional architecture, an Execution Platform and a third hierarchy of elements (called Mapping model) representing the deployment of the Functional architecture onto the Execution one. The genericity of SysML model elements were restricted applying Stereotypes from MaRTE®, which is an OMG® Profile specific for Real-Time Embedded Systems. The implementation was conducted internally by experienced ELT software engineers in collaboration with the TeCiP institute of Scuola Superiore Sant'Anna. Although this solution enabled a first transition in the adoption of models as a mean to encapsulate and share knowledge among software stakeholders, it must be mentioned that models are not just about software. One of the main concerns of an Engineering Company shall be to introduce system engineers to system-level modelling, adopting formal languages as a vehicle of information and as a mean to enrich technical documentation.

3 A project example with MBSE

Bolstered by the achieved consensus with MDE, ELT has experimented MBSE on internal pilot projects (partially or totally self-financed) in the context of the Company innovation process named BELT (short form of Building ELT together). The System of Systems (SoS) adopted as use-cases for the modelisation are aimed to support armed forces in the operations of integrated missions that cover the following domains:

- Electronic Warfare
- Spectrum Management
- Signal intelligence
- Cyber Operation

In this context, MBSE can provide a consistent advantage to manage the complexity caused by: the intrinsic scalable and reconfigurable shape of these SoS; the high number of actors (internal and external to the System); the high number of program's stakeholders with vested interest to be kept into consideration along the project lifespan.

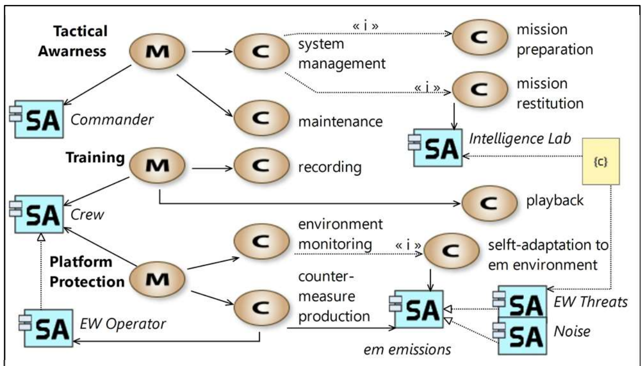


Figure 2- EW Mission/Capability example diagram

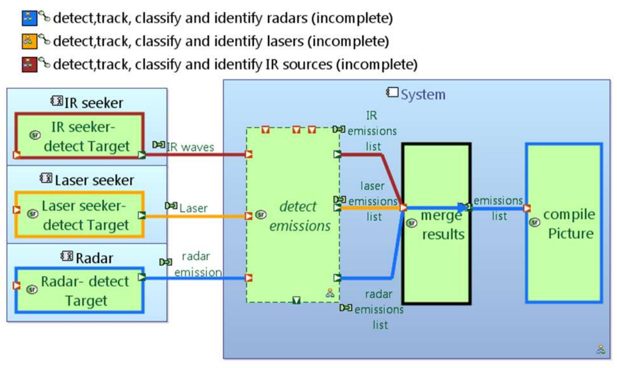


Figure 3- System Architecture diagram with functional chains

The model-based approach selected for this purpose was ARCADIA (ARChitecture Analysis and Design Integrated Approach) with its open-source model-editor Capella and the Requirements Viewpoint to import requirements from IBM Doors. In one of the MBSE experimentations performed within BELT, a team of six system-engineers with different specialties and not prior knowledge of modelling based languages, supported by one Modeling Expert and one Project Manager, managed to enclose a portion of the system knowledge into a model-based representation with enough details to run basic model-checking activities on it. The Mission/Capability diagram in figure 2 shows some of the classical challenges that EW SoS are required to perform in a reliable and accurate manner: self-adaptation to the electromagnetic (e.m.) environment, tactical awareness, mission and data management and platform protection. The System Architecture diagram provided as example in figure 3 provides a quick overview of three simplified Functional Chains associated to environment-monitoring Capability. Specifically, the detection, classification and identification of Infrared (IR), Laser and Radar guided weapons (also called Targets) in a synthetic representation of the electromagnetic environment.
Major benefits reported at the end of the design stage were:

- Mapping of targeted use cases toward the developed architecture.
- Inheriting interfaces from high-level to system-level architecture.
- Automatic verification of interfaces consistency.
- Justification of the physical architecture toward the functional one.
- Impact analysis to evaluate complete and consistent propagation of requirements toward the final architecture.

In particular, at the end of the medialisation activity it had been possible to investigate a number of issues just by validating the model. We identified the absence of a physical connection to carry data exchanged among functions originally thought to be deployed on two unconnected nodes. This issue were tackled restructuring functional deployment and physical architecture so as to optimise the overall design in terms of costs and performances. The Arcadia methodology and Capella model-editor resulted of easy understanding for the team. An initial two-weeks training period was enough to make the team self-confident and autonomous in the basic modelling activities, which were performed in accordance to processes specific for a document-centric SE workflow. This experience demonstrated how the adoption of formal notations could support system engineers to reason about architectural choices and their impact on stakeholders.

4 Conclusions

A takeaway message from this experience is the possibility to use model-driven solutions to automate processes that have a direct return on investment and do not need the full MBSE to be implemented at the beginning. Given the additional cost of creating models, it is of primary importance to create a modelling ecosystem in which models can be exploited to automatically produce valuable artefacts such as low-level embedded code, documentation, adapters, simulators and other supporting facilities for Validation & Verification purposes. As a way forward, the availability of model-based solutions readily available to European Space Companies, such as the Open-Source Reference Architecture (OSRA), could enable a quicker transition to MBE and MBSE as a strategy to increase quality and productivity while reducing development and maintenance costs.

Acknowledgements

This abstract takes contributions from the ELT website and presentations internal to ELT for which we thank **Daniela Pistoia** (Corporate Chief Scientist at ELT), **Antonio Tafuto** (Head of Research & Innovation at ELT) and **Francesco Chirico** (Principal Engineer at ELT). Additional guidelines were provided by **Andreas Jung**, **Javier Fernandez Salgado** and **Marcel Verhoef** from ESA/ESTEC.

References

[1] Parrott, E., Trase, K., Green, R., Varga, D., Powell, J. NASA GRC MBSE Implementation Status.

[2] Chami, M., Bruel, J.M., (2018) A Survey on MBSE Adoption Challenges

[3] Di Natale, M., Perillo, D., Chirico, F., Sindico, A., Sangiovanni-Vincentelli, A., "A Model-based approach for the synthesis of software to firmware adapters for use with automatically generated components", Software and Systems Modeling (SoSyM) Journal 17(1): 11-33 (2018)

[4] Malavenda, C. S., Menichelli, F., & Olivieri, M. "Wireless and Ad Hoc Sensor Networks: An Industrial Example Using Delay Tolerant, Low Power Protocols for Security-Critical Applications", in Applications in Electronics Pervading Industry, Environment and Society: 153-162 (2014) Springer International Publishing

Author/Speaker Biographies

David Perillo , PhD. He recently joined TEC-SWF in ESA/ESTEC as Vitrociset Contractor, with technical responsibility over software and model-based projects. Before joining ESA he was technical responsible in ELT for software lifecycle activities of ELINT systems. He was also in charge to support ELT with its transition toward MBSE with Arcadia.

Claudio Santo Malavenda , PhD, MBA. He covered several roles in STMicroelectronics, Selex ES (actual Leonardo) and ELT dealing with Project and Contract management. His works activity deals first series products management and marketing, Wireless Sensor Network, C4I systems for fire control in network centric environment and Cyber Electromagnetic Activities.

Lessons learned on the use of MBSE in the preliminary design of space systems at CT Paris

Author : Julien Morane (julien.morane@ctingenierie.com), Cedric Dupont (cedric.dupont@ctingenierie.com)

Formerly part of Bertin Technologie, the Paris office of the CT Engineering Group has a strong expertise in innovative space system design and High-performance computing simulation.

During the last year, the group has started to implement the Arcadia method supported by the Capella tool in several projects. The choice of the Arcadia method was driven by its 'customer-friendly' first steps (Operational Analysis & System Needs Analysis), that allow to check the adequacy of the prototype system relatively to customer requirements and expectations. Another expected benefit was the completeness and the consistency of the created system. Eventually, the possibility to capitalize in the created system for later development phase, and especially for the interaction with potential supplier, was also a main driver for change.

We developed models for different projects including:

- Our patented space debris mitigation system INSIDeR (an inflatable net aiming to capture space debris)
- An innovative space braking system for just-in time collision avoidance
- Launcher ground segments

Although the main concepts were already conceptualized, the use of the Arcadia methodology in support of the preliminary studies has revealed relevant, particularly to prepare the way forward for those low-TRL systems.

The modeling of the nominal situations and components has enabled a refinement of the preliminary requirements thorough the study.

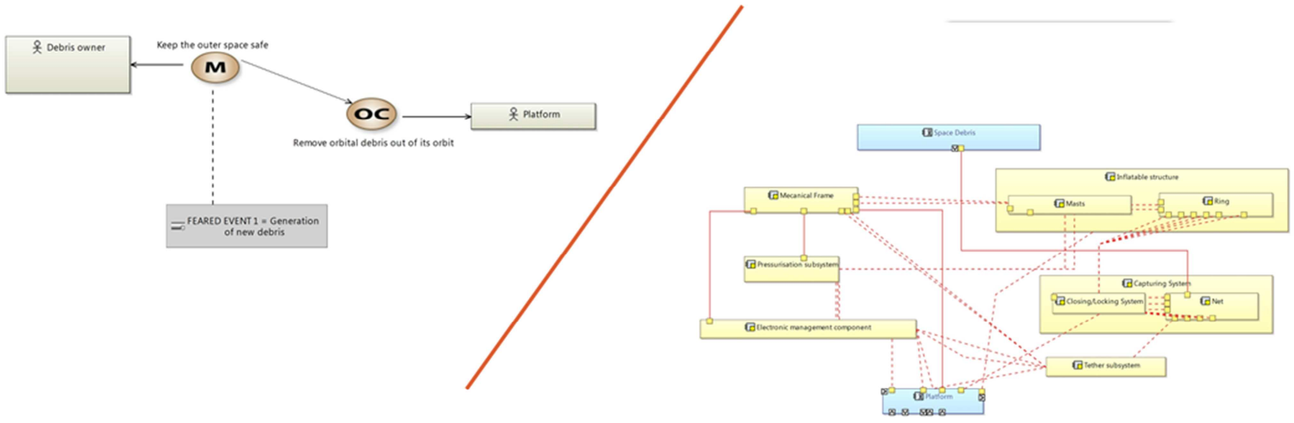
The modeling of non-nominal situations and feared events has enabled to consider new problematics. For instance, in INSIDeR, a system aiming to close the net and capture the debris has been designed. The analysis of a feared event 'creation of new debris' has conducted to further studies on the system design to ensure that a single link break in the system will not entail the separation of any subsystem of the inflatable structure.

The paper will also present how the traditional value analysis methodology and the Arcadia method are complementary. One may cite among others:

- The management of the feedback loops between customer expectations and real system behavior is facilitated by the centralization of all the system characteristics.
- The management of the heterogeneity in subsystem conception, both in the liberty Arcadia method gives to build the models and in the maturity of the different subsystem.
- Automatic and formal verification of the completeness of the work, with the automatic validation and transitions between Capella layers.

Once created, the model has been (or will be) used for several purposes:

- Description of the dynamic behavior of Insider for communication purpose at various levels.
- Management of interfaces with the satellite in which the system will be embedded.
- Streamlined description of sub-systems to be realized, with their associated requirements, functions, interface and behavior, allowing the communication with potential suppliers and used for the development roadmap of the system.



Insider and Arcadia / Capella : from Customer mission to first system design

The Arcadia/Capella approach is a complementary method to the classical preliminary design methodologies. The use of MBSE at the early stage of a project allows to prepare the next development phases with a common system architecture that can be share by all future stakeholders of the project.

Joe Gregory[1], Lucy Berthoud[1], Theo Tryfonas[2], Ludovic Faure[3]

1: Department of Aerospace Engineering
2: Department of Civil Engineering
University of Bristol, Queens Building, University Walk, Bristol BS81TR, UK

3: Airbus Defence and Space, 31 Rue des Cosmonautes, 31400 Toulouse, France

Applying the 'Spacecraft Early Analysis Model' to the Biomass Mission

Model-Based Systems Engineering (MBSE) represents a move away from the traditional approach of Document-Based Systems Engineering (DBSE), and is used to promote consistency, communication, clarity and maintainability within systems engineering projects. In previous work, industry focus groups have indicated that one way this can be achieved is by performing early functional validation of elements of the spacecraft avionics.

This paper presents an extended approach and model template, introduced in a paper previously published by the authors, to enable early functional definition and analysis of a spacecraft. The approach uses the 'Spacecraft Early Analysis Model' (SEAM), a SysML-based model framework for the definition, development and analysis of a space-based mission and corresponding space system. In using this model, the traditional Mission Operations Concept Document is replaced with a model-based representation of the design that can be executed, interrogated and quantified. The objective of this model template is to improve the clarity, consistency and quality of the design information, and to structure this information in such a way as to enable the high-level simulation of the design much earlier in the system life cycle. This approach focusses on the definition of the concept of operations during Phase B of the spacecraft system lifecycle.

The SEAM pulls together different, traditionally disparate, analysis tools and enables them to work together, producing an integrated system model spanning multiple tools. It facilitates the definition and simulation of the mission using dedicated orbit modelling software System Tool Kit (STK), complex mathematical analysis using MATLAB, spreadsheet-based data manipulation using Microsoft Excel, and can be extended to incorporate IBM DOORS for the handling of requirements. At its core, the SEAM utilises Cameo Systems Modeler (by No Magic).

The structure of the core SysML-based model builds on the principle described by Stephane Estable in the 'Federated and Executable Models' approach – the preservation of separation between the mission definition and the system definition. The SEAM builds on this by introducing a third distinct layer: the operational definition. Maintaining separation between these three aspects of the model allows for greater flexibility of modelling and clarity when looking to analyse, modify or validate the mission, operations, and system definitions. The SEAM uses a complementary systems engineering methodology to derive appropriate functional and logical architectures.

The SEAM has been developed iteratively by applying it to case studies taken from real spacecraft under development by Airbus, refining the capabilities of the template accordingly, and subsequently generalising the model. The resulting version of the SEAM contains multiple reusable and customisable MBSE patterns that will ultimately provide users with a comprehensive, consistent and intuitive SysML-based structure to follow when applying the SEAM to a specific mission.

The case study presented herein focusses on the Biomass mission – an ESA-led, low-Earth orbit, Earth-observation mission due to be launched in 2022. The primary mission objectives are to determine the distribution of above-ground biomass in the world forests and to measure annual changes in this stock over the period of the mission. To achieve these objectives, a P-band (435 MHz) Synthetic Aperture Radar (SAR) has been selected as the payload. The Biomass space segment consists of a single low-Earth orbit spacecraft (Biomass) carrying the SAR instrument. The spacecraft will utilise a large deployable reflector, and this must be deployed during the early phases of the mission. This deployment process is an example of a critical sequence, characterised by an intricate decision-making process and subject to a complex relationship between the ground and space segments where communication can be limited. The MBSE approach adopted enables the definition and analysis of this critical sequence, pulling together multiple analysis tools to analyse the design of the system and the concept of operations, generate a deployment sequence timeline, and assess this against the mission needs.

The preliminary results of this work demonstrate that the deployment timeline is heavily influenced by the orbit chosen (which affects the availability of communication windows). In fact, the spacecraft is functionally active for only ~20% of the total time required to complete deployment. A significant amount of time is spent establishing communications windows and making continuation decisions on the ground. The case study has successfully demonstrated the SEAM's ability to model critical sequences and validate this spacecraft functionality and the concept of operations against the mission needs.

Next steps in the development of the SEAM include its application to a wider variety of case studies and missions to develop and demonstrate its versatility, and the development of metrics to measure its perceived value among practitioners. For example, the SEAM has also been applied to ExoMars, a Mars rover mission due to launch in 2022. Future applications may include constellations and crewed missions.

This project has received funding from the UK Engineering Physical Sciences Research Council and Airbus, grant no. 16000151. The authors would like to acknowledge support from Alexandre Cortier, Stephane Estable, Thomas Fenal, Joanna O'Rourke, Antonio Prezzavento and Alain Rossignol of Airbus.

MBSE APPROACH APPLIED TO LUNAR SURFACE EXPLORATION ELEMENTS

Jasmine Rimani [a], Stéphanie Lizy-Destrez [b], Jean-Charles Chaudemar [c], Nicole Viola [a]
[a]Department of Mechanical and Aerospace Engineering, Politecnico di Torino, Torino, Italy, *Email: name.surname@polito.it*
[b]Department of Aerospace Vehicles Designs and Control, ISAE-SUPAERO, Toulouse, France, *Email: name.surname@isae-supaero.fr*
[c]Department of Complex Systems Engineering Department, ISAE-SUPAERO, Toulouse, France, *Email: name.surname@isae-supaero.fr*

***Abstract*—The last two decades have shown that among the new drivers of the design of space systems the level of autonomy is a key element to ensure the success of a mission. The final aim is to monitor and direct the operations or counteract unforeseen events as efficiently as possible, even without the man in the loop. To effectively accomplish these new tasks, the decision making layer of the spacecraft should be able to evaluate the available resources and the overall state of health of the system. The Model-Based System Engineering (MBSE) framework can help to understand the general behavior of a complex system as it is an autonomous space platform. The MBSE scheme exhibits the links and the interdependency between the different phases of mission analysis and between the components. The study proposed in this paper follows the MBSE methodology to design an autonomous guidance, navigation, and control (GNC) subsystem of a planetary exploration rover and its collaborative drone. The study starts from the high-level requirements of a lunar exploration mission and ends with the preliminary design of a state-machine, that describes the behavior of an autonomous GNC. To ensure a high level of autonomy, the decision-making layer of the GNC takes into account the outputs of the failure detection, identification, and recovery (FDIR) subsystem and the overall health state of the rover. The FDIR subsystem embodies the idea of a multidisciplinary design where different inputs should be managed to ensure the safety of the overall system under study. The novelty of this analysis lays in using the MBSE to define the design box of the autonomous GNC. The logic behind the MBSE enables the designer to keep track of the effects of the high-level mission-related decisions and of the FDIR on the overall behavior of an autonomous GNC subsystem.**

In the application presented in this paper, the preferred mean to study the mission and behavioral analysis is MBSE software *Genesys 7.0* of Vitech Corporation [1]. While the state machine and the related artificial intelligence algorithms are designed in *Robot Operating System* (ROS). The described approach is applied to the case study of a collaborative rover and drone on the lunar surface. The mission is designed as a ”precursor mission” to assess the safety of the lunar lava tubes as possible future human settlement.

I. INTRODUCTION

In the specific case study presented in this paper, the reference mission aims to explore the lunar lava tubes. More in detail, the preferred target is the lava tubes' *skylight* locate in Marius Hills, in the equatorial zone of the Moon. The pit gives access to a lava tube at fifty meters below the surface of the Moon, which can be used as future human settlement [2]. The logic flow starts with the definition of the high-level requirements for the lava tubes exploration mission as (i) assure the safety inside the lava tubes, (ii) map the environment outside and inside the lava tubes, (iii) communicate with Earth. The rover should accomplish to requirement (iii) with the help of a relay satellite, while the piggy-back hopping/flying drone aid to fully accomplish the first and second tasks. The system context is presented in fig.1. It defines the design box of the rover. In the ”Moon Environment” box of fig.1 the qualitative and quantitative impact of radiation, temperature, and illumination are defined. In the ”Satellite” box the planning of the available communication windows is inserted and it defines how long the system is required to be completely autonomous before contacting Earth. The ”drone” box contains the components and the functions that permits to augment the rover exploration capabilities. The ”rover” box is refined by the definition of its subsystems and their functions. Hence, the study progresses with the functional analysis and the identification of the main components for each subsystem of the rover and the constraints placed by the environment for each of them. The main subsystems considered for the rover are: (i) the mobility and GNC system; the structure and mechanism; (ii) the passive thermal control system; (iii) the power system; (iv) the communication system; (v) the command and data handling system; (vi) the payload that comprehends the drone and the scientific instrumentation to map the lava tubes from outside. This logical set-up phase is performed using the MBSE software *Genesys 7.0* of Vitech Corporation, [1]. The *Genesys 7.0* helps to describe the mission and the layers dependency. Actually, each component in the architecture is associated with a function and at least on state. Moreover it can be included in an operational mode, or it can cause a risk. The ”linguistics” links used in the software ease the comprehension of the system dependencies. At the end of the study, a preliminary functional architecture of the autonomous navigation task can be obtained. This functional scheme can be exported as a Simulink model and it constitutes the skeleton of a preliminary state machine in Python to study the impact of the failure propagation of other hazards on the GNC system, [3], [4]. The state machine is then refined with a python code exploiting ROS (Robot Operating System), as middleware, and Gazebo, as the main simulation environment. The use of ROS enables an easier transition from the simulation to the real platform testing of the failure, detection, identification, and recovery algorithms. The methodology supported by Vitech *Genesys 7.0* is called STRATA, an abbreviation of strategic layers [5]. As suggested by the name, the process is based on layers that gradually becomes more detailed at each design iteration [6]. The approach defines a first sizing box starting from the high-level requirements, constraints and boundaries. Usually, this first structure is called ”system context”, fig.1. After delimiting the design environment, the process focuses on the definition of the expected behaviours: what the system in the analysis should do and how well. This logical flow leads to the definition of the subsystems, or components, that can comply with the expected behavior. At the end of the loop, the overall architecture is verified against the expected performances and validated against the requirements. The process is then replicate with an increasing granularity up until the design team is satisfied with the results. Each new ”layer” starts from the outcomes of the previous one [5]. In our case, the STRATA methodology is particularly appealing for its intrinsic characterization of constraints in an early stage of the design [5]. A good and clear picture of design boundaries helps understanding which can be the behaviours to avoid and which are the related risks. Therefore, the STRATA framework helps to develop the right mindset to analyse the behaviour of systems in contingency situations. Any change in the boundaries and constraints affects the complexity of the system under study and how it interfaces and interacts with the ”system context”. [6]. Similar study on a fault-tolerant or reconfigurable GNC have been presented in [4] and [7]. However, the main novelty of this project is the development of GNC algorithms and design boxes, keeping in the loop the mission requirements, functions, and operations. The point of view is that of the integrated health system management, where a decision at functional level may have a great impact on the component level.

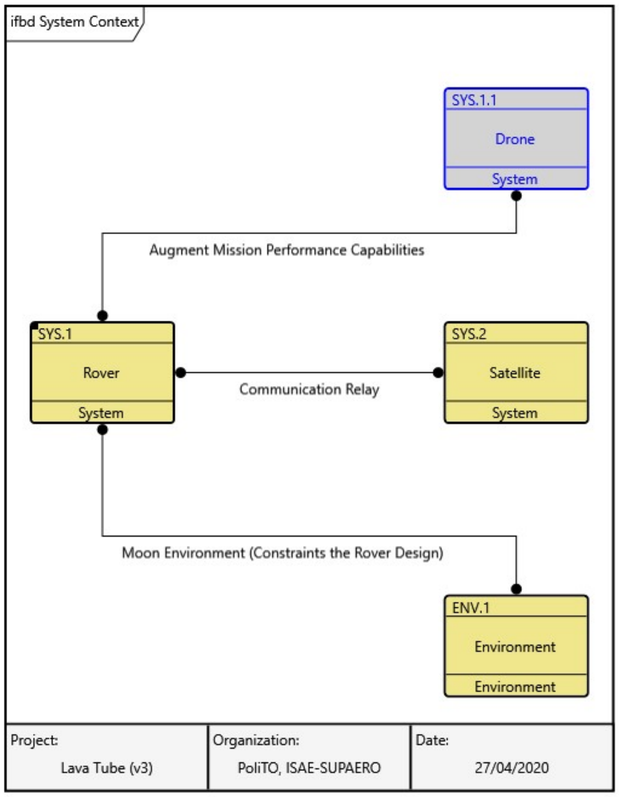


Fig. 1. Lunar exploration rover interfaces between the environment and other subsystems in Genesys 7.0 [1] analysis framework.

II. Project Overview

In the previous section, it was briefly explained how the rover interacts with the other systems in the mission design and which tasks it is expected to perform. This section presents

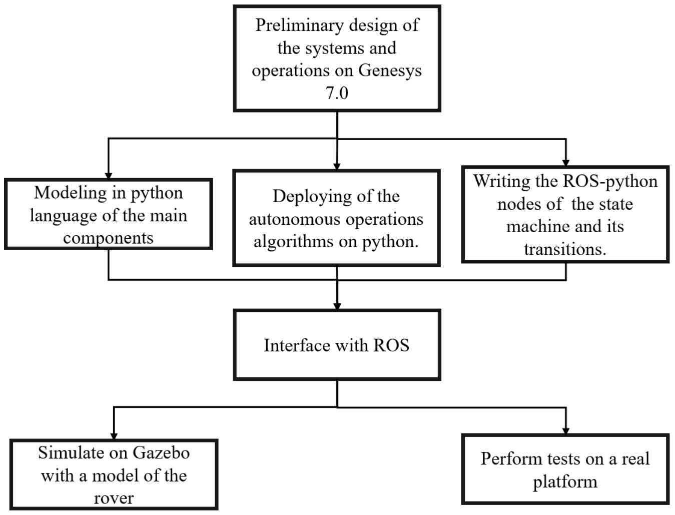

Preliminary design of the systems and operations on Genesys 7.0
Modeling in python language of the main components
Deploying of the autonomous operations algorithms on python.
Writing the ROS-python nodes of the state machine and its transitions.
Interface with ROS
Simulate on Gazebo with a model of the rover
Perform tests on a real platform

Fig. 2. Interaction between Genesys 7.0, ROS and python language modelling.

the autonomous navigation functions and how they are linked together. The main two assumptions of the analysis presented can be summarized as: (i) in between the communication windows the rover should be completely autonomous; (ii) the modes of operations consider only the Moon surface operation and not the launch, traverse, landing and disposal operations. The surface mission operational modes are derived with the help of [8] and [9]. In our specific case, the second requirement of mapping the surroundings of the lava tubes generates the need for a ”traverse mode”. During this mode most of the power is dedicated to generating a map, to compute the rover trajectory and effectively move the rover. Therefore, it was identified as the most demanding scenario for the GNC. Inside the ”operations” block, it is possible to define which faults can affect the system during its traverse on the lunar soil. These faults can then be associated with a ”risk” or a ”constraint” that affects the functional level of the analysis of the autonomous navigation architecture. The traverse related faults have been identified as goal errors (off-track) and system-related errors (one of the system parameters is off track) [10] [11]. From the most common ”faults”, it is possible to understand which are the important sizing parameters for the GNC: (i) the power available for the mobility system (that is limited by the battery’s voltage, current, temperature and charge level); (ii) the terrain characteristics that impact the wheel slipping and the wheel sinkage; (iii) the overall weight of the rover; (iv) the wheels motor available torque; (v) the maximum traversable obstacle height; (vi) the steering characteristics; (vii) the goal velocities, (viii) the typical drifting from the global planned trajectory during ”dead reckoning” navigation. These characteristics are coupled with the functional analysis of the autonomous navigation task following the guidelines of [12] and [13]. The first level functions are affected by the concerns raised by the goal and the system-related errors. The identified high level functions are: (i) map generation; (ii) global path planning; (iii) rover localization; (iv) local path planning computation; (v) obstacle avoidance; (vi) trajectory control; (vii) path execution; (viii) resources estimation. In the ”resource estimation” block, the health of the overall rover, and its effects on the GNC are defined, eg. power level. The preliminary scheme of high-level functions and their connections for the *traverse mode* is presented in fig.3. The *Genesys 7.0* output is used as input for designing the hierarchical state machine in *ROS* to study the impact of FDIR on the GNC. Each first-level function is defined by a series of tasks. Therefore each block is a state machine per se in which the output influences the overall autonomous navigation behavior. The layered approach of STRATA [5] helped to define the functional interfaces and the physical links needed to understand the impact of failures and degradation on the rover during traverse operations. The different levels of detail aided with the understanding of the overall behaviour without detailing each component of the GNC subsystem. Actually, following the flow of requirement-behaviour-component, it was easier to identify which component needed to be modelled to simulate nominal and contingency scenarios while studying operations. The overall simulation framework is based on python’s language. The components with their state equations have been defined as classes following the inputs-output flow defined in *Genesys 7.0* through interfaces and links. The definition in python classes is useful to immediately cross-check the logical flow with the one defined in *Genesys 7.0* and to easily set up the python nodes that communicate through ROS toward the real or the simulated rover, fig.2. In the end, the *Genesys 7.0* model output the inputs for the component design, the algorithms to estimate the best path based on the resources of the system and the state machine modes and functions. All those python-based classes are then build up to constitute a ROS node and interfaced directly to simulation and test, future work. The python code is then interfaced with ROS to send command, to simulate the failure or degradation of various subsystems and see the overall impact on the operations. These simulations are then used as feedback in the design to see if the architecture matches the expected behaviour during contingency situations.

