Systems Modelling and Simulation of the ESA e.Deorbit Space Debris Removal Mission

Stéphane Estable, Théo Granger, Tim Lochow, Till Zoebelein, Norbert Brauer
(Airbus, Germany);

Ilya Tolchinsky
(Phoenix Integration, France);

Sam Gerené
(RHEA Group, The Netherlands);

Hans-Peter Koning
(ESA/ESTEC, The Netherlands)

Abstract

In the frame of the ESA e.Deorbit study (Architecture Definition phase), a highly integrated and collaborative MBSE process related to Systems Modelling and Simulation has been developed and applied for supporting the iterative generation and maturation of the system requirements, architectures and system budgets at phase B1 level. e.Deorbit is a compelling mission concept that addresses the most pressing debris challenge for Europe: the post-life disposal of ESA’s environmental satellite Envisat. The e.Deorbit robotic-based chaser is characterized by strong safety (avoid collisions between chaser and target) and autonomy (capabilities to operate automatically with ground supervision) features. Additionally, the system complexity w.r.t.

1. the dependencies in and between the architectures,

2. the number of mission scenarios including the contingencies and

3. the system responses according to changes in the mission environment and the system configuration

calls for system modelling and simulation to support the proper definition of the mission and the system and architecture requirements.

During the e.Deorbit study, complemented with Airbus internal studies, a MBSE process called "Federated and Executable Models" was developed with further support of Phoenix Integration and RHEA. The main motivation of this approach is to maintain systems thinking aspects over the complete SE methodology.
MBSE patterns, data loops, data models and data sprints build the overall Systems Engineering (SE) process based on models, operability between the models and SE strategies. They produce together iteratively and successively the organization of the information leading to an optimal system definition for the system environment. They synthesize a unified whole, i.e. an open system with properties, capabilities and behaviours corresponding to the expected emergent properties, capabilities and behaviours. Using new standards for interoperability (OSLC, Linked Data) and high-end visualization technologies (VR) the MBSE toolchain is driving and fostering core characteristics of agile development tactics, such as improved collaboration and stakeholder interaction through transparency and communication. They provide a holistic, multi-disciplinary and collaborative approach to designing and maintaining complex systems.

Furthermore, this MBSE process strategy allows eliminating or mitigating the threads associated with a system definition. The architectural and design aspects in the requirements and vice versa are covered, consistent data for the different system analyses are used, the architectures are described in the same context reducing the risk to not catch the relevant relationships and the system complexity, and it is verified early in the process by simulation, whether the system fundamental properties are generated by the developed solutions.

This MBSE process is also considered as an industrial production means for system definition in studies and projects. The process might lead to a higher efficiency in the Systems Engineering work. The objective at Airbus Defence and Space is to iteratively implement and experiment this MBSE process.

W.r.t. the full MBSE process addressed in the paper, only a part of it related to architecture was implemented so far. The presented MBSE patterns and data models will need further definition and validation work in experiments before their full application in studies.

1. Introduction

In the frame of the ESA e.Deorbit project (Architecture Definition phase), data models were developed to support the definition of the architecture requirements, starting from the mission and systems requirements generated in the previous System Requirements Definition phase. Concretely, these requirements with use cases of the debris removal mission were provided by ESA in a SysML model to the industrial consortium. Three main models were then developed in the study:

1. SysML model of the CONOPS, system capabilities, functional architecture, safety and architecture requirements,
2. Product Tree model of the physical decomposition of the chaser vehicle and management of its system budgets,
3. Matlab dynamic behaviour model of the chaser navigation to the target, the robot arm and their dynamic coupling.

This initial implementation of a preliminary MBSE process has well supported the engineering team in the definition of the architecture requirements and for exchanging with ESA on the system definition work. The architecture requirements are fully defined in the further developed SysML model and linked to the architecture definition in the same model. At that stage, there was no interoperability between the models. This MBSE process is presented in [1].

In parallel to the e.Deorbit study, Airbus is working on the definition of advanced MBSE processes. In this internal Airbus activity, the focus was on the generalisation and interoperability of the MBSE process applied in e.Deorbit using the e.Deorbit engineering data for its evaluation. This paper aims at presenting the definition of a general MBSE methodology, that can more specifically be utilized and tailored for the definition of space systems requirements and solutions.

The Systems Engineering methodology at large is composed of method, process and tools. The method for Systems Engineering and its characteristics are well known [2]. The paper is more dedicated to a specific MBSE process called "Federated and Executable Models" and a selection of tools as a possible realisation of the Systems Engineering (SE) process. The "Federated and Executable Models" MBSE process will be evaluated with the e.Deorbit application using specific tools for space applications.

