OPEN ACCESS

University of BRISTOL

Timperley, L. R., Berthoud, L., Snider, C. M., & Tryfonas, T. (2024). Mapping the MBSE Environment and Complementary Design Space Exploration Techniques. In *2024 IEEE Aerospace Conference* (IEEE Aerospace Conference Proceedings). Institute of Electrical and Electronics Engineers (IEEE). https://doi.org/10.1109/AERO58975.2024.10521188

Peer reviewed version

License (if available):
CC BY

Link to published version (if available):
10.1109/AERO58975.2024.10521188

Link to publication record on the Bristol Research Portal
PDF-document

This is the accepted author manuscript (AAM) of the article which has been made Open Access under the University of Bristol's Scholarly Works Policy. The final published version (Version of Record) can be found on the publisher's website. The copyright of any third-party content, such as images, remains with the copyright holder.

University of Bristol – Bristol Research Portal

General rights

This document is made available in accordance with publisher policies. Please cite only the published version using the reference above. Full terms of use are available:
http://www.bristol.ac.uk/red/research-policy/pure/user-guides/brp-terms/

# Mapping the MBSE Environment and Complementary Design Space Exploration Techniques


Louis Timperley
University of Bristol
Queen's Building,
University Walk,
Bristol, BS8 1TR
lt17550@bristol.ac.uk

Lucy Berthoud
University of Bristol
Queen's Building,
University Walk,
Bristol, BS8 1TR
lucy.Berthoud@bristol.ac.uk

Chris Snider
University of Bristol
Queen's Building,
University Walk,
Bristol, BS8 1TR
chris.snider@bristol.ac.uk

Theo Tryfonas
University of Bristol
Queen's Building,
University Walk,
Bristol, BS8 1TR
Theo.tryfonas@bristol.ac.uk



***Abstract*— Today's MBSE tools and environments are highly varied and therefore present a challenge for organizations looking to implement MBSE. Furthermore, while MBSE environments are highly capable of supporting the description of design baselines, the current capabilities within these environments could be further refined for exploring alternative designs. As a result it is important to gain an understanding of the limitations of current MBSE tooling in performing the valuable activity of design space exploration, and identify a set of candidate techniques to combat these. This paper reviews the various options available to MBSE practitioners by comparing some of the most common MBSE languages, tools and methods. The possible issues that can be encountered when exploring different designs have been identified and assigned a severity rating. A set of design space exploration techniques are presented, and where possible these have been sourced from existing literature. A knowledge graph has been constructed to collect all this data into a structured format, containing all the MBSE languages, tools, methods, design space exploration-related issues and techniques, as well as the relationships between each of these. This knowledge graph, implemented as a Neo4j graph database, allowed deeper insights to be drawn from the collected information. By defining a selected MBSE environment, including language, tool and method, the knowledge graph could be used to identify the least troublesome sequence (with minimum number of related issues) to arrive at a desired design artifact, for example a set of optimized system parameters. Beside this, the knowledge graph could be used to display the relationships and clusters of MBSE languages, tools and methods, to assist organizations with selecting suitable MBSE environment elements. Future work will bring greater depth to the analysis available with the knowledge graph, for instance, differentiation between different types of design space exploration issues and techniques.**


## TABLE OF CONTENTS



## 1. INTRODUCTION

Model Based Systems Engineering (MBSE) is a maturing method, the tools used to setup a capable and successful MBSE environment are still evolving. As a result, it can be difficult to select the language, tool, method and support tools that are suitable for an organization's or project's goals and context. Further to this, methods for exploiting MBSE's unique benefits when performing design space exploration are not yet fully understood. Some work in this area has already been performed [1],[2],[3],[4],[5] however there are still many untested candidate techniques for design space exploration with MBSE and further investigation is necessary to fully justify wide adoption in industry.

The goal of this study was to firstly review and map the primary elements of today's MBSE environments in a way that would be useful to new practitioners setting up their own environments. Secondly this study reviews the short-comings in these MBSE environments when exploring alternative designs. With reference to a theoretical framework for design variability discussed in the next section, these issues are mapped to candidate techniques for possible solutions, some of which have already been applied in literature and others as yet untested. The rest of this paper is split as follows; section 2 presents the variability framework used as part of this study, section 3 describes the findings of the review, section 4 presents the capture of the review findings in the form of a knowledge graph, section 5 discussion and future work and finally section 6 gives conclusions.

## 2. VARIABILITY FRAMEWORK

To enable a more precise categorisation of how designs can vary between alternatives, a variability framework was established. Shown in table 1, this framework has been used to categorise the capabilities of different design space techniques and more precisely understand where issues lie in current MBSE practice.

It is expected that the most frequently sought level of variability in design problems is 'parameter'. Element type level is likely the second most important. Topological is largely constrained by the specific components included in a design, and multiplicity can in many cases be approximated by parameter level variability in a model. This framework does not cover aspects such as variation in exploration


979-8-3503-0462-6/24/$31.00 ©2024 Crown

objectives or requirements themselves. Further work is planned to refine the framework, discussed later.

**Table 1 Design variability framework**

| Variability Level | Description | Example |
|---|---|---|
| **Parameter** | Numbers, e.g., mass | Mass, Orbit Altitude |
| **Multiplicity** | Number of elements in a group | Number of Redundant Reaction Wheels |
| **Topological** | Configuration of interfaces between parts | Centralised topology vs. federated |
| **Element type** | Type of parts used in the system | Solar array or radio-isotope generator |

## 3. REVIEW

A literature review has been carried out, covering a wide range of options available for MBSE environments, as well as issues with current practice and candidate design space exploration techniques.

This review considered MBSE environments to be comprised of three elements [6]:

Language – The way that information is captured, used to specify all the various aspects of the system [7].

Tool – The software or programme used to create the model.

Method – The series of steps followed to populate the model using the language in the tool.

Any formal model will follow an ontology, or a description of a set of concepts and their inter-relations [8], this will specify how the abstract language is used to describe the real system. Though it focuses on the language, it tends to be specific to particular methods, therefore ontologies have been considered part of the method for this review.

*Languages*

Table 2 gives a summary of the range of modelling languages covered in the review. Green cells are positive, yellow are non-ideal, red is negative.

**Table 2 Common MBSE Languages**

| | Developer/ Organisation | Tool | Methodology | Variability Modelling | Simulation Capability | Customisation |
|---|---|---|---|---|---|---|
| **UML** | OMG | Flexible | Flexible | Generalization and multiplicity across relationships | Good (though can be slow) | Can be extended via profiles |
| **SysML V1** | OMG | Flexible | Flexible | Similar to UML plus parametric diagrams may also be used to support alternative analysis | Good (though can be slow) | Can be extended via profiles |
| **SysML V2** | OMG (Future release) | Flexible | Flexible | Product line engineering and variability points | Good | Can be extended via extensible model libraries |
| **Capella** | Eclipse | Capella | ARCADIA | None inbuilt | Limited (on third party tools) | Open Source |
| **Object-Process methodology** | OPM/ISO | Flexible | OPM | Limited/none | Animation | Limited |

| | Developer/ Organisation | Tool | Methodology | Variability Modelling | Simulation Capability | Customisation |
|---|---|---|---|---|---|---|
| **System Definition Language (SDL)** | Vitech | GENSYS | Vitech MBSE Methodology | Limited/none | Behaviour simulation in GENSYS | Is extensible |
| **Valispace language** | Valispace | Valispace | Valispace | Limited/none | Parametric | Web based, has API |
| **RePoSyD Language** | RePoSyD | RePoSyD | RePoSyD | Limited/none | Limited/none | Open source |
| **AADL** | SAE | Flexible | Flexible | Limited/none | Can be simulated, e.g. in TASTE for model base verification | Can be extended via language annexes |
| **SLIM** | COMPASS | COMPASS | COMPASS | Limited/none | Performability analysis and others, focussed on safety and correctness | Limited |

*UML*—UML is a diagram-based language, designed with software development in mind. Its objective is to allow engineers to analyse, design, and implement software systems as well as modelling of business and similar processes[9]. There are several diagrams available, such as class diagrams, composite structure diagrams, package diagrams, activity diagrams, state machine diagrams and sequence diagrams among others. It has only limited provision for describing design variability, whereby generalization relationships can be defined between elements. This enables specific variants to inherit properties from a "general" element type, that may then be further refined. Multiplicity may also be specified for association, aggregation and composition relationships. While suitable for modelling product lines with well defined and finite differences (for example a range of car models with varying features), these methods become cumbersome to use when exploring design alternatives not explicitly modelled by the user.

*SysML V1*—Has been widely used across MBSE tools and applications in industry, and has become somewhat synonymous with the concept of MBSE [10]. Developed from UML, it has good coverage of systems engineering concepts, but with similarly limited variant modelling capability (using the same generalization and relationship multiplicity methods). However the SysML spec does include an extension for trade studies shown in table 3 [11, p. 6]. Other extensions can be built to provide new types for the language via profiles, that are sets of custom "stereotypes", meaning custom types specific to the users needs and domain.

**Table 3 SysML V1 Extension for Trade-Studies**

| Stereotype | Base class | Description |
|---|---|---|
| **Objective Function** | Constraint Block | An objective function is used to determine the overall value of an alternative in terms of weighted criteria and/or MOE's (measure of effectiveness) |
| **MOE** | Property | A measure of effectiveness (MOE) represents a parameter whose value is critical for achieving the desired mission cost effectiveness |

*SysML V2*—An upcoming version of SysML V1, with many major changes hoped to address the shortcomings identified with the original [12]. Firstly, Version 2 has a more layered approach compared to V1. These layers are; System syntax, Kernel syntax (this has its own specification), Core syntax and Root syntax. Besides this, it will add a textual version of the language, that is intended to be equally expressive. For better integration with ontology tools and languages, it will have integration with the OWL/RDL ontology languages. This allows it to have good mapping to data ontologies such as ESA's Overall Semantic Modelling for System Engineering (OSMoSE)[13]. On top of product line engineering capabilities similar to UML and version 1, a major benefit of version 2 will be more support for variability modelling, better enabling trade-offs and alternative configurations as well as decision points [14].

