POLITECNICO DI TORINO
Repository ISTITUZIONALE

A Concurrent Model-Based Systems Engineering Methodology for SmallSat Lunar Mission Design

Original
A Concurrent Model-Based Systems Engineering Methodology for SmallSat Lunar Mission Design / Campioli, S., Deiana, C., My, C., Corpino, S.. - In: AEROSPACE SCIENCE AND TECHNOLOGY. - ISSN 1270-9638. - (2026). [10.1016/j.ast.2026.113865]

Availability:
This version is available at: 11583/3016135 since: 2026-09-30T08:39:58Z

Publisher:
Elsevier

Published
DOI:10.1016/j.ast.2026.113865

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

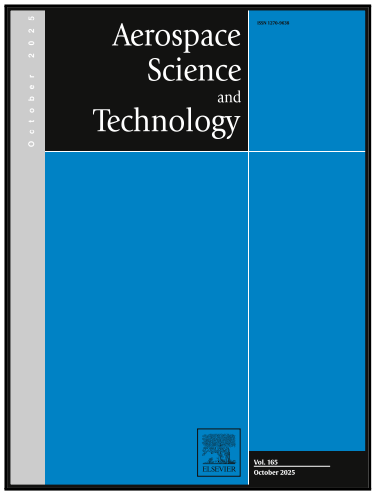
Publisher copyright

(Article begins on next page)

01 November 2026

Communicated by Dr Chen Anthony Siming

Journal Pre-proof



A Concurrent Model-Based Systems Engineering Methodology for SmallSat Lunar Mission Design

Serena Campioli , Carlotta Deiana , Chiara My , Sabrina Corpino

PII: S1270-9638(26)02241-8
DOI: https://doi.org/10.1016/j.ast.2026.113865
Reference: AESCTE 113865

To appear in: *Aerospace Science and Technology*

Received date: 30 April 2026
Revised date: 3 September 2026
Accepted date: 21 September 2026

Please cite this article as: Serena Campioli , Carlotta Deiana , Chiara My , Sabrina Corpino , A Concurrent Model-Based Systems Engineering Methodology for SmallSat Lunar Mission Design, *Aerospace Science and Technology* (2026), doi: https://doi.org/10.1016/j.ast.2026.113865

This is a PDF of an article that has undergone enhancements after acceptance, such as the addition of a cover page and metadata, and formatting for readability. This version will undergo additional copyediting, typesetting and review before it is published in its final form. As such, this version is no longer the Accepted Manuscript, but it is not yet the definitive Version of Record; we are providing this early version to give early visibility of the article. Please note that Elsevier's sharing policy for the Published Journal Article applies to this version, see: https://www.elsevier.com/about/policies-and-standards/sharing#4-published-journal-article. Please also note that, during the production process, errors may be discovered which could affect the content, and all legal disclaimers that apply to the journal pertain.

© 2026 The Author(s). Published by Elsevier Masson SAS.
This is an open access article under the CC BY-NC-ND license
(http://creativecommons.org/licenses/by-nc-nd/4.0/)

Journal Pre-proof

Highlights

A Concurrent Model-Based Systems Engineering Methodology for SmallSat Lunar Mission Design

Serena Campioli, Carlotta Deiana, Chiara My, Sabrina Corpino

- Lack of an integrated system representation limits consistency and traceability in concurrent space mission design.
- Current approaches struggle to manage complexity and data continuity across lifecycle phases in complex missions and systems development, such as interplanetary missions.
- Concurrent Model-Based Systems Engineering (CE-MBSE) methodology supports coordinated multidisciplinary iterations through a shared model-based engineering backbone, improving coherence across mission development phases.
- Application to the BioMoon lunar CubeSat mission shows improved traceability, coordination, and early verification.

Journal Pre-proof

Journal Pre-proof

A Concurrent Model-Based Systems Engineering Methodology for SmallSat Lunar Mission Design

Serena Campioli[a], Carlotta Deiana[a], Chiara My[a], Sabrina Corpino[a]

[a]*Department of Mechanical and Aerospace Engineering, Politecnico di Torino, Corso Duca degli Abruzzi 24, 10129, Torino, Italy:* serena.campioli@polito.it , carlotta.deiana@polito.it, chiara.my@studenti.polito.it, sabrina.corpino@polito.it

*Corresponding author. *serena.campioli@polito.it*

Abstract

This paper presents a Concurrent Model-Based Systems Engineering (CE–MBSE) methodology aimed at improving the design of complex space missions by combining the collaborative nature of Concurrent Engineering (CE) with the rigor and traceability of Model-Based Systems Engineering (MBSE). The proposed approach is centred on an integrated system model acting as a single source of truth across all lifecycle phases, enabling consistent management of requirements, system architectures, and multidisciplinary data. The methodology adapts concurrent workflows to different maturity stages, from early mission definition to verification and operational phases, while progressively increasing model fidelity. A modular and interoperable toolchain is introduced, integrating the Capella MBSE environment with the Altium Requirements & System Portal (RSP) through Python-based synchronization mechanisms, ensuring digital continuity and real-time data consistency across domains. The approach is applied and preliminarily assessed through its application to the *BioMoon* mission, a lunar CubeSat constellation designed to investigate the radiation environment and its effects on biological systems. Results demonstrate improved traceability across abstraction levels, enhanced multidisciplinary coordination, and the integration of quantitative engineering analysis within the verification workflow, while highlighting some challenges related to toolchain complexity, synchronisation overhead, and scalability to larger and more complex systems. Overall, the CE–MBSE methodology represents a promising framework to support coherent, knowledge-driven, and lifecycle-oriented design processes for next-generation space missions.

Keywords: Concurrent Engineering, Model-Based Systems Engineering, Lunar Mission & Systems Design, Small Satellite

Journal Pre-proof

1. Introduction

Systems Engineering (SE) provides a structured interdisciplinary framework to support the development of complex systems across the lifecycle, from their conceptual design up to operations and disposal [1]. In the space sector, this lifecycle perspective is essential to manage the strong interdependencies across all contributing technical and programmatic domains. Traditional space missions' development has long relied on sequential processes, often represented through the V-model [2], and formalised in the ECSS-M-ST-10C rev 1. [3]. While robust and well-established, this approach can be time-consuming for space systems, as strong subsystem interdependencies often cause cascading changes propagating across the entire architecture, leading to costly iterations.

To address these limitations, Concurrent Engineering (CE) has been progressively adopted in space

mission and system design as a complementary paradigm [4]. CE promotes the simultaneous involvement of multidisciplinary experts and stakeholders from the earliest design phase, when design decisions remain flexible and the cost change is still relatively low. By fostering continuous interaction among disciplines, CE supports the early identification of critical issues, inconsistencies, and design gaps which might otherwise emerge only in later stages of the development.

Despite its advantages, CE does not by itself guarantee continuity or consistency of engineering information [5]. In practice, information is often distributed across disconnected tools, documents, spreadsheets, and discipline-specific models: the absence of a shared and formalised system representation may lead to fragmented knowledge, limited traceability, and difficulties in managing interdependencies across disciplines. These issues become even more critical in multi-partner projects, where different standards, toolchains and working practices hinder coordination and collaboration. In addition, effective concurrent work requires close collaboration across disciplines and organisations, which may be constrained by communication barriers, resistance to change, and data-sharing concerns [5], [6].

In this context, Model-Based Systems Engineering (MBSE) provides a structured approach to formalise system knowledge and manage complexity across the lifecycle [2]. By capturing explicit relationships among stakeholder needs, requirements, functions, components, interfaces, and engineering properties, MBSE supports traceability, dependency management, and change-impact assessment. These capabilities are particularly valuable in concurrent environments, where engineering information evolves rapidly and must remain accessible and consistent across disciplines. However, the practical integration of MBSE into fast-paced concurrent workflows remains nontrivial, especially when architectural models must remain connected to specialised requirements management, engineering data, and analysis environments.

The resulting challenge is not merely the adoption of MBSE within a concurrent environment, but the definition of a structured methodology able to integrate model-based systems engineering principles into concurrent design activities while preserving information continuity, traceability, and consistency as the system evolves. In this context, the main critical aspect is the operational implementation of MBSE during fast multidisciplinary iterations characterising the concurrent workflow, where design changes may simultaneously affect, among the several, requirements and verification activities, architecture, operating modes, engineering parameters and analysis. Existing CE and MBSE approaches provide relevant methodological foundations, but the way in which concurrent workflows, model refinement, requirements management, multidisciplinary engineering activities, and verification processes should be coherently combined across different levels of system maturity remains only partially addressed.

Therefore, this paper proposes a CE-MBSE methodology that combines concurrent workflows with a model-based approach to ensure consistency, traceability, and lifecycle coherence in space mission development. Section 2 reviews the evolution of Concurrent Engineering, relevant MBSE principles and methodologies, and previous MBSE applications within Concurrent Engineering environments, and derives the research gap addressed by this work. Section 3 presents the proposed CE-MBSE methodology, including the concurrent workflow adaptation, model-based integration logic, and a supporting toolchain. Section 4 describes the *BioMoon* case study and discusses the application and assessment of the methodology, including a preliminary quantitative workflow efficiency evaluation and the observed benefits and limitations. Finally, Section 5 summarises the main conclusions and identifies directions for future work.