III. Conclusions

The study presented in this extended abstract follow the logic of MBSE to design an autonomous GNC system starting from the mission requirements. Exploiting The *Genesys 7.0* software has an internal diagnostic tool to verify that all the objects, instance in the database are justified, rightly connected and make sense. as an analysis platform, it is possible to ensure the traceability and the impact of high-level decisions on the component and functional levels. The first step is to give a system context to the rover. Then the operational modes and the functional analysis at system and subsystem level are conducted in order to define the expected outputs and the concerns associated with the autonomous navigation task. The case study of a mission for the exploration of the lunar lava tubes is used to explain the logical process. In this analysis, the ”traverse” operational mode is investigated as well as the tasks related to the autonomous GNC and the related faults. Eventually, the output of this analysis is a preliminary layout of the hierarchical state machine that can be implemented in ROS and simulated with Gazebo or tested in the robotic laboratory. The MBSE scheme adopted in the project has helped the understanding of high-level boundaries and constraints at the very begging of the mission definition. Therefore, it was useful and crucial to understanding which contingency situations were interesting to study from an operational point of view. Moreover, it helped the definition of the inputs and outputs of each GNC function in the traverse mode and the related components. It eased the definition of the software architecture used during the simulations and the analysis of the operational layer of the rover. The most significant difficulty lies in the change of point of view: it was difficult to adopt and understand the logic of STRATA methodology at first. However, this approach helped understand which components where vital and which can be doubled in their use to keep on with the mission even during contingency situations. Overall, the management of the multidisciplinarity typical of MBSE has been of great asset in the study. The future work will focus on the three main branches for both the rover and the drone: the mission analysis, the study of failures and faults, and the study of the autonomous GNC. These three ingredients are highly intertwined together to assess fully autonomous operations. The aim is the creation of a comprehensive design framework to study the autonomy of surface robotics systems. More in detail, the simulation outcomes will be fed back to the MBSE model to verification that the real performances match the expected ones. The direction is to continuously iterate between the early design layer and the output on the behaviour of the system to derive sizing rules or good practices to define the autonomy level and better the performances of the system during contingency operations.

References

[1] Vitech Corporation. Genesys: Enhancing systems engineering effectiveness, 2020.

[2] T. Kaku et al. Detection of intact lava tubes at marius hills on the moon by selene (kaguya) lunar radar sounder. *Geophysical Research Letter*, 2017.

[3] Darwish A. Abdelghafar S Hassanien, A.E. Machine learning in telemetry data mining of space mission: basics, challenging and future directions. *Artificial Intelligence Review*, 2019.

[4] Peter Zane Schulte. A state machine architecture for aerospace vehicle fault protection. *PhD Dissertation, Georgia Institute of Technology*, 2018.

[5] Zane Scott David Long. *A Primer for Model-Based System Engineering*. Vitech Corporation, 2011.

[6] Brian London. A model-based systems engineering framework for concept development. *Master of Science Dissertation, Massachusetts Institute of Technology*, 2012.

[7] Edward Balaban et al. A mobile robot testbed for prognostics-enabled autonomous decision making. *Annual Conference of the Prognostics and Health Management Society 2011*, 2011.

[8] Lancaster R. Clemmet J. Silva, N. Exomars rover vehicle mobility functional architecture and key design drivers, 2013.

[9] Hélène PASQUIER (Editor). *Space Operations: Inspiring Humankind’s Future*. Springer International Publishing, 2010.

[10] Mark Maimone. Challenges for planetary rover navigation.

[11] Stephen B Johnson (Editor). *System Health Management: with Aerospace Applications*. Wiley, 2011.

[12] Yang Gao (Editor). *Contemporary Planetary Robotics: An Approach Toward Autonomous Systems*. Wiley, 2016.

[13] Alex Ellery. *Planetary Rovers*. Springer-Verlag Berlin Heidelberg, 2016.

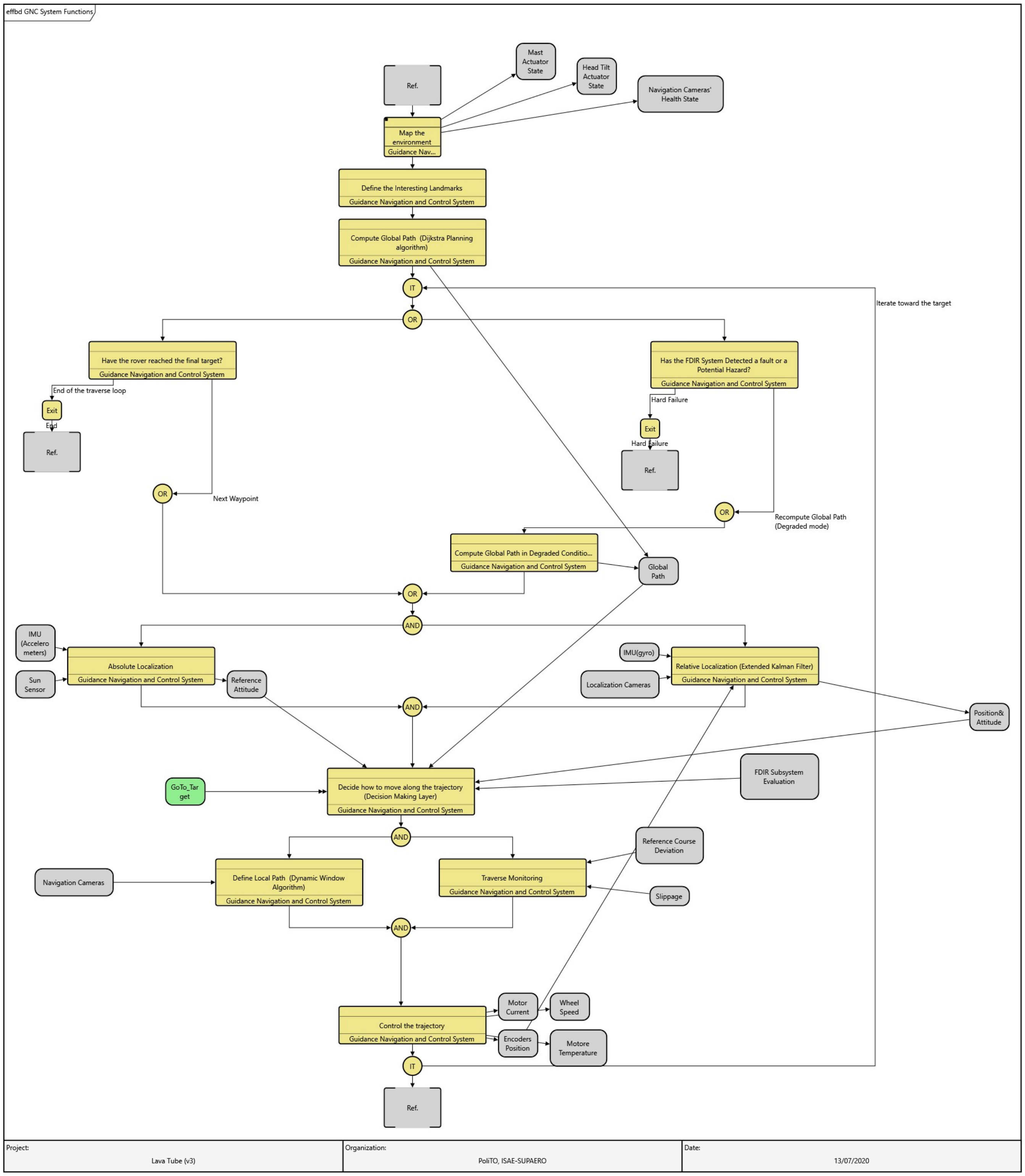

effbd GNC System Functions
Ref.
Mast Actuator State
Head Tilt Actuator State
Navigation Cameras' Health State
Map the environment
Guidance Nav...
Define the Interesting Landmarks
Guidance Navigation and Control System
Compute Global Path (Dijkstra Planning algorithm)
Guidance Navigation and Control System
IT
OR
Iterate toward the target
Have the rover reached the final target?
Guidance Navigation and Control System
End of the traverse loop
Exit
End
Ref.
Has the FDIR System Detected a fault or a Potential Hazard?
Guidance Navigation and Control System
Hard Failure
Exit
Hard Failure
Ref.
OR
Next Waypoint
OR
Recompute Global Path (Degraded mode)
Compute Global Path in Degraded Conditio...
Guidance Navigation and Control System
Global Path
OR
AND
IMU (Accelero meters)
Sun Sensor
Absolute Localization
Guidance Navigation and Control System
Reference Attitude
IMU(gyro)
Localization Cameras
Relative Localization (Extended Kalman Filter)
Guidance Navigation and Control System
Position& Attitude
AND
GoTo_Target
Decide how to move along the trajectory (Decision Making Layer)
Guidance Navigation and Control System
FDIR Subsystem Evaluation
AND
Reference Course Deviation
Navigation Cameras
Define Local Path (Dynamic Window Algorithm)
Guidance Navigation and Control System
Traverse Monitoring
Guidance Navigation and Control System
Slippage
AND
Control the trajectory
Guidance Navigation and Control System
Motor Current
Wheel Speed
Encoders Position
Motore Temperature
IT
Ref.
Project:
Lava Tube (v3)
Organization:
PoliTO, ISAE-SUPAERO
Date:
13/07/2020

Fig. 3. Autonomous GNC high-level functions during the "traverse" operational mode using Genesys 7.0 [1]. The grey and green boxes are the inputs or outputs to each behavior, while the yellow boxes comprehends the high-level function. The "Ref." at the start and at the end of the logical flow indicates the starting and the ending of the traverse mode functions.

Successful MBSE landing on a CNES operational use case

Jonathan LASALLE and Benoit VIAUD
ARTAL Technologies, 1 rue Ariane, 31520 Ramonville-Saint-Agne, {jonathan.lasalle, benoit.viaud}@artal.fr

Martine JOURET, Anthony JUIN, Fabienne SCHAFFHAUSER and Raymond SOUMAGNE
CNES, 18 avenue Edouard Belin, 31400 Toulouse {martine.jouret, anthony.juin, fabienne.schaffhauser, raymond.soumagne}@cnes.fr

1 Introduction

The Space Variable Objects Monitor (SVOM) is a space system dedicated to gamma ray detection and study, under development by China National Space Administration (CNSA) and the French Space Agency (CNES), to be launched in 2021. The system shall be able to trigger alerts of Gamma Ray Burst (GRB) in real-time with a maximum of associated data. The space segment consists in a set of sensors going from large angle of view for detection to narrow angle of view for data measurements. Since GRB are very transient events, it requires the satellite to autonomously (i.e. without communication with ground) point on target the different sensors that, each in turn, provide more accurate position and data. As an addition to the system scientific and technical challenges, the organisation of the system operation by the two agencies introduces some more complexity.

The design of this system was conducted within the framework of the CNES engineering process, based on a set of documents cascading the textual requirements from the high-level concept of operations to the technical specification of equipment. The validation of the obtained specification mainly relies on human expertise and on the validation campaign.

The complexity of the system makes it a perfect candidate for an experimentation of MBSE. This paper presents the results of a study that has been led after the design has been already defined but while the system was still in development and the topic still fresh in the heads of the architects. The study tried to assess the benefits that MBSE could bring in this specific context.

2 MBSE-oriented objectives

Why injecting the MBSE methodology inside an existing process that proved its efficiency several times? Three main objectives are often associated to MBSE:

1) Communicate: to improve the communication between stakeholders by using a rigorous and yet reader-friendly language, and thereby reducing ambiguities.
2) Secure: to assist the system definition validation by using traceability and coverage mechanism to ensure consistency, completeness…
3) Generate: to take advantage of the formal description of the system to generate engineering assets (documents, code, database schema, etc.) or to assist the specification refinement by automatically initializing sub-level representations.

The current fully-operational CNES engineering process can thus be potentially improved, along these axes, by injecting a pinch of MBSE on it. Based on this conjecture, two projects took place successively. The first one was an R&T study, dedicated to the analysis of the current process and the evaluation of the potential benefits that MBSE could bring. Due to promising results, a second project, based on the models realized during the first study, was dedicated to the operational capture of the system validation.

Artal worked in close collaboration with the CNES in order to provide its MBSE expertise to the SVOM project and to CNES specialists. The MBSE activities of these projects were realized using the Capella tool [2], an open-source graphical modeller based on the Arcadia Method [1] (Arcadia is a model-based engineering method that defines high-level concepts). Capella is mainly based on four representation layers, dedicated to the system needs capture (Operational Analysis (OA) and System Analysis (SA) layers) and to its associated solution specification (Logical Architecture (LA) and Physical Architecture (PA) layers). The different representation layers are linked together in order to being able to apply traceability and coverage mechanisms.

3 MBSE-driven interface engineering

The first step consisted in analysing the CNES engineering process through the in-progress SVOM case. The main goal was to identify the capability and the relevance of capturing the system specification using Capella. In order to guide the modelling activity, we decided to focus on the interfaces specification. Indeed, it is a crucial step in the design of a complex system and the international collaboration context called for even more rigor in the definition process.

By analysing the existing specification of the system, associated to several co-modelling sessions, we were able to capture in the Capella model almost all the system description. We captured the main objectives of the system (using System Analysis (SA) layer) by specifying its interactions with external actors. We then obtained a quite bright view of the public interfaces needs.

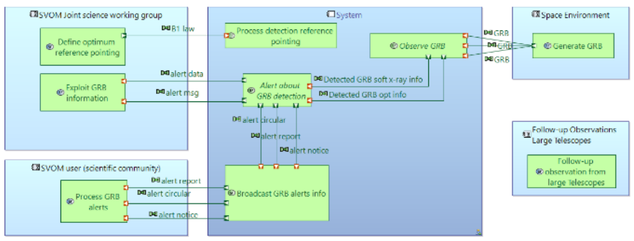

SVOM Joint science working group
Define optimum reference pointing
Exploit GRB information
System
Process detection reference pointing
Observe GRB
Alert about GRB detection
Broadcast GRB alerts info
Space Environment
Generate GRB
Follow-up Observations Large Telescopes
Follow-up observation from large telescopes
SVOM user (scientific community)
Process GRB alerts
R1 law
alert data
alert msg
Detected GRB soft x-ray info
Detected GRB opt info
alert circular
alert report
alert notice
GRB

Figure 1. Partial System Analysis of the SVOM system

Then, using the Logical Architecture (LA) layer, we captured the internal system definition by representing all sub components, their functions, and their associated exchanges.

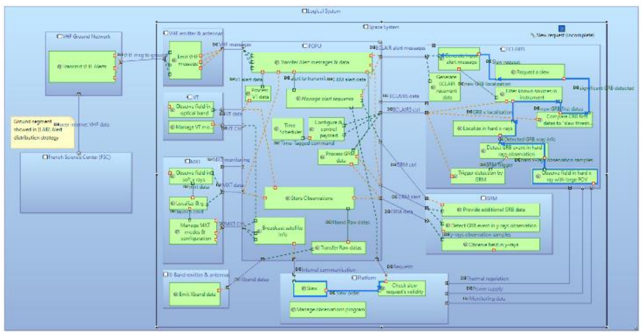


Figure 2. Partial Logical Architecture of the SVOM system

In order to address the interface engineering goal, different strategies were identified. It is possible to simply add textual description on the exchanges or on the associated communication ports. It is particularly relevant in case of subjective interfaces or if the interface detail is not required Otherwise, it is possible to refine functional exchanges by capturing the data structure associated to them. Then, it offers a clearer and more complete representation, whose only limits are those of the modeling language.

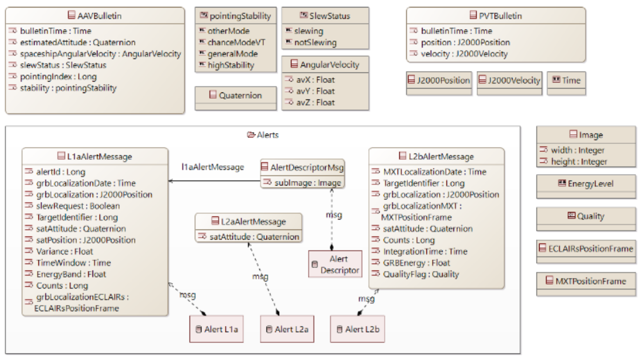


Figure 3. Data structure definition

Following the capture of the system itself, we studied the means to capture the specification of the simulator of a sub-part of the system. Using Capella internal tools, a new model, inherited form the original one, was initialized in order to derive the architecture specification into its associated simulator specification while maintaining traceability links.

Using the analysis of the obtained model and considering the three MBSE objectives defined in the section 2, the following conclusions have been drawn:

1) Considering the communication goal, the MBSE process gave us, in this context, a promising communication structure and a formal specification of the interfaces. Nevertheless, the SVOM project being in progress (and the interfaces specifications being already captured using the historic CNES process), it was not possible to clearly evaluate the capital gain.

2) As regards to the secure objective, the traceability links between the LA and the SA Capella layers gave us direct evaluation of the coverage of the capabilities by modelling items. The capture of interface detail also provide controls about the completeness of the specification (an exchange without associated data structure has to be completed).

3) Regarding the generation process, using the M2Doc tool (that allow the generation of Word documents including modelling items), we were able to mainly generate the traditionally manually filled document, the non-formal schema of the original document being replaced by formal Capella diagrams.

Due to the encouraging results of this first project, a second one, in an operational field, was dedicated to the capture of the V&V specification.

4 MBSE-driven V&V

Based on the models realized during the first project, the goal was to specify the V&V objectives and the corresponding tests sequences using modelling activities. As references, some test procedures of other CNES project were analysed and working session gathering Artal and the CNES allowed the identification of the V&V modelling needs. A dedicated Artal Capella viewpoint (called VVO) was customized to answer these needs.

First of all, the validation needs must be expressed by defining Functional Chains (succession of functional exchanges), each one representing one behaviour of the system to validate. Then, the definition of global validation objective (VVO) allows to groups them. For example, to validate the communication between two components (example of VVO), it will be necessary to satisfy a set of validation needs (e.g. all the possible connections between these components). In the context of a VVO, each function chain can then be derived in order to convert "abstract test objective" into "concrete test sequence". This step allows to specify the executable version (boxed in Figure 4) of a part of the validation sequence to be simulated (in purple in Figure 4).

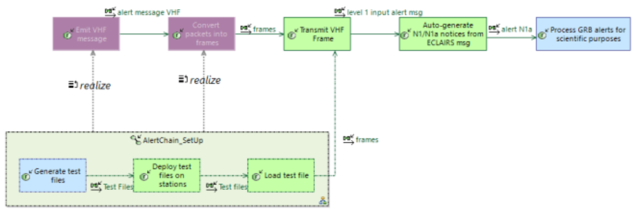


Figure 4. Concretization of the test

Using this data, an embedded tool allows the generation of a test sequence that can be annotated in order to specify the interactions steps and the success criteria to be manipulated by the test operator. Each step or criteria can embeds configuration parameter that will be valued during the test sequence instantiation.

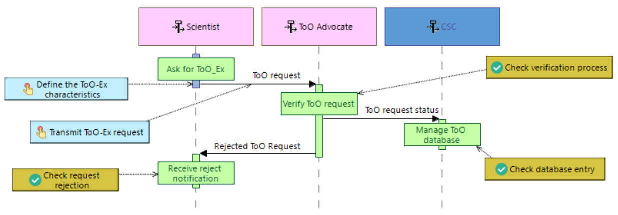


Figure 5. Annotated concrete test sequence

Using this toolchain, it was possible to capture all required validation data, the evidence being that all the required V&V specification document were fully generated from the model. Indeed, all along this collaboration, the MBSE has gained ground gradually. Initially, it was experimented in parallel of the classical process, in order to prove its worth. Then we planned to gather the two "ways of working" by generating, from the model, the document usually manually filled. The proof having been provided, progressively, the CNES engineers relied on the model and used it as data reference to conceive the V&V data, which were then integrated in the model. Finally the writing of the operationally used V&V specification document was fully delegated to the implemented tools.

The SVOM experts, MBSE and Capella inexperienced people, received this new process positively and were unanimous regarding the benefits of such approach. The operational gain was notable thanks to the strong stakeholders' involvement in this project and the real consideration of the model as the specification reference.

5 Going further

Around this main flow, several "on-the-edge" points were considered. First of all, we confronted the model and the 579 textual requirements in order to evaluate their overlap. Less than half of them can be strongly linked to the model (either fully covered by it or completing it, by adding performance constraints for example), the others being either too technical or, on the contrary, too abstract. An independent and autonomous requirement engineering process remains then needed and cannot be fully integrated to this described MBSE process.

Another point consists in the managing of specification version. In the original CNES process, the produced documents themselves embed their version and the change tracking report (manually filled). To transpose such capability in the MBSE world requires to being able to support such feature:

- The versioning of each stage of the model by saving the model stable copies.
- Storing the description of the changes associated to a new model version in order to facilitate impact analysis and to carry out reviews on a limited scope.
- Tracking the author and the modification dates.

The usage of some tools and connectors gravitating around the Capella platform (Github, Jira, Mylyn…) associated to the suitable method seems to be a satisfying answer.

Finally, in order to ensure a complete data continuity along the development process, it would be necessary to link the experimented modelling phase with the following steps namely the system building including software implementation. Concerning such goal, only a small incursion concerning the link with the satellite database was achieved. Starting from the Capella model, we well generated a skeleton of it (which has to be filled manually). Based on the “Mapping” API, this demonstrator supports iterative processes, in other words : allowing to progressively update the database content according to the successive version of the Capella model, while allowing manual database edition in parallel. A specific interface being dedicated to conflict resolution.

6 Conclusion

The smooth incursion of MBSE in CNES engineering process was undeniably well received. The SVOM experts were converted to this new way of working. Even if their professional schedule were fully charged, they did not hesitate to invest time to completely follow this experience until the end. The building of an operational model-based toolchain to capture the VV specification is an achievement which opens the door to a wider reach of MBSE within CNES. The data continuity is a powerful help in order to track inconsistence and to compute impact.

The three identified MBSE pillar seems to enter into resonance with the CNES needs:

- The communication between engineers will be lightened while remaining rigorous.

- The specification process will be secured thanks to the generalization of data continuity including a strong link between the validation specification and the system under test specification itself.

- All required documents will be automatically generated from the model, avoiding time waste in the heavy task of writing document.

Capella perfectly answered the CNES needs in this context and could be easily incorporated in a larger engineering framework to cope with transversal engineering concerns that rapidly arise.

Acknowledgement to SVOM project including CNES and SECM

References

[1] VOIRIN, Jean-Luc. *Model-based System and Architecture Engineering with the Arcadia Method.* Elsevier, 2017.

[2] ROQUES, Pascal. *Systems Architecture Modelling with the Arcadia Method: A Practical Guide to Capella.* Elsevier, 2017.

Model Based Space Systems and Software Engineering Workshop

-MBSE 2020

Title: International Cooperation on Model Based Development For Spaceflight Assurance: The TACS Test Case

Authors: Isabelle Conway, Silvana Radu, ESA

Lui Wang, John W. Evans, NASA

Naoki Ishihama, JAXA

Michel Izygon, Tietronix

Arthur Witulski, Vanderbilt University

Martin S. Feather, Jet Propulsion Laboratory, California Institute of Technology

Abstract:

As this workshop attests, Model Based Systems Engineering (MBSE) is moving to the forefront of spacecraft development. The benefits of SysML® as language for the elucidation of the system architecture is well understood and is being demonstrated across programs, such as the NASA Europa Clipper currently in Phase C of the life cycle [1]. Concurrently, the benefits of the evolving development of MBSE for assurance have been recognized and are emerging as Model Based Mission Assurance (MBMA), which promises the development of integral assurance stakeholder views into the model as well as the production of useful products from the model [2,3]. In this regard, the assurance organizations of NASA, ESA and JAXA have setup the MBMA Task Force within the established trilateral Safety Mission Assurance (SMA) working group to explore jointly the potential benefits of MBSE and MBMA in anticipation of future joint projects in which an architecture for a flight mission will be shared in a SysML model. This paper presents the goal and content of this cooperation and reports upon current results.

The cooperative project goal is to develop a model based mission assurance reference model suitable for representing faults and failures and allow automatic generation of Reliability Availability Maintainability and Safety (RAMS) analyses. To ground this effort, the project is using a CubeSat as a target system. Figure 1 represents this CubeSat, dubbed the Trilateral Assurance CubeSat (TACS). The base model for the project was derived from the INCOSE CubeSat standard model built in Magic Draw for demonstration purposes. The project model was derived at Johnson Space Center as the lead organization, with ESA and JAXA as international partners in the design. The preliminary TACS model has been shared among our agencies in order to ensure a common set of requirements, system architecture, functions, and failure modes. Specifically, the Trilateral MBMA Task Force has adopted the ESA ECSS Parts Failure modes Catalog (Annex G - ECSS-Q-ST-30-02C) and generic failures identified in

CubeSats for assigning faults to the system components. The Trilateral Task Force reviewed and refined the model and approach, and came up with an initial mission assurance meta-model.

It has long been recognized that useful system products are forthcoming from SysML. The emphasis in this project is on the generation of fault management and reliability artifacts. These include Failure Modes and Effects Analysis (FMEA) and Fault Trees (FTA) based on early mission design, using TACS as an example. Leveraging on a consistent SysML model developed by the system design team using a NASA SysML Profile, the MBMA modeling framework extends the nominal system models and behaviors by adding failure modes and effects using a combination of SysML state machine and activity diagrams representations.

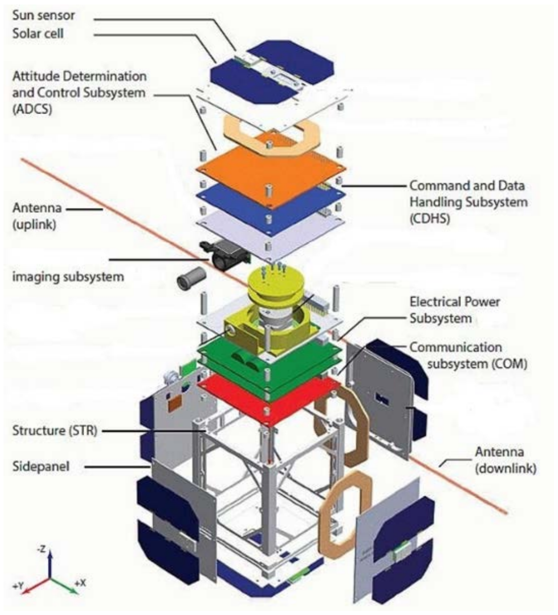

Sun sensor
Solar cell
Attitude Determination and Control Subsystem (ADCS)
Antenna (uplink)
imaging subsystem
Structure (STR)
Sidepanel
Command and Data Handling Subsystem (CDHS)
Electrical Power Subsystem
Communication subsystem (COM)
Antenna (downlink)
-Z
+Y
+X

Figure 1, TACS CubeSat - adapted from ESTCube-1.

In this approach, state machine diagrams represent the possible transitions between nominal and faulty states of the system's components, together with the effects those faults have upon the components' functions. Figure 2 illustrates interactions among state machines that capture the transitions between the nominal <ON> state to off-nominal <Failed> states using a combination of Signals, Activity and Guard Conditions.