Section 2 will recall the systems thinking and Systems Engineering precepts to motivate the definition of the MBSE process. Section 3 will propose a process based on MBSE patterns implementing the selected Systems Engineering method and considering the systems thinking precepts. Section 4 will propose an instantiation of the defined MBSE process with specific tools for space applications. Section 5 will evaluate the proposed Systems Engineering methodology with the e.Deorbit application.

2. Systems Theory Foundations

2.1 Systems Thinking precepts

In order to better understand the underlying concepts for the presented MBSE process we recall the principles of Systems Thinking referring to the works of Derek Hitchins [3].

According to Hitchins “Systems thinking is thinking, scientifically, about phenomena, events, situations, etc., from a systems perspective, i.e., using systems methods, systems theory and systems tools.” Systems Engineers should
thereby consider not only the System under Development itself but also its environment, which in itself presents an ever evolving system.

In recent years, as we can observe a paradigm shift towards digital continuity, Industry 4.0 or Internet of things where our products have become parts of a whole, which can be considered as an open system or System of Systems. For our studies a system follows the definition of “an open set of complementary, interacting parts, with properties, capabilities and behaviours emerging, both from the parts and from their interactions, to synthesize a unified whole.” [3].

There are a couple of references which give valuable definitions of Systems Thinking precepts such as holism, organicism, synthesis, variety, emergence, etc. [3]. These precepts reveal the roots of the future challenges of designing new products and services. The key takeaway of these definitions could be that today’s open systems are greater than the sum of its individual parts. Their value emerges from the symbiosis of an undetermined (i.e. large variety) set of individual constituent systems: i.e. some large and open systems exhibited properties (so-called emergent properties) that were not exclusively evident in any of their individual parts, and it was found that these properties could be synthesized by engaging the right constituent systems in the right way to create a unified super-system that creates a new value in itself.

As systems engineers we face an ever increasing challenge to predict the behaviour of such systems and finding the best balance (optimum) of the interacting parts, while ensuring robustness, viability and delivering the intended purpose.

Consequently, processes of engineering open systems have become even more dynamic as they have ever been. The arising challenges for companies to develop disruptive technologies and smart service innovations right-first-time and in short response times to the market are quite similar across traditional industries such as automobile, transport, communications and aerospace. In the Systems Engineering community there is an increasing interest for methods and tools that allow companies being faster, more flexible and more agile in their development and manufacturing. Accordingly, there is a strong need for the true adaptation of our traditional way of developing systems to new methods, processes and tools, which can leverage these characteristics. We could observe already a strong trend for more virtual engineering means such as modelling and simulation in the last decade. But today even these methods need to be advanced considering the high dynamicity of the market and to be aligned with the new envisaged processes.

Yet, the core principles of engineering methods [2] remain valid: we need to explore the problem space; we conceive one or more potential architectural solutions; design, model or prototype the best answer as response to the problem. Later one includes cross cutting methods such as optimisation and
simulation. Finally we validate and test the robustness of the solution against the test cases which lead to the acceptance by all relevant stakeholders.

The challenge though is actually to implement this schema into highly dynamic and iterative Systems Engineering processes which are required to be faster and more agile.

2.2 Systems Engineering methods

In the following we briefly review the foundations of Systems Engineering which provide the baseline of the proposed new MBSE process. The focus in this paper is on the system definition (left part of the Vee) only and not on its realization (right part of the Vee). The aim is to keep the Systems Thinking aspect as presented previously as central part of the Systems Engineering processes.

There are several definitions of Systems Engineering, e.g. [2] [3], which highlight that the main focus is on the coordination of an interdisciplinary set of processes enabling the realisation of a successful (complex) system. There is a lot of work (handbooks, body of knowledge, etc) available that describes what practitioners should consider when doing Systems Engineering right. INCOSE provides a Handbook [2] that describes what should be done in a quite exhaustive list of reference processes. Also recommendations for methods and concrete practices are given. What is not explained in those references though is a guideline how the Systems Thinking precepts as mentioned in the previous section can effectively be addressed by the processes. In our understanding these precepts essentially have to be considered though when developing a new effective SE method.

For the left side of the Vee Hitchins and INCOSE present a sound set of activities to be performed. INCOSE addresses the needs for each process such as Business or Mission Analysis, Stakeholder Needs and Requirements Definition, System Requirements Definition, Architecture Definition, Design Definition and System Analysis Process (orthogonal to part of the others). Hitchins puts more emphasis on applying system science in a sequence of iterations while addressing the problem, exploring the problem space, exploring the solution system purpose, developing the concept of operations, designing the solution system, optimizing the solution system design, and creating and proving the solution system.

In order to create an agile responsive MBSE process for our program we identified the need to find a way combining the two of them while applying to the new approach state-of-the-art techniques (methods and tools) for model-based Systems Engineering, and systems analysis and optimisation.
3. "Federated and Executable Models" Systems Engineering process

This section will describe the "Federated and Executable Models" SE process developed at Airbus. This Systems Engineering process relies on the SE method and applies the systems precepts introduced in the previous section. The focus will be on the core SE process. Information Technology aspects like the configuration management, which shall also be part of an operational MBSE SE process in its full deployment, will not be addressed here.