*Capella*—The Capella language (used by the Capella tool) is similar to SysML but with some notable differences. Capella functions have no control flow description (meaning conditions for particular sequences of functions) as Arcadia (the Capella method) is purely focused on architectural design and interface definition [15]. There is no Capella requirement diagram, but requirements can be shown in any diagram in Capella. No parametric diagram exists in Capella, but can generally handle the concepts it requires in other ways. Finally, the Capella language has no variant modelling.

*Object Process methodology (language elements)* —First developed in 1995, the Object Process Methodology (OPM) uses a conceptual modelling language for capturing knowledge and designing systems [16]. In a similar fashion to SysML V2, OPM has two versions of the language. Firstly object-process diagrams, or the diagram-based version of the language and the object-process language, or the text based version of the language. OPM is focussed on modelling both objects and processes including the links between them [17]. Both objects and processes are modelled as OPM "things" and links are either modelled as structural or procedural [18].

Unlike UML/SysML that follows a multi-model structure (with behaviour spread through 13 types of diagram) OPM is a single model structure. This helps with understanding of the model and avoids the model-multiplicity problem, where concurrent understanding of multiple sub models is required [19].

Some examples of using the OPM language to model variants exist in literature, but with no application to systems engineering [20].

*System definition Language*—The System Definition Language (SDL) is used as part of the Vitech MBSE Methodology, STRATA. SDL is based on the following concepts [21]:

- SDL *Entities* are similar to nouns in English and refer to objects
- *Relationships* are similar to verbs and refer to a link between two entities
- *Attribute* are further descriptions of entities in the same way adjectives modify nouns
- Adverbs also have a corresponding element in SDL with *Attributed-Relationships* (i.e., attributes on relationships)

*Valispace language*—The language used by the Valispace tool is highly data driven [22] and so largely focusses on describing parameters and their relationships.

*RePoSyD Language*—This is another language tied to its method and tool. Its definition is based on a Meta model, essentially a description of allowed elements and relationships in models and similar to an ontology. Meta models may also be found in UML/SysML models.

*AADL*—The Architecture Analysis and Design Language (AADL) was originally designed for architecting software and hardware systems for avionics [23]. It facilitates direct code generation and can be used as a form of documentation itself.

*SLIM*—Based on AADL, SLIM is another tool-tied language for the COMPASS toolset [24].

*MBSE Tools*

Table 4 gives a summary of the modelling tools covered in the review. Green cells are positive, yellow are non ideal, red is negative.

**Table 4 Common MBSE Tools**

| | **Developer/Year of latest release** | **Language** | **Methodology** | **Simulation** | **Customisation** |
|---|---|---|---|---|---|
| **Cameo** | Dassault | SysML V1 | Flexible | Good (though can be slow) | Closed source, Open API, custom plugins |
| **Capella** | Thales | Capella | ARCADIA | Limited (reliant on third party tools) | Open Source |
| **Papyrus** | Eclipse | SysML V1 | Flexible | None | Open Source |
| **Valispace** | Valispace | Valispace | Valispace | Parametric | Web based, Closed source |
| **Rhapsody** | IBM | SysML V1/UML | Flexible | Good | Closed source, Open API, custom plugins |

| | Developer/Year of latest release | Language | Methodology | Simulation | Customisation |
|---|---|---|---|---|---|
| **Enterprise Architect** | Sparx | SysML V1/UML | Flexible | Limited to simulation of behavioural diagrams | Closed source |
| **OPMCloud** | OPMCloud | OPM | OPM | Limited animation style simulation, some optimisation tools inbuilt | Web based, closed source |
| **GENSYS** | GENSYS | (Ecosystem of tools, SDL for MBSE) | STRATA | System behavioural simulation | Closed source |
| **RePoSyD** | RePoSyD | RePoSyD | RePoSyD | Limited | Open source |
| **TASTE** | ESA | SDL, AADL | Flexible | System behavioural simulation and testing for validation | Open Source |
| **COMPASS 3.0** | COMPASS/ESA | SLIM | COMPASS | Performability analysis and others, focussed on safety and correctness | CLI interface and some open-source aspects |

*Cameo Systems Modeler (No Magic)* —Cameo Systems Modeler (CSM) is a widely used MBSE tool aimed for use with UML/SysML V1 (it does not enforce a particular method). Now called Magic Systems of Systems Architect, it is a closed source programme but can be customized and extended via plugins. CSM includes powerful simulation capabilities via the Cameo Simulation Toolkit that allows direct execution of SysML models. Further to this it includes integrations with MATLAB, python and provides an open API. All of this capability has ensured CSM is widely used in industry, for example by JPL [25] and Airbus [26]. The integrations offered by CSM could be leveraged to support analysis of alternatives, for example as done in [27].

*Capella (Arcadia)* —Capella [28] is an open-source tool for use with the Capella language and Architecture Analysis & Design Integrated Approach (ARCADIA) method. Its simulation capabilities are smaller compared to CSM, but it is able to simulate state machines and functional flows. MATLAB can be integrated with the tool, but generally simulation is heavily reliant on third party tools. Capella is the standard tool used by ESA and Thales Alenia Space, who were both involved with its development.

*Papyrus*—An open source SysML based tool. Much like CSM it has no specific related method, however unlike CMS it has extremely limited simulation capability, making bottom-up style analysis difficult.

*Valispace*—An example of web-based MBSE tool is Valispace. While it may be more difficult to customise (being web based), it does offer an API for further integrations. As previously discussed, it is highly oriented towards capturing engineering data, rather than being concerned by specific semantics, leading to it sometimes being to referred to as a tool for Data-Driven Systems Engineering (DDSE).

*IBM Rhapsody*—IBM Rhapsody is a further UML/SysML tool that has been in use since 1996 and has reached a high level of maturity. It has simulation capabilities built in "for design-level debugging" [29].

*Sparx Systems Enterprise Architect*—Another UML/SysML modelling tool is Sparx Systems Enterprise Architect, capable of simulation of behavioural diagrams such as State machines, interaction (sequence) diagrams, activity diagrams etc. It can generate executable code for state machines for application and simulation and supports Hardware Description Languages (HDL, such as Ada, VHDL and Verilog).

*OPMCloud*—OPMCloud is the web-based tool that supports the object process method language. Unlike most other tools, it incorporates means to optimise the system model, based on pareto front, Design-Structure Matrix (DSM)-based methods, or even graph database queries [30].

*GENSYS*—GENSYS is a manufacturing-focussed MBSE tool, covering capabilities such as; product data management, production planning and control and shop floor control

*RePoSyD*—The RePoSyD tool is tied to the RePoSyD

method/ language

*TASTE*—TASTE is a freely available toolset for development of embedded real-time systems [31], developed by ESA and various partners. It primarily focusses on modelling of embedded systems and can generate code from external modelling languages, such as SDL or Simulink, as well as supporting typical embedded languages such as C++, VHL, ADA etc. It has other typical MBSE tool capabilities built in such as validation of models [32].

*COMPASS 3.0*—COMPASS (Correctness, Modelling and Performance of Aerospace Systems) 3.0 a toolset providing an integrated model based approach for system software co engineering in the aerospace domain [33]. It uses the SLIM language and includes various analysis capabilities [34] such as; functional correctness checking, Safety and dependability analysis, performability analysis, fault detection identification and recovery effectiveness analysis and finally requirement validation.

There is an ongoing effort to combine TASTE and COMPASS, to form COMPASTA [35].

*MBSE methods*

Table 5 summaries the MBSE Methods covered in the review. Green cells are positive, yellow are non ideal, red is negative.

**Table 5 Common MBSE Methods**

| | **Developer/Initiator** | **Language** | **Tool** | **Design Space Exploration** |
|---|---|---|---|---|
| **ARCADIA** | Eclipse/Thales | Capella | Capella | Limited |
| **OOSEM** | Lockheed Martin and SSCI | SysML V1 | Flexible | Limited |
| **Harmony** | IBM, Hoffman, Douglass | SysML V1 | Rhapsody | Supported within the design synthesis stage |
| **RePoSyD** | Hoppe | RML | RePoSyD | Limited |
| **STRATA** | Vitech | SDL | GENSYS | Limited |
| **OPM** | Dori | OPM | OPMcloud | Limited |
| **Valispace** | Valispace | Valispace | Valispace | Limited |
| **MOFLT** | Airbus | SysML V1 | Flexible | Limited, but has tailoring in mind |
| **COMPASS** | COMPASS | COMPASS | COMPASS | Limited |
| **SEAM** | University of Bristol | SysML V1 | Flexible | Limited |

*ARCADIA*—The Architecture Analysis & Design Integrated Approach (ARCADIA) is the method used by the Capella language/tool. It follows 5 stages all intended to produce a system architecture and flow down requirements to the component level [36].

- Define the Problem – Customer Operational Need Analysis,
- Formalisation of System/SW Requirements – System/SW Need Analysis,
- Development of System/SW Architecture – Logical Architecture,
- Development of System/SW Architecture – Physical Architecture,
- Formalize Components Requirements – Contracts for Development and IVVQ (Installation, Verification, and Validation Qualification)

It has very little design space exploration provision besides one example in literature using a third party tool [37].