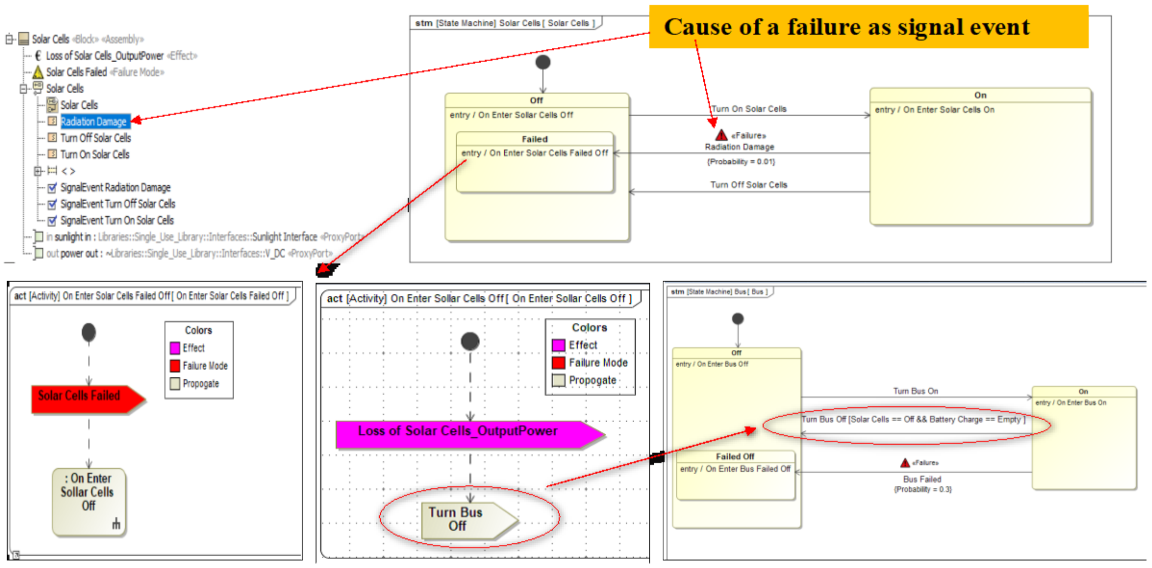
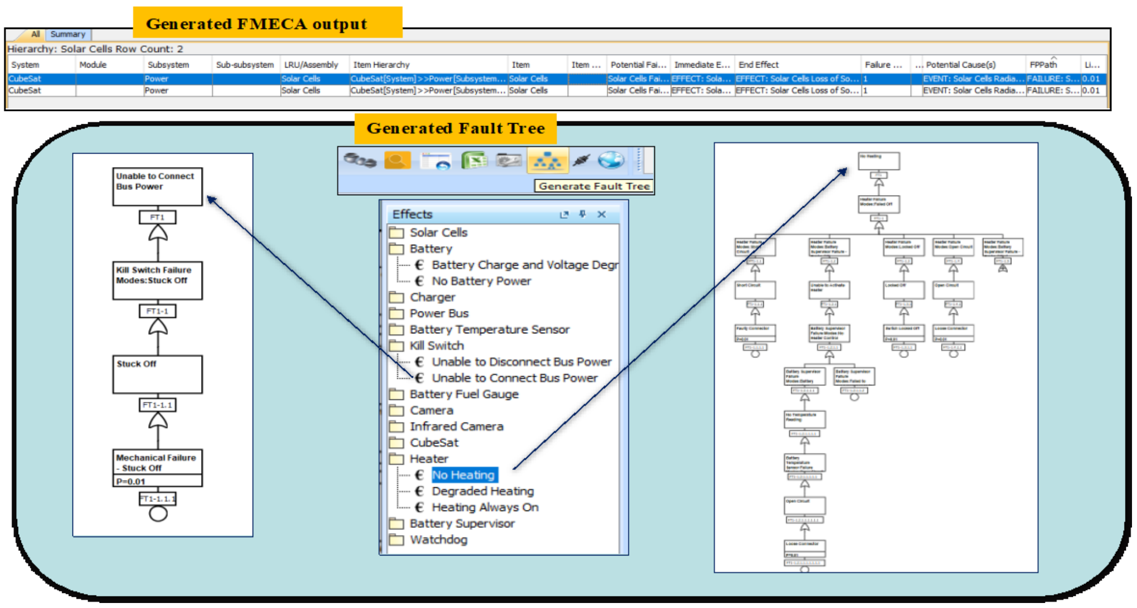

Solar Cells «Block» «Assembly»
Loss of Solar Cells_OutputPower «Effect»
Solar Cells Failed «Failure Mode»
Solar Cells
Solar Cells
Radiation Damage
Turn Off Solar Cells
Turn On Solar Cells
< >
SignalEvent Radiation Damage
SignalEvent Turn Off Solar Cells
SignalEvent Turn On Solar Cells
in sunlight in : Libraries::Single_Use_Library::Interfaces::Sunlight Interface «ProxyPort»
out power out : ~Libraries::Single_Use_Library::Interfaces::V_DC «ProxyPort»
stm [State Machine] Solar Cells [Solar Cells]
Cause of a failure as signal event
Off
entry / On Enter Solar Cells Off
Failed
entry / On Enter Solar Cells Failed Off
Turn On Solar Cells
«Failure»
Radiation Damage
{Probability = 0.01}
Turn Off Solar Cells
On
entry / On Enter Solar Cells On
act [Activity] On Enter Solar Cells Failed Off[On Enter Solar Cells Failed Off]
Colors
Effect
Failure Mode
Propogate
Solar Cells Failed
: On Enter Sollar Cells Off
act [Activity] On Enter Sollar Cells Off[On Enter Sollar Cells Off]
Colors
Effect
Failure Mode
Propogate
Loss of Solar Cells_OutputPower
Turn Bus Off
stm [State Machine] Bus [Bus]
Off
entry / On Enter Bus Off
Turn Bus On
On
entry / On Enter Bus On
Turn Bus Off [Solar Cells == Off && Battery Charge == Empty]
Failed Off
entry / On Enter Bus Failed Off
«Failure»
Bus Failed
{Probability = 0.3}

Figure 2, Electrical Power System (EPS)-Solar Cells State Machine & Activity Diagrams of failure due to Radiation

After enhancing the nominal system model with the failure information, FMECAs and FTAs can be generated automatically from the enhanced SysML model. Figure 3 illustrates the FMECA and FTA outputs and different failure effects can be interactively selected from a system component hierarchy and displayed in a graphical user interface within the MagicDraw® application.



Hierarchy: Solar Cells Row Count: 2

System	Module	Subsystem	Sub-subsystem	LRU/Assembly	Item Hierarchy	Item	Item …	Potential Fai…	Immediate E…	End Effect	Failure …	…	Potential Cause(s)	FPPath	Li…
CubeSat		Power		Solar Cells	CubeSat[System]>>Power[Subsystem…	Solar Cells		Solar Cells Fai…	EFFECT: Sola…	EFFECT: Solar Cells Loss of So…	1		EVENT: Solar Cells Radia…	FAILURE: S…	0.01
CubeSat		Power		Solar Cells	CubeSat[System]>>Power[Subsystem…	Solar Cells		Solar Cells Fai…	EFFECT: Sola…	EFFECT: Solar Cells Loss of So…	1		EVENT: Solar Cells Radia…	FAILURE: S…	0.01

Figure 3, TACS EPS FMECA & FTA Output

This modeling approach has been demonstrated by NASA on projects such as NASA Cascade Distiller System [4] and has been the basis for some efforts on the NASA Europa Clipper [5].

The plan for this effort is to present the recommendation of a meta-model for the representation of faults and failures at the Trilateral Safety and Mission Assurance Conference (TRISMAC) in June 2021 in Tokyo and to work towards standardization of the framework across agencies.

References

1. T. Bayer et al., "Europa Clipper Mission: Preliminary Design Report," 2019 IEEE Aerospace Conference, Big Sky, MT, USA, 2019, pp. 1-24, doi: 10.1109/AERO.2019.8741777.
2. J. Evans, S. Cornford and M. S. Feather, "Model based mission assurance: NASA's assurance future," 2016 Annual Reliability and Maintainability Symposium (RAMS), Tucson, AZ, 2016, pp. 1-7, doi: 10.1109/RAMS.2016.7448047.
3. M. Izygon, H. Wagner, S. Okon, L. Wang, M. Sargusingh and J. Evans, "Facilitating R&M in spaceflight systems with MBSE," 2016 Annual Reliability and Maintainability Symposium (RAMS), Tucson, AZ, 2016, pp. 1-6, doi: 10.1109/RAMS.2016.7448031.
4. M.J. Sargusingh, M.R. Callahan, S. Okon, “Cascade Distillation System Design for Safety and Mission Assurance,” 45th Int. Conf. on Environmental Systems, Bellevue, Washington, 2015.
5. Castet, J.F., Bareh, M., Nunes, J., Okon, S., Garner, L., Chacko, E. and Izygon, M., 2018, March. Failure analysis and products in a model-based environment. In 2018 IEEE Aerospace Conference (pp. 1-13). IEEE.

Acknowledgements: this research was carried out at ESA, NASA, JAXA, Tietronix, Vanderbilt University, and the Jet Propulsion Laboratory, California Institute of Technology under a contract with the National Aeronautics and Space Administration.

Experiences and Expectations with Model Based System and Software Engineering

Andreas Wortmann, Martin Beet, Dirk Roßkamp
OHB System AG, Bremen, Germany
<Firstname>.<Lastname>@ohb.de

A brief position paper summarizing obstacles and requirements on the application and introduction of model-based engineering from the perspective of and with an emphasis on (space segment) software engineering.

Projects for a long time have engaged ideas of model-based engineering in various flavors in order to optimize quality and efficiency of development. In general, there activities have been carried out in isolated applications not sharing data, structure and processes. At ADCSS 2016 some examples have been presented. They range from flight software development with Rhapsody in C, simulator software based on SMP, SMP2 and ECSS-SMP, hardware/software codesign for specific functions to requirements engineering and systems engineering with SysML.

Common misconceptions and obstacles

While MBSE is around for quite some time there are still quite some misconceptions present. From a language point of view most prominent is the term 'model' itself. With respect to MBS(S)E a model is not a simulator and a model does not refer to a spacecraft model like the EM, PFM or FM. Secondly, MBSE commonly is mistakenly put on a level with using UML or SysML. Many new terms with unclear or overlapping semantics have been introduced, including: digital clone, digital twin, digitalization, digital continuity and many more. Their use sometimes seems rather arbitrary.

The introduction of MBSE methods is a significant change in the way engineers interact and think their projects. It's a long-term process, but there is an implicit expectation that the Return of Invest is quickly achievable in short term. Furthermore, it's blinded to think that everything will be better, more efficient and cheaper, but we don't have to change our way of thinking and invest in related development processes. The assumption that all elements (e.g. software source code, configuration tables) with heritage can be continued to be used without modification in general is not true. Different tools and infrastructure call for different artifacts. On the other side, it's a misconception that replacing artifacts like software code with models inadvertently lead to loss of heritage and previous knowledge. In fact, the opposite is true: If correctly realized the prior knowledge and good design pattern are rigorously applied to all functional code (functions with pre-existing code and new functions) and the heritage is in reuse of implementation concepts and pattern rather than in source code. This is possible by raising the level of abstraction when actually implementing. With the SAVOIR/OSRA ESA is doing exactly this for harmonizing onboard software architecture.

Ongoing activities

At ESA a plurality of activities are carried out striving towards model centric engineering, including the MB4SE and OSMoSE initiatives as well as the various SAVOIR groups and the EGS-CC. At OHB such activities are supported and the internal organization and workflows are aligned. Preceding activities are identified and connect whenever possible in terms of processes and data flow. Various tools and approaches are assessed, including UML based modelling and domain specific languages, and serve a step-wise improvement of established processes and tools.

Essential Requirements

From the experience made so far, some basic requirements against an envisioned model-based engineering environment and platform can be drawn. These are presented in the following.

Todays' systems are large in terms of size and complexity and they are expected to grow even more in future. The engineering platform required must be capable to handle such large systems and keep up with it expected future growth.

Collaboration is essential in large engineering projects. A multitude of different engineering disciplines are working together on a shared model. Two needs arise that at a first glance seem contradicting: On the one hand an engineer requires a stable baseline to base his work on as continuous changes introduced by colleagues while working will stifle progress. On the other hand, an engineer always requires the latest information in order to not design the wrong system. In software engineering transaction-based collaboration tools like git have been introduced and solve these challenges very successfully. In cooperative systems engineering environments such tools will be beneficial as well. In addition to pure transaction based revisioning systems it must be possible (for instance in a concurrent engineering session) to collectively edit a model in a google docs style, where one actually can observe the colleague's cursor.

The tools user interface (UI) is important for acceptance and efficient operation. Overwhelming complexity with 1000-button menus for doing simple jobs are not suitable and error prone. The UI should be scalable with the use case and customized to the engineering task. While the user expects rigorous failure and consistency checks as well as simple analyses to be carried out interactively the tooling must remain live even with very large model being handled. A system that fades for 30 seconds while editing or that requires a "make" button to trigger long-lasting activities that put the engineer on hold will not be accepted in the long term. This specifically holds for models that represent executable systems and test cases.

It's important to model more domain specific aspects of a system than what is achievable when using UML/SysML as these are general purpose languages. While they can be extended via profiles etc. respective models are not very intuitive to read. Multiple paradigms and multiple notations will be required to optimize meta-models and languages for their application in their respective domain. This ranges prose-style (high level requirements) declarative (type system), behavioral (test, math expression) and structural (deployment) languages with textual (requirements), tabular (lookup), graphical (state machine, deployment) or symbolic (math, chemistry) notations.

Outlook

So, how could such an engineering framework look like? Tools including MS Excel and Word that currently are used for connecting artifacts from different engineering disciplines can easily be substituted by a model-based environment. But from the pure amount of highly specialized tools, it seems obvious that there will not be the one "BIG NEW TOOL" that will do everything. Established tools need to collaborate and share a common model. The focus should be on a shared model that can either be directly maintained or that external tools interface with, rather than on an exchange of models among external tool. Such integration calls for a data repository/hub that provides the mentioned collaboration features and it requires the specialized tools to provide interfaces that allow transaction-based or continuous exchange of data.

Due to the specifics of space engineering and peculiarities of the various stakeholder, it is not expected that a suitable out-of-the-box tool will ever be on the market. However, in a collective effort agencies and industry should be able to establish a customizable framework meeting the requirements.

MBSE-2020 workshop abstract submission

Abstract : "Experience Report: History and State of the Practice of Model-Based Software Engineering in Thales Alenia Space in France"

Authors: Régis De Ferluc, Marco Panunzio

Thales Alenia Space in France

Introduction

Model-Based Software engineering methods and tools have been used in Thales Alenia Space in France for more than a decade, benefiting from active and efficient R&D efforts, including collaboration with space agencies, and accompanied by a pragmatic and incremental deployment. In this paper, we summarize the major steps of adoption of MBSE: 1) emergence of modeling; 2) consolidation and maturity; 3) link to other disciplines beyond SW development; 4) application to payload software development.

We highlight the main factors that have made the adoption possible. We describe the current state of the practice of Model-Based Software Engineering in Thales Alenia Space in France, and provide some insights about challenges ahead in the next future.

Emergence of modeling for on-board software

In the late 90's and early 2000s structured design methods were already adopted in the space domain to address challenges of hard real-time software development (HRT-HOOD[1] was finalised in 1994). The biggest merit of those methodology was to stimulate the emergence of an initial software reference architecture within the company, which would permit to support the development of real-time embedded software, while supporting domain-specific aspects (i.e., first adoption of PUS, specificities of the avionics). At the same time however, there was a large space for improvement in the design support of these methods.

[1] HRT-HOOD: A Structured Design Method for Hard Real-Time Ada Systems - de A. Burns, A. Wellings

The first emergence of satisfactory use of software modeling for on-board software has its origin in self-funded R&D activities in collaboration with Thales Group, which led to the definition of a model-based engineering environment for satellite platform software: Melody CCM.

Melody CCM was originally devised so as to target OMG's CORBA Component Model, yet it evolved and adapted to support space-specific considerations: support for PUS in the design space, support for precise data type modeling (mirroring the expressiveness of the Ada programming language), and use of a target run-time adapted for embedded space software. The software design environment was extended so as to support code generation for Ada, and targeting TAS' own reference architecture. The work capitalized on early advances in software modeling, such as those of the EU FP6 ASSERT research project, with ESA as project coordinator, or attempts to use UML as modeling language; yet the result was a decisive step forward, in particular thanks to the domain-specific nature of the modeling space, and the generated code, which was factorizing reference code patterns already in use.

The engineering environment (known as "CCM for Space") was operationally deployed for the first time in 2009 for the development of the Sentinel 3 platform software.

Consolidation and maturity

In the years 2012-2016, every new platform software was developed using more and more advanced evolutions of the MBSE methodology and toolset (Exomars TGO and EDM, Iridium Next, Spacebus NEO, and Earth Observation satellites, including SWOT). Evolution of the toolset was driven by two main factors: i) to quickly respond to specific program needs (adaptations and performance optimizations); (ii) to extend the capabilities of the toolset with additional capabilities.

Factor i) led to the decision of maintaining full control within the on-board software department of the development of the toolset, as only a dedicated team with knowledge both of modeling and toolset development and of the target software architecture could support this goal, and with the necessary reactivity.

Factor ii) was possible thanks to a synergy of self-funded R&D and participation to several R&D activities funded by ESA and CNES. Among all the topics, those that demonstrated good potential and results in prototypal developments were progressively added to the operational toolsuite (e.g. generation of export files to the Satellite Data Base; auto-coding of configuration / missionisation software based on the content of the Satellite Data Base; generation of TM-TC ICD; MMU support in the modeling tool; Model-Based Test Campaign specification; Model-Based testing of OBSW missionization. Some other features were considered promising, but have not found (yet) a path towards operational deployment: support for Time and Space Partitioning; Model-based Schedulability analysis; Model-based test behavior specification).

Link to other disciplines beyond on-board development

The growth of the Software Factory within the software area (on both development and test sides) was resulting from a pragmatic and local improvement of engineering practices. At the same time, model-based techniques were adopted in related disciplines such as the Satellite Data Base (SDB-Next), the Operations Preparation Environment (SCOPE), the AOCS team (full GNC modeling with Matlab-Simulink), or Data Handling Teams (FDIR Design and Avionics Unit Specification in Capella). This new ecosystem has brought new opportunities (System to Software transition for equipment management SW, full auto-coding of GNC software, harmonization of test and operation environments, or digital continuity from design models to Satellite Data Base), but also raises new challenges considering the strong heritage of practices in the company. Just as a few examples: need to set-up co-engineering practices, need to align tools and technology, need to coordinate configuration management, …).

In this context, the Software Factory cannot been seen anymore as an independent asset, but needs to be considered together with many other external assets with in the so-called concept of “Model-Based System-Software Factory”. This interesting step in the evolution of the Model-Based adoption in TAS in France required a paradigm change in terms of organization and governance. Instead of local optimization, the Model-Based System-Software Factory seeks for global optimization throughout the whole process, where more effort is needed in the early stages of the V cycle in order to save significant amounts of time and money on the latter ones. It is particularly true when dealing with topics such as Electronic Data Sheets, Early Validation and Verification analysis, or System to Simulation transitions.

Application to Payload Software development

Model-Based practices have taken more and more importance for the development of avionics / platform on-board software. A recent trend foresees partial or full application of the same methods also for payload software. Payload software complexity has greatly increased in the past years, thanks also to the trend of moving function implementation from costly and often be-spoke ASIC implementations, to FPGAs, or SW executing on a general purpose space processors. Unfortunately, this ramp-up was not sustained by the formalism that comes with model-based practices, and the lowest reliability expectations associated to (part of) payload software have not permitted to justify early adoption of similar practices. However, it appears nowadays that Payload Software development can benefit substantially from the existing Model-Based System-Software factory, in particular for parts related to (Payload) Command and Data Handling, real-time behavior, communication protocols and resource management, i.e., aspects in common with platform software. In turn this requires adaptation to this new context (possibly different programming languages or software execution platforms, different underlying hardware, different performance/ predictability / reliability needs, …).

Initial deployments in this context were performed for the Payload Execution Platform of the MTG FCI and IRS payloads, and a sizeable telecom payload.

Challenges for the near future

As a continuation of the trend highlighted in previous sections, model-based software engineering requires to be used in context that show three new trends:

- A systematic search for a solution to the "digital continuity" challenge, i.e., the capability of *meaningfully* model a **full system** from the early design phases (i.e., 0, A/B1), and to transition modeling data in the design and implementation phases (B2, C, D) and later into operations, without loss of data, and maintaining flexibility of adjusting the abstraction of representation to the level meaningful to the actual phase of development
- The reconciliation of heterogeneity internal to the project, which derives from approaches, methodologies and technologies best suited for individual disciplines (i.e., AOCS, thermal, structures, avionics / SW), which should now be able to fit together,

breaking existing walls in the free, meaningful circulation of data between and beyond disciplines, throughout the whole development lifecycle

- The reconciliation of heterogeneity brought by the collaboration of several company (or several Agencies) within the same project, which may hinder effective communication and engineering work.

Recent work on methods and concepts such as Model-Based System Engineering, Ontologies, Engineering PDMs, Digital Twins, all reflect the desire of the engineering community to overcome those challenges.

These challenges will be all present in the Gateway project, a multi-agency endeavor lead by NASA, with contributions of ESA, JAXA and CSA for the development of the future human base in orbit around the moon. Thales Alenia Space in France is one of the partners selected for the realization of the Gateway I-HAB module. We will report on how we plan to deploy modeling approaches in such context, in particular for the areas of software and avionics, and what adaptations to our practices we foresee in this new development context.

Lessons learned from the use of SysML in Space Systems at SENER Aeroespacial

L. Tarabini-Castellani, V. Gómez, J. Fombellida, S. Ramirez, N. Puente, R. Contreras, R. Haya
Sener Aeroespacial S.A., C. Severo Ochoa 4, 28760, Tres Cantos (Spain)

EXTENDED ABSTRACT

The practical application of the SysML language in engineering processes of stablished organizations is a relevant feedback to steer the evolution and consolidation of the associated methods and tools. SENER started using SysML for Space Systems in 2014. The first project to use SysML was the ESA Proba-3 formation flying demonstration mission. SysML methodology was used for the system design of the ground segment and operations. Since then the number of SENER projects adopting this technique has grown, bringing to a cross fertilization and to the internal standardization of the SysML modeling approach (Figure 1). This paper deals with the evolution of the SysML use in SENER describing for representative projects, covering from full flight systems and subsystems to equipment, the reasons to implement this standard, the benefit achieved and the main lessons learned from its adoption.

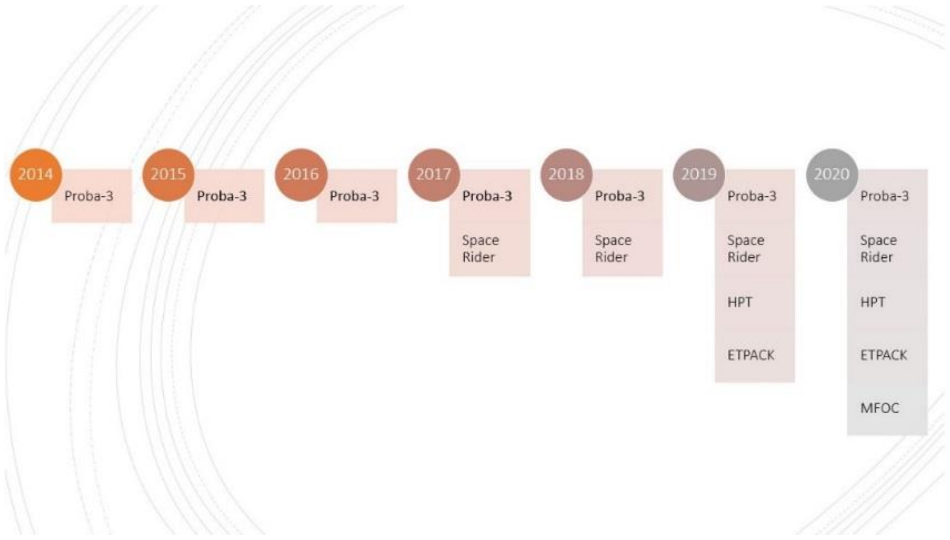


Figure 1: SENER SysML adoption timeline

Proba-3 is a complex ESA formation flying demonstration mission. In 2014, at the beginning of phase C, SENER as prime contractor was in charge to define the ground segment and operations approach. The challenge was to adapt the Redu ground station and the Proba1&2 operations environment to the demanding and highly autonomous Proba-3 formation flying operations. A Mission Operation Concept Document (MOCD) was required in a very short time frame since these activities were not included in the previous phase for budgetary reasons. Thanks to SysML diagram sharing and agile design methodology, ground segment use cases were prepared in a very reduced time. Flight autonomy versus Ground autonomy versus manual operation discussion was possible by the use of SysML activity diagrams. Final consensus was reached in due time resulting in a solid MOCD that is still currently used. This case of success showed the critical importance of using unambiguous semantics understandable by system engineers, software engineers and operations engineers [1]. SysML was also used at component level in Proba-3 for defining unit requirements from the use cases. At system level SysML was used to model the complex mode architecture including spacecraft modes, Formation Flying modes and the GNC modes at spacecraft level. This logical model allowed to simulate and verify the logic correctness and the mode transitions. SysML was also used to compile and maintain Proba-3 power budget. This budget was particularly complex due to the large number of units and operating modes. Additionally, the SysML simulation feature was used for the independent design verification of the Failure Identification and Recovery (FDIR) system (Figure 2). The main lesson learned from Proba-3 is that SysML models are of great help in the system design. SysML should be adopted in the initial phase of the project and the model evolved during its development [2].

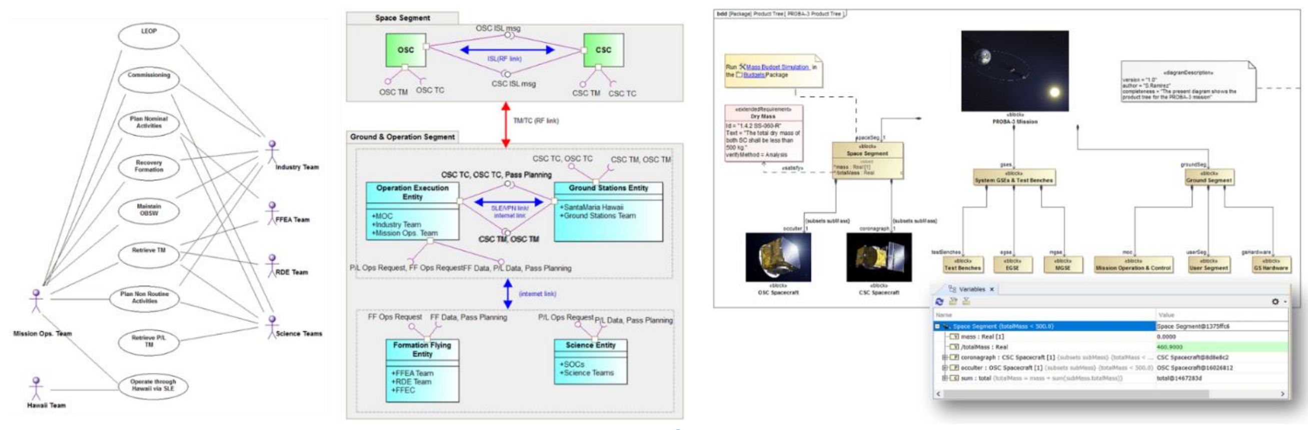


Figure 2: Proba-3 MOCD (left) & System Budgets (right)

Space Rider is the space program managed by ESA for the development of a reusable robotic spacecraft. In 2017, during phase A/B1, Space Rider was under Thales Alenia Space and CIRA co-primeship and SENER was responsible for the GNC including the requirements definition. Adopting SysML methodology, SENER defined GNC use cases starting from the mission requirement. For each use case a dedicated activity diagram led to identify the critical requirements. In a second step, the GNC system was designed ad-hoc to satisfy the requirements [3]. Moreover, the high degree of autonomy, scenarios and phases for the GNC of Space Rider called for a systematic approach to move from the high level mission requirements to the allocation of functions at component level, which motivated the adoption of SysML at GNC subsystem level. In 2018 Space Rider mission and system was substantially updated. The modified VEGA AVUM was selected as external orbital module and SENER, that designed an integrated GNC system suitable for the orbit and re-entry phase should adapt the GNC exclusively for the Re-Entry Module. Thanks to the SysML digital design, the change was absorbed with limited impact. SysML orbital GNC modules were removed and GNC re-entry module were reused and improved in order to obtain the detailed operations definition. Despite the complex consortium organization and the split of the GNC development responsibility by phase, using SysML, SENER managed to provide and maintain a consistent implementation of the Re-Entry GNC functions interfaces (Figure 3). The clear interface design was of paramount importance to identify additional requirements and analysis to be performed [4]. Again, SysML handled the system complexity and demonstrated high flexibility to adapt the design.