2. Background and Related Work

2.1 Evolution of Concurrent Engineering in Space Mission Design

Formally, CE was first defined by Winner et al. [7] as the "*systematic approach to the integrated, concurrent design of products and their related processes, including manufacturing and support. This approach is intended to cause the developers from the outset to consider all elements of the product life cycle from conception to disposal, including quality cost, schedule, and user requirements*". This definition highlights the key shift introduced by the concurrent approach: the transition from sequential execution of design activities to their concurrent and integrated development. Rather than treating design, manufacturing, testing, operations as separate phases, CE promotes their simultaneous consideration from the earliest stages of the development process. Later, Bandecchi et al. [4] has set the well-known ESA Concurrent Engineering definition as "*a systematic approach to integrated product development that emphasizes the response to the customer expectations. It embodies team values of cooperation, 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*". Thus, three fundamental principles characterize the CE paradigm: (i) *early involvement of stakeholders* ensures that all relevant customer perspectives are considered from the beginning; (ii) *multidisciplinary team-based approach* enables continuous interaction among disciplines, fostering shared understanding and coordinated decision-making; (iii) *parallel execution of activities* allows for faster design convergence through iterative refinement and real-time feedback.

In the past three decades [5], [8], [4], significant effort has been spent to enhance concurrent design tools and team management techniques. This approach finds a suitable application in the space mission and system design, especially during its preliminary phases, for *Phase 0* and *Phase A* [3]. In fact, CE responds to a need to reduce timescales and costs and to increase and improve communication between interested parties involved in the process, from stakeholders to customers, to designers and users. This need is particularly evident during the preliminary stages of design, when the design is more fluid and susceptible to change.

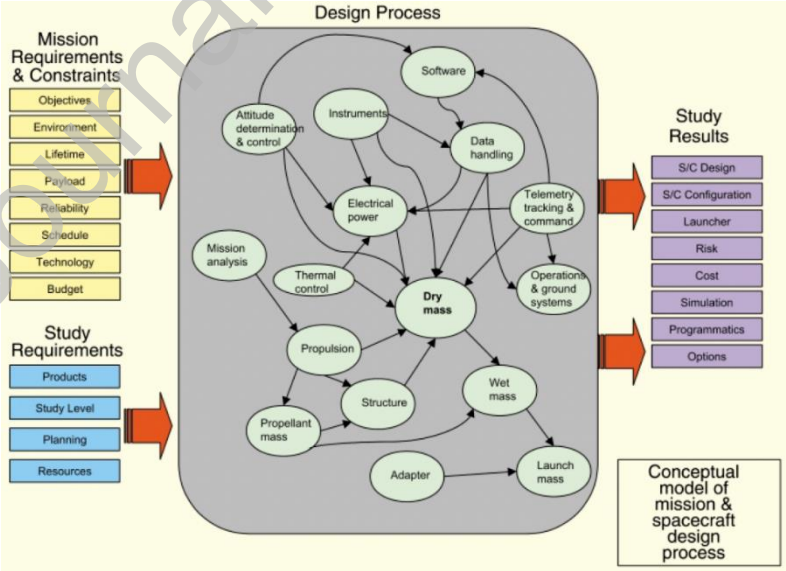


Figure 1 CE process [4]

The process, as shown in Figure 1, typically begins with preparatory meetings involving the customer, team leader, and systems engineers to refine mission requirements, constraints, design

drivers, and resource assumptions. The core study then proceeds through iterative multidisciplinary sessions in which specialists and stakeholders work in parallel, assess technical and programmatic aspects, explore alternatives, and correct emerging issues at an early stage. Its effectiveness depends on a collaborative team able to exchange information, update analyses, and support decisions in real time under the coordination of an experienced team leader.

Additionally, the process is *model-driven* and relies on integrated, often parametric, models that combine discipline-specific data, support rapid scenario updates, and formalise common design assumptions and responsibilities. These activities are usually conducted in a dedicated facility equipped with shared visualisation, communication, modelling, analysis, and data-exchange tools, with a dedicated workstation assigned to each discipline.

Overall, CE supports a faster and more integrated design process by promoting early interaction among disciplines and stakeholders. However, these benefits are not automatic outcomes of concurrent work. They depend on the capability of the concurrent environment to maintain shared assumptions, synchronized data, traceable decisions, and consistent interfaces across the different domains involved in the study. If the exchanged information is fragmented, outdated, weakly traced, or interpreted differently by each discipline, the same multidisciplinary interaction that makes CE valuable can become a source of confusion, rework, and design inconsistency [5]. In fact, CE requires the continuous exchange of large amounts of heterogeneous data, including requirements, assumptions, budgets, interfaces, design parameters, analysis results, risks, and programmatic information. However, these data are often distributed across spreadsheets, discipline-specific tools, databases, documents, presentations, and informal discussions.

As a result, a concurrent study may successfully support rapid multidisciplinary interaction, but still produce outputs that are difficult to trace, reuse, configure, or evolve after the session [6]. The challenge is therefore not only to make experts collaborate in real time, but to ensure that the information generated during this collaboration remains coherent, traceable, and reusable across disciplines, sessions, and lifecycle phases.

2.2 Model-Based Systems Engineering: Principles, Methodologies, and Enabling Technologies

The limitations identified in concurrent engineering environments have encouraged a transition from document- and file-centred information management towards Model-Based Systems Engineering (MBSE). MBSE formalises the use of models to support requirements definition, system design, analysis, verification, and validation throughout the lifecycle [2]. Models provide a common and authoritative source of truth, representing system functions, architectures, and behaviours across lifecycle phases. By organising knowledge in a structured and consistent way, MBSE supports the management of complexity and interdependencies that characterise space missions, while improving communication among disciplines and stakeholders. When integrated within a concurrent framework, MBSE enhances the CE by improving traceability, consistency, and transparency throughout the design process.

2.2.1 MBSE Principles and Information Continuity

MBSE shifts the authoritative representation of engineering information from disconnected documents and discipline-specific files towards structured and interrelated digital models. A model-based environment generally combines interconnected system models, a modelling language with defined syntax and semantics, and a shared information base through which engineering data can be managed and exchanged.

Indeed, a model is a simplified representation (graphical, mathematical or physical) of a concept, phenomenon, relationship, structure or system [9]. Within an MBSE environment, different model

types support complementary engineering purposes. Descriptive models represent system requirements, functions, behaviours, architectures, and interfaces, whereas analytical models provide quantitative evaluations of system properties and performance. Their controlled integration is essential to maintain consistency between the architectural definition of the system and the analyses used to support design decisions.

The value of MBSE therefore extends beyond the use of multiple engineering models. In an MBSE approach, relationships among stakeholder needs, requirements, functions, components, parameters and verification artefacts are explicitly represented and maintained. This enables traceability across abstraction levels, supports trade-offs analysis and improves interface management, as well as facilitates the reuse of system knowledge throughout the lifecycle. Thus, engineering with disconnected models may still rely on inconsistent assumptions and baselines.

These capabilities are particularly relevant to concurrent design, where large amounts of heterogeneous information evolve rapidly during multidisciplinary sessions. However, the introduction of MBSE also requires careful scoping of modelling objectives, appropriate levels of fidelity, clearly assigned information ownership, and effective integration with the discipline-specific tools used for quantitative design and analysis.

2.2.2 Modelling Languages and MBSE Methodologies

A modelling language provides the syntax, rules and semantics [9] used to express system information, whereas an MBSE methodology defines the processes, methods, and supporting tools through which model-based systems engineering activities are performed [10]. This distinction is important because modelling languages, methodologies, and software environments address different aspects of model-based development.

Among the available standards, *Unified Modeling Language* (UML) [11] and *Systems Modeling Language* (SysML) [12] are the most widely adopted, with SysML becoming the reference language for systems engineering applications. The language is organized around four main modelling domains: (i) *Structure*, characterized by block definition diagrams and internal block diagrams; (ii) *Behavior*, captured through activity, sequence, use-case, and state-machine diagrams; (iii) *Requirements*, describing requirement hierarchies and traceability relationships; (iv) *Parametrics*, used to define quantitative constraints on system parameter values such as performance, reliability and mass properties. These modelling domains constitute the core pillars of SysML and provide a unified framework for describing complex multidisciplinary systems. More recently, SysML has evolved towards SysML v2 [13], which introduces a more rigorous semantic foundation together with both textual and graphical notation and enhanced support for model interchange and tool integration. These developments are relevant for multidisciplinary engineering environments; however, its adoption within established aerospace engineering process as well as its integration in industrial toolchains is still in progress.