*OOSEM*—The Object-Oriented Systems Engineering Method is a top down, scenario-driven process for SysML. It supports analysis, specification, design and verification of systems and is generally applied recursively at each level of the system [38].

*Harmony*—Intended to be used with IBM's Rapsody, the method covers the system development from requirement analysis to system acceptance [39]. The main steps in the process are:

- Requirement Analysis – Collection and completion of requirements and categorisation as functional or non-functional. Requirements are allocated to use cases. The models at this stage are not executable
- System Functional Analysis - The use cases are analysed and executable use case models are created and verified

against requirements

- Design Synthesis – This is the stage where most trade-offs and design exploration may be performed. Behaviour from the use case models are merged into a single system model. Functionality is allocated to subsystems, allowing the formation of an architectural concept model, including each subsystem. The resulting executable model is then verified against requirements.

- SW Analysis and Design - The model is now handed off to SW and HW development. The specifics of this stage is not covered explicitly by the harmony process.

- SW implementation and unit Test - The implementation of SW and subsystems is not covered explicitly by the harmony process.

- Module Integration and Test, followed by (sub-) System integration and test – the various SW and HW modules are integrated and verified against requirements, followed by the subsystems and then full system.

- System Acceptance – Finally, if the full system is successfully verified against requirements, it may be accepted, finishing the process.

*RePoSyD*—The Requirements Engineering, Project Management, and System Design (RePoSyD) method, developed in 2017 [40], is based on the IEEE-1220 systems engineering process definition [41]. It has its own cloud based tool and language (RML) and includes some provision for basic trade-offs[42]

*STRATA*—STRATA, provided by Vitech, uses the SDL language and covers four main domains [17]:

- Requirements – starting from top level system requires, these are further refined through the process [43].

- Functional/Behaviour – Description of what the system must do to achieve its mission. Top level functions are decomposed as the process advances [44].

- Architecture/Synthesis – Physical Architecture of the system, where physical components are assigned functions, ensuring the physical design is separated from the logical design and avoid the premature allocation of behaviour to physical components [43].

- Verification And Validation – Each physical component and behaviour is verified against relevant requirements already included in the model.

*OPM*—The object process method, used by OPMCloud, uses two main mechanics. Firstly in-zooming is a refinement, typically of a process, that produces a timeline of events occurring in that process. Unfolding is a process that analyses the system structure, identifying objects in the system. Dynamic and Structural aspects can be represented in the model. Dynamic aspects modelled by relationships between processes and objects in an Object-Process diagram (OPD), and structural represented by "relations between two things of the same class (Process-to-Process) or (Object-to-Object) within an OPD" [17]. There is no dedicated diagram for requirements, instead they are modelled as a state of an object and a "satisfies/is implemented by/derives" link is added to relate the requirement to its associated elements. [45]

*Valispace*—The Valispace method is intended to address many aspects of systems engineering, such as requirement engineering, detailed design, simulation, test, review, technical change management and documentation. The main focus of the whole language, tool and method ecosystem is to facilitate concurrent design and the early phases [46]. Concurrent design "is a systematic approach to integrated product development that emphasises the response to customer expectations. It embodies team values of co-operation, trust and sharing in such a manner that decision making is by consensus, involving all perspectives in parallel, from the beginning of the product life-cycle" [47]. It has seen wide application across the spacecraft industry and, amongst others, ESA has become a significant supporter of this way of running early phase design studies, setting up a dedicated Concurrent Design Facility (CDF) [47].

*MOFLT*—The MOFLT method (Mission Operations Functional Logical Technical), developed at Airbus Defence and Space, is language and tool agnostic. Compliant with ECSS and with tailoring points for extensibility, the method has 5 layers [48]; mission, operational architecture, functional architecture, logical architecture, technical/physical architecture. The mission layer covered mission definition and determining of potential mission realisations. The operational analysis layer covers operational concept definition and operational scenarios. The functional architecture includes functions, sequence of functional execution and structural arrangement of functions. The logical architecture defines the system logical components and allocates functions to each. Finally, the physical architecture is the physical (or real) implementation of the system. There are strong similarities between MOFLT and ARCADIA, with the primary difference being that the functional architecture is given its own layer, instead of being spread across the system analysis, logical and physical architecture layers [49].

*SEAM*—The Spacecraft Early Analysis Model framework (SEAM), was developed at the University of Bristol, in partnership with Airbus Defence and Space [50]. Much like ARCADIA and MOFLT, SEAM is made up of five packages, or sub models, each capturing a different aspect of the system, Mission, Operation Logical architecture, Functional architecture and Simulation. Firstly, the Mission package contains information about the system purpose, requirements, external entities and external interfaces. The operations package describes what actions the system shall perform

while it is operating. The two architecture packages are closely intertwined and reference each other at many levels of the system. The logical architecture defines the constituent parts of the system (logical elements), such as subsystems, assemblies and components, while the functional architecture describes the functions available to the system and what logical elements perform them. The simulation package contains information about the simulations, for example the mission phases to be stimulated and start and stop mission times. SEAM is targeted for use with Cameo Systems Modeler and SysML, therefore the simulations package can make use of the Cameo Simulation Toolkit for direct simulation of the system model.

*System simulation*

Table 6 presents an overview of the dedicated simulation tools considered for this review. Green cells are positive, yellow are non-ideal, red is negative.

**Table 6 Common Simulation Tools**

| | **Developer/Year of latest release** | **Language** | **Interfacing/model generation** | **Customisation** |
|---|---|---|---|---|
| **Modelica** | Modelica | Modelica | Common (e.g. cameo) | Open Source |
| **Simulink** | MathWorks | Simulink/MATLAB | Common (e.g. cameo) | Closed Source |
| **Collimator** | Collimator | Collimator (strong coupling with Python) | Less Common | Closed Source (limited python API) |
| **Cameo Simulation Toolkit** | NoMagic/Dassault Systems | Sys/ML V1;UML | Directly from SysML or UML | Open API |
| **Rapsody** | IBM | Sys/ML V1;UML | Directly from SysML or UML | Open API |

*Modelica*—Modelica is an object-orientated modelling language, with integration for many tools including cameo systems modeller. It has the potential to allow a dual model approach [51] with a high level system model, for example in SysML, for requirements and Modelica model for simulation. Open Modelica is an example of free Modelica environment [52].

*Simulink*—Simulink provides 7 primary capabilities [53], namely, intuitive description of system architectures, domain specific tools and prebuilt blocks for modelling multi-domain systems, integration of multiple team's tools into one simulation, simulation and analysis of system behaviour, parallel simulation for batch analysis and finally deployment of simulation as standalone applications. Further to this, Simulink has integration support for many tools, but as the developer, MathWorks, does not provide an API (and Simulink is closed source), these integrations can be somewhat unreliable.

*Collimator*—Collimator [54] is a closed source, web based, data driven systems modeller with similar diagramming to Simulink. Besides simulation and analysis, it can be used to generate C and Python code. It has built in version control and a python API for loading models and running simulations. Furthermore, Collimator can import data, in CSV, JSON, Text and XML formats for creating series, for example, describing input parameter variation through time.

*Cameo Simulation Toolkit*—Cameo simulation toolkit [55] is built into Cameo Systems Modeller (now Magic Systems of Systems Architect) and is capable of full behavioural simulation and parametric modelling, allowing execution of all relevant SysML/UML diagram types. The toolkit has a capable API for accessing simulation elements.

*Rhapsody*—Similarly to cameo simulation toolkit, Rhapsody's built-in simulation capabilities allow animated execution of all relevant SysML/UML diagram types, alongside an API for interfacing to external tools. Unlike cameo simulation toolkit, simulations in Rhapsody do not have a simulation clock, however it is still possible to model asynchronous behaviour and some language constructs such as state machine joins and forks not available in cameo simulation toolkit are available in Rhapsody.

*Domain Issues*

With all of the above information collected, it was possible to identify a set of issues, or shortcomings of the current state of MBSE practice related to design space exploration. These issues are listed in table 8.The list is by no means exhaustive but includes the most immediate issues. Note that the rules in table 7 were used to identify the issue severities.