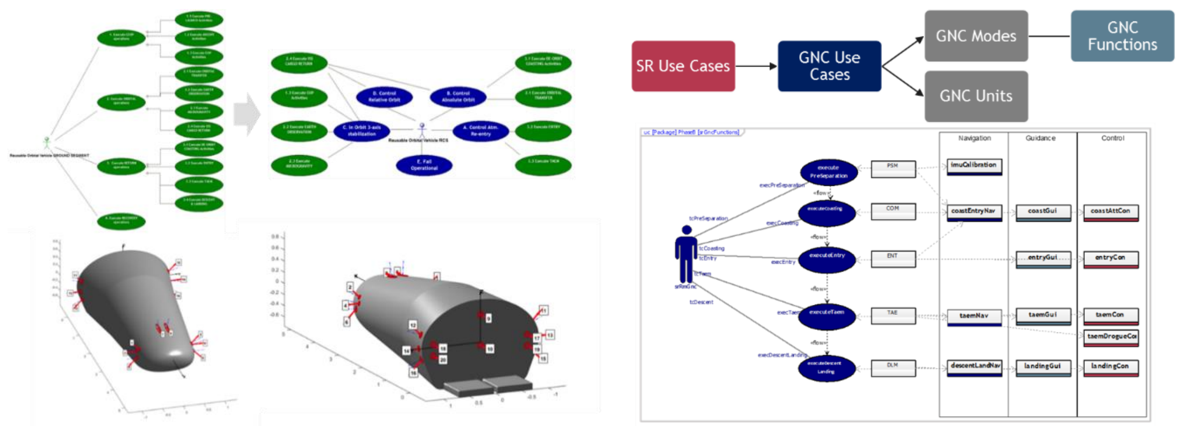


Figure 3: Space Rider GNC requirement definition (left) and GNC function architecture (right)

The **Helicon Plasma Thruster** (HPT) is a radio frequency-powered plasma propulsion technology that can perform well while eliminating many issues that have affected Electric Propulsion Systems (EPSs) to date. SENER started the development of the HPT in 2013 in collaboration with the University Carlos III of Madrid, based on internal funding and ESA's GSTP support programmes. Since then, several prototypes have been built to increase the technology TRL. In 2020, SENER is leading a consortium to evolve the HPT system to TRL 6 in the frame of an EU-funded project called HIPATIA. In order to optimize the project efficiency and to speed up the design loop review, in 2019 SENER started to implement and maintain the complete system design of the HPT (Figure 4). In this model based oriented project, all the project reviews are performed directly on model's views reducing to the barely minimum the technical documentation. At the same time external reviewer have continuous full access to the detailed design [5].

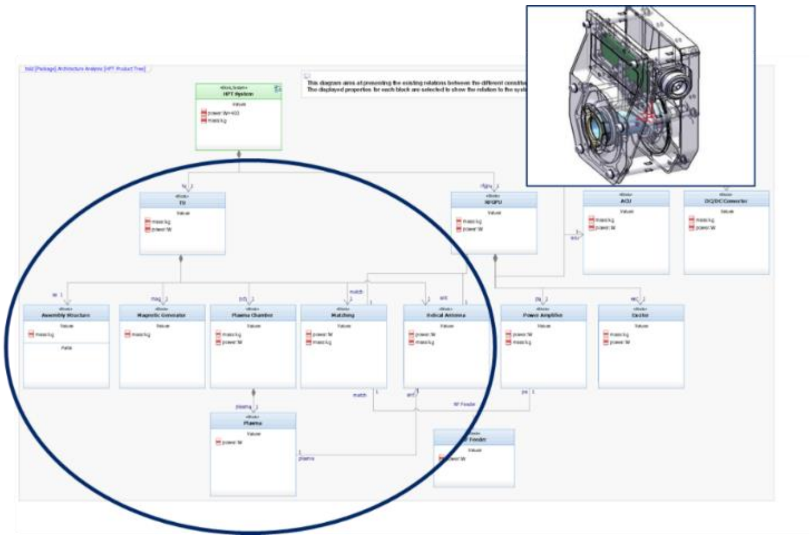


Figure 4: HPT SysML model

E.T.PACK is an EU funded project aimed to design a deorbit kit device based on electrodynamic tether and develop a prototype up to TRL4 by 2022. The project will follow the successful HPT SysML implementation scheme taking full advantages of the lessons learned and building on it (Figure 5). In this project SysML design will mimic the prototype to build a digital twin. The objective is to reduce to the minimum the cost of the technology development that will hopefully end in a demonstration flight in 2025 [6].

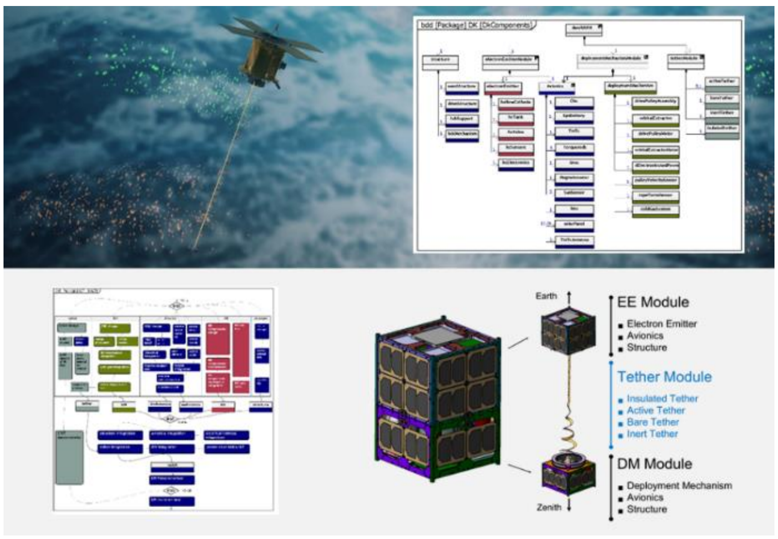
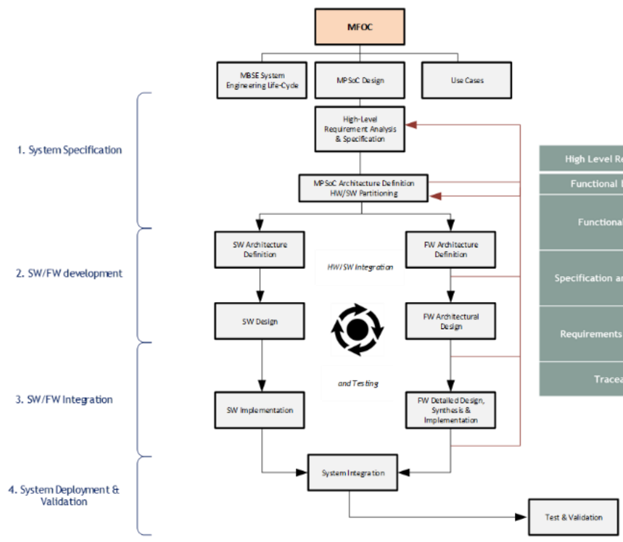
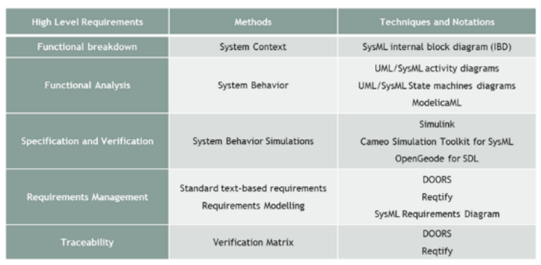


Figure 5: ETPACK system design with SysML

The **Madrid Flight on Chip** (MFOC) is a project funded by Comunidad de Madrid and the European Union to develop an execution platform based on MultiProcessor System on Chip (MPSoC) for future new space applications and satellites [7]. MFOC started in 2018 and includes work packages dedicated to the advanced use of MBSE and in particular to SysML, integrated in a complete engineering design environment. Within this activity, SENER is with The Reuse Company to maximize the exploitation of the SysML tool and its connectivity to other system design tools. Currently, this project is actively supporting SysML standardization activity in SENER.

Within MFOC, SysML formal modeling has been adopted at the System Specification phase of a hardware-software co-design and co-verification approach. This approach has been implemented to better exploit trade-offs between firmware and software partitioning and to design architectures conforming functional and stringent performance requirements with a shorter design cycle (Figure 6).


MFOC
MBSE System Engineering Life-Cycle
MPSoC Design
Use Cases
High-Level Requirement Analysis & Specification
MPSoC Architecture Definition HW/SW Partitioning
SW Architecture Definition
FW Architecture Definition
HW/SW Integration
SW Design
FW Architectural Design
and Testing
SW Implementation
FW Detailed Design, Synthesis & Implementation
System Integration
Test & Validation
1. System Specification
2. SW/FW development
3. SW/FW Integration
4. System Deployment & Validation



High Level Requirements	Methods	Techniques and Notations
Functional breakdown	System Context	SysML internal block diagram (IBD)
Functional Analysis	System Behavior	UML/SysML activity diagrams UML/SysML State machines diagrams ModelicaML
Specification and Verification	System Behavior Simulations	Simulink Cameo Simulation Toolkit for SysML OpenGeode for SDL
Requirements Management	Standard text-based requirements Requirements Modelling	DOORS Reqtify SysML Requirements Diagram
Traceability	Verification Matrix	DOORS Reqtify

Figure 6: MFOC project aiming at integrating SysML with the complete electronic development environment

In the frame of the ESA Open Space Innovation Platform (OSIP), SENER proposed an idea for designing **ESA AOCS/GNC with SysML**. If selected, the activity goal would be to digitalize the AOCS/GNC design process. With SysML the final user would be capable to easily follow and operate the subsystem by the originated diagrams avoiding the need of very detailed documentation and hard-to-follow texts. The final outputs would be a set of guidelines for a SysML based AOCS/GNC Design, the definition of the relations amongst the different model elements, diagrams and views, the generation of templates and a roadmap for the reuse of AOCS/GNC data [8].

As conclusions, SysML is considered a mature methodology in SENER and is widely used for internal developments, proposals, ESA and EC projects. SENER is also teaching SysML for Space at the University Carlos III of Madrid. SysML allows mastering the complexity with a reduced number of graphical elements and associated documentation. The standardized SENER working procedure guides the engineer in the early task of requirement definition up to the level of definition of the component detailed design. The increase in the engineer's productivity results has demonstrated to lead to higher project efficiency with consequent saving of money. SysML is well accepted by customers and brings to considerable optimization of the work. The key for methodology acceptance is that different projects have adopted SysML at different levels according to their needs and expectations, concurrence with the rest of stakeholders, communication and training as well as a definition of the scope within already stablished engineering processes

References

[1] L. Tarabini Castellani, S. Llorente, J.M. Fernandez, A. Agenjo, A. Mestreau-Garreau, A. Cropp, A. Santovincenzo. **Proba-3 - Achieving Formation Flying Millimeter Accuracy.** 9th International ESA Conference on Guidance, Navigation & Control Systems, Porto (Portugal). 06/2014

[2] S. Ramirez. **Suitability Assessment of a MBSE Procedure for Modelling, Analysing and Validating a Spacecraft within Systems Engineering Applications**, ETSIAE UPM, Madrid. 09/2017

[3] L. Tarabini Castellani, R. Haya, A. Ayuso, **Space Rider Thruster Configuration and Control Strategy Optimisation.** 10th International ESA Conference on Guidance, Navigation & Control Systems, Salzburg (Austria). 05/2017

[4] R. Haya, L. Tarabini Castellani, A. Ayuso. **Re-Entry GNC Concept For A Reusable Orbital Platform (Space Rider).** 69th International Astronautical Congress, Bremen, Germany. 10/2018

[5] J. Navarro-Cavallé, M. Wijnen, P. Fajardo, E. Ahedo, V. Gómez, A. Giménez, M. Ruiz, **Development and Characterization of the Helicon Plasma Thruster Prototype HPT05M**. 36th International Electric Propulsion Conference, Vienna, Austria. 09/2019.

[6] L. Tarabini Castellani, A. Ortega, A. Gimenez, E. Urgoiti, G. Sánchez-Arriaga, G. Borderes-Motta, E. C. Lorenzini, M. Tajmar, K. Wätzig, A. Post, J.F. Plaza, **Low Work-Function Tether Deorbit Kit.** 1st International Orbit Debris Conference (IOC), Houston (Texas). 12/2019

[8] Madrid Flight On Chip (https://flightonchip.es/)

[7] Open Space Innovation Platform. (https://ideas.esa.int/)

Abstract for MBSE2020 Workshop

Title: “What to Expect from SysML Version 2?”

Author(s): Hans Peter de Koning (hanspeter.dekoning@dekonsult.com)

Affiliation: DEKonsult, Amsterdam, The Netherlands (retired from ESA per 31 Dec 2019)

In the ongoing adoption of Model-Based Systems Engineering (MBSE) the Systems Modeling Language (SysML) standard from the Object Management Group (OMG) plays a major role, as it is de facto the only global standard for MBSE. Since SysML version 1.0 was released in 2008 and has increasingly been deployed in industry and government agencies across many industry sectors to support the development of complex systems. In addition in 2017 it was adopted as the ISO/IEC 19514 standard. SysML is also used in the European space sector, e.g. on the ESA projects e-Deorbit, Euclid, PLATO and Mars Sample Return.

In the last 10 years the standard has seen a number of gradual upgrades from version 1.2 in 2010 – that marked the start of real industrial use – to the current version 1.6 release in November 2019 [1]. SysML version 1 is strongly based on Unified Modeling Language (UML) v2, and therefore inherits a number of concepts and standardization patterns from this object-oriented software engineering standard. This can be considered both a strength and a weakness. A strength because it meant that mature UML tools could be adapted with reasonable investment to support SysML, and provided good coverage for software-intensive systems. However, a weakness too, because the software engineering heritage created barriers for the uptake by systems engineers without a strong software engineering background. A general complaint on SysML v1 is that the learning curve is too steep, and that the language unnecessarily complicates modelling a number of key systems engineering concepts, such as interface connections between nested components.

This was also acknowledged at OMG and in 2015 work was started on collecting user needs and requirements for a RFP (Request for Proposal) for SysML version 2. The goal was to ensure that all lessons learned from the initial years of industrial usage would be taken into account. Since such a major overhaul of the standard can only be afforded every now and then, this preparation was taken very seriously and performed by a working group with broad representation by end-users from different industry sectors (large and small enterprises, government agencies, research institutes, academia) as well as SysML tool vendors over the course of 2 years. It resulted in two extensive RFPs: one for the SysML v2 language itself [2], and another one for the Application Programming Interface (API) and Services [3]. All discussions and prototyping that went into the preparation can be found at [4].

Since the beginning of 2018, a team of more than 100 experts from around 60 organisations -- the so-called SysML v2 Submission Team (SST) -- has been developing the second version of SysML. This work is culminating into the first full public release – for both the language and the API and Services – planned for September 2020.

The current presentation will provide an overview of the new and enhanced capabilities of SysML version 2, including but not limited to:

1. New simplified SysML meta-model, which is founded on a minimal set of key concepts.
2. The new normative and informative model libraries including the upgraded way of handling quantities, units and scales, which now have a very rigorous underlying information model, that also allows for automated unit / scale conversion, which is important when integrating models coming from different partners. Also basic geometric modelling is supported to represent e.g. the specification of enveloping shapes for system components.
3. The new textual notation, including a standardized and very powerful expression and constraint language, as well as the upgraded graphical notation (diagrams), and the integral, flexible viewpoint

/ view capabilities. These overcome many limitations of the SysML v1 Block Definition Diagrams and Internal Block Diagram, and provide much better and more precise ways to define interfaces and connections, also in deeply nested structures. Then there is the integrated approach to model behaviour (activities, functional architecture, time-based sequences, finite state machines, 4D lifecycle objects), both in precise textual notation and in diagrams that can be mixed and matched, to answer the needs of particular domains.

4. Support for variant modelling and product line engineering built into the language and/or connectable to external variant modelling tools.
5. The much improved support for integrating SysML v2 models with external analysis and simulation paradigms and tools, founded on much more precise execution semantics.
6. The prototype implementations of the textual and graphical language, on the Eclipse Modeling Framework as well as in Jupyter Notebooks.
7. The new API and Services that provide a much better and richer capability to interact with SysML models than SysML v1 XMI files. The technology neutral API specification allows for both static whole model transfers and simultaneous dynamic interaction of many client tools with one or more SysML repositories. In the current SST prototype implementation, a REST, an OSLC, and a Java API are supported.
8. The way compatibility with SysML v1 is ensured via a SysML v2 profile as well as via the new API.

The presentation will highlight how the learning curve is expected to be reduced for systems engineers. Time will be dedicated to explain the new so-called usage-focused modelling approach, which allows to directly model deeply nested architectures, in a way that feels more natural for most systems engineers. This new capability is in addition to the SysML v1 “definition/type first” approach, and still maintaining ways to ensure a rigorous modular architectures. The same definition / usage and composition patterns are consistently applied throughout all aspects of the language, for requirements, structure, behaviour, interfaces, parametrics, constraints, verification.

Finally an outlook on the deployment schedule of SysML v2 will be provided.

The author has been a member of the SysML v1 task forces since 2009 as well as the SysML v2 RFP working group, and is an active member of the Submission Team for SysML v2.

References

[1] Systems Modeling Language v1.6, OMG, November 2019, https://www.omg.org/spec/SysML/1.6/

[2] Systems Modeling Language (SysML®) v2 Request For Proposal (RFP), OMG, December 2017, https://www.omg.org/cgi-bin/doc.cgi?ad/2017-12-2

[3] Systems Modeling Language (SysML®) v2 API and Services Request For Proposal (RFP), OMG, June 2018, https://www.omg.org/cgi-bin/doc.cgi?ad/2018-6-3

[4] OMG SysML v2 RFP Working Group Wiki at http://www.omgwiki.org/OMGSysML/doku.php?id=sysml-roadmap:sysml_assessment_and_roadmap_working_group

[5] Hans Peter de Koning, “Progress on SysML v2”, 13th ESA Workshop on Avionics, Data, Control and Software Systems (ADCSS2019), November 2019, ESA/ESTEC, https://indico.esa.int/event/323/contributions/5057/attachments/3756/5215/11.55_-_Progress_on_SysML_v2.pdf

[6] General information on the OMG Systems Modeling Language (SysML), see http://www.omgsysml.org

[7] General information on MBSE across all industry sectors, INCOSE/OMG MBSE Wiki at http://www.omgwiki.org/MBSE/doku.php

MODEL-BASED SYSTEMS ENGINEERING IN SPACE ROBOTICS: THE ADE EXPERIENCE

J. Ocón [(1)], I. Dragomir [(1)], R. Jalvo[(1)], R. Marc [(2)], M. Foughali [(3)], L. Kunze [(4)], R. Dominguez [(5)]

[(1)] GMV Aerospace and Defence, Isaac Newton 11, PTM, Tres Cantos, 28760, Spain, Email: jocon@gmv.com
[(2)] Airbus Defence and Space Ltd., Gunnels Wood Road, Stevenage, SG1 2AS, UK, Email: gnc.uk @airbus.com
[(3)] Verimag, Univ. Grenoble Alpes 700, 38401 St. Martin d'Hères Email: mohammed.foughali@univ-grenoble-alpes.fr
[(4)] Oxford Robotics Institute, Dept. of Eng. Science, Univ. of Oxford, Oxford OX1 3PJUK, Email: lars@robots.ox.ac.uk
[(5)] DFKI GmbH, Robert-Hooke-Straße 5 28359 Bremen, Germany Email: raul.dominguez@dfki.de

ABSTRACT

Model-based systems engineering (MBSE) is the adopted practice for taming the increased complexity and heterogeneity of today's (systems-of-) systems under development. System modelling allows one to obtain abstract representations of the system by focusing only on the crucial aspects needed at the different development stages. These aspects can tackle for instance the design of the components performed independently by different engineering teams, the integrative design of the system in terms of communications, and the system design subject to formal verification and validation. Integrated in a model-driven development process, such as the waterfall model, and supported by many tools, MBSE provides a complete solution that aims to derive, possibly (semi-)automatically, implementations from high-level specifications. MBSE offers many benefits during the development phases: modularity and independent development of the different systems/components, reuse of components, compatibility with other systems/framework, and formally checked reliability and resilience.

In this paper, we present the MBSE formalisms, approach, and tools used in the H2020 Autonomous decision making in very long traverses (ADE) project (https://www.h2020-ade.eu/). The aim of the ADE project is to develop a demonstrator for a planetary rover capable of performing very long traverses (kilometres per sol), taking autonomously decisions required to progress, reducing risks, and seizing opportunities for data collection (opportunistic science). The rover will be able to perform high-level goals requested from ground, decompose these high-level goals into low-level activities, and perform these activities in real-time, while reacting to any hazardous situations and adapting the activities to the current conditions.

More specifically, the ADE design of the demonstrator involves the use of many models and technologies in order to achieve such crucial goals, some of them being beyond the state-of-the-art in space robotics. For autonomous decision taking, the ADE system integrates an on-board planner based on artificial intelligence (AI) techniques. This component uses the Problem Domain Definition Language (PDDL) for modelling the world and computational logic for reasoning and finding solutions. For opportunistic science, ADE integrates a scientific detector also based on state-of-the-art AI. This component uses trained neural network models that detect and classify scientific targets of interest. For long traverses, ADE integrates a rover guidance supported by a perception and localisation system that allows for autonomous path planning and hazard avoidance. Additionally, ADE uses a robotic arm for sample caching. These components use and implement control models to provide the basic functionalities of the robotic platform. For reacting to hazardous situations, ADE integrates fault detection, isolation, and recovery (FDIR) based on formal methods. This component uses Behavior, Interaction, and Priority (BIP) to formally model the system and check its correctness, at both at runtime and offline.

Finally, for the real-time execution of all these functionalities as well as creating the integrative design, ADE uses the TASTE tool. TASTE is an open source framework developed by ESA that enables the development of embedded, real-time systems based on MBSE. A TASTE system design is produced with standardized modelling languages (e.g., ASN.1 and AADL) describing different views of the system including views for

data types, components, as well as for the deployment. The tool-chain generates code for the target deployment platform (while enforcing real-time properties) and produces the system executable(s), among other features.

ADE develops other components, integrated in the considered demonstration scenario. A Ground Control Station enables the control of the system in different autonomy modes, as well as bookkeeping the results of the operations for further assessment. The navigation system is supported and checked by ground truth. Other offline assessments include the traversability of the terrain (soil), the simulation of the mission(s) and the replay of the operations performed by the robotic platform. Soil traversability is based on neural network models trained with data logged prior by the robotic platform. The simulation includes a model of the robotic platform in terms of kinematics, controlled by the ADE developed system. The aim is to evaluate and correct the functionalities of the demonstrator before field trials. The replay mode adds the assessment of the system performances from real logged data.

This paper will describe the models, approach, and tools used in ADE for the development of the planetary rover. We will present both the challenges encountered during the development, mainly related to the integration of many formalisms into a common design, and the approaches taken to address them. Finally, we will report on the experience of using different technologies and tools as well as the lessons learned

ACKNOWLEDGMENTS

We would like to thank the European Commission and the members of the PERASPERA programme support activity (ESA as coordinator, ASI, CDTI, CNES, DLR, and UKSA) for their support and guidance in the ADE activity. We also thank the ADE Consortium for their collaboration and support in this activity.

This project has received funding from the European Union’s Horizon 2020 Research and Innovation programme under Grant Agreement No 821988.

Capella to TASTE MBSE bridge

Project Team:

Presenter: Michał Kurowski
N7 Space Sp. z o.o.
mkurowski@n7space.com

Other project contributors:

Arkadiusz Wójcik
was with *Creotech Instruments S.A.*
arkadiusz.wojcik@creotech.pl

Michał Kocon
N7 Space Sp. z o.o.
michal.kocon@n7space.com

Daniel Kaczmara
Creotech Instruments S.A.
daniel.kaczmara@creotech.pl

Bartłomiej Juszczyk
Creotech Instruments S.A.
bartłomiej.juszczyk@creotech.pl

Michał Mosdorf
N7 Space Sp. z o.o.
mmosdorf@n7space.com

ESA team:

Hans Peter de Koning
ESA (retired)

Maxime Perrotin ESA
Maxime.Perrotin@esa.int

Short Abstract: The presented project provides a bridge between Capella, an open-source MBSE tool supporting the Arcadia method, and TASTE, ESA's open-source MBSE toolchain. The bridge is implemented via a Capella plugin, which translates the Capella's data and physical architecture models into TASTE-compatible ASN.1 and AADL models, which can be further enhanced with behaviour definitions through C, Ada or SDL, and compiled into deployable binaries. The bridge was validated by implementing software for a Mass-and-Thermal Mockup running on STM32 MCU.

Keywords: Capella, TASTE, Arcadia, ASN.1, AADL, SDL, MSC, MBSE, STM32, plugin

Background. Capella [1], originally implemented by Thales, is an Eclipse-based tool implementing the Arcadia method [2], allowing to perform operational need analysis, system analysis, logical architecture design, physical architecture design and finally define a product breakdown structure, providing an alternative to UML and SysML. It allows to capture requirements and other project specific metadata, delivering a high-level cross-domain MBSE solution. While it does not provide any code generation capabilities by itself, it is highly extensible through Java plugins. TASTE [3], managed by ESA, is a set of tools focused on supporting model-based software development. In particular, it allows to generate code from ASN.1, AADL and SDL models, which can be then compiled and deployed onto the supported platforms, including x86, Leon3 and ARM STM32. The resulting software can be then tested using executable (via Python) MSC diagrams. A bridge connecting the two solutions, in the form of a Capella plugin, was implemented during the *MBSE_Implement* project founded by the European Space Agency and carried out by Creotech Instruments (prime contractor) and N7 Space (subcontractor). It allows to apply an MBSE based approach throughout the entire software product lifecycle, from high-level cross-domain analysis to implementation, testing and deployment.

Bridge implementation. After discussions held between Creotech Instruments, N7 Space and ESA, N7 Space analysed the scope and explicitness of Capella model elements with respect to the capabilities and requirements of TASTE toolchain. The following was considered – data model, architecture and behaviour.

Capella's data model focuses on the data semantics. TASTE on the other hand models both the semantics, via ASN.1, and encoding, down to the bit-level, via additional ACN definitions or by application of default UPER rules. While Capella's data model could be enhanced with additional metadata for bit-level encoding specification, therefore enabling ACN generation, it was considered complicated and unnecessary. N7 Space implemented ASN.1 generation, by mapping Capella's Packages, Classes, Unions, Collections, Numeric Types into ASN.1 Modules, Sequences, Choices, Sequences Of and Integers or Reals respectively. The user can additionally choose, through custom properties, an encoding specification from between UPER, platform native or ACN. In case of UPER and platform native, encoding is handled automatically by TASTE. In case of ACN, the additional rules must be provided by the user separately. Capella's data model supports data types, values and expressions. On the other hand, ASN.1 supports only data types and values. In order to partially resolve this limitation, a simple evaluator was implemented in the plugin to translate integer expressions into concrete values. As Capella's Units, relevant to Physical Quantities, do not have a corresponding construct in ASN.1, they were implemented via ASN.1 type name postfixes. Similarly, Capella's class inheritance hierarchy is translated into a set of ASN.1 Choices.

Capella's architecture model is expressed via logical and physical architectures. The former is usually considered a "principle", coarse-grained, general architecture. The latter is the finalized architecture. As TASTE requires a concrete architecture definition, and the tracing between the physical architecture and the logical architecture is maintained in Capella anyway, the physical architecture was chosen as the base for AADL generation. N7 Space implemented a mapping from Capella's physical Nodes, Components/Actors, Links/Paths, Ports, Functions and Functional Exchanges into TASTE Nodes, Partitions, Buses, Devices, Functions and Interfaces respectively. The TASTE concepts are expressed via standard AADL constructs such as Packages, Systems, Processes, Subcomponents, Connections, Features and Subprograms. As time-and-space partitioning is not supported in the plugin, all components residing on a single node are merged into a single partition. An extensible and explicit mapping to target processors and drivers is provided through custom string properties. The developed mapping allows the generation of TASTE Interface and Deployment Views.

While Capella supports the modelling of behaviours through Sequence, as well as Mode and State diagrams, N7 Space deemed their translation into SDL (or any other executable language) infeasible without significantly extending the Capella's model. Consequently, the concrete behaviour definition is to be performed directly in TASTE, e.g. via SDL, C or Ada. Unambiguous, formal and user-friendly software behaviour definition within Capella can be a subject for future work.

The plugin implemented by N7 Space first checks the Capella model for consistency and completeness from TASTE's perspective (thus constraining the Capella's model to a subset with well-defined semantics), provides feedback, allows the user to select data or architecture models' subsets and then generates the corresponding ASN.1 and AADL artefacts. The plugin allows to naturally follow the Arcadia method (implemented in Capella) with TASTE based implementation within a fully model based workflow. A partial approach is also possible by using only the generated ASN.1 [4].

Validation. In order to validate the plugin and the MBSE approach, a use case scenario was established jointly by Creotech and N7 Space. Creotech modelled a Mass-and-Thermal Mockup software in Capella. The model was then iteratively refined using feedback from N7 Space, illustrating the benefits of model-based design formalization and disambiguation for achieving common understanding across different industrial partners. After the finalization, the model was automatically translated into ASN.1 and AADL files using the plugin. N7 Space implemented the software behaviour in SDL (high-level functionality), C (peripheral drivers) and Ada (RS-485 communication device driver). The C code was based on an alternative, twin software, manually coded by Creotech Instruments. The software was then successfully deployed and tested on a physical Mass-and-Thermal Mockup designed and produced by Creotech for an in-house developed satellite platform. The test scenarios, defined by Creotech, were implemented by N7 Space in Python using the code automatically generated from MSC diagrams created in TASTE. Additionally, as a part of the applied MBSE approach, Creotech Instruments used a freely available M2Doc [5] add-on to generate documentation from an internally developed Capella model.