3.1 Purpose of the MBSE process

The purpose of the Systems Engineering process is understood as developing systematically and in parallel requirements and solutions to a problem or an opportunity. According to the system engineering method from section 2 the requirements and solutions are generated in 3 main steps:

1. Mission and System Requirements together with Conceptual solutions
2. Architecture Specification Requirements together with Architecture solutions
3. Design Specification Requirements together with Design solutions

The purpose of each of the systems definition steps is proposed as follows:

*Purpose of the conceptual solution and associated requirements:* Identify the system of interest purpose and emergent properties, capabilities and behaviours (systems fundamental properties) for the mission context including scenarios and adjacent systems.

*Purpose of the architectural solution and associated requirements:* Define a coherent architecture and its sizing considering all the main domains of a socio-technical system, which can generate the system purpose and emergent properties, capabilities and behaviours.

*Purpose of the design solution and associated requirements:* Instantiate the architecture to the target cost and schedule with the appropriate technologies, equipment reuse opportunities, verification and production means and quality rules.

3.2 Executable Data Models federated in Data Loops, Patterns and Sprints

The MBSE process is described as "Executable Data Models federated in Data Loops, Patterns and Sprints". Data models of different types can be executed like the simulation of states and activities or the execution of analysis flows. They are federated and not integrated, i.e. the data models remain independent and exchange data using standard interfaces in data loops (exchange of data between 2 models to produce new information). According to the system definition phase, the data loops can be rearranged in different manners typical for Systems Engineering activities and leading to MBSE patterns. Finally, the
strategy to run the data loops and the patterns is selected for a specific project and applied in the team organization in sprints. These different aspects of the proposed MBSE process will be depicted in this section.

Basic concepts of the MBSE process

The "Federated and Executable Models" process follows a sequence of 4 main steps:

1. Develop in parallel requirements and solutions up to the system detailed design for:
   a. Mission and System Requirements / Conceptual solutions
   b. Architecture Specification Requirements / Architecture solutions
   c. Design Specification Requirements / Design solutions
2. For each requirements / solution definition, the systems method is applied:
   a. Explore the problem space.
   b. Conceive one or more potential solutions (hypotheses).
   c. Represent the potential solutions interacting with other systems in the solution space.
   d. Prove the adequacy of solutions w.r.t. emergent properties in a variety of relevant contexts.
3. At each step of the systems method, tailor and apply the appropriate MBSE pattern from existing data models and data loops.
4. At each step of the systems method, select a strategy and arrange the chosen MBSE patterns in data sprints to produce the requested information with the engineering team.

Develop requirements and solutions

The prime purpose of the Systems Engineering process is to develop successively requirements and solutions for the system of interest. The process relies on the generation of 3 requirements/solutions iterations for conceptual solutions, architectural solutions and design solutions, as defined in the Systems Engineering method along the left part of the Vee.
Figure 1: The prime purpose of the Systems Engineering process is to develop requirements and solutions.

**MBSE patterns**

Each combination of SE move (concept, architecture, design) and systems method gives a unique MBSE pattern (each cell in the table below corresponds to 1 MBSE pattern), i.e. a specific set of data loops linking data models of different types. A MBSE pattern might be reused after a tailoring in different projects, reinforcing the reuse capability. A MBSE pattern generates new information on the system to be used in the following pattern. In principle, the full Systems Engineering process for system definition follows the sequential execution of the MBSE patterns from the top-left to the bottom-right of the table below. The starting MBSE pattern can be chosen according to the system heritage.