Beyond their formal definition, modelling languages play different roles depending on the engineering activities and lifecycle stages they support. In space applications, a recent literature review [14] shows that SysML is widely used to represent mission objectives, requirements, spacecraft architectures, operational scenarios, and system behaviour, particularly during requirements management and model-development activities. UML and SysML are also frequently combined, while domain-specific languages such as Modelica are used where more specialised behavioural or physical modelling capabilities are required. More broadly, programming and data exchange languages are increasingly used to connect system models with simulation, analysis, and configuration management environments, reflecting the growing need for interoperability within

space MBSE toolchains [14].

SysML-based methods, such as the Object-Oriented Systems Engineering Method (OOSEM) [15], provide flexible and comprehensive support for complex system development, although they require significant modelling expertise and tailoring to the selected project and tool environment. In the European space sector, the ESA MBSE Methodology [16] defines a recursive and iterative process linking mission objectives, stakeholder needs, requirements, functional analysis, system architecture, and verification information. Although intended to remain tool-independent, its practical implementation generally requires project-specific tailoring and suitable modelling environments. While these approaches provide flexible MBSE frameworks, ARChitecture Analysis & Design Integrated Approach (ARCADIA) [17] provides an alternative architecture-centred methodology organised around Operational Analysis, System Analysis, Logical Architecture, and Physical Architecture. Its explicit separation of abstraction levels supports functional decomposition, allocation, and architecture consistency. Operational Analysis defines the mission context, stakeholders, operational capabilities, and interactions independently of the system solution. System Analysis then introduces the System of Interest, its externally visible functions, and interfaces. Logical Architecture decomposes these functions and allocates them to logical components, while Physical Architecture refines the solution towards implementation-oriented system elements. This progression supports the transition from the problem space to the solution space while maintaining functional and architectural continuity.

ARCADIA is implemented in the open-source Capella environment through a dedicated domain-specific modelling language. Compared with general-purpose SysML-based approaches, this provides a more prescriptive architecture-development workflow, although quantitative analysis and requirements or verification management generally requires extensions and integration with external tools.

A summary of the leading MBSE methodologies, as detailed in [15], is presented in Table 1.

Methodology	**Main Strength**	**Main Limitation**	**Tool(s)**
OOSEM: Object-oriented and scenario-driven methodology based on SysML and Vee-model principles	Strong traceability; suitable for complex systems; supports reuse and verification activities.	Requires SysML expertise and significant modelling effort.	Cameo, Rhapsody, Enterprise Architect
ESA MBSE Methodology: Recursive and iterative methodology tailored for space systems engineering	Strong support for verification and traceability in space applications.	Requires tailoring according to project needs.	Ideally tool-agnostic but mainly implemented in Cameo and Enterprise Architect
ARCADIA: Architecture-centric methodology based on Operational, System, Logical and Physical analyses.	Excellent complexity management and functional allocation.	Limited quantitative analysis capabilities without external tools.	Capella

Table 1 Comparison of main MBSE methodologies

More broadly, no single modelling environment can replace the complete engineering toolchain used in space mission design. MBSE implementations therefore increasingly rely on requirements repositories, engineering databases, analytical and simulation tools, and controlled data-exchange mechanisms such as application programming interfaces and model transformations. Tool interoperability and semantic consistency consequently become essential conditions for

maintaining the system model as an authoritative engineering backbone.

While these methodologies, languages, and technologies provide the foundations for an integrated model-based environment, their effectiveness within concurrent engineering depends on how they are embedded into real-time multidisciplinary workflows. The following section therefore reviews the principal experiences of MBSE adoption within space Concurrent Engineering Centres (CECs).

2.3 MBSE in Concurrent Engineering Environments

In recent years, several space CECs have investigated the integration of MBSE into their established workflows. These experiences demonstrate the potential of model-based approaches to improve traceability and information consistency but also highlight the difficulty of maintaining modelling productivity during fast-paced multidisciplinary sessions.

NASA JPL's Team X [8] traditionally relies on linked spreadsheets, discipline-specific tools, and a central database to support rapid mission concept studies. Initial experiments [18] with SysML showed that a generic modelling environment could not provide sufficient productivity for direct use during concurrent sessions. The subsequent approach therefore relied on customised modelling views, domain-oriented interfaces, and reusable model elements intended to reduce the effort required from discipline specialists. This experience highlighted that MBSE must provide immediate and visible value to users; otherwise, it risks being perceived as an additional documentation activity rather than as active support to design.

NASA Langley's Engineering Design Studio followed a progressive adoption strategy. Its original workflow combined shared OneNote workspaces and Excel-based workbooks with centralised information consolidation. A reusable SysML template [19] was subsequently introduced to connect mission objectives, requirements, architectures, interfaces, activities, and study planning information. Starter models and predefined views reduced the effort required during sessions, while dedicated systems engineers or modellers captured design decisions and architectural updates. This experience also showed that familiar tools should not necessarily be replaced but progressively connected to the authoritative system model.

At ESA, the Concurrent Design Facility has historically relied on discipline-specific worksheets and a central repository, initially supported by the Open Concurrent Design Tool (OCDT) [20] and later by COncurrent Model-based Engineering Tool (COMET) [21]. SysML-based pilots [22] and the Model-Based methodology to support Space Systems Engineering (MBSSE) were subsequently investigated to connect requirements, scenarios, functional analysis, architectures, interfaces, and operating modes. These applications improved traceability and multidisciplinary communication, but also revealed difficulties in managing multiple configurations, alternatives, and model updates during concurrent sessions. Although ESA has progressively formalised its MBSE practices through the ESA MBSE Methodology, their systematic operational integration within the CDF workflow remains an evolving area.

Overall, the reviewed experiences indicate a progressive move from generic MBSE adoption towards tailored implementations designed to preserve modelling productivity within concurrent studies. Across NASA JPL, NASA Langley and ESA, recurring solutions include reusable templates, simplified views, dedicated modelling responsibilities, and integration with existing discipline-specific tools. At the same time, current implementations consistently highlight unresolved challenges related to modelling effort, tool interoperability, configuration management, and the limited connection between descriptive system architectures and quantitative analysis environments.

Journal Pre-proof

Journal Pre-proof

2.4 Literature Synthesis and Research Gap

Concurrent Engineering and Model-Based Systems Engineering address complementary needs in complex system development. CE enables rapid multidisciplinary interaction and parallel design evolution, whereas MBSE provides the structured information relationships required to maintain traceability, consistency, and knowledge continuity. Existing applications at NASA and ESA demonstrate the benefits of combining these approaches but also show that effective integration requires substantial workflow and tool tailoring.

Current implementations remain affected by modelling effort, heterogeneous toolchains, semantic and configuration inconsistencies, and weak integration between descriptive architecture models and quantitative analysis environments. Moreover, concurrent engineering applications remain primarily focused on conceptual design, while existing MBSE methodologies do not explicitly define how models, roles, tools, and information exchanges should be adapted to the pace of concurrent sessions and progressively evolved across lifecycle phases.

The identified gap therefore concerns the absence of an operational framework connecting concurrent workflows, architectural models, engineering repositories, requirements, simulations, and verification evidence through controlled and traceable information exchanges. To address this gap, this work proposes a lifecycle-oriented CE-MBSE methodology combining progressive model refinement, phase-adapted concurrent activities, and controlled information exchanges among the engineering environments involved in the study. Rather than replacing ARCADIA or the ESA MBSE Methodology, the proposed approach operationalises their model-based principles within concurrent mission design.

3. Methodology

3.1. CE-MBSE methodology overview

The proposed methodology adopts a Concurrent Model-Based Systems Engineering (CE-MBSE) approach to support the development of complex space missions and systems across their entire lifecycle. The methodology combines the collaborative and multidisciplinary nature of concurrent engineering with the rigour, traceability and continuity provided by MBSE, enabling a structured yet flexible development process.

Its main characteristic lies in the adaptation of model-based engineering activities to the iterative and multidisciplinary nature of concurrent design. Indeed, the proposed approach integrates model refinement, requirements management, engineering assessment, and verification within recurring concurrent iterations, so that design changes and their cross-disciplinary impacts can be progressively identified, assessed, propagated, and consolidated as the system evolves.

The methodology is based on five principles: (i) model-centred information management; (ii) progressive refinement across abstraction levels; (iii) explicit traceability among the different elements such as needs, requirements, functions, components; (iv) controlled exchange with discipline-specific environments; (v) lifecycle continuity between design phases and later project phases with a focus on verification and validation.

At the core lies an integrated system model, which acts as an authoritative model-based engineering backbone throughout all mission phases. This integrated model represents the mission context, including stakeholder needs and mission objectives, requirements, functional behaviour, physical architecture, and relevant meta-data for Modelling & Simulation (M&S) as well as for Verification & Validation (V&V). It is continuously refined as the system development progresses. All technical and programmatic domains, spanning from the different mission segments disciplines

to manufacturing, testing, costs and schedule, etc., interact concurrently with the integrated system model, which provides the common reference for identifying dependencies, assessing change impact and consolidating updated engineering information.