Summary. The developed plugin allows to translate a well-defined subset of Capella's data and physical architecture models into ASN.1 and AADL models compatible with TASTE. These models can be then further enhanced with behaviour definition via SDL, C or Ada, and compiled into executable software for target platforms. The plugin and the MBSE approach were successfully validated via the implementation of software for a Mass-and-Thermal Mockup. Feedback from the validation was propagated as suggestions or remarks to the TASTE project.

References:

[1] Capella, https://www.eclipse.org/capella/
[2] Model-based System and Architecture Engineering with the Arcadia Method, Jean-Luc Voirin, ISBN 978-1-78548-169-7
[3] TASTE, https://taste.tools/
[4] Maxime Perrotin et al.. TASTE in action. 8th European Congress on Embedded Real Time Software and Systems (ERTS 2016)
[5] M2Doc Capella add-on https://www.m2doc.org/

CoRA-SAGE: The lessons learnt from AOCS/GNC algorithms deployment in TASTE

A. Figueroa[1], J.M. del Cura[1], S. Lozano[1], G. del Valle[1], G. Rodríguez[1], J. Corchero[1], J. Cardín[1], L. Tarabini Castellani[1], E. Rodríguez[1], D. Oddenino[2]

[1] SENER Aeroespacial, Flight Systems Division, Calle Severo Ochoa, 4, 28760 Tres Cantos, Madrid, Spain

[2] ESA/ESTEC – GNC, AOCS & Pointing Division, 1 Keplerlaan 2201AZ, Noordwijk, Netherlands

Abstract

Compact Reconfigurable Avionics (CoRA) is a co-engineering activity involving AOCS&GNC, software engineering and on-board data handling whose aim is to prototype an in-flight reconfigurable avionics system. In this context, SAGE (CoRA-SAGE) is a multidisciplinary activity aimed to implement AOCS&GNC functional chains with a large suite of space sensors and actuators in parallel with the Model Based Avionics Design (MBAD) activity and the Reconfigurable Data Handling Core (RDHC) activity. CoRA-SAGE has been responsible of developing the AOCS/GNC exercised in the reconfigurable avionics and the ground support equipment, including simulated units and a piece of flight hardware used to test the overall CoRA system.

CoRA-SAGE selected Space Rider reusable servicing vehicle as strawman test configuration. Space Rider is a unique ESA mission with several application scenarios including Earth Observation, telecommunication, science and demonstration missions. Space Rider orbital and re-entry phases requirements allows to stress the need to fit very different AOCS/GNC modes in the on board computer. From the full set of the Space Rider mission, SENER has included in CoRA-SAGE the Fine Pointing Mode (FPM), Safe Mode (SM) and Re-Entry Mode (REM), which are representative of the AOCS/GNC modes present in any space mission. CoRA-SAGE EGSE provides the electrical interfaces to the AOCS components, combined with the simulation of the spacecraft environment and behaviour in order to verify the functionality of the AOCS/GNC and the stimulation of the hardware unit selected, an AURIGA star tracker which is operative in one of the orbital modes. CoRA-SAGE EGSE provides a mixture of simulation and sensor stimulation capabilities, including:

- Simulation of AOCS components for flight software verification and to support incremental integration, i.e., simulation of absent components
- Simulation of spacecraft environment and dynamics for open and closed loop testing of AOCS algorithms
- Characteristics that cannot be provided by the real devices, such as simulation of erroneous behaviour due to device failure or degradation
- Integrated external developed models as part of the simulation environment
- Real sensors (e.g. star tracker, etc.) so that they can generate representative data for the test scenarios used for AOCS verification
- Interfaces of the spacecraft on-board buses
- Interfaces of stimulated AOCS components (sun acquisition sensor, gnss, imu, flush air data system, reaction wheels, reaction control system)
- Optical ground support equipment of the star tracker HW unit

The architecture of the CoRA-SAGE system is depicted in Figure 1.

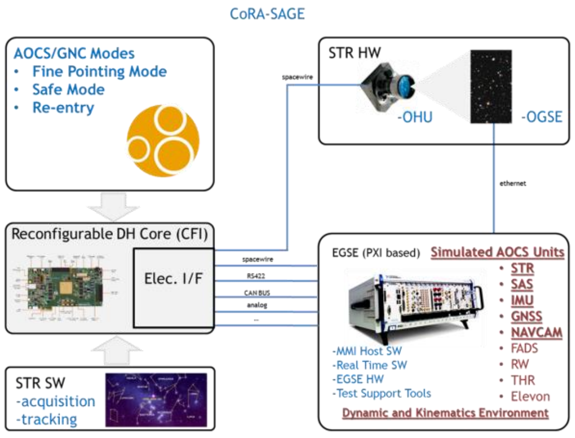
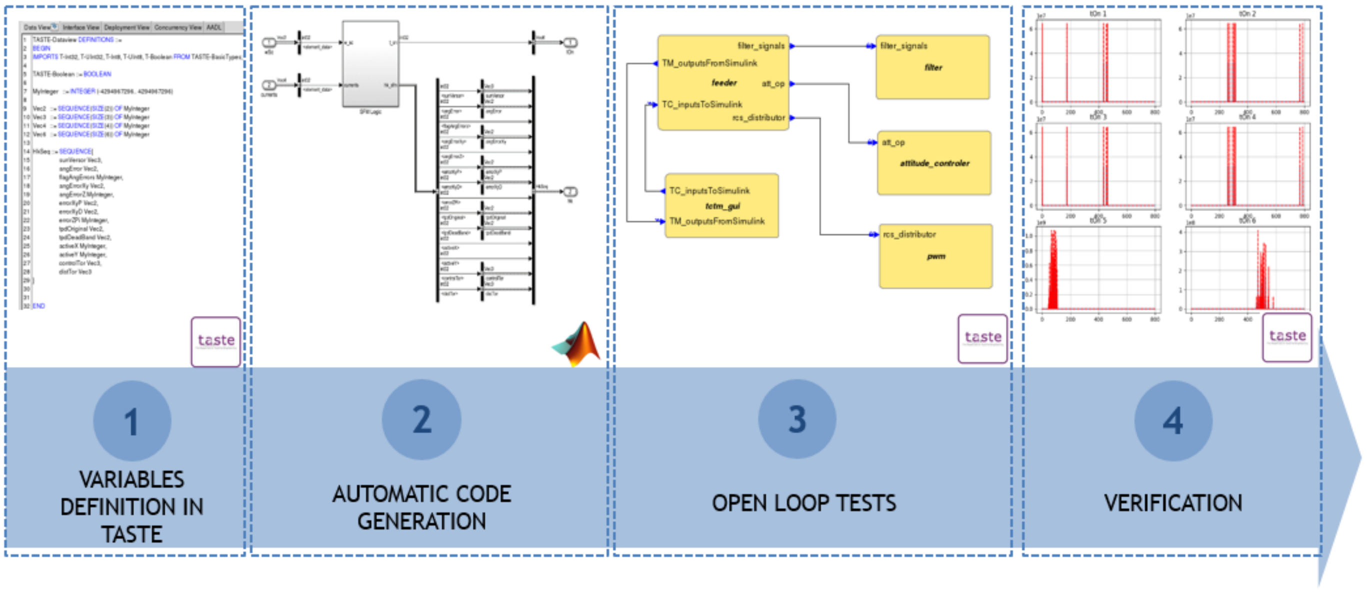


Figure 1: CoRA-SAGE Architecture

The design of the selected AOCS/GNC modes is modular and allows to reconfigure the AOCS/GNC functions partitioning between hardware and software implementations, hence allowing the implementation of the same algorithm both in FPGA or in the processor and following a Model Based approach. The design and performance assessment were conduct in Matlab/Simulink while the verification and deployment have been done via TASTE. The use of these tools, and specially TASTE, have facilitated the interaction with the other CoRA teams to specify and implement the AOCS/GNC software/hardware in the Data Handling Core. From CoRA-SAGE point view, TASTE served to capture the AOCS/GNC modes architecture in the same environment in which the algorithms were finally deployed with the rest of the CoRA system and to centralize the interfaces of the AOCS/GNC modes and sub-functions. Within the CoRA-SAGE team, TASTE allowed the creation of a constantly updated and exchangeable database for the variable names, variable length and data type easing the communications between of the AOCS/GNC and software teams. Moreover, this approach has been paramount to verify the AOCS/GNC deployment.

CoRA-SAGE team decided to verify the AOCS/GNC modes incrementally. First, in the Functional Engineering Simulator (FES), the transformation of the algorithms to use fixed-point representation was done. Secondly, Open Loop tests were designed and executed directly in TASTE (with an internally available emulated GR740 processor), prior to the Closed Loop tests conducted with the CoRA-SAGE EGSE and the COTS-Bread Board (preliminary BB procured by RDHC to test CoRA-SAGE before having available all CoRA elements) and the whole CoRA system acceptance tests. The open loop tests verification in TASTE paved the way to deploy successfully the AOCS/GNC modes both on the COTS-BB and on the Elegant-BB (definitive BB designed for CoRA). Indeed, no flaw due to the AOCS/GNC modes functioning was detected during the overall system verification conducted after the verification of AOCS/GNC modes through open loop tests via TASTE. The TASTE features for early verification and testing of the generated software (GUIs and Python scripts) were employed to verify the deployment. The objective was not testing again the full functionality, previously verified in a dedicated Matlab/Simulink Functional Engineering Simulator, but a subset of representative cases selected in order to validate the successful migration of the algorithms to the target platform.

AOCS/GNC modes code were generated automatically from Simulink, uploaded in TASTE and run in an open loop simulated environment with the inputs generated in the FES. For this, the inputs/outputs definitions automatically generated by TASTE were loaded in Simulink, according to the interface database defined within TASTE. Finally, the GUI application available in TASTE allowed iterating with the algorithms via Python external scripts feeding the AOCS/GNC modes with input reference data and verifying the integration. A sketch of the verification process is shown in figure 2.


```
TASTE-Dataview DEFINITIONS ::=
BEGIN
IMPORTS T-Int32, T-UInt32, T-Int8, T-UInt8, T-Boolean FROM TASTE-BasicTypes;

TASTE-Boolean ::= BOOLEAN

MyInteger  ::= INTEGER (-4294967296..4294967296)

Vec2  ::= SEQUENCE(SIZE(2)) OF MyInteger
Vec3  ::= SEQUENCE(SIZE(3)) OF MyInteger
Vec4  ::= SEQUENCE(SIZE(4)) OF MyInteger
Vec6  ::= SEQUENCE(SIZE(6)) OF MyInteger

HkSeq ::= SEQUENCE{
        sunVersor Vec3,
        angError Vec2,
        flagAngErrors MyInteger,
        angErrorXy Vec2,
        angErrorZ MyInteger,
        errorXyP Vec2,
        errorXyD Vec2,
        errorZPi MyInteger,
        tpdOriginal Vec2,
        tpdDeadBand Vec2,
        activeX MyInteger,
        activeY MyInteger,
        controlTor Vec3,
        distTor Vec3
}


END
```

Figure 2: Verification process of CoRA-SAGE AOCS/GNC implementation

This process facilitated to achieve the consistency between AOCS/GNC algorithms and implemented software, the generation of code and the partitioning of the AOCS/GNC into elementary functions to fit the FPGA capacities. Moreover, it served to verify the AOCS/GNC algorithms deployment in the early phases of the project.

Model-based techniques for space microcontroller applications

Steve Duncan, Thales Alenia Space UK Ltd

Mixed-signal microcontrollers have become dominant in terrestrial electronics applications because of their low cost, high integration and rapid development cycle. The recent advent of rad-hard and rad-tolerant devices with onboard analogue interfacing has started a similar revolution in the space domain, with potential reductions in component count, board space, power consumption, harness mass and, above all, cost.

Typical microcontroller application programs are simple, deterministic and repetitive, which makes them simple to analyse and relatively straightforward to develop. However, they are often deployed as part of a larger distributed system, in which case a substantial proportion of the behavioural complexity arises from the interaction between the nodes rather than the nodes themselves. This is especially true for bus-based command and control protocols, particularly where the communications medium is not perfectly reliable and there is a possibility that messages between nodes may be lost or corrupted.

The management of emergent complexity is an important part of distributed system design. Experience has shown that despite high test coverage, it is still possible for systems to contain latent faults that cause an unrecoverable state when confronted with rare but unfortunate events, for example the permutation of the address field in a message to an incorrect but legal value. A very desirable goal would be the ability to prove formally that the system is free from such defects, i.e. it will always recover itself to a known state following any sort of upset.

Conventionally, the dynamic interactions between nodes in a system are designed using Message Sequence Charts. These diagrams are good for capturing stationary sequences (the “happy path”) but tend to become unmanageable when branching due to nondeterministic decisions or message errors is included.

SDL, as used in ESA’s TASTE toolchain, is a formal language for specifying and modelling the behaviour of systems. It was developed by an ITU working group in the 1970s and was widely used in the design of call control schemes for circuit-switched landline and mobile telephony. SDL represents a distributed system as a collection of interacting finite state machines (FSMs). It has a graphical from that lends itself readily to visual design capture and analysis, and a textual form that can be compiled into an executable. Both representations are equivalent and each may be readily transformed into the other.

SDL encourages the development of predictable systems by allowing key simplifying constraints on the programming model to be enforced:

- Execution occurs only during state transitions
- The execution path through any state transition is acyclic.
- An FSM may only affect another through tightly constrained points of interaction.

In this presentation, we will describe the results of our research into the model-based specification, design and validation of microcontroller-based subsystems using the TASTE toolset. We cover the following topics:

Behavioural Design

The decomposition of the system into SDL Finite State Machines is described.

Data Modelling

The modelling of data structures in ASN.1 is described, and the application of the data model in the generation of secondary representations (e.g. object dictionary, Electronic Datasheets, spacecraft database) is discussed. Particular attention is paid to the separation of system and protocol software from application, leading to a configurable model that can be easily reused for new applications and targets.

Model Validation

The construction of an executable simulation model is described, including the modelling of abstracted system components related to the interacting elements (e.g. data buses, application software). Validation of the system within this model is discussed. Specifically, the use of scripting and other automated techniques to explore and/or enumerate the state space of the system.

Code Generation

The automatic generation of code for heterogeneous systems of 32-bit and 16-bit processors is described, together with the necessary procedures for ensuring correct encoding from ASN.1 representations to legacy packet structures. We then show how it is possible to generate memory-efficient data encodings that are compatible with the small program memory spaces generally available on microcontrollers.

Subsystem Integration

The incremental integration of a system is demonstrated, whereby the TASTE model is replaced, one component at a time, with real subsystems, in order to validate the implementation against the model. The possibility for the TASTE tool to become the basis of the EGSE is discussed.

Case Studies

Finally, we present the results of three breadboard projects developed using the MBSE approach with TASTE:

- A CANbus Backplane with heterogeneous nodes based on the Thales Alenia Space DPC microcontroller and the Cobham Gaisler GR712,
- A distributed thermal control System based on the Cobham Gaisler GR716,
- A microcontroller-based Rate Gyro using the Microchip SAMV71.

Tiny Runtime to Run Model-Based Software on CubeSats

Rafał Babski
N7 Space Sp. z o. o
rbabski@n7space.com

Michał Kurowski
N7 Space Sp. z o. o
mkurowski@n7space.com

Konrad Grochowski
N7 Space Sp. z o. o
kgrochowski@n7space.com

Maxime Perrotin
ESA
Maxime.Perrotin@esa.int

Short abstract: The presented project extends TASTE, ESA's open-source model-based software development toolchain, with support for MSP430, a family of cheap low-power space-grade MCUs used in small satellites. The goal is achieved via establishing a mapping between FreeRTOS and TASTE constructs, implementing code generation via templates, optimizing/replacing the existing middleware and integrating an Ada compiler for the target platform. The results are validated by implementing in TASTE a demonstration software for a mock CubeSat, based on a MSP430 kit.

Keywords: TASTE, MSP430, MBSE, FreeRTOS, Kazoo, CubeSat, ASN1SCC, Capella

Background. ESA's TASTE [1] MBSE toolchain uses AADL, ASN.1, SDL and MSC languages to describe a system's architecture, data model, behaviour and test cases respectively [2]. AADL, SDL and MSC, while stored textually, can be manipulated via graphical tools. ASN.1 code can be edited using an IDE [3] or generated from a model created in Capella. The models can be used for documentation generation, static analyses, testing and generation of deployable binaries. The latter requires TASTE support for the given platform – code generation templates, middleware, compilers, etc. At the beginning of the project, only several 32- and 64-bit targets were supported, such as RTEMS Leon 3 and Linux x86. MSP430 is a family of cheap, ultra-low-power, mixed-signal microcontrollers, which includes space-grade radiation hardened parts (e.g. MSP430FR5969 rated for 50krad, with non-volatile FRRAM). The above traits make them good candidates for deployment in small satellites. Features distinguishing MSP430 from the existing TASTE targets are 16-bit architecture and very small amount of memory (2kB SRAM and 64kB FRAM for MSP430FR5969). N7 Space is an active TASTE Steering Committee member, bringing experience from perspectives of both user (deployment of ASN.1 models in PROBA3 payload for TC/TM transcoding) and contributor (ASN1SCC improvements and asn1scc.IDE development, PUS C ASN.1 generator and Capella-to-TASTE plugin).

Extending TASTE with MSP430 support. Before the recent introduction of the Kazoo tool, the TASTE method to generate code from AADL models relied heavily on the hardcoded use of Ocarina [2]. Kazoo is a new tool in TASTE toolchain and it implements a new approach to generate code: it uses a templating engine to generate code, build scripts, derived models, and other artefacts giving much more flexibility to the end user, including the possibility to create backends to support new platforms.

The code generated by Ocarina uses PolyORB-HI/Ada or PolyORB-HI/C as a platform independent middleware which provides constructs required by TASTE. One of the disadvantages of PolyORB is huge memory requirements, which makes impossible to use it on small platform like MSP430.

The newly implemented support for MSP430 in TASTE is based on FreeRTOS real-time operating system for microcontrollers [4]. All constructs required by TASTE were implemented using FreeRTOS features. This approach allows to reuse templates created in this project for other platforms (including tiny ones) which are currently supported by FreeRTOS or could be easily added in the future. Even the MSP430 support itself required adding a new port, which proved to be a reasonably simple process.

For entities modelled in SDL (Specification and Description Language) TASTE toolchain uses OpenGEODE [2] to generate Ada source code. While creating TASTE model of a realistic CubeSat test application targeting MSP430, the support for Ada language was not yet ready for the platform, therefore, OpenGEODE's feature to generate C source code was refreshed and improved.

One of the big challenges after abandoning PolyORB in the generated code was to provide compatibility for communication with the code which still uses old middleware. To solve this issue new compatible device drivers were created: one for MSP430 and one for PolyORB. These drivers utilize a new simplified protocol to exchange messages.

After establishing generic code patterns required for mapping models into FreeRTOS objects, the tools forming the TASTE toolchain required some improvements and fixes:

- capabilities of Kazoo were extended to allow generation of code for all TASTE constructs;
- the ASN1SCC tool was extended with support to generate code for 16-bit platforms;
- OpenGEODE C code generation issues regarding case-sensitivity were fixed and support for cases where more than one function is modelled in SDL was added.

The documentation of all the tools and the aforementioned process is available on TASTE wiki [2].

TASTE, while AADL, ASN.1 and SDL based, supports also several other implementation languages such as C and Ada. Additionally, SDL is integrated into the final binary via intermediate transformation to C or Ada code. The existing freely available Ada compilers either do not support MSP430 or are considered legacy software. In order to provide a seamless open-source support for Ada user-routines and SDL-to-Ada generation, an Ada compiler was assembled. AdaCore's GNAT LLVM [5] is used to translate Ada into LLVM bytecode, which is then translated by LLVM [6] into MSP430 assembly, finally compiled by Texas Instruments GCC [7]. The entire process is wrapped via a Python script serving as a frontend.

Validation. In order to validate the developed target support, a mock CubeSat-class satellite was designed in Capella [7], demonstrating basic power supply monitoring, thermal management, mode management and simple payload handling. This model was then manually translated into TASTE interface and deployment views, as well as SDL diagrams. PUS-C compliant-by-construction TC/TM data model was prepared in PUS-C Population Tool [8] and automatically transformed into ASN.1. Low-level hardware handling was implemented using C. The resulting binaries were deployed and tested on a flatsat build around the MSP430FR5969 LaunchPad Evaluation Kit.

Summary. TASTE is now extended with support for MSP430FR5969 MCU, enabling an MBSE-based approach in small/low-cost satellites, such as CubeSats. The performed work validates the benefits of the Kazoo template-based approach and can be a starting point for supporting other MCUs, as well as reducing the memory footprint on the existing targets.

References

[1] TASTE - A tool-chain targeting heterogeneous embedded systems, using a model-based development approach (https://taste.tools/)
[2] TASTE Wiki (https://taste.tuxfamily.org/wiki/)
[3] asn1scc.IDE - Qt Creator plugin for asn1scc - ASN.1/ACN compiler for embedded systems (https://n7space.github.io/asn1scc.IDE/)
[4] FreeRTOS - Real-time operating system for microcontrollers (https://www.freertos.org/)
[5] GNAT LLVM (https://github.com/AdaCore/gnat-llvm)
[6] The LLVM Compiler Infrastructure (https://llvm.org/)
[7] GCC - Open Source Compiler for MSP Microcontrollers (http://www.ti.com/tool/MSP430-GCC-OPENSOURCE)
[8] Capella (https://www.eclipse.org/capella/)
[9] *Deployment of the PUS-C Standard in Projects supported by an Automatic Generation Toolset (PUS-Gen)* Maxime Perrotin, Serge Valera, Michal Kurowski, Arnaud Bourdoux - ADCSS 2018

SHARED DATA TYPES FOR OSRA AND TASTE

- MBSE 2020 -

28-29 September 2020

Noordwijk, Netherlands

Jan Sommer(1), Andreas Gerndt(1), Daniel Lüdtke(1),

(1)DLR (German Aerospace Center), Institute for Software Technology
Lilienthalplatz 7, 38108 Braunschweig, Germany
Email: jan.sommer@dlr.de
Email: andreas.gerndt@dlr.de
Email: daniel.luedtke@dlr.de

ABSTRACT

Demand for more complex on-board functionality for future spacecraft continues to rise, increasing the burden on often small software teams. To handle the complexity of modern on-board software, model-driven methodologies can help to capture the overall architecture and design of the software. In a later step, they also allow auto-generating source code and documentation artifacts from the model, thereby relieving software developers from monotonous tasks.

In order to support model-based software development, the European Space Agency (ESA) provides *The Assert Set of Tools for Engineering* (TASTE) and the *On-Board Software Reference Architecture* (OSRA). Both share some common design concepts like separation of concerns, component-based modeling and graphical tooling for the design tasks. However, OSRA targets mostly the design of spacecraft on-board software. At the same time, it leaves the concrete implementation of the code generators to the entity using OSRA. TASTE, on the other hand, provides a more generic framework, includes code generators for the C and Ada language and has also been applied in robotics applications as well. Unfortunately, the interworking between the two frameworks lacks a mechanism to exchange data easily without duplicating the data type information.

In complex software projects, data is exchanged by a plethora of software modules, potentially developed by different software teams. To ensure safe data exchange inside a single as well as across many different modules, essentially three basic problems need to be solved: First, there has to be a common source defining the data types, ideally with the option to define constraints. Secondly, there needs to be some way to determine if a value is valid or not. Finally, there needs to be a shared understanding about how to encode values of these data types. TASTE solves the first problem by describing its data types through the language agnostic *Abstract Syntax Notation One* (ASN.1) notation. The second one is addressed by generating test routines for validity checks during runtime. The last problem is solved by implementing encoding rules of ASN.1 or by using a custom description for the encoding of data types using TASTE's *ASN.1 Control Notation* (ACN). OSRA currently only solves the first problem through its graphical data type editor.

Our goal is to allow the exchange of data between software developed with both tools without the need for manual interventions. In the modeling phase this is mainly achieved by adding ASN.1 capabilities to OSRA: data exchange is carried out between OSRA and TASTE applications by using the same ASN.1 representation for the data types. In OSRA, data types are modeled graphically and are part of the overall model which is based on the *Eclipse Modeling Framework* (EMF). We added plugins which auto-generate ASN.1 data types from this internal representation, allowing TASTE to use them. The code-generators are implemented using the xtext/xtend framework. To make the integration bi-directional, we also implemented a basic set of the ASN.1 grammar with the xtext programming framework. It provides an editor with syntax highlighting for ASN.1, but more importantly, it allows to parse existing ASN.1 data type descriptions, e.g. from TASTE, and register them as *external types* in the OSRA model. In the end, all available data types, the ones generated by the graphical editor as well as textual ones, are captured in this model. With this approach, the first problem is solved for OSRA and TASTE by the same way using an ASN.1 notation to describe data types.

For solving the second problem for OSRA, we developed an alternative concrete implementation for the data types generated from the ASN.1 type description. In contrast to TASTE's *asn1scc* generator, the target language is not C but modern C++ using features introduced in the new standards C++11 to C++17. DLR had successful missions with on-board software written in a subset of the C++ language, such as the TET-1 and Eu:CROPIS satellite missions, the MAIUS-1 sounding rocket mission, and continues this path in the upcoming ReFEx re-entry vehicle. With today's wide

availability of compilers with support for modern C++ standards, even for embedded targets, it is now possible to leverage the newly introduced features. At the same time, common constraints for spacecraft on-board software, e.g. the avoidance of dynamic memory allocation, are kept.

The much more powerful templating system of modern C++ allows developing templated meta-classes for the common base types with build-in constraint checking. For example, numerical types can check the values based on their ranges and choice type variables can track their active field. In general, constraint compatibility is checked during type conversion. Template meta-programming techniques, i.e. variadic templates, type traits, constant expressions and static assertions, play a major role in the implementation. They direct the C++ compiler to generate the type checking for a concrete type instance. Since template resolution in C++ needs to be carried out at compile time, also many checks will produce a compile time error if certain conditions are not met, e.g. assigning numerical types with incompatible ranges, thereby directly prohibiting the introduction of possibly dangerous code into the source tree.

In situations where compile-time checks are not possible, runtime checks are carried out. They are triggered automatically by the type variable itself when its value changes, freeing the developer from remembering it manually and avoiding code clutter. Finally, with the complexity for value checking captured in the type meta-classes, the C++ code produced by the code generator from the ASN.1 input is comparably simple. This increases maintainability for the code generator. For numerical types, this can often mean simply a one line *using* statement. For structured types, only the mapping between field name and type needs to be generated. This approach leads to a clean API for the users of the type system.

To solve the problem with shared encoding rules between TASTE and OSRA, we added a prototype implementation for a serialization mechanism to the aforementioned type system. It aims to be compatible with the ASN.1/ACN encoded binary streams of TASTE. With this step, software developed with both frameworks can now exchange data based on the same data type description. TASTE also supports additionally several ASN.1 encoding rules. This is currently not the case for our data type framework. ACN was chosen for the first implementation since it allows the definition of encoding rules which makes it applicable for existing communication protocols. However, the framework is flexible enough to add further encoding rules in the future. The *Basic Encoding Rules* (BER) and *Packed Encoding Rules* (PER) of ASN.1 are good candidates to add next.

This work shows the additions to the OSRA infrastructure in order to allow the exchange of data between OSRA and TASTE based on the same data type descriptions in ASN.1. This includes enabling OSRA to read and write ASN.1 data type descriptions, the implementation of the data types in modern C++ and the serialization of the data types into an encoded binary compatible to TASTE. Some first results about the exchange of data between both frameworks are presented in this work as well. Of course, the work presented here is not only useful for the data exchange between both frameworks but also builds the basic type system on which our OSRA based code-generation can build upon in the future.

TASTE: a toolchain for multicore TSP applications

Laura Gouveia[1], Maxime Perrotin[2], Thanassis Tsiodras[3], Jérôme Hugues[4], Daniel Silveira[5]

[1] Laura Gouveia, GMV, lasequeiragouveia@gmv.com
[2] Maxime Perrotin, ESA, maxime.perrotin@esa.int
[3] Thanassis Tsiodras, ESA, thanassis.tsiodras@esa.int
[4] Jérôme Hugues, CMU/SEI, jhugues@andrew.cmu.edu
[5] Daniel Silveira, GMV, daniel.silveira@gmv.com

Keywords: Model-Based, Model Transformation, TASTE, AIR, AADL.