<table>
<thead>
<tr>
<th>Conceptual solutions and requirements</th>
<th>Explore the problem space Code E</th>
<th>Conceive solutions Code C</th>
<th>Represent the potential solutions Code R</th>
<th>Prove the potential solutions Code P</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pattern C-E: Identify issues symptoms or opportunity</td>
<td>Pattern C-C: Characterize the solution space.</td>
<td>Pattern C-R: Represent properties, capabilities</td>
<td>Pattern C-P: Prove that issues symptoms are</td>
<td></td>
</tr>
<tr>
<td><strong>Identify system purpose, and emergent properties, capabilities and behaviours.</strong></td>
<td>Characterize system environment. Define system prime directive and services provided by the system to its environment. Analyse system activities and capabilities in all life cycles.</td>
<td>Find emergent optimum properties in the solution space. Find emergent capabilities. Find emergent behaviours. Characterize CONOPS.</td>
<td>Represent the system properties. Represent the system capabilities. Represent the system behaviours. Define CONOPS.</td>
<td>Prove that the concepts can eliminate the issues symptoms. Prove that the concepts can meet the CONOPS.</td>
</tr>
<tr>
<td>Code C</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Architectural solutions and requirements</strong></td>
<td><strong>Pattern A-E:</strong> Gain detailed information on the system context. Analyse mission Analyse capabilities Analyse safety Analyse complexity Analyse behaviours Analyse system dynamic Analyse system sizing Analyse technologies Analyse costs</td>
<td><strong>Pattern A-C:</strong> Conceive solution hypotheses. Select timeline Select system level functions Concept Define system partitioning Select states Select actuators Select geometries Select technologies Trade-offs</td>
<td><strong>Pattern A-E:</strong> Represent the architectures and the requirements. Represent system states Represent functional architecture Represent physical architecture Reliability Safety Represent verification architecture Represent relationships between architectures Represent functional chains Optimize the solutions</td>
<td><strong>Pattern A-P:</strong> Prove that the system emergent properties are generated by the architectural solution. Prove in simulation that the architectures can generate the expected emergent properties, capabilities and behaviours and meet the CONOPS.</td>
</tr>
<tr>
<td><strong>Code A</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Design solutions and requirements</strong></td>
<td><strong>Pattern D-E:</strong> Analyse the design drivers and context. Analyse procurement Analyse COTS</td>
<td><strong>Pattern D-C:</strong> Identify design alternatives. Select software architecture Select visible</td>
<td><strong>Pattern D-R:</strong> Represent design alternatives and requirements. Represent technical details of</td>
<td><strong>Pattern D-P:</strong> Prove that the system emergent properties are generated by the design solution. Prove in simulation that the design can</td>
</tr>
</tbody>
</table>
Data loops

The MBSE patterns are run using a combination of data loops to produce new information in the system definition. A data loop is represented by a data model which is able to interact with other data models. Data loops are therefore dealing with interoperability aspects (see section 4.5). The table below identifies for the MBSE patterns related to the architecture solutions, the data loops involved and the type of data model.

<table>
<thead>
<tr>
<th>MBSE patterns</th>
<th>Data Loops</th>
<th>Data Models</th>
</tr>
</thead>
<tbody>
<tr>
<td>A - C</td>
<td>Mission timeline, System functions, System states, Geometries, System properties, Costs</td>
<td>SysML, Geometry model, Product Tree model, Cost model, Trade-offs</td>
</tr>
<tr>
<td>A - P</td>
<td>System architecture simulation</td>
<td>Simulator for system response analyses, sensitivity analyses, &quot;What if&quot; analyses and requirements checks</td>
</tr>
</tbody>
</table>
Table 2: Data loops implemented in data models for the MBSE patterns of the architecture solutions.

Data loops for the MBSE Pattern "Architecture - Explore"

The purpose of this pattern is to explore the problem space at architecture level to gain detailed information on the system and its environment.

The pattern is shared in 2 main parts. One dedicated to the further analysis of the physical properties w.r.t. the vehicle trajectories, the orbital parameters and system sizing. The other part further analyses the functional system capabilities, behaviours and the CONOPS in SysML use cases to identify the detailed capabilities of the system. The capabilities are then associated to the system stakeholders and actors. From the detailed capabilities the system top-level capabilities are identified in an inducing process. Potential accidents and
hazards related to the system are analysed from the use cases in the SysML model.

*Data loops for the MBSE Pattern "Architecture - Conceive"*

The purpose of this pattern is to explore the solution space for conceiving hypotheses on the architectural solutions from the previous exploration of the problem space.

A set of options on architectural solutions is generated from the analyses of the orbital parameters, trajectories and potential solutions in the parameter space, as well as from the system capabilities. A trade-off may be applied to the identified solutions to select the architectures to be represented in detail in the next pattern.

*Data loops for the MBSE Pattern "Architecture - Represent"*

The purpose of this pattern is to represent in detail the architectures and to generate the architecture requirements starting from the options identified in the previous pattern.
This pattern keeps the share between the physical and functional representations of the previous patterns. The physical architecture is extended with domain analyses for sizing precisely the system and with detailed geometries. The functional architecture includes a functional decomposition covering the interaction between the functions, the definition of the ports and the states of the system. This representation of the functions is continued in system functional failure mode analysis. The initial software architecture relies on the functional analysis results w.r.t. interactions, functional blocks and modes. A high-level integration and verification architecture is defined which integrates the physical and functional components with the test model philosophy. Once all the architectural components have been defined with their relationships, the allocations between them can be analysed in virtual reality to perform some optimization on the architectures. Finally the information generated from the architecture definition and the analysis is formulated as requirements. Iterations between architectures and requirements shall allow reaching consistency and completeness between the architectures and the requirements and within the architectures. This is better achieved when all the architectures are represented in the same modelling environment.
Data loops for the MBSE Pattern "Architecture - Prove"

The purpose of this pattern is to prove in simulation that the system emergent properties, capabilities and behaviours modelled and expressed as requirements are generated by the architectural solution.