The methodology is therefore inherently lifecycle oriented. During early phases, it supports mission definition and architectural trade-offs, while in later stages it enables design consolidation, simulation-based verification, and preparation for manufacturing and testing. Model fidelity progressively increases, supporting analysis, verification, and operational activities throughout the system lifecycle.

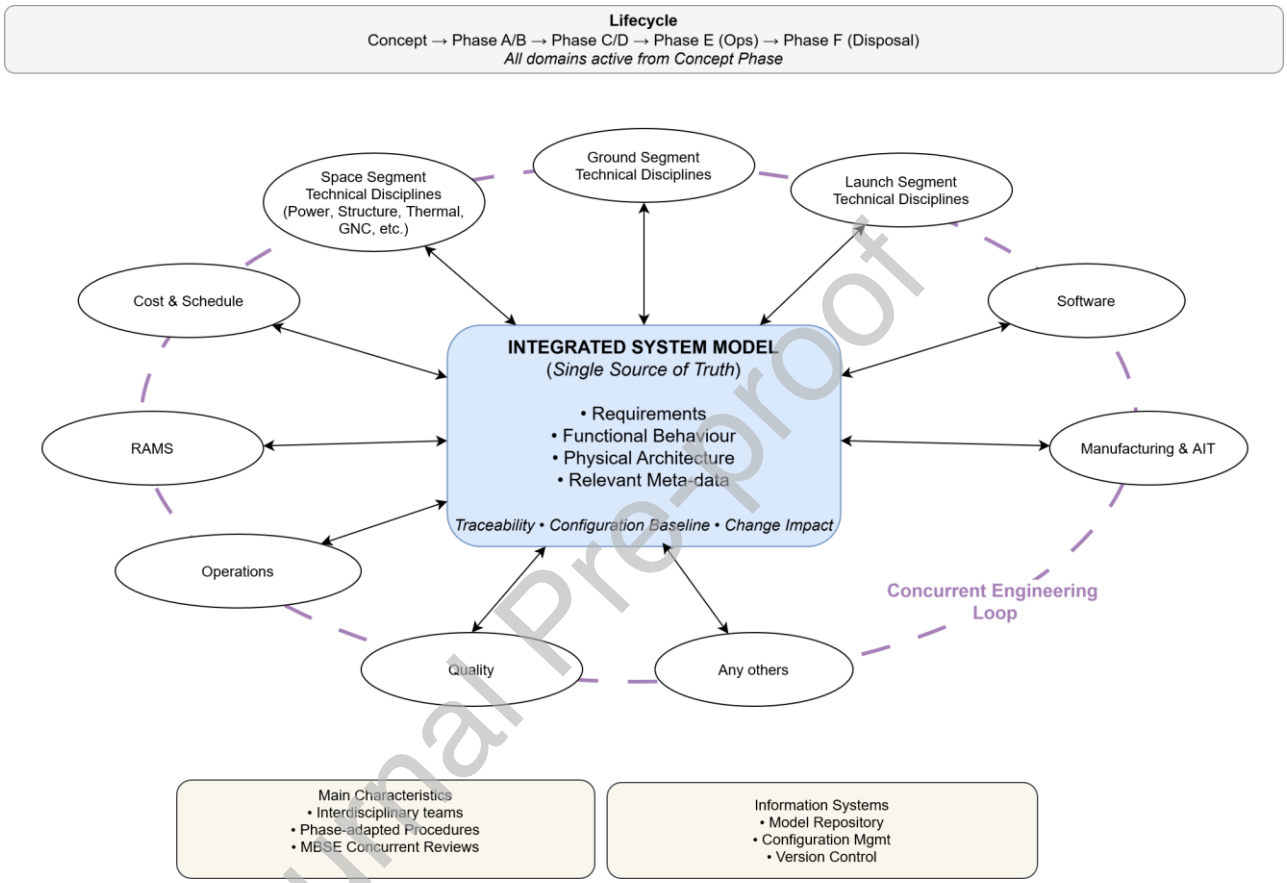


Figure 2 CE-MBSE methodology high level overview

3.2. Concurrent workflow adaptation

The concurrent workflow is adapted across lifecycle phases to account for increasing system maturity and verification needs. The main characteristics of the CE-MBSE approach are summarised in Table 2.

Phase	Objective	CE Adaptation	MBSE Role	Key Output
0/A	Mission definition and trade-offs	Standard CDF approach [4] with multidisciplinary concurrent sessions enabling rapid iterations and early design exploration	Capture mission context, stakeholder needs, and high-level system architecture [23], [24], [25]	Mission concept and preliminary architecture
B/C	Design consolidation	Integration of higher-fidelity models and simulation-driven	Ensure traceability and consistency across	Validated system architecture and

Journal Pre-proof

Phase	Objective	CE Adaptation	MBSE Role	Key Output
	and validation	verification [26] within concurrent sessions	requirements, functions, and system architecture	design baseline
D	Integration & testing	Test-driven concurrent activities integrating hardware and software in the loop	Update system models based on test data and support verification and validation planning	Verified system and test validation results
E	Operations & monitoring	Extension towards digital twin [27] enabling real-time monitoring and operational analysis	Maintain synchronisation between the system model and the operational system	Operational digital twin and mission performance data

Table 2 Concurrent workflow per phase(s)

As the system evolves, the methodology progressively shifts from exploratory design activities to verification-driven processes, with increasing model fidelity and tighter integration between system architecture and simulation environments.

The current work focuses on demonstrating the application of the proposed methodology to the early design and preliminary verification phases, with particular emphasis on Phases 0/A and the transition towards Phase B/C. Later lifecycle stages, such as Phase D integration and testing and Phase E operations, are included in the methodological framework to illustrate its intended lifecycle extension, but their implementation and validation are outside the scope of the present case study.

3.3. Model-Based Approach Integration

The model-based component of the proposed CE-MBSE methodology builds on the recursive and iterative principles of the ESA MBSE methodology whilst adopting ARCADIA as the architecture-development framework within the open-source Capella. These established approaches are not replaced; instead, they are integrated into the concurrent workflow to structure the progressive development of a space system. In particular, it is important to notice that, for the present work, ARCADIA Capella tool was preferred over a SysML-based implementation because the objective was to support a structured architecture development workflow directly usable during concurrent design. Its predefined progression from Operational Analysis to System and Logical Architecture reduces methodological tailoring and provides a consistent framework for functional decomposition and allocation. SysML would provide broader modelling expressiveness and interoperability potential but would currently require additional methodological configuration to reproduce the same architecture development workflow. With the recent evolution of SysML v2, the authors are currently exploring its application and implementation within the toolchain.

The resulting framework, as shown in Figure 3, encompasses: (i) the analysis of the stakeholders' needs as well as the definition of mission objectives; (ii) a structured functional analysis, with an increasing level of decomposition up to the unit level; (iii) the development of system architectures, with an increasing level of decomposition and a direct link to the functional analysis; (iv) a tailored virtual simulation environment; (v) a transversal requirements management for V&V activities.

This top-down approach evolves from the mission objectives to the detailed design of physical components through an iterative and recursive process. Starting from the definition of the mission context and stakeholder needs, system functions are progressively identified and decomposed. These functions are then allocated to system components, establishing a direct link between functional behaviour and physical architecture. This process ensures consistency across abstraction levels and supports the progressive development of the system architecture.

Journal Pre-proof

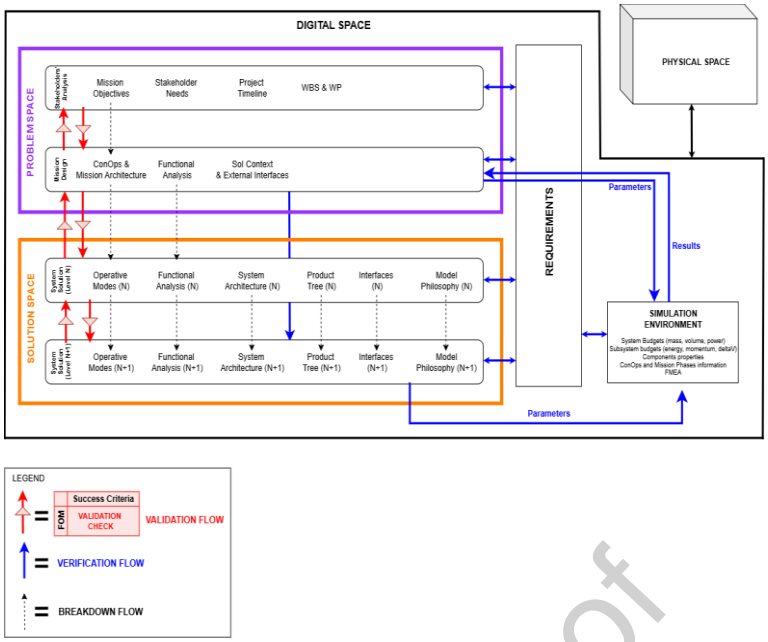


Figure 3 Model-Based approach integration with a focus on V&V

Requirements management is treated as a transversal activity throughout the modelling process. Requirements are directly linked to system elements, enabling traceability between mission objectives, functions, and components. This integration supports continuous verification by comparing system parameters with defined constraints, ensuring alignment between design solutions and stakeholder needs.