1 Introduction

TASTE, “The ASSERT Set of Tools for Engineering” [1], is a development environment dedicated to embedded, real-time systems. It can be used to design small to medium-size systems, relying on formal languages and based on the concept of building "correct by construction" software. It has been recently improved to include support for Time and Space Partitioning (TSP) architectures, specifically GMV’s AIR hypervisor [2], and improve code generation performance and tool expandability. In this, we addressed the challenge of generating an Execution Platform with support for multiple partitions on a multicore CPU. TASTE’s new TSP functionalities are being implemented in a complex use case, the EagleEye OBSW, deployed on a LEON4-N2X board [4] using AIR with TSP and RTEMS RTOS in multicore [5].

2 Deploying a multicore TSP application using TASTE

Deploying a TSP application using TASTE is not different that a regular TASTE application. In the following, we detail the extensions we performed on the various steps of the TASTE process. We recall the main steps:

- Interface View: The Interface View (IV) defines the logical functions and their interactions within the system. On the Interface View, functions are defined and their interfaces are specified. TASTE is then capable of generating the application code skeletons, clearly identifying where user defines the behaviour of the function. The user can specify the function behaviour either in a programming language (Ada, C, C++ and Micropython are supported), or using a graphical modelling language (SDL, Simulink, etc.), for which code cam be generated and integrated automatically. The Interface View remains unchanged for both TSP and non-TSP applications.

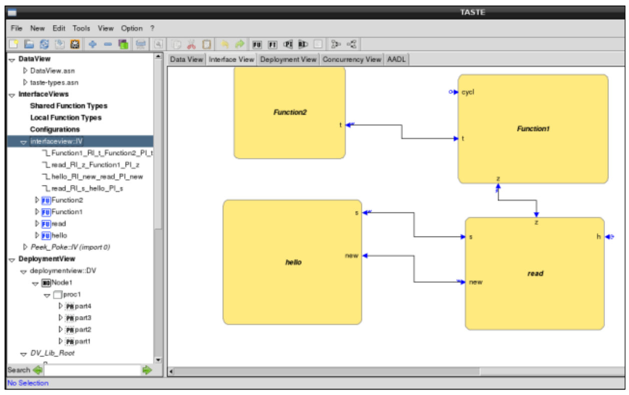
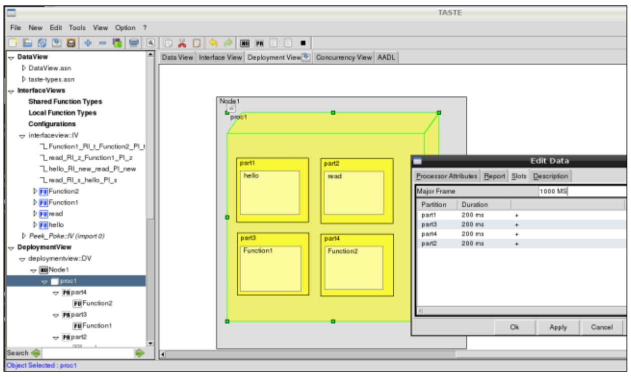


Figure 1 - TASTE Interface View

- Deployment View: The Deployment View (DV) shows how the logical functions of the system are deployed on the target hardware. The Deployment View reuses predefined hardware component descriptors that are available within an AADL library (HW Library). This library contains configuration parameters for the operating system (processor) or the communication libraries (endpoints). These elements are used by the Ocarina code generator and PolyORB-HI middleware to configure the system on the target platform.

A set of additional entities and attributes has been added to the Deployment View to support TSP architectures. Time partitioning is defined by additional scheduling attributes within the Processor, whereas space partitioning requires the definition of memory segments associated with each Partition. Additional information such as the criticality level of each Partition can be also specified.



Major Frame 1000 MS

Partition	Duration	
part1	200 ms	+
part3	200 ms	+
part4	200 ms	+
part2	200 ms	+

Figure 2 - TASTE Deployment View with partition timing slots definition

- Concurrency View: The Concurrency View is the result of an automatic model transformation whose inputs are the Interface and Deployment Views and the output is a new AADL model including a multi-threading architecture complying with the Ravenscar Computation Model (RCM). The concurrency view is used to perform code generation, but also used scheduling analysis providing two scheduling analysis functions by using Cheddar [6] and MAST [7]. In the Concurrency View, properties can be adjusted to finely tune SMP usage, such as task allocation to core and priority.

3 Code generation and build system

3.1 PolyORB-HI

PolyORB-HI is the main execution platform used in TASTE. It provides the code that interacts with the underlying operating system: RTEMS, GNAT, Linux, FreeRTOS, etc. PolyORB-HI was upgraded to support the latest version of RTEMS that is compatible with multicore platforms and the AIR hypervisor. In particular, PolyORB-HI can now interface its own communication mechanisms (queues, semaphores) with the inter-partition ports provided by AIR.

3.2 Kazoo

Kazoo is the build system of TASTE. It is in charge of computing the set of runtime resources that are needed to deploy the system on target according to the requirements from the Interface and Deployment Views. Kazoo generates the Concurrency View, together with code that ensures the system orchestration together with PolyORB-HI. In the scope of this work, Kazoo was extended to enable the deployment of threads on TSP partitions. This was made possible by the flexible design of Kazoo, which allows creating new code generators via a powerful templating engine.

4 Results and way forward

The main result of the study is an augmented MBSE toolchain that allows to specify and design multi-partition communicating systems. It benefits from a mature MBSE process that abstracts away a lot of complexity and facilitates the prototyping and deployment of TSP systems. The work is not over yet: support of I/O partitions will shortly allow to have isolated hardware-software interactions ; scheduling analysis of TSP systems based on the models ; finer-grain specification of the processor core usage in combination with multi-partitions ; integration with system-level models (via OSRA), etc. MBSE allows for moving step by step from a manual, error-prone development lifecycle to a much more solid and consistent process supported by tools.

References

1. TASTE (The ASSERT Set of Tools for Engineering) Website: http://taste.tools
2. AIR Website: http://www.gmv.com/en/Products/air/
3. ESA Contract No. 4000121551/17/NL/FE for ITT AO/1-8834/17/NL/FE – Multicore implementation of the On-Board Software Reference Architecture with IMA capability:
4. Andersson, J., Hjorth, M., Habinc, S., Gaisler J.: Development of a functional prototype of the quad core NGMP space processor. In Proceedings of Aerospace Conference DASIA (2019).
5. RTEMS real time operating system (RTOS), 2020. https://www.rtems.org/
6. F. Singhoff, J. Legrand, L. Nana, L. Marcé. “Cheddar: a Flexible Real-Time Scheduling Framework”, ACM SIGAda Ada Letters, 24(4):1-8, ACM Press. 2004
7. M. Gonzalez Harbour; J.J. Gutierrez Garcia; J.C. Palencia Gutierrez; J.M. Drake Moyano. MAST: Modeling and analysis suite for real time applications, Proceedings 13th Euromicro Conference on Real-Time Systems, IEEE, 13-15 June 2001.

"CORA-MBAD FOR ZYNQ 7000: MODEL BASED DEFINITION AND IMPLEMENTATION OF RECONFIGURABLE COTS AVIONICS"

Tiago Jorge [(*)], Laura Gouveia [(*)], Rubén Domingo [(*)], Fernando Pousa [(*)], David Arjona [(*)], Elena Alaña [(*)], Fabrizio Ferrandi [(+)], Thanassis Tsiodras [(‡)], Christophe Honvault [(‡)]

(*) GMV Aerospace and Defence
(+) Politecnico di Milano
(‡) European Space Agency

1. INTRODUCTION

The CoRA-MBAD activity ("*Compact Reconfigurable Avionics - Model Based Avionics Design*") was aimed at developing a HW/SW co-design toolchain providing functionality to easily deploy functional blocks in either HW or SW implementations, from identical source models. The toolchain developed for CoRA-MBAD was based on the TASTE toolset [1] and targeted a GR740 general-purpose processor coupled to a BRAVE reconfigurable FPGA. In this follow up activity, "*CoRA-MBAD for ZynQ 7000*", we adapt said toolchain to a ZynQ 7000 SoC target, motivated by low-cost missions that will use platforms based on COTS components such as this Xilinx SoC – which includes a dual-core ARM processor and a large reconfigurable FPGA.

2. OVERVIEW

To switch between HW and SW forms, the toolchain implements the automatic transformation of C source code (*whether manually written or generated by a code generator like those in Matlab/Simulink*) into Hardware (VHDL) source files. It additionally performs an automatic generation of the needed consistent communication interfaces supporting the exchange of commands and data between functional blocks executed on the processing system (PS) and on the programmable logic (PL) sides of the Xilinx SoC. This required adding support for the ARM Cortex A9 development toolchain (*RTEMS ARM support*), the Xilinx FPGA development toolchain (*Vivado*), and for the Advanced eXtensible Interface (*AXI*) communication interface for on-chip communication.

The toolchain was adapted to leverage the latest TASTE enhancements. The TASTE's Kazoo tool [2] was adapted to build the modeled systems with significantly increased build performance, especially in rebuilds. It efficiently produces derived models, code and scripts using AdaCore's "*templates-parser*" for templates processing and files generation.

For Matlab/Simulink models, the MBAD System relies on model-to-code transformation performed by MathWorks Embedded Coder [3] and on high-level synthesis of C code performed by Bambu [4]. Note that both TASTE and Bambu are open-source SW tools, so subsystems built in pure C can be synthesized and executed on the FPGA with no external dependencies. Bambu is FPGA vendor independent, hence it can be used with minor adaptations needed for each FPGA specific component.

The demonstrator use case is based on a computer vision algorithm that is used for vision-based navigation in the HERA project.

3. MODEL-BASED APPROACH

The complete automation and resulting cost effective extensibility made possible by the toolchain is not an easy feat to achieve since several elements need to come together, namely (see also Figure 1):

1) Basic SW and HW reusable elements: a) (SW) The RTEMS 5.1 custom built cross-compiler for ARM Cortex A9, equipped with the needed BSPs and validated on target. Additionally, some verification efforts necessarily have to target low level real time concerns such as CPU and task management,

clock frequency configuration, etc.; b) (SW) An AXI IP core control driver providing the necessarily interface configuration, initialization and read/write access to the SW applications. c) (HW) An AXI interconnect IP Core to manage and connect the AXI ports in the PS with the AXI ports of each of the modules/IPs implemented in the PL. At the same time, an AXI DMA IP controller to manage stream data transmissions between PS and PL.

2) Extensible model-based environments with high degree of automatism: a) TASTE provides heterogeneous application level modeling and implementation facilities, and importantly transparent and robust middleware level automation capabilities, in particular for communication aspects. TASTE's Kazoo allows for simple expansion and update of the supported targets, while improving code generation and build time; b) Vivado is an EDA (Electronic Design Automation) tool for FPGA and SoC, developed by Xilinx, with capabilities for low and high level synthesis, bitstream generation, timing analysis, simulation, etc. c) Matlab Simulink commonly used by domain engineers to design dynamic systems, e.g. the control and guidance of satellites designed by a GNC team, producing cyclically actuation data from sensor data, are best modelled with mathematics, data flows or functional models. This environment is extensible to e.g. incorporate as well autocoding facilities such as Embedded Coder.

3) A pivot open toolchain gluing all elements together: TASTE, being an open framework targeting heterogeneous systems, is particular suitable to integrate and orchestrate all the other necessary elements. E.g. from a common ASN.1 data model and an AADL minimalistic component interface model it consistently and automatically exports: a) interface definition in the target language of choice with consistent inputs and outputs (in our demonstrator a Simulink model); b) SW and HW wrapper interface code that transparently guarantees the correct communication between the target's functions. Importantly these interface wrappers automatically grow or shrink according to the number and type of inputs and outputs; c) SW device driver to provide SW-HW communication with the HW implementation of the target function; d) "Bridge" code with the necessary adaptations and extra inputs needed in the transition between two autocoding tools, in this case between Embedded Coder and Bambu. Additionally, TASTE e) integrates the custom cross-compiler (1-a) as part of a new deployment target, f) links with the necessary bus drivers (1-b), g) maps with the needed HW BSP exported from (1-c, 2-b), h) orchestrates the calls to all needed autocode and compilation tooling - e.g. forwarding the Embedded Coder output as a Bambu input together with the generated consistent bridge (3-d) and finally calling the synthesis facilities of Vivado (2-b).

4) Multifaceted team in co-engineering: The high technical degree of the activity required the diverse skills and close collaboration of a: a) SW engineer (1-a/b, 3-c/f), HW engineer (1-c, 2-b, 3-b/g), design environment engineer (2-a, 3-a/b/c/d/e/h), and domain engineer (2-c).

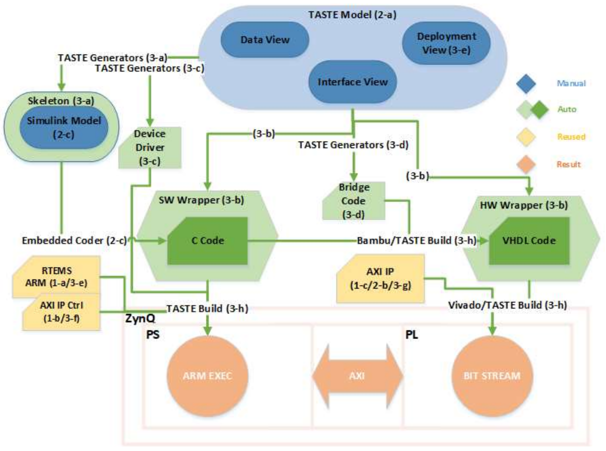


Figure 1 Toolchain overview

4. HW PROCESSING CAPABILITIES

CoRA-ZynQ makes use of AXI bridges of Zynq-7000 architecture to connect PS with PL. Three independent interfaces are implemented in order to provide different capabilities: One AXI interface used to write and read configuration registers, one AXI interface fully devoted to write and read large blocks of memory inside FPGA, and finally, one AXI stream interface to support stream data processing. The number of registers or memories addressed through AXI interfaces can be configured to optimize the resource allocation of the FPGA. In addition, AXI stream transmission can be directed to achieve up to 32 different destinations through the same interface.

5. USE CASES

The use cases implemented have as prime objectives to 1) demonstrate the toolchain new target support and to 2) support a space representative application. Objective 1 was fully achieved with simple use cases. Objective 2 is presently partially achieved with work still ongoing. A preliminary version of the HERA mission computer-vision Lambertian sphere matching of asteroid body algorithm was re-used in this context. The Matlab design reused is not tailored for a HW implementation (e.g. no parallel nor fixed-point design) which naturally represents some challenges to the autocoding facilities and HW resource usage. Such tailoring was not yet performed due to project scope and time availability. Targeting prototyping activities, the present approach is instead leveraging to the maximum possible extent the configurability and autocoding strengths of the toolchain, avoiding any manual work, e.g. by exploring the rich Embedded Coder and Bambu options, types of possible SW-HW interfaces generated (e.g. external memory access, streaming type parameters) and resulting HW resource allocation.

[1] "TASTE," European Space Agency, [Online]. Available: https://taste.tools.

[2] [Online]. Available: http://taste.tuxfamily.org/wiki/index.php?title=Kazoo.

[3] [Online]. Available: https://www.mathworks.com/products/embedded-coder.html.

[4] P. d. Milano, "PandA," [Online]. Available: https://panda.dei.polimi.it/.

ENABLING COMBINING MODELS AND TOOLS IN AN ONLINE MBSE COLLABORATION PLATFORM

Peter Gorm Larsen (1), Georgia Soulioti (2), Hugo Daniel Macedo (1), Vagelis Alifragkis (2), John Fitzgerald (3), Nikolaos Livanos (2), Holger Pfeifer (4), Mauro Pasquinelli (5), Martin Benedict (6), Casper Thule (1), Stefano Tonetta (7), Bernd Stritzelberger (8), Angelo Marguglio (9), Lorenzo Franco Sutton (9), Martin Obstbaum (8), Sergio Gusmeroli (10), Frank Beutenmüller (8), George Suciu Jr.(11), Quirien Wijnands (12) , Prasad Talasila(1)
(1) DIGIT, Aarhus University, Finlandsgade 22, 8200 Aarhus N, Denmark
(2) EMTECH SPACE P.C., 32 Korinthou & S.Davaki, 14451, Athens, Greece
(3) Newcastle University, Newcastle NE1 7RU, UK
(4) Fortiss, Guerickestraße 25, 80805 Munich, Germany
(5) Thales Alenia Space Italia S.p.A, Strada Antica di Collegno 253, 10146 Torino, Italy
(6) Virtual Vehicle, Inffeldgasse 21a, 8010 Graz, Austria
(7) FBK, Via Sommarive, 18 – POVO, 38123 Trento, Italy
(8) TWT GmbH Science & Innovation, Ernsthaldenstr. 17, 70565 Stuttgart, Germany
(9) Engineering Ingegneria Informatica S.p.A., Piazzale Dell'Agricolture 24, 00144 Roma, Italy
(10) Politechnico di Milano, Piazza Leonardo da Vinci, 32, 20133 Italy
(11) BEIA Consult International, Str. Poiana Narciselor, Nr. 12, Sect. 1 Bucharest, Romania
(12) ESA/ESTEC, Keplerlaan 1 2200 AG Noordwijk, The Netherlands

EXTENDED ABSTRACT

This paper provides an introduction about two projects and suggests how the models and tools from these could potentially be combined in the future. The projects are ACoSim supported by ESA and HUBCAP supported by the EU H2020 programme.

Since the domain of Space System Engineering, started to move away from the document-centric approach towards a Model-Based System Engineering (MBSE) approach, many standardization efforts have targeted the facilitation of Simulation Model's exchange and reuse. During these efforts, it was quickly recognized that not only the source/executable code of a model are important but also the complete life-cycle data needs to be considered to make reuse truly possible. In this respect, ECSS-E-TM-10-23 attempts to standardize topics like data exchange, semantics of the data, and repository, in order to facilitate exchange between stakeholders.

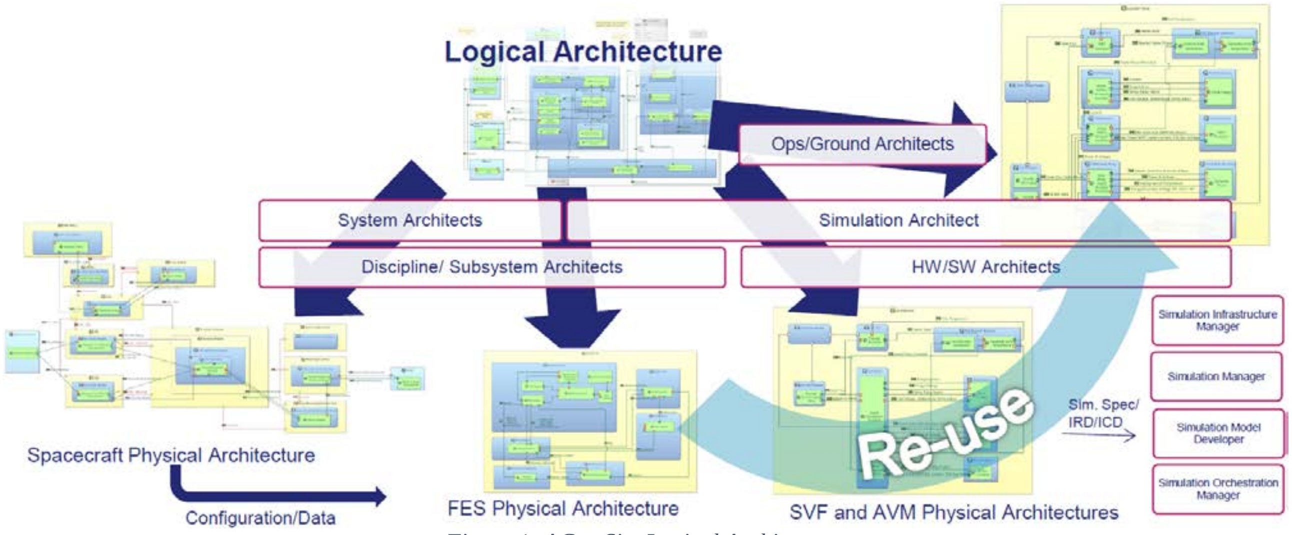
However, as there is a very tight relationship between the Product/System that is under design/development and needs to be verified, and the System Simulation Facility (SSF) that is built as part of this verification, to truly make reuse possible, it is important that the MBSE approach is extended to also consider the SSFs and the corresponding simulation models. Also, whereas in the past the SSFs development was carried out by dedicated teams based on requirements specifications, currently there is an increased tendency to incorporate and integrate the domain specific simulation models. This joint simulation can provide a higher level of fidelity. These situations are just some examples of the areas in which Co-Simulation could potentially play an important role.

The main challenge in the Co-Simulation of space simulators is to create a solution that would integrate a variety of simulators, modelling frameworks, databases, visualizers and reporting systems into a simulation that is distributed across various nodes. Preferably not excluding the heritage of space domain modelling and simulation, e.g. the used simulation kernels, standards and reference architectures. The idea is the creation of a Co-Simulation in which each application (consisting of models and solvers) executes in its own native environment, with the highest possible fidelity representative, for that purpose, of the specialized tool(s) used in the respective discipline. In order to do so, the process of setting up co-simulation needs to start during the early phases of the space systems' development, and the above challenges have to be taken into account while eliciting the requirements and performing the architectural design of the system models (during the MBSE process).

In this regard, the Application of Co-Simulation to support Tests and Operations (ACoSim) tries to bridge the gap between model-based system representation methods and the cross-domain SSFs, using combinations of MBSE and Co-Simulation enablers. For this study, three SSFs are focused that according to ECSS-E-TM-10-21 are very often recurring and amongst which much commonality exists. These SSFs are the Functional Engineering Simulator (FES), the Software Verification Facility (SVF) and the Training, Operations and Maintenance Simulator (TOMS).

A model-based approach is used to demonstrate concepts of: "how the system level architecture can be mapped down to a simulation architecture" and "how a formal model can be used to derive simulation-related information". This can be done through the usage of a common model-based definition by the system engineering team and the simulation team.

For the System Level Modelling, the Capella open source tool, developed by Thales has been used. Capella implements ARCADIA, a system engineering method based on the use of an MBSE model. The study provides an analysis of different initiatives and the relationship with standardization initiatives such as the ECSS-E-TM-10-23. Without loss of generality, only three disciplines were taken into account: GNC, Thermal and Electrical Power Management.
In the ARCADIA method, the Logical Architecture defines the functions, the exchanges between the functions and the allocation to each logical component. This is the starting point for the definition of the spacecraft physical architecture, but at the same time can also be used for the Functional Engineering Simulator. The Physical Architecture defines the HW/SW implementation. Mapping between logical components and HW/SW implementation provides the input for the mapping between the FES and the SVF in terms of control functions implemented by software or other means. The SVF is mapped to the Physical Architectures (and so, indirectly, also to the Logical Architecture, so to be able to analyze the transition between FES and SVF) The SVF and the FES can be enhanced (through bottom-up approach) to cover also AIV and TOMS needs.


Logical Architecture
Ops/Ground Architects
System Architects
Simulation Architect
Discipline/ Subsystem Architects
HW/SW Architects
Simulation Infrastructure Manager
Simulation Manager
Simulation Model Developer
Simulation Orchestration Manager
Sim. Spec/ IRD/ICD
Re-use
Spacecraft Physical Architecture
Configuration/Data
FES Physical Architecture
SVF and AVM Physical Architectures
Figure 1: AComSim Logical Architecture

The ACoSim Consortium is currently implementing the End-to-End concept with main principle to validate and enhance the proposed Co-Simulation Verification & Validation (V&V) methodology. In this regard, the Functional Mockup Interface (FMI) used in automotive industry to support Co-Simulation has been explored in ACOSIM as a candidate for facilitating the Co-Simulation in the space domain [Blockwitz14]. The advantages as known from the non-space domain when it comes to FMI-based Co-Simulations are easy: exchange of component/models, IP protection mechanisms, gaining robustness in the workflows, and coupling-possibilities of different domains and tools.
In doing so, the thermal and power discipline models as generated for the FES in MATLAB/Simulink, as well as the C/C++ GNC model, have been exported as FMI-compliant models e.g. Functional Mockup Units (FMUs), and the underlying FES scenario was executed as an FMI-based Co-Simulation. For the execution purposes, the INTO-CPS (Cyber-Physical Systems) Co-Simulation Orchestration Engine (COE) developed by Aarhus University [Thule&19], is used as the software controlling the simulation execution.
Furthermore, the reuse of these FES FMU models into SVF and TOMS environments is demonstrated within the context of ACOSIM. To achieve such reuse, generic software components have been developed. These are the so-called Enablers of the Co-Simulation that can act as building-blocks for future Co-Simulation developments. These Enablers mainly concern the interfaces/bridges of the SMP2 compliant facilities used (EuroSim for SVF and SIMULUS for TOMS) with the FMI Co-Simulation Orchestration Engine Maestro (developed by AU) as well as a bridge between SIMULUS and EuroSim environments. The latter will be developed based on the SimBridge application; a tool developed by EMTECH to enable communication between SMP2-compliant environments and external COTS tools.
Another Enabler developed by TWT, bridges the gap between the Capella modelling and the system repository with respect to the Simulation Meta Model: an automated procedure analyzes the system architecture and design as modelled in Capella and detects possible design changes. By applying various logics and identifying dependencies between the different artefacts, the system design as well as the Verification Model and Verification Environment using the FMUs and Maestro, can be tracked using the SADM server functionality, developed in the NMM (New modelling Methods in Simulation, Verification and Validation) ESA project. Next to the dependency tracing it is also shown that an automated initialization of the Co-Simulation setup becomes feasible.