**Table 7 Rules used for issue severity**

| Condition | Severity |
|---|---|
| Issue prohibits any form of design exploration | 5 |
| Issue prohibits most forms of design exploration | 4 |
| Issue prohibits a few forms of design exploration or presents a challenge to carry out the exploration process | 3 |
| Issue limits the quality of the design exploration process in some way | 2 |
| Issue has only a minor impact on some aspect of the design exploration process | 1 |

**Table 8 Identified design space exploration issues**

| Name | Summary | Affected Aspects | Severity |
|---|---|---|---|
| **Parameter variation language** | The language lacks convenient ways to describe how a design's parameters change between different design options | UML, SysML V1, Capella Language, OPM Language, SDL, Valispace Language, RML, AADL, SLIM | 5 |
| **Topology variation language** | The language lacks convenient ways to describe how a design's topology changes between different design options | UML, SysML V1, Capella Language, OPM Language, SDL, Valispace Language, RML, AADL, SLIM | 4 |
| **Multiplicity variation language** | The language lacks convenient ways to describe how a design elements' multiplicity changes between different design options | UML, SysML V1, Capella Language, OPM Language, SDL, Valispace Language, RML, AADL, SLIM | 3 |
| **Element variation language** | The language lacks convenient ways to describe how a design elements' type changes between different design options | UML, SysML V1, Capella Language, OPM Language, SDL, Valispace Language, RML, AADL, SLIM | 5 |
| **Tool based variation** | The tool lacks convenient ways to switch between different design options | Cameo, Capella, Papyrus, Valispace, Rapsody, Enterprise Architect, OPMCloud, GENSYS, RePoSyD Tool, TASTE, COMPASS 3.0 | 4 |
| **Exploration stage in methodology** | There is no well-defined stage at which exploration of design alternatives (in physical architecture) should take place | ARCADIA, OOSEM, RePoSyD, STRATA, OPM, Valispace Method, MOFLT, COMPASS, SEAM, TASTE Method | 3 |
| **Identification of what aspects to vary and at what level** | There is no well-defined method for selecting aspects of a design to vary (apart from starting from nothing) | ARCADIA, OOSEM, RePoSyD, STRATA, OPM, Valispace Method, MOFLT, COMPASS, SEAM, TASTE Method | 3 |
| **Simulation time** | Simulations can become expensive to run for many design alternatives, especially if the analysis model used cannot be simplified | Cameo, Capella, Papyrus, Valispace, Rapsody, Enterprise Architect, OPMCloud, GENSYS, RePoSyD Tool, TASTE, COMPASS 3.0 | 4 |
| **Selection of designs** | The designer cannot select the "best" design among many without rigorous metrics | ARCADIA, OOSEM, RePoSyD, STRATA, OPM, Valispace Method, MOFLT, COMPASS, SEAM, TASTE Method, UML, SysML V1, Capella Language, OPM Language, SDL, Valispace Language, RML, AADL, SLIM | 2 |
| **Synthesis of new alternatives** | Efficiently numerating design aspects between alternative designs is difficult | Cameo, Capella, Papyrus, Valispace, Rapsody, Enterprise Architect, OPMCloud, GENSYS, RePoSyD Tool, TASTE, COMPASS 3.0 | 2 |
| **Synthesis of Novel alternatives** | It is difficult to encourage an "algorithmic" process to incorporate the inventiveness of a human designer in synthesising new design options | Cameo, Capella, Papyrus, Valispace, Rapsody, Enterprise Architect, OPMCloud, GENSYS, RePoSyD Tool, TASTE, COMPASS 3.0 | 2 |
| **Design Space Coverage** | Guaranteeing that the design space has been fully explored is a highly open ended and complex task | ARCADIA, OOSEM, RePoSyD, STRATA, OPM, Valispace Method, MOFLT, COMPASS, SEAM, TASTE Method | 4 |
| **Constraints on feasible designs** | It is not possible to describe what designs are feasible, or compliant with requirements | UML, SysML V1, Capella Language, OPM Language, SDL, Valispace Language, RML, AADL, SLIM | 3 |

*Design Space Exploration techniques*

With the identification of current MBSE practice covering languages, tool and methods, alongside design space exploration related issues, it was possible to start exploring candidate techniques to integrate into current practice that would address its short comings. The exploration techniques included both methods of generating new design alternatives, and the evaluation of these alternatives. Various domains were considered, not necessarily related to MBSE, with the expectation that many techniques used in other domains could be adapted.

The necessary inputs and outputs as well as applicable types of design space (type of variability) vary for each technique. For example some may require simulation or some form of system analysis, and some are only capable of varying continuous parameters while others can be adapted to various types of variability. The selection of techniques should be driven by firstly what kind of design exploration is being done, in reference to the variability framework presented in table 1, what inputs are available and what are the desired outputs. Each technique, required input, output and variability type is recorded in table 9.

**Table 9 Candidate design space exploration techniques**

| **Name** | **Summary** | **Variability type/** Issues Solved | **Inputs** | **Outputs** | **Advantages** | **Disadvantages** |
|---|---|---|---|---|---|---|
| **Surrogate modelling** [56] | Mathematical approximation model of full parent model, useful when it is expensive to simulate the parent model | **Parameter**/ Simulation Time | Training points from parent model | Mathematical approximations of parent mode | Significantly reduce simulation time of complex models | Needs training, not robust to discontinuities |
| **Combinatorial optimisation** [57] | Discrete type of optimisation- find an optimum set from a finite set of objects (e.g. travelling sales man problem) | **Discrete**/ Selection of designs | Finite set of possible system element variants, (element type variation or discretised parameters etc.) | Optimum set of system element types/ discretized parameters | Relatively simple to implement | Can be NP complete (Nondeterministic Polynomial complete) leading to long simulation time |
| **Gradient based optimisation** [58] | Follows direction of steepest descent to find minimum | **Parameter**/ Selection of designs | System model, objectives | Local/global Minimum | Faster convergence on minima | If not agent based, can get trapped in local minimum |
| **Genetic optimisation** [59]**,** [60]**,** [61] | Uses repeated cross over and mutation between a population of individual solutions to find minimum | **Parameter**/ Selection of designs | System model, objectives | (typically) Global minimum | Less susceptible to getting stuck in local minima, Tuneable mutation and crossover | Computationally expensive |
| **Particle swarm optimisation** [62] | A similarly bio inspired method, much like genetic optimisation, that considers a set of particles that behaves in a similar way to a flock of birds in order to find the minimum of a design space | **Parameter**/ Selection of designs | System model, objectives | (typically) Global minimum | Less susceptible to getting stuck in local minima | Computationally expensive |
| **Simulated annealing optimisation** [63] | Models the process of heating then cooling a material to minimise the model's total energy as an analogue for finding the minimum of a design space | **Parameter**/ Selection of designs | System model, objectives | (typically) Global minimum | Less susceptible to getting stuck in local minima | Computationally expensive |

| Name | Summary | Variability type/ Issues Solved | Inputs | Outputs | Advantages | Disadvantages |
|---|---|---|---|---|---|---|
| **Interactive Optimisation** [64] | A human "expert" is able to make inputs to the optimisation process, either by changing search directions or parameters etc. | **Dependant on user interaction**/ Selection of designs | System model, objectives, Expert "intuition" | (typically) Global minimum, and selection of "unmeasured" aspects performed | Can better be fused with "expert" intuition | Requires manual input |
| **Rule based design generation** [65] | Discrete rules for if such and such an aspect of the organisation or requirements are present, then include such and such an element in the design | **All**/ Synthesis of new alternatives | Organisational info, mission requirements | System architecture | Rapidly arrive at solution | Limited to rule domain |
| **Design of experiments** [66] | Statistical process for identifying configuration of tests with the aim of identifying what key factors influence outputs, what setting should they be in acceptable outputs, main interactions in the process, how to minimise variation in the outputs | **Parameter**/ Design Space Coverage, Identification of what aspects to vary and at what level | System model, objectives, simulation results | Test points | Maximise usefulness of training/sample points | Need to run more simulations to identify sensitivities |
| **Evolutionary generative design** [67] | Similarly to genetic optimisation, using a stochastic evolutionary algorithm to generate designs (ideally from scratch), but relying on the designer to perform the selection of the best option | **All**/ Synthesis of new alternatives, Synthesis of novel alternatives | Requirements, Expert "intuition" | Set of varied designs | Possibility of diverse set of designs suggested to designer/engineer | May become challenging for engineer to make decision on best option |
| **Deep Reinforcement learning for design concept generation assistant** [68] | Select a component selection from a database (action) then evaluate this (e.g. by using a parametric model). Calculate a reword for this, then use this to train on a neural network, the action-reward mapping | **All**/ Synthesis of new alternatives | Parametric system model, database of components | Product breakdown (component list) | Identify optimum component selection, generally faster than combinatorial approaches | Must first train Neural net to learn reward-action mapping, limited to component database scope |
| **SysML V1/UML Generalization Relationships** [69] | By defining a "general" element and linking each design alternative with this general element using a generalization relationship, properties can be inherited and refined for each design alternative | **Discrete**/ Parameter, multiplicity and element type variation language | Finite set of possible system element variants, (element type variation or discretised parameters etc.) | System architecture, including a set of design alternatives (per element, not at system level) | Relies only existing language constructs in UML and SysML V1 | Can only include variants explicitly modelled by the user, cumbersome to explore these variants at the systems level |

| Name | Summary | Variability type/ Issues Solved | Inputs | Outputs | Advantages | Disadvantages |
|---|---|---|---|---|---|---|
| **Parametric Variability SysML V1 Stereotypes** [27] | New SysML V1 stereotypes for describing the parameter level variation of design variables | **Parameter**/ Parameter variation language | Logical Architecture, Functional Architecture, Physical Architecture | Parameter Variability | Enables parameter level variability description in SysML models | limited to parameter level |
| **Objective SysML Stereotypes** [27] | New SysML stereotypes for describing the objectives related to the model | **Parameter**/ Description of a good design | Logical Architecture, Functional Architecture, Physical Architecture | Objectives | Enables a simple selection method of designs, by which a better value of objectives means a better design | Can become ambiguous with many conflicting objectives |
| **Design constraint and dependant variable SysML Stereotypes** [27] | New SysML stereotypes for describing dependant variables and constraints related to them to identify compliant/feasible and infeasible/noncompliant designs | **Parameter**/ Description of constraints on feasible designs | Logical Architecture, Functional Architecture, Physical Architecture | Design constraints | Enables description of where the feasible design region is | difficult to describe more complex constraints than a parameter being within a certain range |
| **Swarm intelligence (SI) and multi-agent societies** [70] | Commonly used to model social or collective behaviours, they are capable of autonomous action using their individual priorities. Essentially a set of unsophisticated individuals interact local with their environment and cause a system or global level functional pattern - emergent behaviour | **All**/ Synthesis of new alternatives, Synthesis of novel alternatives | Requirements, Parametric System Model | System architecture | "Regular design evaluation and improvement, Multiple solutions, Optimisation, Disruptive innovation" [70] | heavy computation requirement "Progress slows down after achieving near optima solutions" [70] |

## 4. MBSE KNOWLEDGE GRAPH

To be able to gain further insight into the data gathered during the review, it was decided to model the information in a knowledge base. This would structure the information is a more machine readable way than simple tables and text. A knowledge base describes a set of real world things and the relationships between them, thus is essentially synonymous with the concept of ontology [71]. It should be noted that knowledge bases/ontologies can contain both conceptual definitions of types or classes, and instances (the population of the ontology). The knowledge base followed the basic ontology/schema defined in fig.1. A Neo4j graph database [72] was used to implement this knowledge graph. Neo4j was picked because of its flexibility for data modelling, access to built-in network analysis tools, and its powerful query language called Cypher [73]. [74] presents an approach for using Neo4j in support of system analysis in conjunction with MBSE.