To complement the descriptive architectural model, simulation is integrated [28] within the MBSE framework as a predictive feedback mechanism supporting architecture development. In this context, simulation is not limited to computer-executable models, but more broadly refers to model-based reasoning used to anticipate system behaviour and assess the consequences of design decisions. As illustrated in Figure 4, this broader perspective allows simulation to be considered as a component of a control mechanism, guiding the development of system architectures that align with both system objectives and stakeholder expectations.

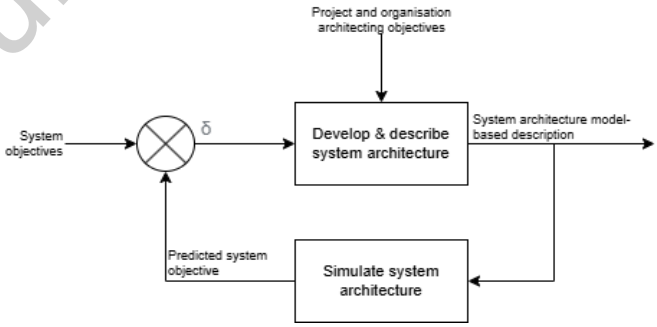


Figure 4 Simulation in architecture design process [29]

Executable and computer-assisted simulations provide quantitative evidence for assessing functional performance, dynamic interactions, and mission feasibility. By comparing analytical results with requirement thresholds and constraints, they support verification activities from the early design phases, when corrective actions are generally less costly.

Journal Pre-proof

Journal Pre-proof

3.4. Toolchain

The proposed CE-MBSE methodology is instantiated by a modular and interoperable toolchain that connects architectural modelling, engineering information management, and quantitative analysis through controlled information exchanges and mappings between corresponding engineering information classes. This integration supports digital continuity, traceability, and synchronisation across disciplines and lifecycle phases. Rather than relying on a single monolithic environment, the toolchain integrates complementary tools, each addressing a specific aspect of the systems engineering process, while preserving authoritative information management across the federated environment. In a concurrent environment, in particular, the purpose of tool integration is not only to exchange data among software platforms, but also to support rapid multidisciplinary interaction, exposing dependencies and possible inconsistencies, as well as propagating selected design changes without introducing excessive overhead. In particular, the toolchain shall support the study before, during, and after multidisciplinary sessions. Before a session, the architectural model and engineering repository provide a common baseline and make assumptions, requirements, and engineering parameters available to the participating disciplines. During the session, selected changes and analytical inputs can be propagated or compared across tools, supporting rapid impact assessment and the identification of inconsistencies. After the session, the resulting information is consolidated, aligned where supported by the interfaces, and released as an updated engineering baseline.

Therefore, as no single environment provides all the capabilities required for space mission and system design, the proposed implementation adopts a modular and federated architecture. Specialised tools remain responsible for discipline-specific activities, while controlled interfaces connect architectural information, requirements, engineering parameters, operating modes and analytical models. Within this broader framework, a primary integration is established between Eclipse Capella and Altium Requirements & System Portal (RSP) tools. During architectural definition and consolidation, Capella provides the authoritative architectural backbone and supports the shared representation of mission context, functions, components, interfaces, and operational behaviour. Additionally, the *ViVA (Verification & Validation Assessment*) viewpoint [30] extends this representation with requirements and verification-related information. Altium RSP complements the architectural environment by supporting engineering data, requirements management, trade-offs, and compliance-related information, while external simulation environments provide quantitative analyses and predictive feedback. The integration [31] is achieved through Python synchronisation scripts that leverage the respective APIs of the tools, *Python for Capella* and the RSP *Python API*, to facilitate the transfer of data. In particular, for the Capella-to-RSP direction, an intermediate Excel workbook (.xlsx) is generated from Capella and processed by local Python scripts, while communication with RSP is performed through HTTPS REST API requests using JSON payloads. The reverse exchange follows the corresponding workflow in the opposite direction: information is retrieved from RSP through the HTTPS REST API, converted into a compatible .xlsx format, and subsequently imported into Capella.

As shown in Figure 5, during architectural development and consolidation, the Capella-to-RSP synchronisation follows a two-step approach: an automated export of selected data from the Capella model, followed by its import into the Altium RSP environment. This process is performed throughout the ARCADIA levels, allowing the RSP engineering repository to evolve consistently with the progressive refinement of the Capella architecture. The synchronisation scripts support the exchange of four main information classes: requirements, components, including Actors and Entities, operating modes, and engineering properties. Requirements, Components, and operating

modes are represented in RSP as Requirements, Components, and Modelists, respectively, while technical metadata defined through the Property Value Management Tool (PVMT) in Capella are mapped to the corresponding Valis in RSP. To maintain the integrity of the system hierarchy, components and entities are exported from the earliest design levels, preserving their inherent structure, and specific "bridge" scripts verify the consistency between the two environments, ensuring that the Altium database automatically adapts to any modifications performed in Capella.

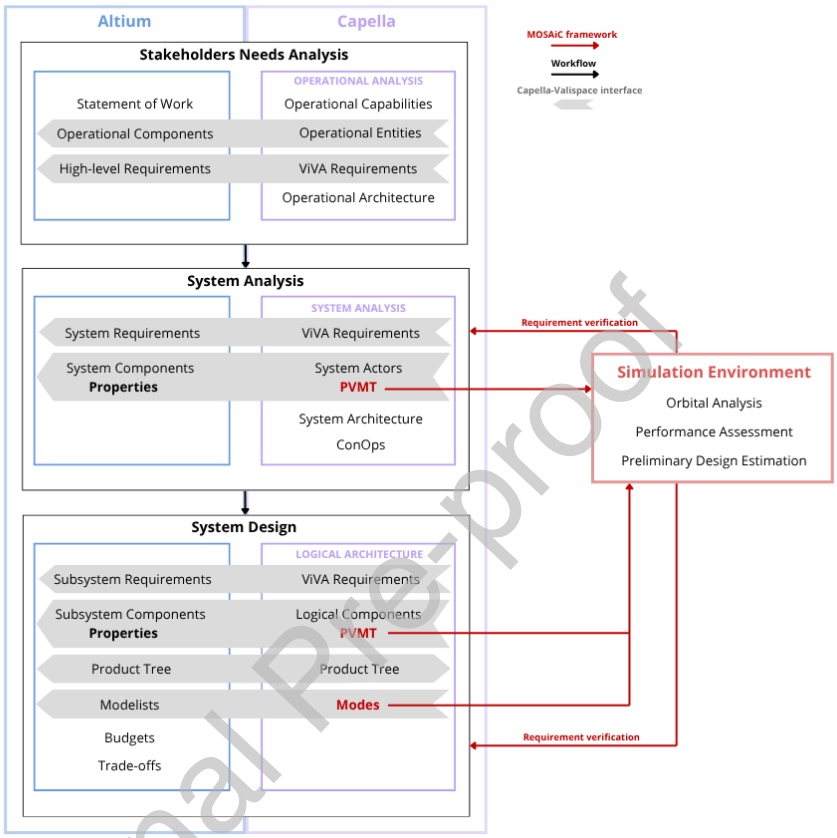


Figure 5 Toolchain

Once the architecture reaches a sufficiently consolidated state, however, the synchronisation direction can be configured according to the ownership of the information being managed. For each supported information class, including components, requirements, engineering properties, and operating modes, the user can designate either Capella or RSP as the authoritative environment. The synchronisation workflow then propagates the selected information from the designated source to the other environment, while consistency checks are used to identify discrepancies before updates are applied. This mechanism maintains a unique authoritative source for each information class while allowing information ownership to evolve according to the concurrent workflow.

A specific application of this information-management approach concerns the connection between architectural modelling and requirements management. To support the formulation and management of technical requirements according to the principle defined in ECSS-E-ST-10-06C [32], the dedicated viewpoint named *ViVA* was developed within Capella, providing a lightweight and immediate mechanism for deriving and structuring requirements in direct connection with the evolving system model. The viewpoint incorporates the main requirement characteristics defined

Journal Pre-proof

by the standard, including unique identification, requirement type, traceability, justification, responsible entity, verification information, and links to the relevant system elements. In particular, ECSS-E-ST-10-06C requires technical requirements to be identifiable, backward- and forward-traceable, unambiguous, singular, complete, and verifiable. These principles were used to define the attributes and templates implemented in ViVA, allowing requirements to be created while analysing stakeholder needs, system functions, architectural elements, interfaces, and operating conditions, thereby preserving the context from which they originate.

As shown in Figure 6, *ViVA* introduces predefined attributes and templates for defining requirements, helping users to reduce the time needed for documentation. These templates promote consistency in requirement formulation and allow requirements to be linked directly to Capella model elements across the different architectural levels. In this way, the viewpoint supports the initial derivation of requirements and preserves traceability between requirements, functions, components, interfaces, and operational scenarios.