The overall added value of the ACoSim project is a new methodology of how Co-Simulation methods and tools are coupled with cross-domain MBSE methods and tools to enhance the System Level Verification & Validation process in the space domain.
In such multi-partner collaborative projects, that are increasingly common in the space sector, getting started with MBSE is a challenge. This is particularly true for SMEs because of the need to acquire and manage unfamiliar tools and integrate them with others in the collaboration. This is made worse by a lack of existing models from which to start, and by the difficulty of accessing experience and expertise. An alternative to a substantial initial infrastructure investment is proposed by the HUBCAP project, which enables potential users to use a 'pay per use' schema, more attractive for SMEs. The platform would: (a) help users select MBSE tools to incorporate in current work; (b) be configurable to allow organisations to exchange models produced using different tools, including co-simulation of heterogeneous models [Gomes&18]; (c) protect IP by permitting sharing as 'black boxes' (e.g., as in the FMI standard); and (d) provide access to existing models as bases for development, with collaboration functionality to help access others' expertise.
FMI is supported by many tools such as OpenModelica [Fritzson14] and the ESA-funded ACoSim project aims to demonstrate how it is possible to incorporate FMI for modelling and simulation at different levels of ESAs Space

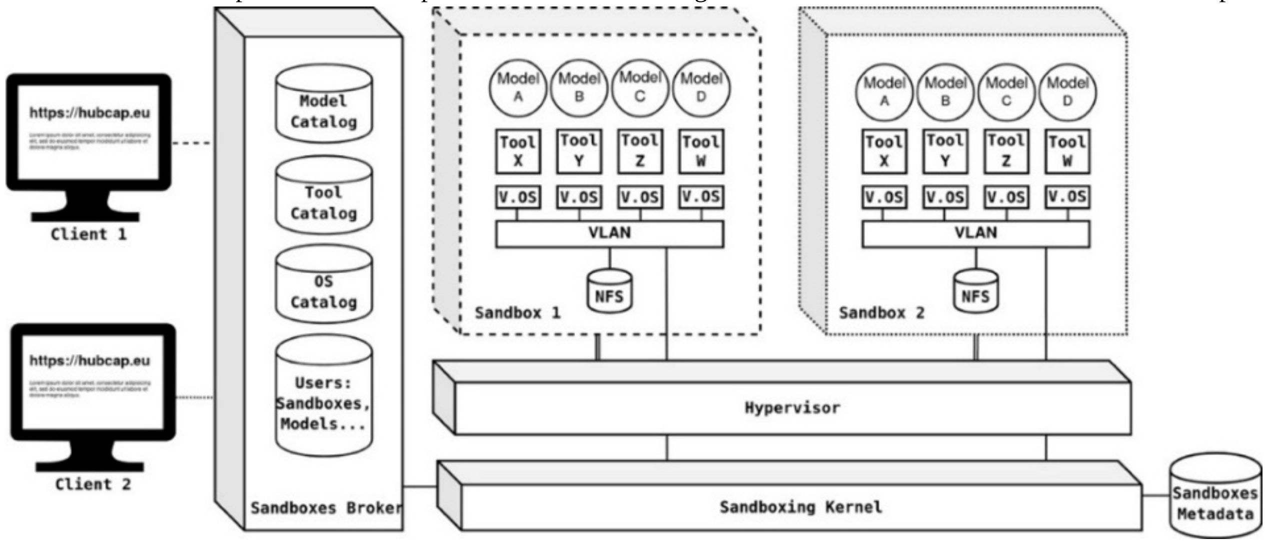


Figure 2: The HUBCAP Sandbox architecture

Simulation Facilities activities.[1] Benefits include faster convergence to collaborative models that can be shared through the supply chain, accommodating impact analysis of proposed changes. The configurability enables integration of physical components with their digital twins, saving production and maintaining V&V fidelity. This includes integration of complex multi-body models, for example during mission feasibility analysis.
The innovation necessary to create the collaboration platform is being supported by the HUBCAP project [Larsen&20][2]. The platform builds on top of the DIHIWARE open source solution[3] developed by ENGINEERING, which supports asset-need matching and joint innovation. DIHIWARE has four main subsystems: an *Identity Manager* manages user authentication and access control; a *Marketplace* handles catalogues in which MBSE assets and services will be shared; a *Knowledge Base* supports semantic indexing and retrieval; a *Social Portal* offers tools for collaboration, matchmaking, and expert search.
HUBCAP extends the DIHIWARE solution with a sandbox capability supporting white-box, grey-box and black-box models, with FMI enabling co-simulation. A sandbox (Figure 1) is a set of Virtual Machines (VMs), each one a CPS tool, interacting over a virtual dedicated subnet and NFS storage. No interaction is permitted between VMs in different sandboxes, but only within the same sandbox. The *Sandboxes Broker* hosts a web application mediating user access over an Internet browser and has access to the catalogues of available assets. Operation of user requests and sandboxing logic are provided by the *Sandboxing Kernel*, which interacts with the system *Hypervisor* to launch the constituents of a sandbox. The *Sandboxes Metadata* stores and tracks sandboxes' states and user ownership of the resources.
The sandbox design itself should ease security auditing and assurance, for example by following a trusted kernel architecture. Moreover, the components of the sandbox kernel are open source and the security will be based on Data-Service Sovereignty principles in order to enhance trust among beneficiaries and to enable use of known malware

[1] See https://digit.au.dk/research-projects/acosim/
[2] EC H2020 Innovation Action starting January 2020. See http://www.hubcap.eu/
[3] Developed in the MIDIH project See http://www.midih.eu/

detection techniques. Secure isolation and Security Information and Event Management can ensure that aggregated data/log records can be analysed giving a picture of what is happening on the platform.
The platform will provide access to assets including models and analytic capabilities of tools as services to be tested in a sandbox. Services will include modelling support with components, contracts, and equations, and analysis based on simulation, model checking, model-based safety analysis, synthesis of HW/SW deployments, fault detection and recovery, and planning. We anticipate that the platform's user community will integrate models to assist newcomers to specific modelling tools and tool combinations. Initially, we would expect to include models from standards and tutorials such as those of the INTO-CPS tool chain and those of the COMPASS tool chain developed in various ESA studies [Bozzano&19].
Models and services will be presented to the user in catalogues, where the users will choose the tool, the kind of analysis they want to try, and existing models associated to it to exemplify the usage. The platform will create a dedicated sandbox with the tool installed and the desired models ready to be used, allowing the user to perform and evaluate the analysis on the chosen model. Users will be able to write their own models and test tools' capabilities. If needed, the users will be able to get support by the tool experts via the collaboration services of the platform.
The HUBCAP Platform is under development, and we expect the first public version in late 2020. Our hope is that the ecosystem supported by this platform might encourage development of MBSE through "servitisation". In the future, users and tool suppliers will explore, share, and buy CPS assets (models, tools, services, training) from across the ecosystem through a 'test-before-invest' sandbox and -- at least in some cases -- integrated 'pay-as-you-go' charging.
We expect that, in populating the platform, we will meet limitations in the capabilities of both tools and the sandbox architecture. There may be challenges in OS licensing, and in tools that have particular hardware support needs that may not easily be supported in a sandbox context. Nevertheless, we hope that the HUBCAP Platform will be extended in several directions enabling true collaboration between diverse participants in major projects of the future.
In conclusion, the ACoSim project has analyzed and is demonstrating how Co-Simulation methods and tools can be integrated into the space domain simulation realm. In doing so, cross-domain modelling and simulation and MBSE methods and tools are considered as part of a proposed "improved Functional System Verification and Validation using Co-Simulation" methodology. Using the HUBCAP technology, it would be possible to include all the necessary models and tools in each their own VM and in this way it would essentially be a manner of combining such VMs in one sandbox and then one would be able to combine the different simulators on one server and access this from a standard browser. It is in theory possible to securely extend the sandbox to include federated and cloud-based simulation units. To securely extend the sandbox, the sandbox network needs to securely connect to hosts / networks running the simulation units [RFC3457]. Such an extension helps include proprietary simulation units in a sandbox environment.

References

[Blochwitz14] Torsten Blochwitz, Functional Mock-up Interface for Model Exchange and Co-Simulation (version 2.0), July 2014.

[Bozzano&19] Marco Bozzano, Harold Bruintjes, Alessandro Cimatti, Joost-Pieter Katoen, Thomas Noll, Stefano Tonetta: COMPASS 3.0. TACAS (1) 2019: 379-385

[Fritzson14] Peter Fritzson, Principles of Object-Oriented Modeling and Simulation with Modelica 3.3*: A Cyber-Physical Approach*, 2014.

[Gomes&18] Claudio Gomes, Casper Thule, David Broman, Peter Gorm Larsen and Hans Vangheluwe, *Co-simulation: a Survey*, ACM Computing Surveys, 51(3), May 2018.

[Larsen&20] Peter Gorm Larsen, Hugo Daniel Macedo, John Fitzgerald, Holger Pfeifer, Martin Benedict, Stefano Tonetta, Angelo Marguglio, Sergio Gusmeroli and George Suciu Jr., *A Cloud-Based Collaboration Platform for Model-Based Design of Cyber-Physical Systems*, SimulTech, July 2020.

[**RFC3457**] Requirements for IPsec Remote Access Scenarios, RFC standard, 2003, http://www.rfcsearch.org/rfcview/RFC/3457.html.

[Thule&19] Casper Thule, Kenneth Lausdahl, Claudio Gomes, Gerd Meisl and Peter Gorm Larsen, Maestro: The INTO-CPS Co-Simulation Co-Simulation Framework, *Simulation Modelling Practice and Theory*, vol 92, pp 45—61, 2019.

CLOUDSF - A CONTINUOUS INTEGRATION FRAMEWORK FOR THE DESIGN AND VALIDATION OF CYBER-PHYSICAL SYSTEMS

Gianmaria Bullegas[1], Anurag Kapur[1], Mini C Saaj[2], Manu H. Nair[2], Adrian Pop[3], Peter Fritzson[3]

Affiliations: [1]Perpetual Labs Ltd, [2]University of Lincoln, [3]Linköpings universitet
Email: gian@perpetuallabs.io, anurag@perpetuallabs.io, msaaj@lincoln.ac.uk, 18710796@students.lincoln.ac.uk, adrian.pop@ri.se, peter.fritzson@liu.se.

Keywords: MBSE, LAST, FMI, FMU, Modelling, Co-simulation

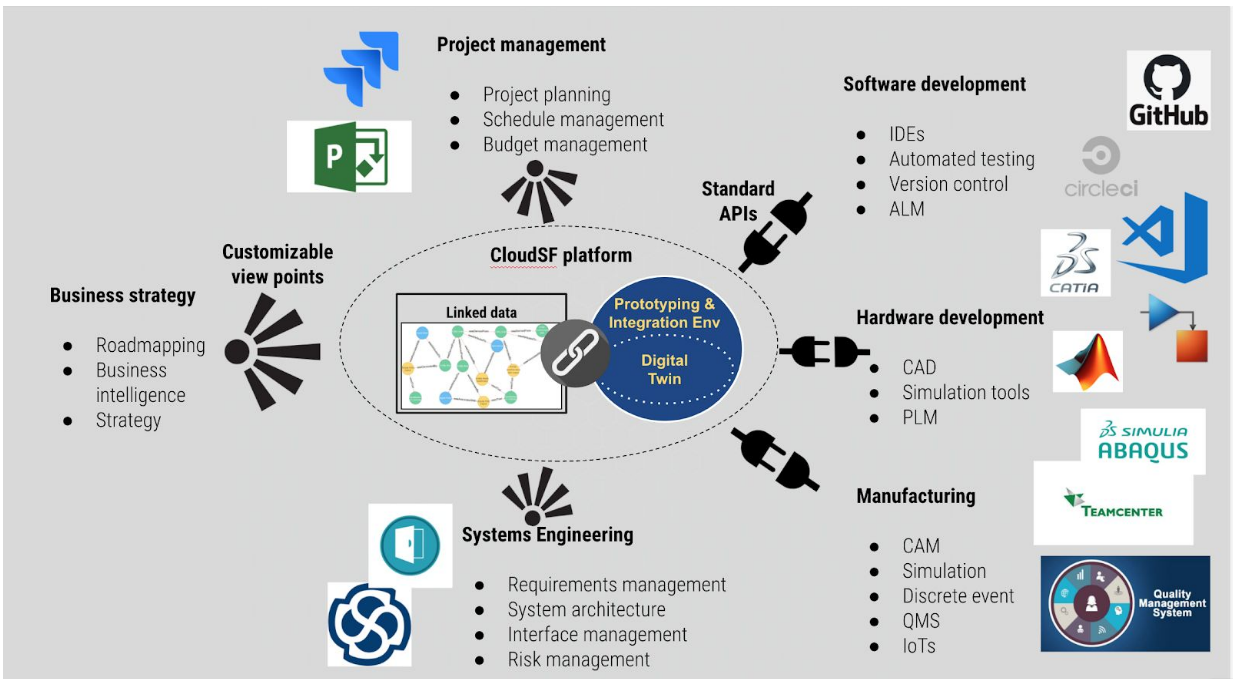


Figure 1. Schematic representation of Perpetual Labs' CloudSF platform in relation to the different engineering disciplines and business functions. These could be part of a single organization or being distributed across the supply chain (i.e. extended enterprise.

Introduction

The extensive use of virtual prototyping methods has become an indispensable tool in the context of Model-Based Design of complex space missions. Modelling the behaviour of such missions often requires considering systems that are composed of physical subsystems (usually from different physical domains) together with computing and networking. These are generally referred to as Cyber-Physical Systems (CPS).

A frequent problem in larger space projects is that, although component-level models and simulations are available, it is a big hurdle to integrate them into larger system-level simulations. This is because different development groups and disciplines, e.g., electrical, mechanical, power, and software, often use their own approaches and dedicated tools for modelling and simulation.

The Functional Mock-up Interface (FMI) [1] has been developed as a standardized exchange format for behavioural models to improve the interoperability of domain-specific models. Model components are exported as Functional Mock-up Units (FMUs) from their respective discipline specific tool. Then a dedicated simulator tool can import the FMUs in a co-simulation environment and integrate them into a composite model of the entire CPS using a suitable master algorithm for coupling the individual units.

However, coupling the different simulator codes contained in the individual FMUs to perform full-system simulation still presents major challenges and is an active research area. Some of the main challenges are:

- **Numerical stability**: modular simulation of a global system by coupling different simulator codes may easily result in an unstable integration [2].
- **Uncertainty quantification**: this is particularly important for large composite system simulations where the uncertainty propagation could rapidly undermine the confidence in the simulation results.
- **Computational scalability**: high-fidelity N-code simulations (FEA, CFD, logical) can take as much as 1 h CPU-time for every real-time second of behaviour prediction [3]. This type of analysis requires significant computational resources which in turn put high-demand on the High-Performance-Computing (HPC) infrastructure.

In addition to the core technical challenges listed above, there are challenges related to the adoption of such virtual prototyping environments at the organizational level. These include:

- **Integration with overarching SE framework and toolchains:** during system development, heterogeneous artefacts are generated, often using different lifecycle modeling languages and simulation tools, leading to integration and interoperability issues.
- **User Interface**: the virtual prototyping environment must be accessible through an integrated Modelling and Simulation Environment.
- **Multi-user collaboration**: design of complex CPS requires expertise in many different domains and often results in large cross-functional teams. Coordinating the exchange of data and information from the different domains is a major challenge and an active research area.

It is understood that collaborative web-based tools and model editors, supported by a powerful system ontology and data infrastructure in the backend, are key to tackle these problems.

The Cloud System Factory (CloudSF) platform

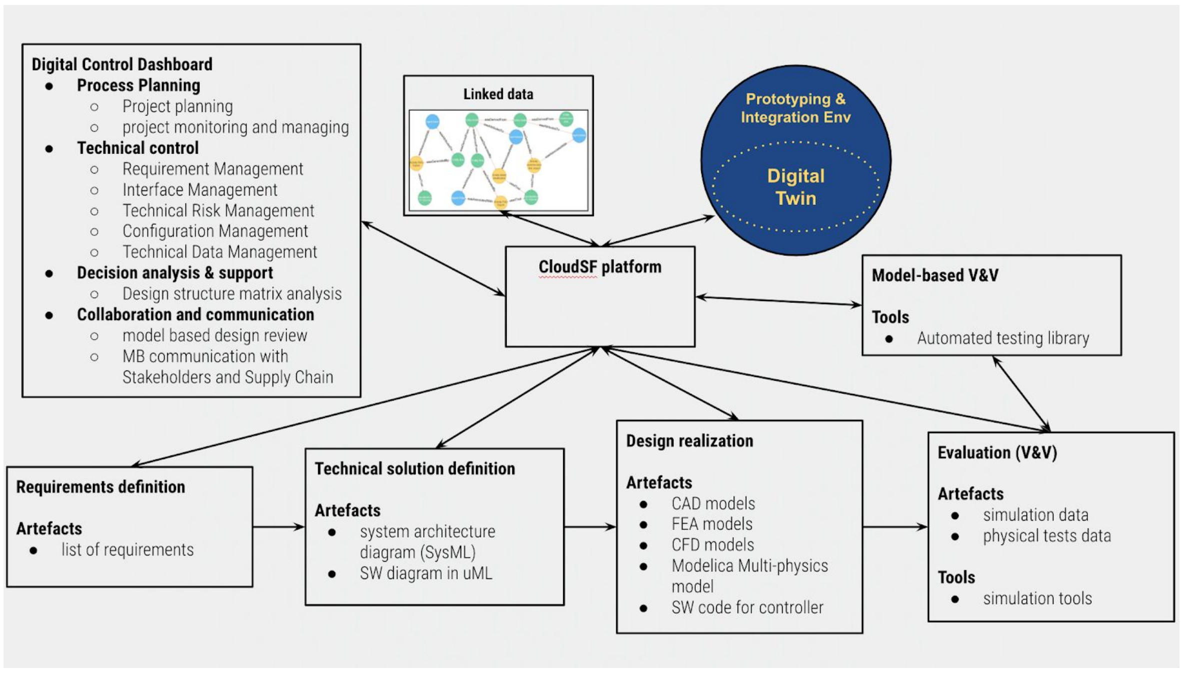

Digital Control Dashboard
Process Planning
Project planning
project monitoring and managing
Technical control
Requirement Management
Interface Management
Technical Risk Management
Configuration Management
Technical Data Management
Decision analysis & support
Design structure matrix analysis
Collaboration and communication
model based design review
MB communication with Stakeholders and Supply Chain
Linked data
Prototyping & Integration Env
Digital Twin
CloudSF platform
Model-based V&V
Tools
Automated testing library
Requirements definition
Artefacts
list of requirements
Technical solution definition
Artefacts
system architecture diagram (SysML)
SW diagram in uML
Design realization
Artefacts
CAD models
FEA models
CFD models
Modelica Multi-physics model
SW code for controller
Evaluation (V&V)
Artefacts
simulation data
physical tests data
Tools
simulation tools

Figure 2 Schematic representation of Perpetual Labs' CloudSF platform in relation to the different stages of the system development life cycle and the corresponding systems engineering processes.

Perpetual Labs is developing a new software platform for collaborative design of CPS called Cloud System Factory (CloudSF). It enables all the stakeholders of a complex engineering system to exchange system data and engineering artefacts independently of the specific tools that they are using (see figure 1). It includes the following key components and features:

- **Conceptual Data Model (System Ontology) environment**. It uses a Linked Data approach to capture traceability information and create semantic links that relate heterogeneous artefacts through the product design lifecycle. The traceability data is stored in a graph database which enables artefacts from different tools to be connected and queried through a standardized interface and language.
- **Virtual Prototyping and system verification environment (ViPro)**: It enables scenario-based simulation at the system-level (composite model) for model-based verification of system requirements. It significantly reduces the need for physical integration and requirement verification testing. It enables the application of Continuous Integration practices to the design of CPSs.
- **Web-based Integrated Design, modelling and simulation Environment (IDE)**: It provides support for the major development phases, such as requirements analysis, system modelling, verification and maintenance through well-integrated and easy-to-use functions. It automates the process of submission of simulation tasks to computing platforms, monitoring, retrieval and analysis of results. It provides real-time collaborative model editing features.
- **Digital Dashboard (Dashboard)**: It enables the creation of customizable viewpoints on the engineering data depending on the specific domain and business function. It allows users to leverage the power of semantic query languages (such as SPARQL) to interrogate the system ontology and engineering database in a fast and intuitive way to support system analysis and automated report generation.

The CloudSF platform supports a Linked Data approach through the adoption of the Open Services for Lifecyle Collaboration (OSLC) standard connectors [4]. This solution enables the definition of the semantic relationships between the different engineering artefacts in a tool-independent fashion and the easy creation and maintenance of a global system ontology. The OSLC standard natively supports any RDF-based ontology language such as WeB Ontology Language (OWL) [5] and, for extension, the Object Role Methodology (ORM) [6] and Ontological Modelling Language (OML) [7]. For an updated list of lifecycle tools that support OSLC APIs, see [8]. The use of Linked Data, supported by a global system ontology, allows to easily establish and maintain a single source of truth across the structure of the extended enterprise and throughout the product development lifecycle (see figure 2).

Technology application

There is an urge for developing novel Robotics, Automation and AI (RAAI) technologies that will facilitate in-space manufacturing and assembly of Large-Aperture Space Telescopes (LASTs), instead of Earth-based assembly (figure 3). Advancements in RAAI is also indispensable for active debris removal missions, spacecraft servicing operations in LEO/GEO (life extension, refuelling, orbit correction), space-based power generation and in-space assembly of other larger structures like super-large Radars and Synthetic Aperture Radars.

The main challenge in the development of RAAI technologies for space missions is the increasing complexity of these systems. In particular, this is due to the rising importance of connectivity and non-deterministic software components (such as Machine Learning and Artificial Intelligence for machine vision and manipulation). Another important challenge is the necessity for collaboration of multiple entities with different design processes and tools.

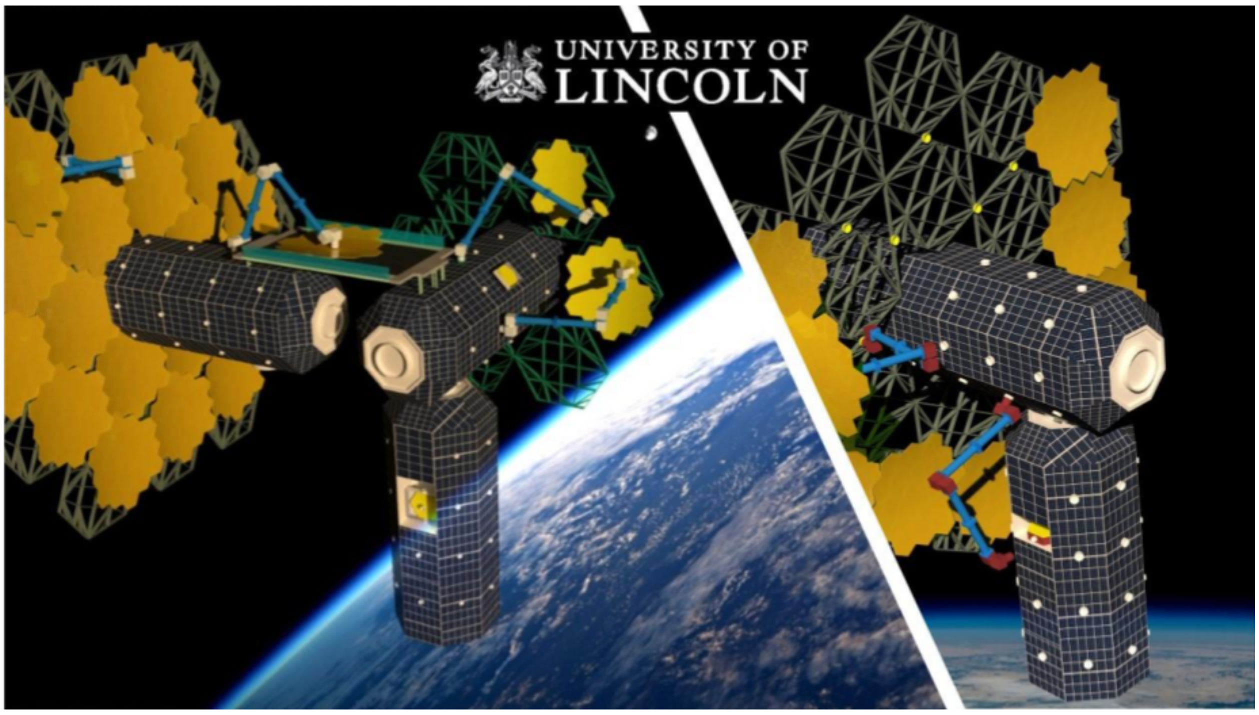


Figure 3. Robotic assembly of LAST mission, rendering courtesy of L-CAS centre at the University of Lincoln.

The proposed benefits of the CloudSF platform will be demonstrated and measured through the application of said platform to the model-based design and verification of a robotic system for on-orbit assembly of telescopic structures using an End-Over-End Walking robot, called the E-Walker (see figure 3) [9,10].

REFERENCES

[1] Functional Mock-up Interface, (n.d.). https://fmi-standard.org/ (accessed March 13, 2020).

[2] T. Schierz, M. Arnold, C. Clauss, Co-simulation with communication step size control in an FMI compatible master algorithm, Proceedings of the 9th International MODELICA Conference, September 3-5, 2012, Munich, Germany. (2012). https://doi.org/10.3384/ecp12076205.

[3] A. Van der Velden, High-Fidelity Simulation Surrogate Models for Systems Engineering, in: Disciplinary Convergence in Systems Engineering Research, Springer, Cham, 2018: pp. 327–339.

[4] Open Services for Lifecycle Collaboration, (n.d.). https://open-services.net/ (accessed August 16, 2020).

[5] OWL - Semantic Web Standards, (n.d.). https://www.w3.org/OWL (accessed August 16, 2020).

[6] Contributors to Wikimedia projects, Object-role modeling, (2004). https://en.wikipedia.org/wiki/Object-role_modeling (accessed August 16, 2020).

[7] opencaesar, opencaesar/oml, (n.d.). https://github.com/opencaesar/oml (accessed August 16, 2020).

[8] OSLC Fest 2020, Day 1, 2020. https://www.youtube.com/watch?v=FxMAyHqEay8 (accessed July 22, 2020).

[9] A. Nanjangud, P.C. Blacker, A. Young, C.M. Saaj, C.I. Underwood, S. Eckersley, M. Sweeting, P. Bianco, Robotic architectures for the on-orbit assembly of large space telescopes, in: Proceedings of the Advanced Space Technologies in Robotics and Automation (ASTRA 2019) Symposium, European Space Agency (ESA), 2019. http://epubs.surrey.ac.uk/851853/7/19_nanjangud.pdf (accessed July 22, 2020).

[10] A. Nanjangud, C.I. Underwood, C.P. Bridges, C.M. Saaj, S. Eckersley, M. Sweeting, P. Biancod, Towards Robotic On-Orbit Assembly of Large Space Telescopes: Mission Architectures, Concepts, and Analyses, in: Proceedings of the International Astronautical Congress, IAC, International Astronautical Federation, 2019: pp. 1–25.

Bridging the Gap between On Board and Ground Configuration Data Bases

Flying an Onboard Software (OBSW) is not just flashing the EEPROM. The binary image has to be delivered together with source code and memory map, but also with many software artifacts such as User and Operation Manual (UOM), Interface Control Documents (ICDs), Test Environments and Satellite Reference Database (SRDB) for configuring the Ground Facilities. Most of these artifacts are developed independently from each other, the coherency of the complete set being verified through long reviews and exhaustive integrated tests that comes at a late stage in the process.

A more efficient way is to ensure coherency by deriving all these products from a single source of truth based on a single formalism, through validated chains of model transformations. Some initiatives have shown the benefits of such approaches. These are however for now mainly limited to flight code, test environment and interface document. It is proposed to extend them to the configuration Ground Facilities.

The PUS-C Gen study has for instance demonstrated that a Generic PUS-C model could be customized and extended for a specific mission. The result of this customization is, amongst others, an ASN.1 definition of the Space-Ground interface. From this definition, the ASN1SCC compiler is able to generate encoders and decoders for the OBSW and for the test environment, but also the ICD in HTML representation.

At the other end of the communication link lies the Ground Facility, with legacy products such as SCOS2000 or more recent ones such as the EGS-CC. In order to communicate with the Space Segment, the Ground Segment has to be configured with the very same TM/TC description as the Space Segment. As today, there is however no direct formal link between the ASN.1 definition and the SCOS2000 Database. These are two different kind of formalisms. On one hand, we have a textual definition supported by a Domain Specific Language. On the other hand, we have a pure Relational Database schema. Is it possible to bridge the gap?

The proposed solution is to capture a decorated version of the ASN1SCC Abstract Syntax Tree (AST) in a model that contains all the information regarding the TM/TC structure and representation. This information is the as the one used to generate the encoders and decoders software and the interface documents. However, the ASN.1 description only covers the structure and representation while the SRDB contains additional information such as calibrations, ground monitoring and display information. The AST model has therefore to be integrated in a larger model that allows capturing this additional information while, for each TM/TC, it shall refer to the AST elements, including fields data type, size and (variable) offset.

A chain of model transformations can then lower the representation into a relational database. During the transformation, all the intermediate models are verified according to constraints and mission specific naming conventions are generated. The user’s comment in the PUS-C Gen ASN1 TM/TC definition can even be automatically flown down into the description of the corresponding field in the SCOS2000 display.

The processing involves model to model transformation through three levels of models: Domains Models, a single Unified Model and Implementations Models. Such a chain of processing is common inside a single tool. What is introduced here is the Unified Model that leverage tools interoperability. The intend of this Unified Model is not to have again one single universal (and complex) model that suggest/enforce a combination of notations and tools. It is to have a light modeling environment allowing small teams to customize the tools they already master to their needs.

Domains models are specific to one aspect of a system (system decomposition and interfaces, behavior, data representation,…).

Front-end tools first analyze these Domain Specific Models, providing their contributions to the customized Unified Model. These tools are Graphical or Textual DSL Editors. Most of them are supported by ‘Solvers’ that generate a solution according to domain constraints. Examples of such tools are: DSL compilers for ASN1, SDL, AADL languages or modeling environment for UML, Matlab, Simulink

The Unified model then puts in relation and extend the output from the previous processing in a common and general purpose modeling environment based on interchange XML files. Cross model validations are then performed.

This Unified model is then lowered in different Implementation models using M2M transformations. Software artefacts are then emitted from Implementation models using final M2T transformations. Examples of such implementation artefacts are: SW or HW configuration tables, technical documentation, relational databases, inter-operability interfaces or FPGA bitfiles, …

With this approach, the SRDB content is, by construction, congruent with the OBSW. All this information is generated from a single high-level source of information, through a trustable chain, reducing system integration and validation effort.

Dominique TORETTE
SPACEBEL

CoCoSim: an automated analysis framework for Simulink/Stateflow [*]

Applying V&V techniques for safety requirement on Simulink/Stateflow models.

Hamza Bourbouh[†‡] Guillaume Brat[‡] Pierre-Loïc Garoche[§]

Abstract

Performing verification and validation (V&V) early in the development cycle of critical systems can help reduce the cost and time of detecting and fixing errors. Thus, performing V&V at the design level helps eliminate potential problems before the software is fully implemented. Our objective is to enable the verification of Simulink (a graphical dataflow modeling language widely used in the design of flight control systems) models with respect to formal properties that represent system requirements. In this paper, we present the CoCoSim toolbox: an open source framework for specifying and verifying user-defined requirements on Simulink models. The open architecture of the tool enables the integration of multiple analyses (ours and promising ones in the research community for instance) in a bid to truly enable the application of formal verification methods to Simulink/Stateflow models.