*Querying and Reasoning Using the Knowledge Graph*

Fig.2 gives an overview of the entire knowledge graph in Neo4j (provided simply for illustrative purposes). Various new insights could be drawn from the graph using Neo4j's query language, Cypher [73]. For example, the following cypher fragment would return the complementary design space exploration techniques associated with the SEAM method (linked by common artifacts):

```
MATCH (technique:Technique)<-[:TAKES_AS_INPUT]-
(artifact:Artifact)<-[:GENERATES]-
(method:Method {uid:'SEAM'})
RETURN technique
```

Similarly for tools, in particular Cameo Systems Modeller:

```
MATCH (technique:Technique)<-[:TAKES_AS_INPUT]-
(artifact:Artifact)<-[:GENERATES]-(method:Method)<-
[:CAN_IMPLEMENT]-(tool:Tool{uid:'Cameo'})
RETURN technique
```

For languages, such as SysML V1:

```
MATCH (technique:Technique)<-[:TAKES_AS_INPUT]-
(artifact:Artifact)<-[:GENERATES]-(method:Method)<-
[:CAN_IMPLEMENT]-(tool:Tool)<-[:AVAILABLE_IN]-
(language:Language)
RETURN technique
```

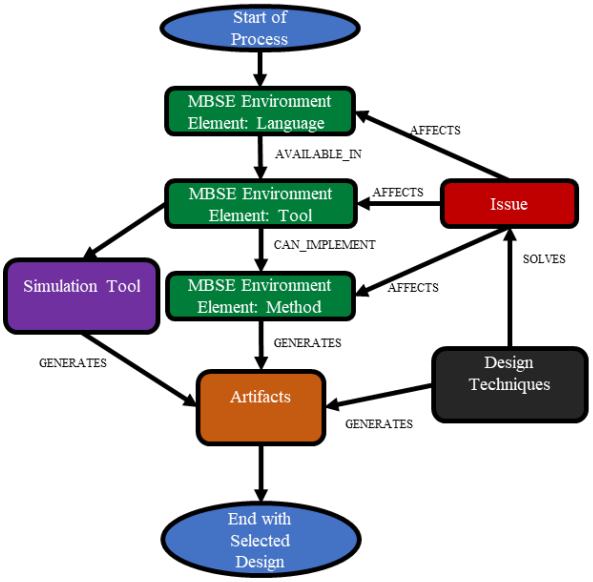


**Figure 1 Knowledge base ontology and MBSE process model.**

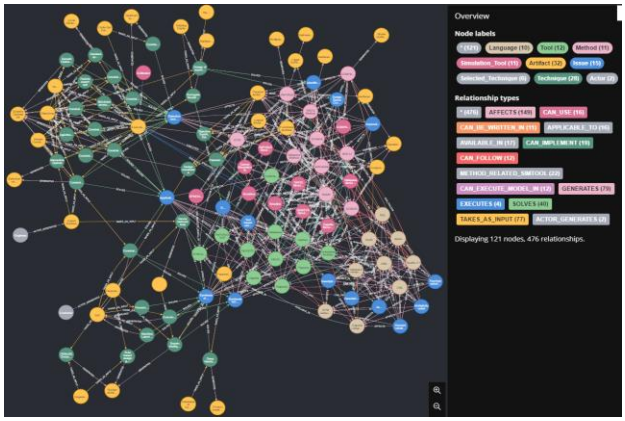


**Figure 2 Overview of the review knowledge graph**

Further to this, simple reasoning could be performed, meaning the derivation of new information from the graph. Specifically by following the ontology in fig.1 it was possible to infer node categories, such as languages, tools and methods all being examples of MBSE environment element, without this needing to be explicitly stored in the knowledge base. This could be performed using the following cypher fragment:

```
CALL n10s.inference.nodesLabelled('MBSE_Environment_
Element', {
catNameProp: "dbLabel",
catLabel: "Type",
subCatRel: "NARROWER_THAN"
})
YIELD node
RETURN node.uid as uid, labels(node) as categories
```

The CALL n10s.inference.nodesLabelled clause calls an inference method that adds the node label MBSE_Environment_Element to nodes that are of a type defined as NARROWER_THAN the supertype MBSE_Environment_Element (meaning that the narrower than relationship exists between them). See figure 3. While this is a simple inference to make, future work is hoped to identify further subtypes to each of the MBSE environment elements, issues and techniques, making this inference capability more useful.

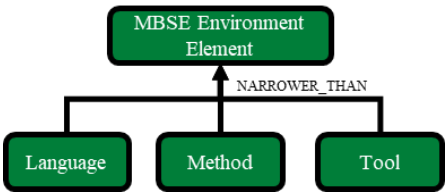


**Figure 3 MBSE environment type definitions in the knowledge graph**

With this reasoning capability the knowledge base was extended to a full knowledge graph, following the definition; a "knowledge graph acquires and integrates information into an ontology and applies a reasoner to derive new knowledge" [75]. Reasoning is a process by which some new piece of information is inferred by a knowledge graph [76], performed by some "reasoner" mechanism.

*Network Analysis*

When undertaking a project using MBSE, the process was modelled starting with the selection of a language, then tool, then method, that in turn generates a set of system engineering artifacts, after which design space exploration techniques may be employed taking certain artifacts as inputs and generating new ones as outputs. The overall MBSE process finishes once a particular artifact has been created, for example, detailed system design. This is shown in fig.1. This allowed different MBSE processes to be explored and the number and significance of related issues compared using the knowledge graph. By adding different design space exploration techniques to the process, certain issues could be "solved", therefore their issue cost as part of an MBSE

process would be set to zero. For example, using surrogate modelling as a technique would address the issue of simulation time, see table 8 and 9. The details of how each issue would be solved varies, and is not explicitly modelled in the knowledge graph, beyond the removal of the specific issue cost when a relevant technique is used.

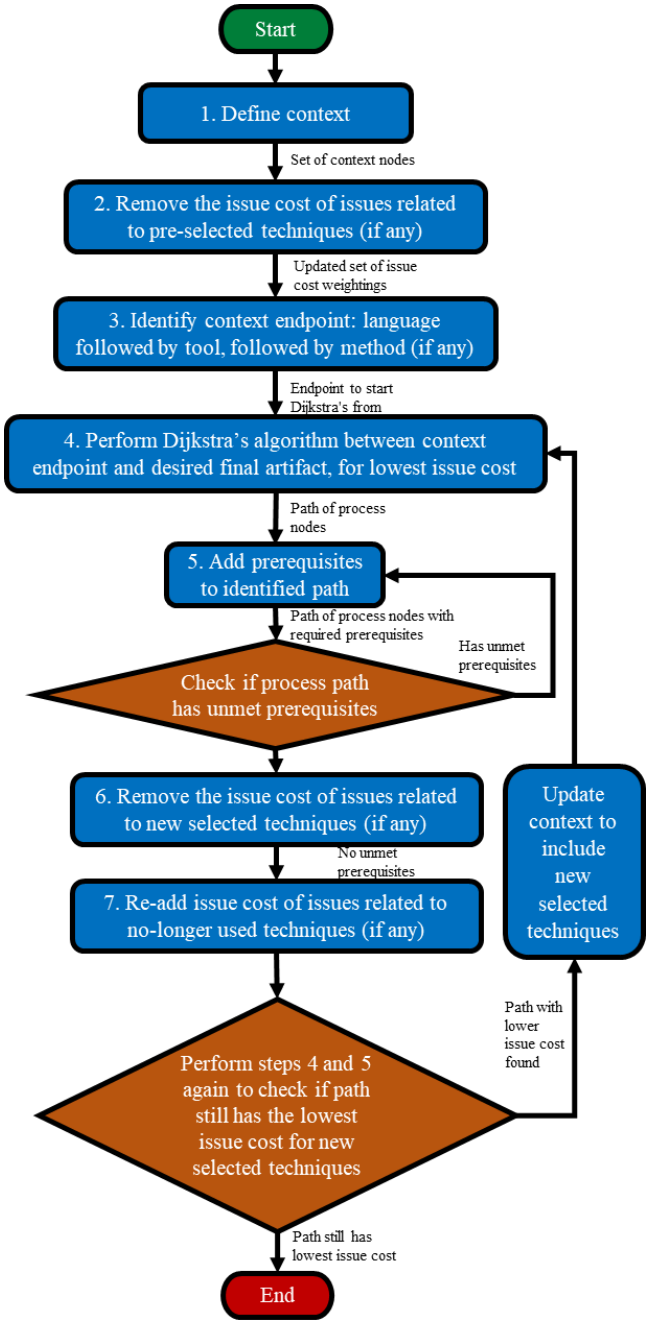


**Figure 4 Flowchart showing the steps used to identify MBSE processes with the knowledge graph**

A search algorithm was developed for identifying efficient MBSE processes using the knowledge graph, leveraging network analysis tools offered by Neo4j. Dijkstra's algorithm [77] was used to identify the shortest path between nodes, thus the most efficient process or process with the lowest issue cost. This algorithm was chosen over alternatives such as the A* shortest path algorithm [78] (that uses a heuristic based approach to guide traversal of nodes) as it was simple to implement and the size of the graph being considered was relatively small.