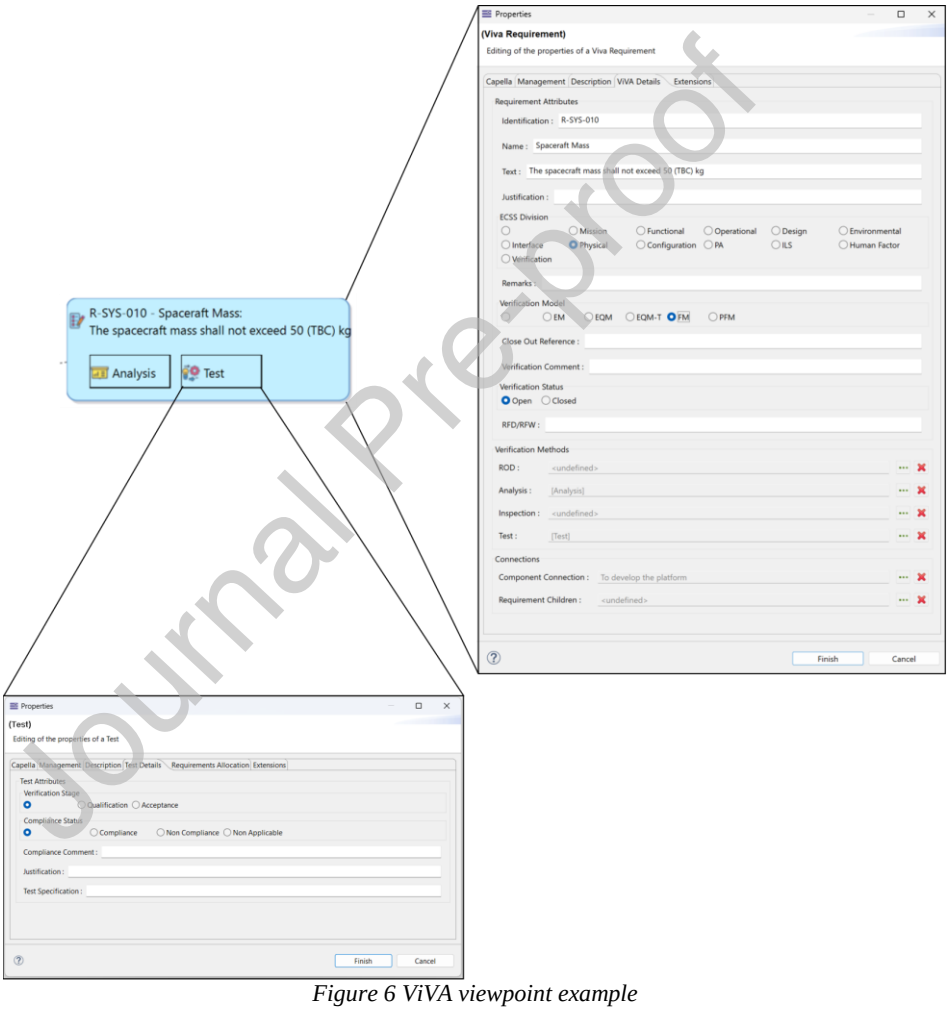


Figure 6 ViVA viewpoint example

ViVA is not intended to replace a dedicated requirement management environment. Its role is to maintain the semantic and traceability link between the requirements and the architectural model during concurrent design activities. Once requirements have been defined or refined within Capella, the Python bridge transfers them to Altium RSP, where formal requirements management is performed. RSP supports requirement hierarchy, versioning, ownership, verification status, engineering values, and the consolidation of compliance-related information.

The combined use of ViVA and RSP therefore separates two complementary responsibilities. Capella and ViVA support requirement derivation in the context of the architecture, whereas RSP provides the specialised environment required for their controlled management and verification. This division allows systems engineers and discipline specialists to formulate requirements while working on the system model without requiring Capella to reproduce the complete functionality of a dedicated requirements-management platform.

While the ViVA–RSP integration connects architectural modelling with requirements management, the other major extension of the toolchain concerns the connection between descriptive system models and quantitative analysis environments. The toolchain can interface with external simulation environments through the *MOSAiC* framework [28]. MOSAiC provides a model-to-simulation workflow in which quantitative properties, operating modes, mission conditions, and selected system configurations are extracted from Capella and transformed into inputs for specialised analysis modules. The integration process is facilitated by Python-based scripts that act as parsers, extracting technical parameters defined in Capella via the Property Value Management Tool (PVMT) and translating them into executable simulation inputs. This capability provides a model-to-simulation pathway in which quantitative outcomes generated by external analyses can be used as verification evidence against system requirements.

4. Results

4.1. Lunar missions as a representative case study

Space exploration beyond Earth orbit is currently being advanced through coordinated international strategies that identify the Moon and Mars as key destinations for future scientific investigation and human settlement capability development. The 2024 Global Exploration Roadmap [33], endorsed by space agencies worldwide, articulates a long-term vision in which short-term activities in lunar orbit and on lunar surface form the basis for progressively more complex exploration, including preparation for human missions to Mars and other deep space missions. NASA's Moon to Mars Architecture [34] defines an evolution of exploration objectives that extends from Earth orbit to deep space, where the Artemis lunar campaign is explicitly designed to advance scientific investigation, mature critical technology and develop the operational experience needed for Martian missions. Similarly, ESA's exploration roadmap [35] emphasises lunar activities as part of a cohesive strategy linking Earth orbit, the Moon, and Mars, ensuring the development of capabilities, scientific understanding and international cooperation that support broader exploration goals.

The progressive and architecture-driven nature of these exploration strategies significantly increases the complexity of mission development, due to the strong coupling between mission objectives, enabling technologies, operational concepts, and international collaboration across multiple phases. The need to ensure consistency, evolvability and interoperability from short-term lunar missions to long-term Martian exploration challenges traditional design approaches. In this context, the proposed CE-MBSE approach becomes a necessary enabler to support the parallel development of multidisciplinary system elements while maintaining coherence across requirements, functions, interfaces, and operational scenarios. Therefore, after implementing the methodology during several missions around Earth [21], [22], [23], [29], the CE-MBSE methodology has additionally been applied to a lunar CubeSat constellation, disclosing the full potential of this approach and identifying further crucial aspects to be integrated into the methodology.

Journal Pre-proof

The *BioMoon* case study was selected because it combines multiple stakeholders, mission segments, operational scenarios, multidisciplinary engineering data, and verification needs. The application was therefore used to assess whether the proposed approach could support: (i) progressive traceability across abstraction levels; (ii) consistency between architectural and engineering repositories; and (iii) the connection between system parameters and requirement assessment; and (iv) change propagation and baseline restoration during concurrent design iterations.

4.2. BioMoon Case Study

BioMoon is a lunar orbital exploration mission aimed at characterising the lunar radiation environment and assessing its effects on biological samples under representative deep-space conditions. The scientific focus is twofold: (i) the measurement and characterisation of the radiation environment in the vicinity of the Moon, (ii) the investigation of the survival and active metabolism of selected biological specimens exposed to this environment. These objectives are addressed through a payload suite composed of a radiation detector and an astrobiological experiment.

A snapshot of the mission concept is depicted in Figure 7 to complement its mission architecture reported in Table 3.