We believe that model-based system engineering combined with tools supporting both code development and V&V activities could make a huge impact on the fast development of space systems. In addition, the open-source feature of the framework eases the integration of state of the art tools and methods from academia, enabling their uses by the industry practionner.

1 Introduction

1.1 Context

Safety-critical systems design requires a thorough development process including formal verification and correct by construction behaviour. In that area, Model-Based Design has been widely used for software development. Such an approach offers the refinement of a system from High Level Requirements down to the embedded code while having an executable model at different stages. Matlab/Simulink[1] from MathWorks, is a *de facto* model-based design standard in industry, offering verification and code generation means.

Nonetheless, other development frameworks are used in addition in some industries, such as aeronautic, railways or space. Indeed, control/command applications have received a particular attention over the years and several synchronous programming languages such as Esterel [1], Lustre [7] or Signal [15] have been defined to help their design. Scade [9] is an industrial and DO 178C qualified Lustre-based framework that provides strong guarantees and proofs well appreciated, in particular for certification.

Offering frameworks linking Simulink and synchronous approaches is thus appealing. CoCoSim belongs to this category as it is an *open source* tool that translates Simulink specification in Lustre while preserving semantics and providing many associated traceability or test capabilities.

This paper gives a brief overview of the CoCoSim architecture and its current capabilities. While Simulink models are more general and could manipulate both continuous time and discrete time systems, their semantics is not as formally defined as it is for a language such as Lustre. In our work we restrict Lustre to the discrete-time subset of Simulink constructs, which is a reasonable assumption when considering models that will be auto-coded into embedded devices.

2 Overview of CoCoSim

CoCoSim is a highly automated framework for verification and code generation of Simulink/Stateflow models. It consists of an open architecture, allowing the integration of different analyses. CoCoSim is structured as a compiler, sequencing a series of translation steps leading, eventually to either the production of

[*]The work was partially supported by projects ANR-17-CE25-0018 and NASA Contracts No. NNX14AI09G and NNA14AA60C.

[†]KBR Inc.

[‡]NASA Ames Research Center

[§]ENAC, Université de Toulouse, France

[1]https://www.mathworks.com/products/simulink.html

source code, or to the call to a verification tool. By design, each phase is highly parametrizable through an API and could then be used for different purposes depending on the customization. The Figure 1 outlines the different steps.

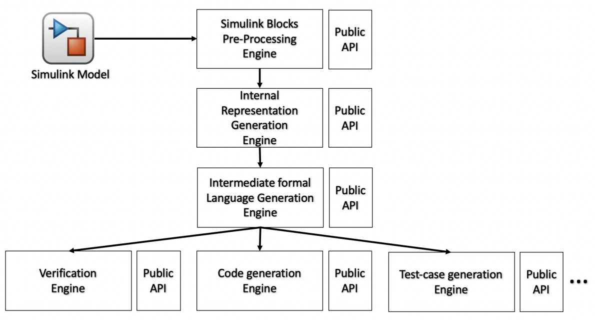

Simulink Model
Simulink Blocks Pre-Processing Engine
Public API
Internal Representation Generation Engine
Public API
Intermediate formal Language Generation Engine
Public API
Verification Engine
Public API
Code generation Engine
Public API
Test-case generation Engine
Public API
...

Figure 1: CoCoSim framework

2.1 Formal semantic

CoCoSim provides a **formal semantic** of a well defined subset of Simulink/Stateflow blocks. This formal representation will permit the use of formal verification methods and code generation.

CoCoSim starts first by simplifying some complex blocks into a set of basic blocks. Then an internal representation of the model is generated containing all information needed for code generation. Based on the work of Caspi et al. [6], Gene-Auto [13, 16] and P [3] projects, CoCoSim translates modularly the pre-processed mono-periodic Simulink model into an equivalent Lustre model. The generated Lustre model has the same hierarchy as the original Simulink model and preserves the initial semantic.

CoCoSim is **customizable** and **configurable**. Indeed, it supports most of frequently used Simulink blocks libraries (around 100 blocks) and new blocks can be easily supported.

2.2 Supported analyses

Once a formal representation of Simulink model is generated, CoCoSim is connected to a set of external tools to provide code generation, formal verification or test case generation. The toolchain is **highly automated** as all the steps of verification or code generation are automated.

The external tools are introduced and linked to the platform in a very generic way. While CoCoSim is built mainly around a specified set of tools, additional ones can be easily locally linked or even distributed as extensions.

All CoCoSim analyses are performed on the compiled artifact and the results are expressed back at Simulink level thanks to **traceability** information. We sketch here the features of the connected tools. At the current moment all tools are open-source and freely available. It **scales** well with large models, therefore various verification techniques and compositional reasoning can be used.

Formal Verification: SMT-based model checking Once requirements have been expressed using CoCoSim library and attached to the Simulink model, different tools can perform SMT-based model checking and check their validity. In case the property supplied is falsified, CoCoSim provides means to simulate the counterexample trace in the Simulink environment. Currently, CoCoSim is connected to Kind2 [8] a powerful tool that implement multiple algorithms including k-induction [14] and IC3/PDR [4] as well as on-the-fly invariant generation. All of these can be performed with various SMT solvers: CVC4, Z3, Yices.

Code generation: Some of CoCoSim backends provide code generation. Eg. LustreC [12] is an implementation of the modular compilation scheme [2] used in Scade. It preserves the hierarchy of the initial model, easing the checking of traceability between Lustre and generated C code.

Test cases generation: CoCoSim generates test cases based on two different methods. In the first method a coverage criteria such as MC-DC is used. The second approach relies on the notion of mutants. A good test suite distinguishes valid program from mutants.

3 Experiments

Since we started this effort of applying Lustre-level analyses to Simulink models, we have had the opportunity to evaluate the approach and the applicability of CoCoSim on reasonably large examples. Among them the NASA Transport Class Model (TCM) [5], the model describing the attitude and orbital control system (AOCS) of the Space Shuttle, the nominal mode of the AOCS of a French scientific satellite (DEMETER) or on other industry-provided examples such as publicly available[2] Lockheed Martin Cyber Physical Systems (LMCPS) challenges [10, 11] which is a set of aerospace-inspired examples provided as text documents specifying the requirements along with associated Simulink models. Examples range from a basic integrator to complex autopilots. The complete case study and analysis results are presented in our technical report.[3]

[2]https://github.com/hbourbouh/lm_challenges

[3]`https://drive.google.com/drive/u/1/folders/1GsKiu_O9_OSK_5XcLZZefi6g9MDAeOCC`

References

[1] Gérard Berry and Georges Gonthier. “The Esterel synchronous programming language: design, semantics, implementation”. In: *Science of Computer Programming* 19.2 (1992), pp. 87–152. ISSN: 0167-6423.

[2] Dariusz Biernacki, Jean-Louis Colaço, Grégoire Hamon, and Marc Pouzet. “Clock-directed modular code generation for synchronous data-flow languages”. In: *LCTES’08*. 2008.

[3] Matteo Bordin, Tonu Naks, Marc Pantel, and Andres Toom. “Compiling heterogeneous models: motivations and challenges”. In: *Proceedings of the 6th International Congress Embedded Real Time Software (ERTS’12)*. 2012.

[4] Aaron R. Bradley. “IC3 and beyond: Incremental, Inductive Verification”. In: *CAV’12*. 2012.

[5] Guillaume Brat, David H. Bushnell, Misty Davies, Dimitra Giannakopoulou, Falk Howar, and Temesghen Kahsai. “Verifying the Safety of a Flight-Critical System”. In: *FM 2015: Formal Methods - 20th International Symposium, Oslo, Norway, June 24-26, 2015, Proceedings*. 2015, pp. 308–324.

[6] Paul Caspi, Adrian Curic, Aude Maignan, Christos Sofronis, and Stavros Tripakis. “Translating Discrete-Time Simulink to Lustre”. In: *Third International Conference on Embedded Software EMSOFT*. 2003, pp. 84–99.

[7] Paul Caspi, Daniel Pilaud, Nicolas Halbwachs, and John Plaice. “Lustre: A Declarative Language for Programming Synchronous Systems”. In: *POPL’87*. 1987, pp. 178–188.

[8] Adrien Champion, Alain Mebsout, Christoph Sticksel, and Cesare Tinelli. “The Kind 2 Model Checker”. In: *Computer Aided Verification - 28th International Conference, CAV 2016, Toronto, ON, Canada, July 17-23, 2016, Proceedings, Part II*. 2016, pp. 510–517.

[9] Jean-Louis Colaço, Bruno Pagano, and Marc Pouzet. “SCADE 6: A formal language for embedded critical software development (invited paper)”. In: *11th International Symposium on Theoretical Aspects of Software Engineering, TASE 2017, Sophia Antipolis, France, September 13-15, 2017*. 2017, pp. 1–11.

[10] Chris Elliott. “An Example Set of Cyber-Physical V&V Challenges for S5, Lockheed Martin Skunk Works”. In: *Safe & Secure Systems and Software Symposium (S5), 12-14 July 2016, Dayton, Ohio*. Ed. by Air Force Research Laboratory. 2016.

[11] Chris Elliott. “On Example Models and Challenges Ahead for the Evaluation of Complex Cyber-Physical Systems with State of the Art Formal Methods V&V, Lockheed Martin Skunk Works”. In: *Safe & Secure Systems and Software Symposium (S5), 9-11 July 2015, Dayton, Ohio*. Ed. by Air Force Research Laboratory. 2015.

[12] Pierre-Loïc Garoche, Temesghen Kahsai, and Xavier Thirioux. *LustreC*. `https://github.com/coco-team/lustrec`.

[13] Ana-Elena Rugina, David Thomas, Xavier Olive, and G. Veran. “Gene-Auto: Automatic Software Code Generation for Real-Time Embedded Systems”. In: *Proceedings of DASIA 2008 Data Systems In Aerospace*. 2008.

[14] Mary Sheeran, Satnam Singh, and Gunnar Stålmarck. “Checking Safety Properties Using Induction and a SAT-Solver”. In: *FMCAD’00*. 2000, pp. 127–144. ISBN: 978-3-540-40922-9.

[15] “Synchronous programming with events and relations: the SIGNAL language and its semantics”. In: *Science of Computer Programming* 16.2 (1991), pp. 103–149. ISSN: 0167-6423.

[16] Andres Toom, Tonu Naks, Marc Pantel, M Gandriau, and Indrawati. “Gene-Auto: an Automatic Code Generator for a safe subset of Simulink/Stateflow and Scicos”. In: *Proceedings of the 4th International Congress Embedded Real Time Software (ERTS’08)*. 2008.

BabyMOD, a Collaborative Model Editor for Mastering Model Complexity in MBSE

Nicolas Hili, Patrick Farail
IRT Saint Exupéry, 3 Rue Tarfaya, CS 34436, 31400 Toulouse, France
{nicolas.hili, patrick.farail}@irt-saintexupery.com

***Abstract*—Modelling is nowadays commonly practiced by system architects. However, it remains a difficult task that requires some advanced User Interface (UI) modelling tools to ease the design of large-scale models. BabyMOD is an interactive and collaborative model editor for mastering model complexity in system engineering. It combines three objectives: visualizing models of systems imported from authoring tools ; reviewing imported models through model annotations ; and editing existing models or creating new models from scratch. In this paper, we present a work-in-progress prototype and discusses some original features, including sketch recognition and enhanced visualization through auto-layout and animation.**

***Keywords*—System Engineering, Model-Driven Engineering, Model-Based System Engineering, Interactive Whiteboards.**

I. Motivation

Despite its proved modeling value in many engineering domains, Computer-Aided Design (CAD) tools have only a moderate acceptance by system engineers and architects to assist them in their day-to-day tasks [Robertson and Radcliffe, 2009]. The complexity of creating, editing, and annotating models of system engineering takes its root from different sources: unsuitable representations, outdated interfaces, laborious modification, and difficult collaboration [Rudin, 2019].

As a result, especially in the early development phases, system architects tend to favor more traditional tools, such as whiteboards, paper, and pencils, over CAD tools to quickly and easily sketch a problem and its solution. Among the different benefits of remaining with traditional tools, whiteboards foster collaboration and creativity as the users do not need to strictly conform to a formal notation.

A common pitfall for using traditional tools, however, is that human users are required to reproduce any sketched solutions inside formal tools when it comes to formalizing them. Modern post-WIMP[1] interfaces (e.g., electronic whiteboards) could help to automatize this task by allowing users working on a digital representation of the model that can be directly exported to be modified via modelling tools. Bridging the informality of the working sketches captured on interactive whiteboards with formal notations and representations, has the potential to lower the barrier of acceptance of CAD tools by the industry [Botre and Sandbhor, 2013], [Alblawi et al., 2019]. This acceptance can be obtained by automatically or semi-automatically translating informal sketches into their corresponding elements using a specified formal notation.

[1]Windows, mouse, and pointer interfaces.

In this paper, we present BabyMOD, a web-based model editor featuring a lighweight and intuitive interface for editing and annotating models of systems in a collaborative way. BabyMOD positions itself between interactive electronic whiteboards for sketching diagrams and model editors. As such, it shares common similarities with other academic and industrial projects, such as OctoUML [Jolak et al., 2016], [Vesin et al., 2017] and MyScript [MyScript, 2020], but it also distinguishes itself from them on various points, including its language-agnostic sketch recognition assistant and its editing and visualization capabilities.

II. BabyMOD Overview

BabyMOD (see Fig. 1) is a web-based multi-modal model editor that has been developped from the ground to adapt itself on different devices equipped with modern browsers. It can run not only on traditional devices, including laptops and PCs, but also on tablets equipped with active stylus, and large multi-touch screens. As such, it intends to cover a large spectrum of scenarios, from single-user modelling to multi-user collaborative reviewing.

BabyMOD targets three main objectives: visualizing models imported from authoring tools, editing the imported models, and reviewing them through model annotations. Minor changes to the imported models can be carried out directly in BabyMOD, but most of the cases, heavier changes will be carried out inside the authoring tools (e.g., Capella). As such, BabyMOD does not intend to replace existing authoring tools (e.g., Capella), but rather complement them with enhanced visualization and interaction features.

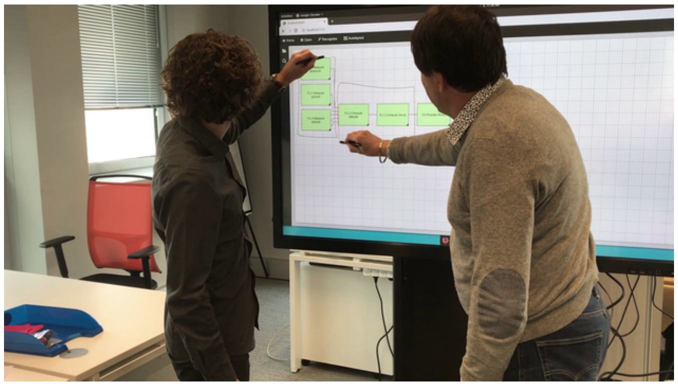
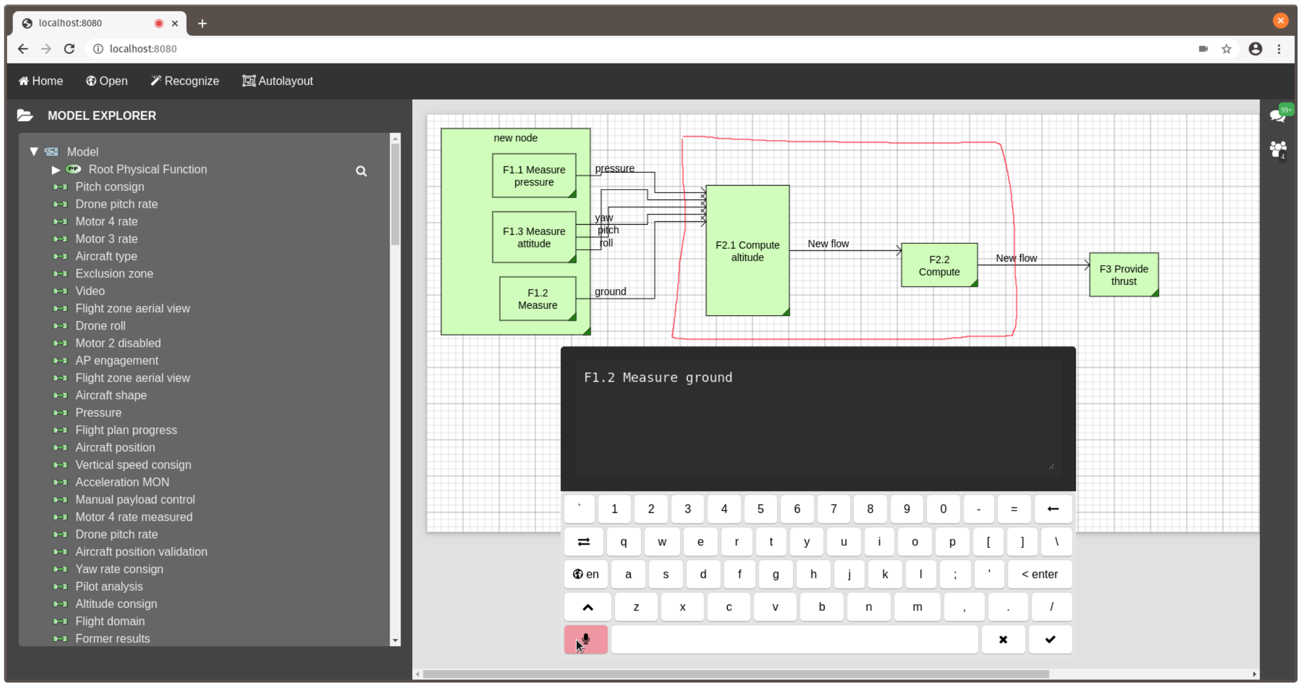


Fig. 1. BabyMOD running on a multi-touch screen where two users are collaboratively editing a functional model.



```
F1.2 Measure ground
```

Fig. 2. Overview of the BabyMOD interface: a model explorer (left-side) allows the user to explore the model hierarchically while a graphical editor (right-side) allows him/her to visualize it and to edit it in a freeform way. Model element's properties can be edited through virtual on-screen keyboard and voice recognition.

The originality in BabyMOD lies in its sketch recognition assistant that allows multiple users to edit or annotate models in a free-form modelling way by sketching elements on an interactive whiteboard (see Fig. 1). Sketch recognition is performed in real-time or on-demand and provides the user with explicable outputs in the form of a selection of choices.

Finally, BabyMOD supports basic editing features, allowing users to add new model elements into the model, remove existing model elements, and modify model element's properties. A screen cast of our current implementation and the different features it provides is available online.[2]

III. Implementation

We implemented BabyMOD using Web technologies (JavaScript and HTML5), standardized Web APIs, and open source third-party libraries, so that it makes it easer to deploy it and to run it on different devices without any prior set-up. The core element is the interface (see Fig. 2) that mainly consists of i) a Canvas-based area for visualizing and editing models graphically ; and ii) a model explorer to display models hierarchically. Fig. 3 shows the architecture of BabyMOD. Besides the core element, BabyMOD relies on different assistants, including sketch and text recognition assistants to recognise hand drawn model elements, a virtual keyboard, an auto-layout algorithm to efficiently render models, and an assistant to import models from existing tools.

[2]https://youtu.be/VRSxZr0VjKQ

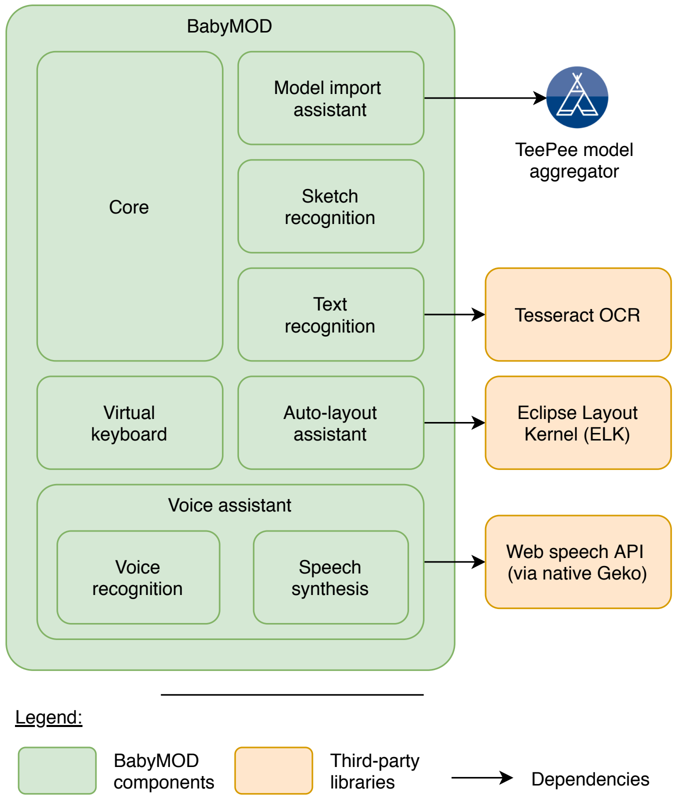


Fig. 3. Overview of the BabyMOD architecture: model import capabilities rely on the TeePee model aggregator [Baclet, 2019a].

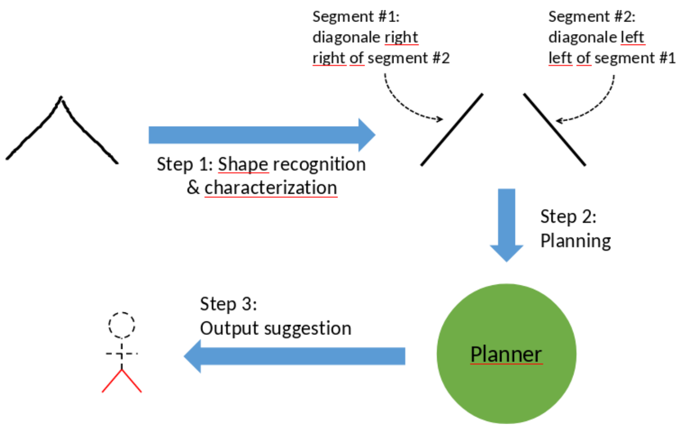


Fig. 4. Overview of the recognition approach [Albore and Hili, 2020]: first, elementary shapes from a partial draw are recognised and characterised ; Second, a planner compares the characterised elements againts a set of goals ; Third, the planner provides te user with explicable suggestions.

A. *Sketch and Text Recognition Assistants*

We implemented our sketch recognition assistant based on an approach of automated Artifical Intelligence (AI) planning [Ghallab et al., 2004] called *Plan Recognition* [Ramírez and Geffner, 2009], that consists, given an initial state and a plan, in recognising the most probable goals to reach. We adapted this approach to our domain where the initial state represents a partial draw on the interactive whiteboard initiated by a user, and a *goal library* describes the set of possible solutions in the form of model elements the user may want to draw. Fig. 4 illustrates the plan recognition approach. Details of the approach and an initial implementation are given in [Albore and Hili, 2020].

In addition to the sketch recognition assistant, text recognition, virtual on-screen keyboard, and voice recognition assistants are provided to the users to edit the different properties of model elements (e.g., the name of a function). The text and voice recognition assistants respectively rely on the Tesseract open source Optical Character Recognition (OCR) engine[3] and the standardized Web speech API[4] available in most recent web browsers. The virtual keyboard is available by the user after selecting a model element in the editor. Voice recognition can be used in different languages and the user may optionally modify the output of the voice recognition assistant using the virtual keyboard if the result is unsatisfactory due to environmental noise or wrong pronunciation.

B. *Model Representation and Auto-Layout*

BabyMOD supports one type of graphical representation of models as graphs, i.e., graphical representation composed of nodes (possibly hierarchical) and links connecting nodes (directly or through their ports). To efficiently display graph models on the screen, BabyMOD uses Eclipse Layout Kernel (ELK) to automatically position model elements in an optimal way preventing model elements from overlapping and efficiently routing the different links between the model elements. Auto-layout is complemented with animation to progressively show the result of the application of a new layout so that users are not confused by a sudden change of their models.

[3]https://github.com/tesseract-ocr/tesseract

[4]https://developer.mozilla.org/en-US/docs/Web/API/Web_Speech_API

C. *Model Import Assistant*

BabyMOD relies on prior results obtained in the MOISE project conducted at IRT Saint Exupéry [Baclet, 2019a]. *TeePee* is a model aggregator with import capabilities to import fragments of models from various off-the-shelf modelling editors (Capella, Cameo Systems Modeler, . . .) and documents (Excel spreadsheets) and aggregate them in the context of Extended Enterprises (EEs) [Baclet, 2019b]. TeePee relies on an internal data representation called *SEIM*[5] to provide a unified representation of models imported from various sources. Fig. 5 illustrates the model import capabilities using TeePee. First, TeePee converts the model into SEIM, then the model can be visualized and edited within BabyMOD. Exporting back the model into the authoring tools is currently not implemented and is a planned activity.

As only an abstract representation of a model is imported into BabyMOD, BabyMOD cannot carry specific changes requiring to have the full knowledge of the internal modelling language of every off-the-shelf editor. One benefit, however, is that users can efficiently focus on editing and reviewing tasks without being distracted by manipulating intricate concepts from the source modelling languages.

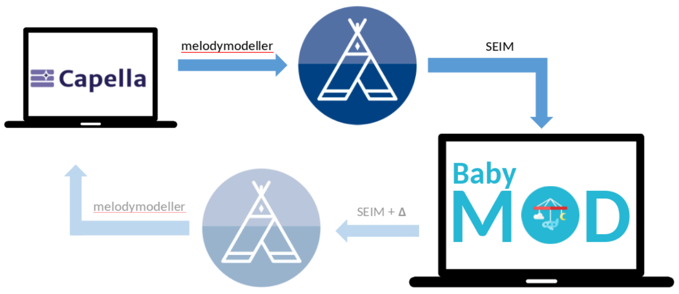


Fig. 5. Illustration of the model import facility using the TeePee model aggregator [Baclet, 2019a] for a Capella model. TeePee supports models from different sources, including Capella (melodymodeller files), Cameo Systems Modeler (Teamwork Cloud), Excel (spreadsheet files), etc.

IV. Conclusion and Future Work

In this paper, we presented BabyMOD, a work-in-progress tool for collaboratively visualizing, editing, and annotating models of system engineering. BabyMOD is a preparatory work for EasyMOD, an IRT Saint Exupéry project planned to start in 2021. EasyMOD extends the perimeter of BabyMOD with: i) multi-site collaboration; ii) support for different types of model representations ; iii) a better integration of modelling assistants – including text and voice recognition –, multi-language modelling, incremental formalization ; and iv) workflow management system integration for model review.

[5]Systems Engineering Information Model

References

[Alblawi et al., 2019] Alblawi, A., Nawab, M., and Alsayyari, A. (2019). A system engineering approach in orienting traditional engineering towards modern engineering. In *2019 IEEE Global Engineering Education Conference (EDUCON)*, pages 1559–1567. IEEE.

[Albore and Hili, 2020] Albore, A. and Hili, N. (2020). From Informal Sketches to System Engineering Models using AI Plan Recognition. In *AAAI 2020 Spring Symposium Series*.

[Baclet, 2019a] Baclet, J. (2019a). Digital Continuity for MBSE – MOISE Project. In *13th European Conference on Software Architecture (ECSA)*.

[Baclet, 2019b] Baclet, J. (2019b). Model-Based Systems Engineering in an Extended Enterprise. In *SAE 2019 AeroTech Europe*.

[Botre and Sandbhor, 2013] Botre, R. and Sandbhor, S. (2013). Using Interactive Workspaces for Construction Data Utilization and Coordination. *International Journal of Construction Engineering and Management*, 2(3):62–69.

[Ghallab et al., 2004] Ghallab, M., Nau, D., and Traverso, P. (2004). *Automated Planning: theory and practice*. Elsevier.

[Jolak et al., 2016] Jolak, R., Vesin, B., Isaksson, M., and Chaudron, M. R. (2016). Towards a new generation of software design environments: Supporting the use of informal and formal notations with octouml. In *HuFaMo@ MoDELS*, pages 3–10.

[MyScript, 2020] MyScript (2020). Myscript home page. https://www.myscript.com/. Accessed: 2020-01-15.

[Ramírez and Geffner, 2009] Ramírez, M. and Geffner, H. (2009). Plan recognition as planning. In *Twenty-First International Joint Conference on Artificial Intelligence*.

[Robertson and Radcliffe, 2009] Robertson, B. and Radcliffe, D. (2009). Impact of CAD tools on creative problem solving in engineering design. *Computer-Aided Design*, 41(3):136–146.

[Rudin, 2019] Rudin, C. (2019). Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206.

[Vesin et al., 2017] Vesin, B., Jolak, R., and Chaudron, M. R. (2017). Octouml: an environment for exploratory and collaborative software design. In *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*, pages 7–10. IEEE.