Issue cost, the proxy for path length used for Dijkstra's algorithm, was computed as the sum of a node's related issue severity. For example the SEAM method that has 4 issues related to it in the knowledge graph, had a total issue cost of 12, counting each issue's severity.

Some logic for identifying pre-requisites for certain parts of a process were added. For example methods creating particular artifacts needed for selected design space exploration techniques. The concept of a "context" was also created, that included optional descriptions of selected language, tool, method, simulation tool and desired destination. Design space exploration techniques could be "preselected" as part of this context, in which case the algorithm would fill in the gaps, by suggesting any additional techniques required to reach the desired destination artifact, or the algorithm could be left to select all of the techniques on its own. As techniques selected as part of the process would "resolve" issues, this led to an update in path issue costs. As a result, an outer loop needed to be added to the Dijkstra's algorithm and prerequisite identification steps to ensure that the process path identified still had the lowest possible issue cost. Therefore the search algorithm could iterate on its selected techniques and eventually converge on the most efficient MBSE process for the given context. This process followed by the search algorithm is presented in fig.4.

The flexibility in information provided in the search algorithm input meant that it could be performed with a range of constraints to suit an engineer working on a project in an organisation that might have access to only certain languages/tools/methods/techniques, and still identify efficient MBSE processes that are relevant to the project context.

Fig.5 shows an example of a path generated using this method, with the context shown in table 10. The path identified includes the techniques; Objective SysML stereotypes (either using measures of effectiveness already included in SysML V1 or the new stereotypes proposed in [27]), parametric SysML Stereotypes [27], design constraint and dependant variable stereotypes and finally constrained genetic optimisation.

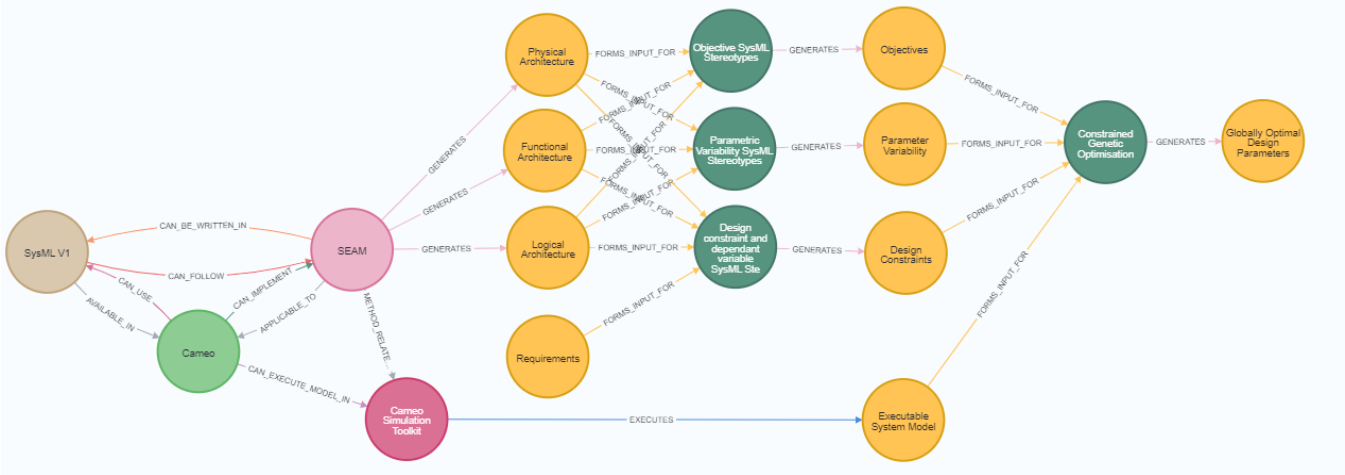


**Figure 5 Example MBSE process identified using the search algorithm, process starts on the left and ends on the right**

**Table 10 Example MBSE Context**

| Context Element | Selection |
|---|---|
| **Language** | SysML V1 |
| **Tool** | Cameo Systems Modeler |
| **Method** | SEAM |
| **Simulation Tool** | Cameo Simulation Toolkit |
| **Pre-Selected Design Space Exploration Techniques** | Constrained Genetic Optimisation, Surrogate Modelling |
| **Desired Destination** | Globally Optimal Design Parameters |

## 5. Discussion and Future Work

The currently available MBSE environments generally fall into two categories; those based on a specialised MBSE language and those using SysML V1 (or in future V2). SysML is the most widely used MBSE language, with many tools and methods developed with it in mind. This, in part, is because of its flexibility, allowing practitioners to tailor the language to better fit their needs. Besides this, the fact that the language is a descendant of UML has likely also helped it to gain traction. Some of the tools and methods that are not related to SysML are focussed on a particular domain, such as COMPASS, with its focus on system-software co-engineering [79], and therefore see more specialisation in capabilities, such as COMPASS's generation of FMEA tables and FDIR assessment features.

SysML V2, while still in the early stages of release, may help alleviate some of the issues raised here, particularly by providing basic design space description capabilities. However, these new features will need to be linked to techniques that can identify and select new designs.

Using the completed knowledge graph presented in section 4, it was possible to consider a wide range of candidate techniques for design space exploration with MBSE and as new ones are identified, these can easily be entered into the knowledge graph and their usefulness measured. The knowledge graph however is still only an approximation of the true problem. As a result, the design of case studies assessing new techniques will still rely on judgement of factors not covered by the knowledge graph, such as access to related expertise and time required to implement versus execute.

The issues presented cover a wide range of aspects of MBSE elements and required capabilities for design space exploration. These could be further categorized alongside the proposed design space exploration techniques, which may assist in identifying issue solutions. For example issues related to the description of the design space (or variant modelling), as opposed to say selection of a particular design in that space, would have clear links to techniques that provide these design space descriptions.

The variability framework used for this study could be expanded to cover the potential for varying the objectives of the design space exploration, and further detail of the current state of the system design (beyond simply the MBSE environment used to produce it). Besides this, provision for sensitivity of different designs would be a useful inclusion, as well as further refined variation subtypes. Work beyond the scope of this paper is currently under way to address these shortcomings.

Finally, the languages, tools and methods presented here do not represent an exhaustive list, and clearly further additions to the knowledge graph will benefit its usefulness.

## 7. CONCLUSION

This study has reviewed a number of today's MBSE languages, tools and methods. The interrelations between these MBSE environment elements have been mapped and a set of related issues and shortcomings for exploring different system designs has been identified. Design space exploration techniques have been proposed as candidate solutions to these issues and all the information collected during the review has been collated into a knowledge graph to allow further understanding and analysis of potential MBSE processes. The code and data used to populate and interact with the knowledge base maybe found in a GitHub repository [80].


## ACKNOWLEDGEMENTS

Many thanks to the Engineering and Physical Sciences Research Council (EPSRC) for providing Doctoral Training Partnership Award funding to facilitate this work, and to Joe Gregory for his advice and work on SEAM.


## REFERENCES


[1] P. Leserf, P. de Saqui-Sannes, and J. Hugues, "Trade-off analysis for SysML models using decision points and CSPs," *Softw. Syst. Model.*, vol. 18, no. 6, pp. 3265–3281, 2019, doi: 10.1007/s10270-019-00717-0.

[2] M. M. LaSorda, J. Borky, and R. Sega, "Model-Based Systems Architecting with Decision Quantification for Cybersecurity, Cost, and Performance," in *2020 IEEE Aerospace Conference*, Mar. 2020, pp. 1–13. doi: 10.1109/AERO47225.2020.9172283.

[3] M. Chodas, R. Masterson, and O. de Weck, "Addressing Deep Uncertainty in Space System Development through Model-based Adaptive Design," in *2020 IEEE Aerospace Conference*, Mar. 2020, pp. 1–17. doi: 10.1109/AERO47225.2020.9172672.

[4] M. M. LaSorda, J. Borky, and R. Sega, "Model-based architecture optimization for major acquisition analysis of alternatives," in *2018 IEEE Aerospace Conference*, Mar. 2018, pp. 1–8. doi: 10.1109/AERO.2018.8396526.

[5] B. P. Douglass, "Chapter 6 - Agile Systems Architectural Analysis and Trade Studies," in *Agile Systems Engineering*, B. P. Douglass, Ed., Boston: Morgan Kaufmann, 2016, pp. 281–311. doi: 10.1016/B978-0-12-802120-0.00006-0.

[6] M. Wäschle, A. Martin, A. Radimersky, M. Behrendt, and A. Albers, "Supporting The Modelling In Mbse By Applying Product Generation Engineering Using Electronic Compact Actuators As An Example," *Proc. Des. Soc. Des. Conf.*, vol. 1, pp. 2425–2434, May 2020, doi: 10.1017/dsd.2020.293.

[7] S. Friedenthal, A. Moore, and R. Steiner, "Getting Started with SysML," in *A Practical Guide to SysML*, Elsevier, 2015, pp. 31–51. doi: 10.1016/b978-0-12-800202-5.00003-5.

[8] A. Gal, "Ontology Engineering," in *Encyclopedia of Database Systems*, L. LIU and M. T. ÖZSU, Eds., Boston, MA: Springer US, 2009, pp. 1972–1973. doi: 10.1007/978-0-387-39940-9_1315.