The system simulation at architectural level combines the mission scenarios and the system modes defined in SysML with the physical representation of the product tree, mission parameters and the vehicle trajectories. The simulation calls for each mission mode the appropriate system configuration from the product tree and the trajectory from the dynamic model. A simulation run gives a response of the system in a specific scenario and mission environment. As the simulation can be executed automatically for a variation of system and context parameters, a sensitivity analysis of the system can be performed. "What if" analyses can allow the detection of non-desired behaviours of the system. With the generation of numerical data for the system parameters and the link to the performance requirements, requirements can be checked at this stage of the system definition, i.e. to verify whether the architecture is able to generate the expected properties and behaviours.

Figure 5: MBSE pattern "Architecture - Prove".
4. Tools selection for instantiating the "Federated and Executable Models" Systems Engineering process

This section will present a selection of tools for instantiating the SE process "Federated and Executable Models". Additionally, an interoperability solution for the physical architecture between Cameo Systems Modeller and CDP is described.

4.1 Tool selection for the data models

All the tools are commercial and rely on established standards to support the data exchange and interoperability. For each data model an adequate tool is selected. The table below summarizes the tools selected for the data models.

<table>
<thead>
<tr>
<th>Data Model</th>
<th>Tool</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unified Architecture Framework</td>
<td>Cameo Systems Modeller with UFDM profile</td>
</tr>
<tr>
<td>SysML</td>
<td>Cameo Systems Modeller 18.2</td>
</tr>
<tr>
<td>Product Tree model</td>
<td>Cameo Systems Modeller 18.2 and CDP 3.11</td>
</tr>
<tr>
<td>Geometry model</td>
<td>CATIA v5</td>
</tr>
<tr>
<td>Mission model</td>
<td>STK 11</td>
</tr>
<tr>
<td>Dynamic model</td>
<td>Matlab R2015a</td>
</tr>
<tr>
<td>Simulator for system response and &quot;What if&quot; analyses</td>
<td>Cameo Systems Modeller 18.2 with ModelCenter 11.2</td>
</tr>
<tr>
<td>Simulator for requirements checks</td>
<td>Cameo Systems Modeller 18.2 with ModelCenter 11.2</td>
</tr>
<tr>
<td>Sensitivity analysis</td>
<td>ModelCenter 11.2</td>
</tr>
<tr>
<td>Optimisation</td>
<td>ModelCenter 11.2</td>
</tr>
<tr>
<td>Cost model</td>
<td>Excel 2010</td>
</tr>
<tr>
<td>Virtual Reality model</td>
<td>HTC Vive / Unity Version 5.4.2 / Cameo Collaborator</td>
</tr>
</tbody>
</table>

Table 3: Tool selection for the data models.

For each domain analysis an adequate tool like Matlab or STK is selected to enable domain computations and relevant studies in the appropriate MBSE pattern.

4.2 Cameo Systems Modeller

Cameo Systems Modeller is a cross-platform collaborative MBSE environment. It permits to define, track and visualize all aspects of systems in standard SysML models. Cameo Simulation Toolkit dynamically solves constraints in the context of a full systems simulation.
4.3 Concurrent Design Platform

The Concurrent Design Platform (CDP™) is a collaborative engineering environment that focusses on the parametric modelling of both the problem specification and the solution. Engineering data is shared between different domains of expertise, each domain of expertise models parts of an architecture and parametrizes that architecture expressing properties such as mass, power consumption etc. The engineering data is shared in near real time through a data repository. The users of the CDP have access to the engineering data and can concurrently update the architecture definition and the values of the specified parameters. The CDP ensures that each user is aware of the current state of the engineering data and facilitates the analysis of the parametric data. This CDP supports an agile and iterative system engineering process where changes, potential issues and solutions can be identified at a fast pace.

![CDP™ Architecture](Image)

*Figure 6: CDP™ Architecture*

For the e.Deorbit project the CDP is used to analyse multiple parametrized architectures. The architectures follow the ECSS standard for system decomposition implemented using decomposition rules modelled in the CDP. Different kinds of budgets are generated using the CDP, e.g. mass/power, to analyse the different solutions. The CDP JSON REST API is used for the integration into the MBSE toolchain.

4.4 ModelCenter

ModelCenter enables a connection between the SysML representation of a system and simulation analyses that can predict its performance. Such a coupling can provide an accurate estimation of the system performance very early in the project cycle. System requirements are linked to performance
characteristics which gives confidence that the selected mission configuration will be successful.

For the e.Deorbit project, ModelCenter presents a view of the state of all the subsystems for every time step in the critical parts of the mission. Instead of simulating the subsystems individually, they are brought together so that unexpected interactions between them can be discovered early on. If such issues were to be discovered later in the Vee, the cost to fix the system could be orders of magnitude more expensive.