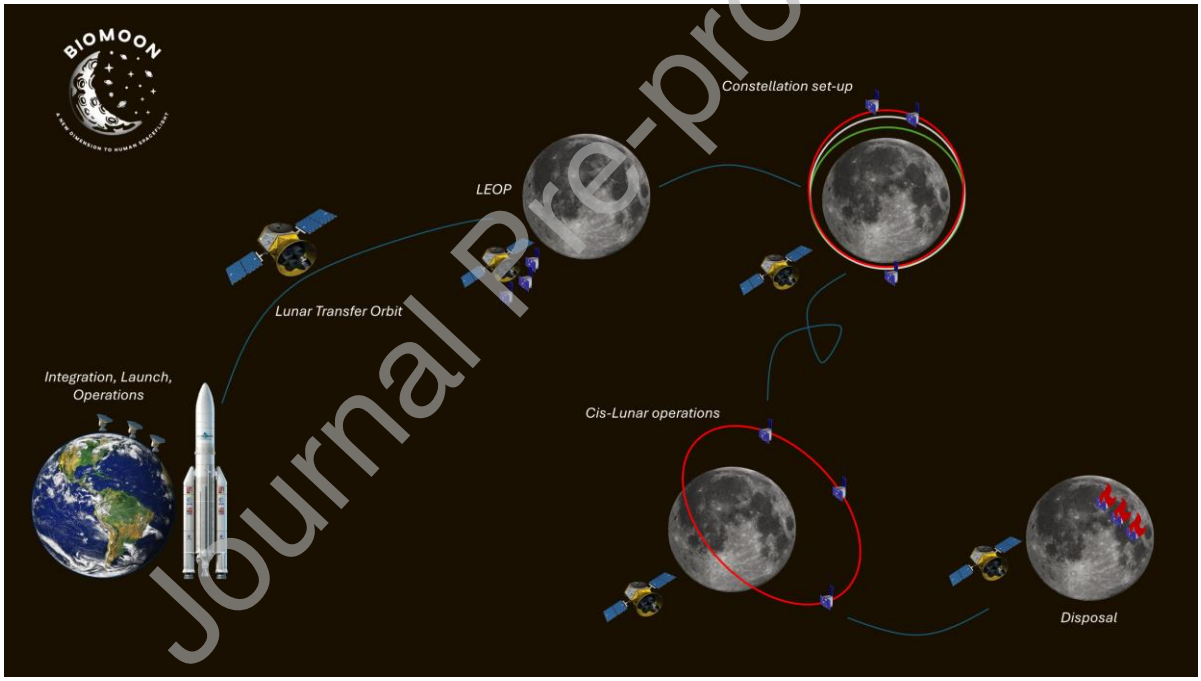


Figure 7 Mission Design Reference

The space segment consists of a constellation of three 12U CubeSats operating in a polar lunar orbit. Following the injection into a circular polar orbit at approximately 500 km altitude, the mission transitions to an operational elliptical polar orbit with a peri-selenium of 150 km and an apo-selenium of 600 km, enabling exposure to varying radiation conditions. At the end of life, a disposal phase is foreseen through a highly elliptical orbit, ensuring compliance with planetary protection and debris mitigation guidelines [36].

Mission element	***Description of baseline***

Journal Pre-proof

Mission element	*Description of baseline*
Subject	Radiation effect on biological specimen Lunar radiation environment
Space Segment	3 CubeSats (12U form factor)
Payload	● Astrobiological experiment ● Radiation detector
Orbit & Constellation	Polar Lunar Orbit @500km (injection) Elliptical Polar Orbit @150km x 600km (operational) Elliptical Polar Orbit @10km peri-selenium (disposal)
Communication Architecture	Store and Forward architecture CubeSat-Orbiter Interlink (UHF)
Ground Segment	ESTRACK compatibility
Mission control centers and operations	ESA MCC On board data processing
Launch Segment	Orbiter TBD Launch Vehicle TBD

Table 3 Mission Architecture

The mission adopts a store-and-forward communication architecture, with CubeSat-to-orbiter interlinks in UHF and downlink to Earth via an ESA-compatible ground segment. Ground operations are supported by ESA infrastructure, with ESTRACK compatibility and centralised mission control through an ESA Mission Control Centre. Science operations are coordinated through a dedicated Science Operations Centre, enabling interaction with academia and research institutions involved in data exploitation and scientific analysis.

From an operational perspective, the mission supports multiple stakeholders, including ESA directorates, space industry partners, and academic and research institutions. The primary mission objectives include the characterisation of the lunar radiation environment and the study of biological response to prolonged radiation exposure. On-board data processing capabilities are implemented to support autonomous operations, payload management, and efficient data handling under limited communication opportunities.

4.3. Application of the methodology

The CE-MBSE methodology has been applied to the BioMoon case study through a series of multidisciplinary concurrent design iterations, while the system model was progressively refined across multiple levels of abstraction. Starting from stakeholder needs and mission objectives, the modelling process evolved towards system architecture and subsystem design, preserving explicit relationships among the information represented at each level. These relationships were then exploited during the concurrent workflow to identify and assess the changes' impact across requirements, architectural elements, operating modes, engineering parameters, and verification activities.

The case study shows the progressive development of the model, but also how it supports multidisciplinary change propagation and restoration of a consistent engineering baseline during concurrent design. In particular, the introduction of an active Thermal Control Subsystem was

selected as a representative design change because it affects several engineering domains simultaneously, including system architecture, product structure, power and energy budgets, operating modes, requirements, and compliance assessment.

The following subsections first describe the progressive model refinement required to establish the dependency structure among system elements and then show how this structure was used to support cross-disciplinary data exchange, consistency assessment and verification within the concurrent workflow.

4.3.1 Concurrent Study Organisation

The *BioMoon* case study was developed through a series of concurrent design sessions involving systems engineering and discipline-specific activities performed in parallel. The workflow followed a general structure, as also adopted in previous concurrent studies [23][24],[25], with an initial kick-off and mission definition stage, followed by mission and system design iterations and a final baseline consolidation. During the sessions, the System Engineers coordinated the evolution of the shared system model, while discipline experts progressively refined subsystem design, including assumptions, requirements, engineering parameters and analysis.

The concurrent sessions were organised around recurring design loops rather than isolated discipline-specific activities. Within each iteration, discipline experts worked in parallel on the affected engineering aspects, while the Systems Engineer coordinated the identification of cross-domain impacts and the progressive consolidation of the shared baseline. Model updates, engineering assessments, and requirement checks were therefore not treated as sequential handovers, but as coupled activities iterated until a consistent multidisciplinary solution was reached. For instance, architectural decisions, engineering budgets, and requirements were progressively reviewed as new information became available, and design changes were assessed with respect to their impact across the affected domains. The integrated Capella-RSP environment was used to support this process by providing a common architectural reference and controlled propagation of information. The introduction of the active Thermal Control Subsystem was used as a representative multidisciplinary change within this workflow. Its effects were assessed across the architectural model, product structure, operating modes, engineering properties, quantitative budgets, and requirement compliance before consolidating the updated engineering baseline.

Table 4 Concurrent workflow stages and supporting CE-MBSE activities

Concurrent stage	Main activities	CE-MBSE support
Kick-off / Mission definition	Stakeholder alignment, mission objectives, initial ConOps and assumptions	Operational model and initial requirements provide a shared starting baseline
Mission design	Parallel assessment of mission architecture, modes, interfaces and constraints	System Analysis and shared engineering data support cross-domain coordination
System design	Concurrent refinement of subsystem architectures, sizing and budgets	Logical model, PVMT properties and RSP data support parallel engineering assessment
Baseline consolidation	Cross-check of architecture, requirements, budgets and compliance	Bridge-supported consistency checks, requirement reassessment and baseline restoration

Journal Pre-proof

The following subsections describe how this concurrent iteration was supported by the CE-MBSE framework, from the identification of affected model elements to the propagation of engineering data, reassessment of quantitative analyses, and consolidation of the updated baseline.

4.3.2 Progressive Model Refinement and Traceability

The progressive model refinement provides the dependency structure required to support further iteration work, including change impact assessment, during a concurrent study.

The process begins with a structured operational analysis, where stakeholder needs and mission drivers are formalised. Main stakeholders were identified, representing institutional, industrial, and scientific actors. Their needs were translated into high-level mission objectives related to radiation characterisation and biological experimentation. Figure 8 captures the initial problem space by linking the identified stakeholders to the mission capabilities required to satisfy the scientific objective.

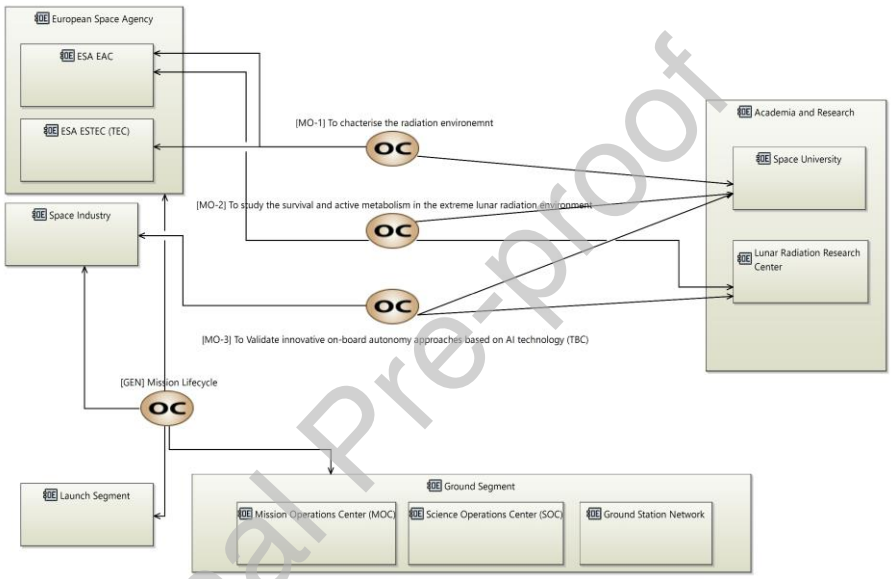


Figure 8 Operational Capabilities Blank (OCB) Diagram

The resulting operational models enable the identification of mission capabilities, stakeholder interactions, and operational processes, providing a structured representation of the mission context. Indeed, Figure 9 provides a representative view of the operational architecture, showing how the identified actors and mission segments participate in the main operational activities and exchanges required to accomplish the mission objectives. These interactions make explicit the dependencies within the mission context before introducing the System of Interest (SoI) as a design solution, and provide the basis for identifying the externally visible system functions and interfaces.