[9] Object Modelling Group, "OMG Unified Modeling LanguageTM (OMG UML), Superstructure." Object Modelling Group, Nov. 14, 2014. Accessed: Aug. 22, 2022. [Online]. Available: https://www.omg.org/spec/UML/2.4/Superstructure/PDF

[10] P. de Saqui-Sannes, R. Vingerhoeds, C. Garion, and X. Thirioux, "A Taxonomy of MBSE Approaches by Languages, Tools and Methods," *IEEE Access*, vol. PP, pp. 1–1, Jan. 2022, doi: 10.1109/ACCESS.2022.3222387.

[11] OMG, "OMG Systems Modeling Language V1.6." Jan. 11, 2019. Accessed: Nov. 29, 2022. [Online]. Available: https://sysml.org/.res/docs/specs/OMGSysML-v1.6-19-11-01.pdf

[12] de Koning, Hans Peter, "SysML Version 2 - Final Stretch," Nov. 2022. Accessed: Nov. 29, 2022. [Online]. Available: https://indico.esa.int/event/407/contributions/7390/attachments/4793/7804/1710%20-%20Abstract%20-%20SysML%20Version%202%20Final%20Stretch.pdf

[13] "OSMoSE - Home." Accessed: Nov. 30, 2022. [Online]. Available: https://mb4se.esa.int/OSMOSE_Main.html

[14] "OMG System Modeling Language™ (SysML®) v2 Release." SysML v2 Submission Team (SST), Oct. 30, 2022. Accessed: Nov. 08, 2022. [Online]. Available: https://github.com/Systems-Modeling/SysML-v2-Release

[15] J.-L. Voirin, S. Bonnet, D. Exertier, and V. Normand, "Simplifying (and enriching) SysML to perform functional analysis and model instances," *INCOSE Int. Symp.*, vol. 26, no. 1, pp. 253–268, Jul. 2016, doi: 10.1002/j.2334-5837.2016.00158.x.

[16] L. Li, N. L. Soskin, A. Jbara, M. Karpel, and D. Dori, "Model-Based Systems Engineering for Aircraft Design With Dynamic Landing Constraints Using Object-Process Methodology," *IEEE Access*, vol. 7, pp. 61494–61511, 2019, doi: 10.1109/ACCESS.2019.2915917.

[17] M. Di Maio *et al.*, "Evaluating MBSE Methodologies Using the FEMMP Framework," in *2021 IEEE International Symposium on Systems Engineering*

*(ISSE)*, Sep. 2021, pp. 1–8. doi: 10.1109/ISSE51541.2021.9582465.

[18] "BSISO/PAS 19450 Automation systems and integration - Object-Process Methodology." Intenational Standards Organisation, Dec. 15, 2015.

[19] M. Peleg and D. Dori, "The model multiplicity problem: experimenting with real-time specification methods," *IEEE Trans. Softw. Eng.*, vol. 26, no. 8, pp. 742–759, Aug. 2000, doi: 10.1109/32.879812.

[20] J. (Yudit) Somekh, M. Choder, and D. Dori, "Conceptual Model-Based Systems Biology: Mapping Knowledge and Discovering Gaps in the mRNA Transcription Cycle," *PloS One*, vol. 7, p. e51430, Dec. 2012, doi: 10.1371/journal.pone.0051430.

[21] GENSYS, "GENSYS Key Concepts." Accessed: Nov. 09, 2022. [Online]. Available: https://vitechcorp.com/resources/GENESYS/onlinehelp/desktop/Key_Concepts.htm

[22] P. Minacapilli, F. Zurita, S. Campo Perez, A. Rodríguez Pérez-Silva, and D. Lasheras, "Small satellites mission design enhancement through MBSE and DDSE toolchain," Nov. 2022.

[23] SAE International, "Architecture Analysis and Design Language (AADL)." SAE International. Accessed: Nov. 30, 2022. [Online]. Available: https://www.sae.org/standards/content/as5506d/preview/

[24] "SLIM Specifaction." COMPASS Consortium, Aug. 27, 2016. Accessed: Nov. 20, 2022. [Online]. Available: https://www.compass-toolset.org/docs/slim-specification.pdf

[25] T. Bayer *et al.*, "Europa Clipper: MBSE Proving Ground," in *2021 IEEE Aerospace Conference (50100)*, Mar. 2021, pp. 1–19. doi: 10.1109/AERO50100.2021.9438186.

[26] Marco Ferrogalini, "MBSE at the heart of Airbus Digital Transformation," presented at the MBSE2022, Toulouse, Oct. 22, 2023. Accessed: Feb. 24, 2023. [Online]. Available: https://indico.esa.int/event/407/contributions/7561/attachments/5009/7810/Keynote%20-%20MBSE%20at%20the%20heart%20of%20the%20Airbus%20Digital%20Transformation.pdf

[27] L. Timperley, L. Berthoud, and C. Snider, "Towards Improving the Design Space Exploration Process Using Generative Design With MBSE".

[28] "Model-based System and Architecture Engineering with the Arcadia Method - 1st Edition." Accessed: Mar. 21, 2023. [Online]. Available: https://www.elsevier.com/books/model-based-system-and-architecture-engineering-with-the-arcadia-method/voirin/978-1-78548-169-7

[29] "Engineering Systems Design Rhapsody - Details." Accessed: Mar. 20, 2023. [Online]. Available: https://www.ibm.com/products/systems-design-rhapsody/details

[30] OPMCloud, "OPMCloud Features," OPCloud. Accessed: Nov. 10, 2022. [Online]. Available: https://www.opcloud.tech/features

[31] "Overview - TASTE." Accessed: Nov. 29, 2022. [Online]. Available: https://taste.tuxfamily.org/wiki/index.php?title=Overview

[32] S. Duncan, "Flight Software Development with TASTE," p. 4.

[33] M. Bozzano, H. Bruintjes, A. Cimatti, J.-P. Katoen, T. Noll, and S. Tonetta, "COMPASS 3.0," in *Tools and Algorithms for the Construction and Analysis of Systems*, T. Vojnar and L. Zhang, Eds., in Lecture Notes in Computer Science. Cham: Springer International Publishing, 2019, pp. 379–385. doi: 10.1007/978-3-030-17462-0_25.

[34] "COMPASS – Correctness, Modeling and Performance of Aerospace Systems." Accessed: Nov. 30, 2022. [Online]. Available: https://www.compass-toolset.org/

[35] A. Bombardelli *et al.*, "COMPASTA: Extending TASTE with Formal Design and Verification Functionality," in *Model-Based Safety and Assessment*, C. Seguin, M. Zeller, and T. Prosvirnova, Eds., in Lecture Notes in Computer Science. Cham: Springer International Publishing, 2022, pp. 21–27. doi: 10.1007/978-3-031-15842-1_2.

[36] J.-L. Voirin, "2 - Main Perspectives Structuring the Modeling Approach," in *Model-Based System and Architecture Engineering with the Arcadia Method*, J.-L. Voirin, Ed., Elsevier, 2018, pp. 15–18. doi: 10.1016/B978-1-78548-169-7.50002-0.

[37] L. Batista and O. Hammami, "Capella based system engineering modelling and multi-objective optimization of avionics systems," in *2016 IEEE International Symposium on Systems Engineering (ISSE)*, Oct. 2016, pp. 1–8. doi: 10.1109/SysEng.2016.7753127.

[38] S. Friedenthal, A. Moore, and R. Steiner, "Residential Security System Example Using the Object-Oriented Systems Engineering Method," in *A Practical Guide to SysML*, Elsevier, 2015, pp. 417–504. doi: 10.1016/b978-0-12-800202-5.00017-5.

[39] Hans-Peter Hoffman, *Model-Based Systems Engineering with Rational Rhapsody and Rational Harmony for Systems Engineering*, Deskbook 3.1.2. 2022. Accessed: Nov. 17, 2023. [Online]. Available: https://www.ibm.com/support/pages/model-based-systems-engineering-rational-rhapsody-and-rational-harmony-systems-engineering-deskbook-312

[40] Hoppe, M, "RePoSyD wiki." Accessed: Nov. 16, 2023. [Online]. Available: https://reposyd.de/start

[41] "IEEE Standard for Application and Management of the Systems Engineering Process," *IEEE Std 1220-2005 Revis. IEEE Std 1220-1998*, pp. 1–96, Sep. 2005, doi: 10.1109/IEEESTD.2005.96469.

[42] Beyerlein,Sophian Theodor Mohamed, "RePoSyD Model Based Systems Engineering: Methodology Assessment Using the FEMMP Framework and a Steam Engine Case Study," Bachelor's, Technische Hochschule Ingolstadt, 2020.

[43] D. Long and Z. Scott, *A Primer for Model-Based Systems Engineering*. Lulu.com, 2011.

[44] J. E. Long, "Relationships Between Common Graphical Representations Used in System Engineering," *INSIGHT*, vol. 21, no. 1, pp. 8–11, 2018, doi: 10.1002/inst.12183.

[45] D. Dori, R. Martin, and A. Blekhman, *Model-based meta-standardization*. 2010, p. 597. doi: 10.1109/SYSTEMS.2010.5482321.

[46] "Valispace Home," Valispace. Accessed: Mar. 21, 2023. [Online]. Available: https://www.valispace.com/

[47] M. Bandecchi, B. Melton, and B. Gardini, "The ESA/ESTEC concurrent design facility," Jan. 2000.