Once the simulation of the baseline system configuration is available, many types of studies can be performed. One such example could be to examine the impact of environmental mission variables. For e.Deorbit, the exact tumbling rate of the target satellite at the time of mission execution is unknown. The simulation can be evaluated for a range of possible tumbling rate values to see if the full system can support all eventualities.

4.5 MBSE toolchain interoperability framework

The aim of the MBSE toolchain interoperability framework is to provide an enabling IT infrastructure and technologies that allow users to effectively deploy the data created during the MBSE process patterns and to ensure the integrity of the data generated by different teams in different tools.

The main idea is to lift the data - usually created locally and hosted by individual tools – to a new shared level, which is theoretically accessible by anyone and from anywhere. This would allow deploying completely new services on the data, which exceed the capability of each individual tool. In addition, as the tool landscape is highly evolutionary and often depending on
preferred skills or prescribed constraints we strive for sustainable and open interoperability standards. Our integration is based on the Open Services for Lifecycle Integration (OSLC), which follows a Linked Data approach imposed by the W3C (www.w3.org/standards/semanticweb/data). It is based on using HTTP REST protocols and representing data e.g. as RDF/XML.

In our case we use this technology to impose the ECSS standard for system decomposition, while ensuring integrity of the Product Tree Model (see table 3) on both sides, the System Architecture definition environment (use of Cameo Systems Modeller / SysML) and the Concurrent Design Platform (CDP). The exposition of the System Specification (SysML to RDF) allows viewing or making connections with various other data sources. Preliminary studies have started to make links on the shared level, based on web-standards, with the PLM/CAD world and more examples digital continuity are continuously evolving.

![Diagram](image)

**Figure 8:** Toolchain interoperability between Cameo and CDP.

## 5. Systems Engineering methodology evaluation with e.Deorbit

This section presents a concrete implementation of some of the MBSE patterns using the MBSE tools described in the previous section for the ESA e.Deorbit study and evaluates the so far implemented Systems Engineering methodology "Federated and Executable Models".

### 5.1 The ESA e.Deorbit mission

e.Deorbit is a compelling mission concept that aims to address the most pressing debris challenge for Europe: the post-life disposal of ESA’s environmental satellite Envisat, which has the highest catastrophic risk-impact of any European spacecraft. This mission is unique in its operational complexity and challenges, and calls for a high fidelity system.
The capture technique based on a robotic arm and gripper combined with a clamping device was selected to implement the e.Deorbit mission. The selection has been made based on a number of key criteria, including performance, safety, cost, and its verification prior to utilization in the mission. This selection takes into account suitability of the selected technique to other possible targets and on-orbit servicing applications, and has been designed with versatility and risk-mitigation in mind. The architecture solution and requirements of the robotic system were successfully reviewed in November 2016 at the ESA System Requirements Review.

5.2 MBSE pattern implementation

In the frame of the ESA e.Deorbit phase B1 study, the activities were specifically dedicated to the definition of the architectures and their requirements. The main MBSE patterns implemented in the study were therefore those related to the architecture solutions and are presented in this section. It should be noted that in the following a subset of the data models implemented in the patterns will be described.

Data Model implementation for the MBSE pattern "Architecture - Explore"

The generation and analysis of the use cases represent the major part of the functional side. The system behaviour is described as its main activities during the mission including the nominal and contingency actions and the decision points, as shown on the diagram below.
From the system high-level capabilities, CONOPS and mission activities, the use cases are generated containing detailed capabilities. The use cases rely on the definition of the stakeholders and the actors from the conceptual MBSE patterns.
The safety analysis is started at that stage with the definition of the mission accidents and the major mission hazards whose logical combination might lead to accidents, as depicted on the following diagram.
Figure 12: Safety diagram on the mission accidents and hazards.

Data Model implementation for the MBSE pattern "Architecture - Conceive"

In the generation of architecture options, the detailed capabilities are analysed in an inducing process to identify what might be the options on the top level capabilities. The diagram below represents all the capabilities from the use cases and the abstracted capabilities up to the top level (top-level capabilities in purple).
The use case diagram below shows one option selected for the top-level capabilities. In the next pattern the functional decomposition will start from these top-level capabilities.
Beside the functional aspect, the options consist of the product tree decomposition started in CDP. Different options for the product tree are defined in CDP with the system parameters. In the next pattern the system parameters are used for the generation and maintenance of the system budgets. The product tree of the chaser element is fully represented in the CDP model and follows the ECSS-E-TM-10-25 guideline, with a decomposition in System / Segment / Element / Assembly / Equipment. A subsystem level is introduced in the model to gather the equipment in logical groups below the assemblies. The chaser element has two assemblies: the platform bus and the payload consisting of the relative navigation sensors, the robotic arm with gripper, the clamps and the payload computer.