[48] Bernaudin,Jean-Baptiste, "MBSE on MSR ERO: a use case," presented at the MBSE2021, Toulouse, Sep. 2021. Accessed: Nov. 29, 2021. [Online]. Available: https://indico.esa.int/event/386/contributions/6227/attachments/4269/6378/1145%20-%20mbse%20on%20msr%20ero%20a%20use%20case.pdf

[49] C. Coelho, "MBSE Best Practices Project CGI Executive Summary." ESA, 2021. Accessed: Nov. 29, 2022. [Online]. Available: https://nebula.esa.int/sites/default/files/neb_study/2520/C4000133517ExS.pdf

[50] J. Gregory, "A Model-Based Framework for Early-Stage Analysis of Spacecraft," PhD Thesis, University of Bristol, 2022. Accessed: Jan. 12, 2022. [Online]. Available: https://www.researchgate.net/publication/363265053_A_Model-Based_Framework_for_Early-Stage_Analysis_of_Spacecraft

[51] A. Dalvi, A. Razban, H. El-Mounyari, T. El-Mekkawy, and R. Promyoo, *Integrated System Model of District Cooling for Energy Consumption Optimization*. 2020.

[52] "Open Modelica." Accessed: Mar. 21, 2023. [Online]. Available: https://openmodelica.org/

[53] Mathworks, "System Modeling and Simulation - MATLAB & Simulink Solutions." Accessed: Nov. 10, 2022. [Online]. Available: https://uk.mathworks.com/solutions/system-design-simulation.html

[54] "Collimator - Data driven design and simulation." Accessed: Nov. 15, 2022. [Online]. Available: https://www.collimator.ai//

[55] "Cameo Simulation Toolkit - CATIA - Dassault Systèmes®." Accessed: Mar. 21, 2023. [Online]. Available: https://www.3ds.com/products-services/catia/products/no-magic/cameo-simulation-toolkit/

[56] D. N. Mavris, K. Griendling, and C. E. Dickerson, "Relational-oriented systems engineering and technology tradeoff analysis framework," *J. Aircr.*, vol. 50, no. 5, pp. 1564–1575, 2013, doi: 10.2514/1.C032079.

[57] F. Peres and M. Castelli, "Combinatorial Optimization Problems and Metaheuristics: Review, Challenges, Design, and Development," *Appl. Sci.*, vol. 11, no. 14, Art. no. 14, Jan. 2021, doi: 10.3390/app11146449.

[58] E. Polak, *Optimization: Algorithms and Consistent Approximations*. Springer Science & Business Media, 2012.

[59] H. K. R. Ong and T. Sortrakul, "Comparison of Selection Methods of Genetic Algorithms for Automated Component-Selection of Design Synthesis with Model-Based Systems Engineering." Accessed: Jan. 21, 2022. [Online]. Available: https://www.semanticscholar.org/paper/Comparison-of-Selection-Methods-of-Genetic-for-of-Ong-Sortrakul/6333ac9f11a25cc6d1fcf68c1a962b2421ddc0ad

[60] S. Paterna, M. Santoni, and L. Bruzzone, "An approach based on multiobjective genetic algorithms to schedule observations in planetary remote sensing missions," *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.*, vol. 13, pp. 4714–4727, 2020, doi: 10.1109/JSTARS.2020.3015284.

[61] IEEE Staff, "Multi Domain optimization with SysML

modeling," IEEE, 2015.

[62] J. Mádar, J. Abonyi, and F. Szeifert, "Interactive particle swarm optimization," in *Proceedings - 5th International Conference on Intelligent Systems Design and Applications 2005, ISDA '05*, 2005, pp. 314–319. doi: 10.1109/ISDA.2005.58.

[63] G. Wu, H. Wang, W. Pedrycz, H. Li, and L. Wang, "Satellite observation scheduling with a novel adaptive simulated annealing algorithm and a dynamic task clustering strategy," *Comput. Ind. Eng.*, vol. 113, pp. 576–588, Nov. 2017, doi: 10.1016/j.cie.2017.09.050.

[64] J. Liu, T. Dwyer, G. Tack, S. Gratzl, and K. Marriott, "Supporting the Problem-Solving Loop: Designing Highly Interactive Optimisation Systems," *ArXiv200903163 Cs*, Sep. 2020, Accessed: Jan. 21, 2022. [Online]. Available: http://arxiv.org/abs/2009.03163

[65] M. Halvorson, J. Fuchs, D. Thomas, A. Blair, S. Patel, and N. Moyers, "Model-Based Mission Planning: Reducing Mission Planning Costs by Generating Mission-Unique Architecture and Process Frameworks," presented at the 73rd International Astronautical Congress (IAC), Paris, France: International Astronautical Federation (IAF), Sep. 2022.

[66] C. R. Hicks, *Fundamental Concepts in the Design of Experiments*. Holt, Rinehart and Winston, 1973.

[67] L. Caldas, "GENE_ARCH: An Evolution-Based Generative Design System for Sustainable Architecture," in *Intelligent Computing in Engineering and Architecture*, I. F. C. Smith, Ed., in Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2006, pp. 109–118. doi: 10.1007/11888598_12.

[68] J.-P. Ceglarek, R. Boumghar, and R. Bertrand, "AI4CE - Automated Space Mission Design Concepts Generation with Reinforcement Learning," p. 5.

[69] S. Meacham, F. Gioulekas, and K. Phalp, "SysML based Design for Variability enabling the Reusability of Legacy Systems towards the support of Diverse Standard Compliant Implementations or Standard Updates: The Case of IEEE-802.15.6 Standard for e-Health Applications," in *Proceedings of the Eighth EAI International Conference on Simulation Tools and Techniques*, Athens, Greece: ACM, 2015. doi: 10.4108/eai.24-8-2015.2261108.

[70] V. Singh and N. Gu, "Towards an integrated generative design framework," *Des. Stud.*, vol. 33, no. 2, pp. 185–207, Mar. 2012, doi: 10.1016/j.destud.2011.06.001.

[71] A. Kiryakov, "Ontologies for Knowledge Management," in *Semantic Web Technologies*, John Wiley & Sons, Ltd, 2006, pp. 115–138. doi: 10.1002/047003033X.ch7.

[72] "Neo4j Graph Data Platform | Graph Database Management System." Accessed: Mar. 16, 2023. [Online]. Available: https://neo4j.com/

[73] "Introduction - Cypher Manual," Neo4j Graph Data Platform. Accessed: Oct. 04, 2023. [Online]. Available: https://neo4j.com/docs/cypher-manual/5/introduction/

[74] F. Schummer and M. Hyba, "An approach for system analysis with model-based systems engineering and graph data engineering," *Data-Centric Eng.*, vol. 3, p. e33, 2022, doi: 10.1017/dce.2022.33.

[75] L. Ehrlinger and W. Wöß, "Towards a Definition of Knowledge Graphs".

[76] C. Feilmayr and W. Wöß, "An analysis of ontologies and their success factors for application to business," *Data Knowl. Eng.*, vol. 101, pp. 1–23, Jan. 2016, doi: 10.1016/j.datak.2015.11.003.

[77] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numer. Math.*, vol. 1, no. 1, pp. 269–271, Dec. 1959, doi: 10.1007/BF01386390.

[78] P. E. Hart, N. J. Nilsson, and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," *IEEE Trans. Syst. Sci. Cybern.*, vol. 4, no. 2, pp. 100–107, Jul. 1968, doi: 10.1109/TSSC.1968.300136.

[79] M. Bozzano, A. Cimatti, J.-P. Katoen, V. Nguyen, T. Noll, and M. Roveri, *The COMPASS Approach: Correctness, Modelling and Performability of Aerospace Systems*, vol. 5775. 2009, p. 186. doi: 10.1007/978-3-642-04468-7_15.

[80] L. Timperley, "Ghipag/MBSE_Design_Space_Problem_Space." Oct. 02, 2023. Accessed: Oct. 04, 2023. [Online]. Available: https://github.com/Ghipag/MBSE_Design_Space_Problem_Space

## BIOGRAPHY

***Louis Timperley*** *is a PhD student studying at the University of Bristol, where he is researching the application of Model-Based Systems Engineering (MBSE) to trade-offs, system optimsation and other forms of design space exploration in the spacecraft domain. He completed his Master's degree in Aerospace Engineering also at the University of Bristol. His previous work has included investigating the use of MBSE for the*

*development and validation of spacecraft flight software as well as reliability analysis of spacecraft systems.*

***Lucy Berthoud** is a Professor of Space Systems Engineering with the Department of Aerospace Engineering at the University of Bristol. She also works for the European spacecraft manufacturer Thales Alenia Space part-time. She has a background in advanced mission concepts and systems engineering for spacecraft. Her research expertise includes CBRN monitoring from space, model-based systems engineering, Mars Sample Return and IR CubeSats.*

***Chris Snider** is a Senior Lecturer in Engineering Design at the University of Bristol. For the last 10 years he's been working at the interface of digital and physical design, specializing in design technologies and the digitization and modelling / simulation of both machines and the processes that create them. Recent work has spanned digital twinning / threading, graph-based representations of complex systems, and integrated physical-digital toolchains in product and machine design.*

***Theo Tryfonas** (PhD (Athens), MSc (Athens), BSc (Crete)) holds the Chair of Infrastructure Systems & Urban Innovation at the Department of Civil Engineering, University of Bristol, UK. He is a computer scientist with expertise in urban data, the IoT and Smart Cities. He is a Chartered Fellow of the BCS, The Chartered Institute for IT, and a Fellow of the Royal Society for Arts, Manufactures and Commerce (RSA). His current interests include future energy systems, air quality monitoring and connected mobility.*