Figure 15: Modelling of the Product Tree options in CDP.
Data Model implementation for the MBSE Pattern "Architecture - Represent"

The system options identified in the previous pattern are developed here in detailed representations that can be simulated in the next pattern.

The top-level capabilities previously identified represent the starting point for the functional decomposition. From the system functions, the lower level functions are decomposed in a deducing process. At each level, the functional ports between functions are defined as well as the black box description (Block Definition Diagram, link of the functional block to other functions at the same level) and white box description (Internal Block Diagram, link to the lower level functional blocks deduced from the function considered). The functional decomposition was performed with a high emphasis on safety, which is a very important system property for this mission due to potential collisions with the target.

Figure 16: Result of the functional decomposition (left) starting from the top-level capabilities (right).
Figure 17: Representation of the system modes in a state machine diagram.

Figure 18: Representation of the interactions between the functions at one given level and one system mode in a sequence diagram.
Figure 19: Functional black box representation at system level with the interaction ports to the actors.

From the identification of the functions, a fault tree analysis for the mission hazards is performed for identifying basic faults and safety requirements.

Figure 20: Fault tree analysis for the catastrophic collision case.
In the current e.Deorbit SysML functional model, the representation shown here at system level is also implemented at the level of the sub-functions. Further diagrams are dedicated to the internal architectures including the connectors between the ports and to the control and data flows.

The CDP model has allowed during the study to keep the chaser configurations and the system budgets under control and to compare the configurations between the iterations.

Figure 21: Product Tree decomposition of the chaser.

In parallel to the definition of the physical architecture in CDP and SysML, the geometries of the chaser were developed in CATIA.
In addition to the product tree representation in CDP, the physical decomposition is also available in SysML for analysing the allocation between functions and hardware, defining the physical ports and connectors and for having the description of the system attributes used in the system simulation. The interoperability aspects between SysML and CDP are described in section 4.5.
Figure 23: Product Tree decomposition of the chaser in SysML.
When the functional, physical and behaviour representations are available in the SysML model, the allocations are defined and displayed in tables in Cameo. Following allocations were analysed: functions to mission activities, functions to mission hazards, functions to chaser modes, functions to chaser hardware and functions to actors. The "functions to chaser hardware" allocation is displayed in the table below.

![Figure 24: Definition of the physical ports and connectors for the payload assembly of the chaser element.](image-url)
An optimization of the architecture sizing w.r.t. the cost for mass reduction was experimented in the Airbus internal MBSE study independently of the e.Deorbit study. It has been analysed where the optimum would be between reducing the mass of the structure using light-weighted and expensive materials and engines with high Isp and lower propellant consumption. Once the system responses are available from the simulation process, optimization analysis is possible. An optimization analysis consists of analysing system responses evolution w.r.t. a variation of system parameters. The optimization process of a set of parameters is composed of 4 main steps available in ModelCenter:
1 – The first step is to sample the design space. For each variation of parameter, the corresponding system response is stored. A Design Of Experiment (DOE) uses a sampling algorithm to properly distribute parameter values on the design space. For each sample, the scenario is simulated to obtain the system response which is stored. The objective of a DOE is to provide an overview of system response variations on the design space.

![Figure 26](image)

*Figure 26: System response of the cost parameter for variations of engine ISP and dry mass.*

2 – The second step is to identify the main parameter contributors in the design space. "Parallel coordinates" representation is used to identify which range of variables can lead to a minimal cost. Each line corresponds to a range of input variable of the study. On this graph, the cheapest solutions are in blue.

![Figure 27](image)

*Figure 27: Selection of the main parameter contributors.*
The "variable influence profiler" below is used to determine which are the driver parameters according to the optimization objectives and to eliminate the other parameters from the further optimization analysis.

3 – The third step is to narrow the search of the optimum in the design space. The "prediction profiler" is used to manually explore the design space around the driver parameter values of the previous step. Each graph below shows what would happen to the output value if a particular input value was changed while keeping other inputs the same.
4 – The fourth step is to find automatically with optimization algorithms the optimum in the reduced area of the design space identified in the previous step.

Figure 30: 3D glyph tool with balancing variables and optimisation tool.

At this stage of the pattern implementation, information is available on the behaviour of the system, its properties as well as its functional and physical decompositions. The system complexity can be related to the dependencies in the architectures and between the architectures, the mission scenarios considering the contingency cases and the system responses to changes in the mission context. The system complexity and dependencies are actually difficult to represent with standard means in a way manageable by the engineering team. It is currently experimented at Airbus whether the Virtual Reality technologies could enable customers to get short-term, up-to-date and realistic impressions of the envisaged product without being lost in the complexity of data. The development team can in such an environment involve customers in the very early verification-of-design proposals against key customer requirements (mission vs. performance, function, safety, etc.) before entering the next iteration. Furthermore, such technologies support remote collaboration of concurrent design facilities. Thereby, the virtual reality environment accesses live project data, which have been made available by the interoperability concept of the MBSE patterns.
Finally, one of the major activity in this pattern is the generation of the architecture requirements. In the current implementation, all the requirements are available in the SysML model to allow defining relations between the requirements and the architecture elements. The model dependencies cover therefore 3 main areas: 1/ the tracing between the requirements of the different levels (e.g. derived from), 2/ the relations between the architectures (e.g. allocation of functions to equipment) and 3/ the dependencies between the requirements and the architectures (e.g. satisfied by, verified by). The recursive establishment of relations between the model elements is accomplished until all the requirements are satisfied by model elements and all model elements satisfy requirements. Missing requirements and model elements are added during the process to achieve a consistent model and complete requirement set.
The aim of the system simulation is to obtain the response of the e.Deorbit chaser system in a mission scenario by producing numerical data for the system parameters. The system simulation can then be handled as a program where the system parameters are varied in Design of Experiments for different studies.
This simulation approach allows combining the data management in the SysML descriptive model and the work flow of data processing in ModelCenter. The SysML descriptive model used in the simulation consists of the behaviour diagrams (mission modes and chaser states) and the block definition diagram of the physical decomposition containing the system attributes.
The internal structure of the simulator is defined in SysML activity and parametric diagrams. The activity diagrams sequence the dynamic execution of the simulation and call parametric diagrams. The constraint blocks of the parametric diagrams are linked to analysis flows defined in ModelCenter. The activity and state machine diagrams are executed by the Cameo Simulation Toolkit. In the simulator, the connection between parametric diagrams and ModelCenter is used to:

- Link physical architecture attributes to CDP parameter values: e.g. the mass and margin of each equipment, and their power consumption and state during a reconfiguration of the chaser in the simulation.
- Link system attributes to a processing chain, e.g. composed of Matlab trajectory simulator, which can compute the evolution of the position and the attitude of the chaser, and STK, to get the sun orientation.
- Do calculations in Matlab like power budget of the chaser for each steps of the simulation.

Figure 35: Simulator configuration linking ModelCenter processing flows to SysML system parameters.
To prove that the system properties, expressed as requirements, are provided by the defined architecture, performance requirements are linked in the SysML model to system attributes of the physical blocks. At the end of the simulation execution, the simulator has checked the fulfilment of the requirements w.r.t. to numerical values generated for the simulated architecture. This result provides the information needed by the engineering team to optimize the architectural solution to meet the expected system properties and to draw conclusions on the architecture option. These results can finally be used to trade the different architecture options and to select the architecture providing the properties closest to the emergent properties defined for the system of interest.

5.3 Evaluation

The architectural MBSE patterns presented in this section are on the one hand implemented operationally in ESA studies like e.Deorbit (functional and physical decomposition including the requirements) and on the other hand still in a preliminary development status (mission simulation and system optimization). Overall, the selected "Federated and Executable Models" MBSE methodology shows an adequate representation of the Systems Engineering method (concept, architecture, design), while integrating over the process system thinking precepts like system synthesis, behaviour, viability and emergent properties. The successive and dependent activation of the patterns allows a tailoring of the process according to the study specificities w.r.t. available information on the system of interest.

Furthermore, this process strategy allows eliminating or mitigating the threads associated with a system definition. The architectural and design aspects in the requirements and vice versa are covered, consistent data for the different system analyses are used, the architectures are described in the same context reducing the risk to not catch the relevant relationships and the system complexity, and it is verified early in the process by simulation, whether the system fundamental properties are generated by the developed solutions.

The interoperability in the data loops between the data models is still not fully deployed and will need further development for operational MBSE patterns.

6. Conclusion

A new MBSE process called "Federated and Executable Models" was introduced in this paper. The main motivation of this approach is to maintain systems thinking aspects over the complete SE methodology.

MBSE patterns, data loops, data models and data sprints build the overall Systems Engineering process based on models, operability between the models and SE strategies. They produce together iteratively and successively the organization of the information (reducing the configuration entropy) leading to
an optimal system definition for the system environment. They synthesize a unified whole, i.e. an open system with properties, capabilities and behaviours corresponding to the expected emergent properties, capabilities and behaviours. They also provide a holistic, multi-disciplinary and collaborative approach to designing and maintaining complex systems.

This MBSE process is also considered as an industrial production means for system definition in studies and projects. The process might lead to a higher efficiency in the Systems Engineering work. The objective at Airbus Defence and Space is to iteratively implement and experiment this MBSE process.

W.r.t. the full MBSE process addressed in the paper, only a part of it related to architecture was implemented so far. The presented MBSE patterns and data models will need further definition and validation work in experiments before their full application in studies.

7. References

[1] Estable, Stéphane et al. (2016), *Generation of chaser requirements and budgets for the e.Deorbit mission applying a MBSE process*: SECESA.