Journal Pre-proof

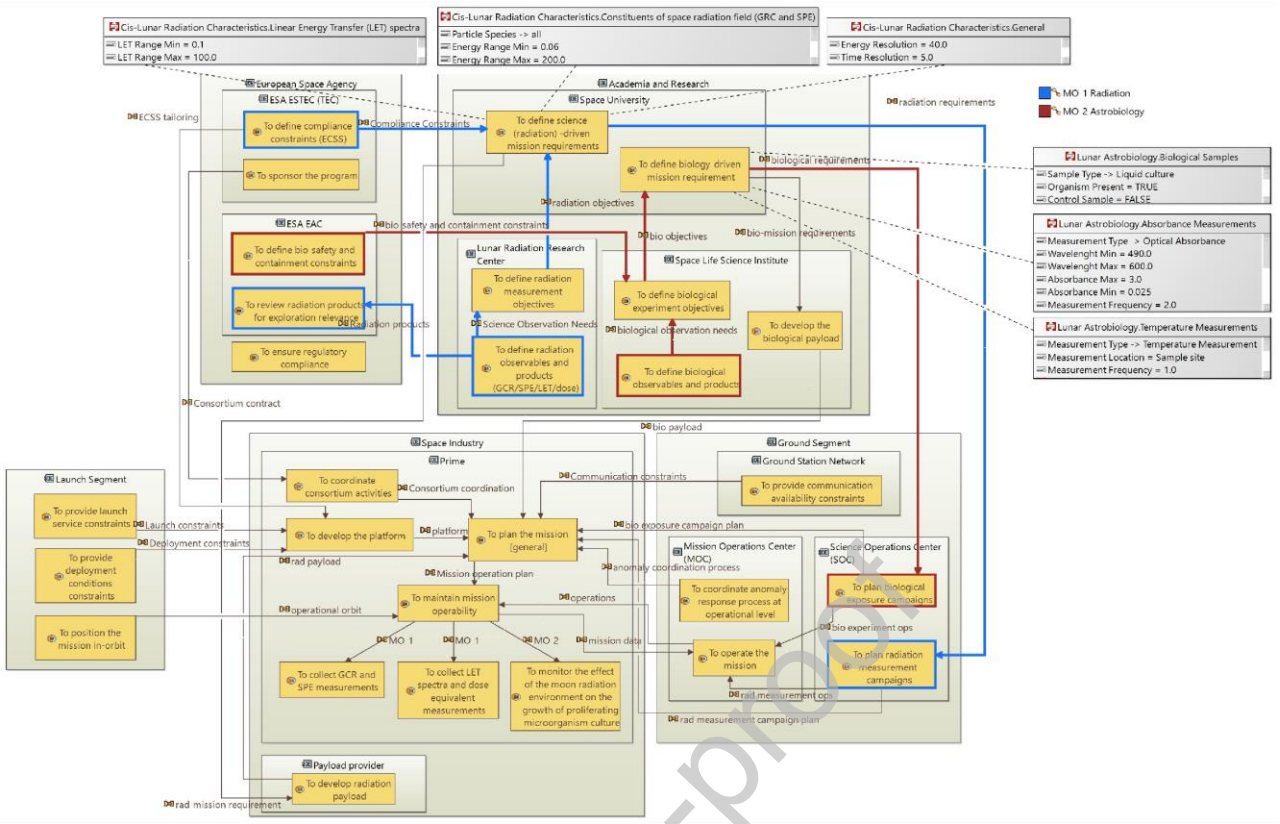


Figure 9 Operational Architecture Diagram (OAB)

The Concept of Operations (ConOps) was then defined at the System Analysis level, where the focus shifts to the SoI, initially treated as a black box. Figure 10 therefore translates the operational scenarios identified at the previous level into the main mission phases and operating conditions, providing the behavioural context for the subsequent definition of system functions and interfaces.



Figure 10 Modes & State diagram (MSM) - Concept of Operation

At this stage, the objective is to define system boundaries, external interfaces, and high-level functions required to satisfy stakeholder needs. System Actors are propagated from the operational level, ensuring continuity and traceability within the model. A progressive functional decomposition is then performed, where high-level functions are refined into lower-level sub-functions and allocated to the SoI or to external actors. Figure 11 makes explicit the functional dependencies and interfaces between the System of Interest and external actors, supporting a consistent and traceable system definition, later used to identify the elements affected by design changes.

Journal Pre-proof

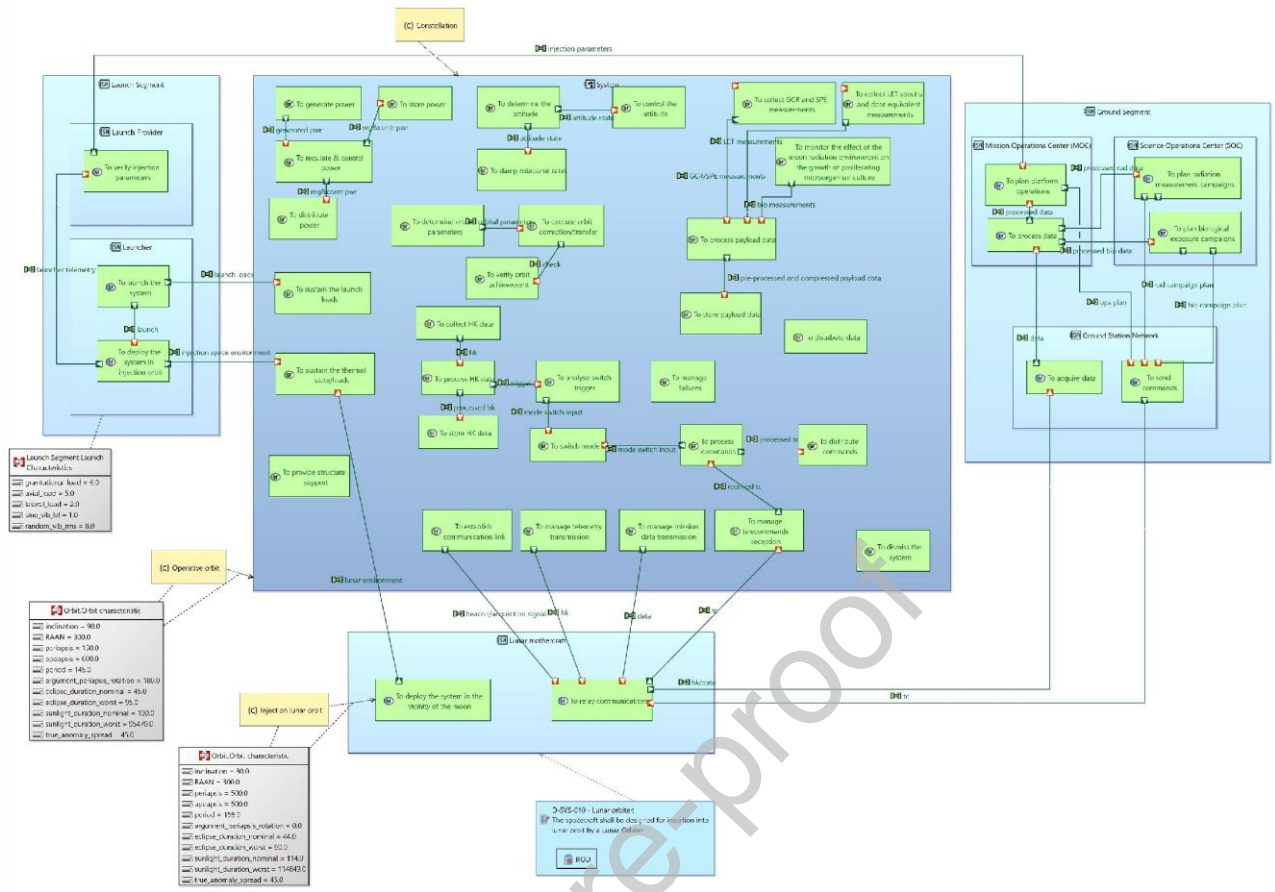


Figure 11 System Architecture Blank (SAB) diagram

At this level, metadata and constraints are associated with both the SoI and system elements, including orbital parameters and launcher-related constraints. This ensures alignment between system architecture and mission-level drivers. In parallel, requirements, components, and associated data are synchronised between the modelling environment and the engineering database, enabling continuous refinement of the product and requirement structures.

Following the definition of high-level functions and preliminary architecture, the development progresses towards detailed design at subsystem level. System functions are further decomposed and allocated to individual components, leading to a refined logical architecture where each element is explicitly linked to the functions it performs. This enables a granular and consistent mapping between functional behaviour and system architecture, supporting traceability and design coherence. Figure 12 shows that functions are allocated to logical components and further dependencies are identified, enabling the impact of a change to be traced from functionality to subsystem-level architecture and associated engineering data and requirements.

Journal Pre-proof

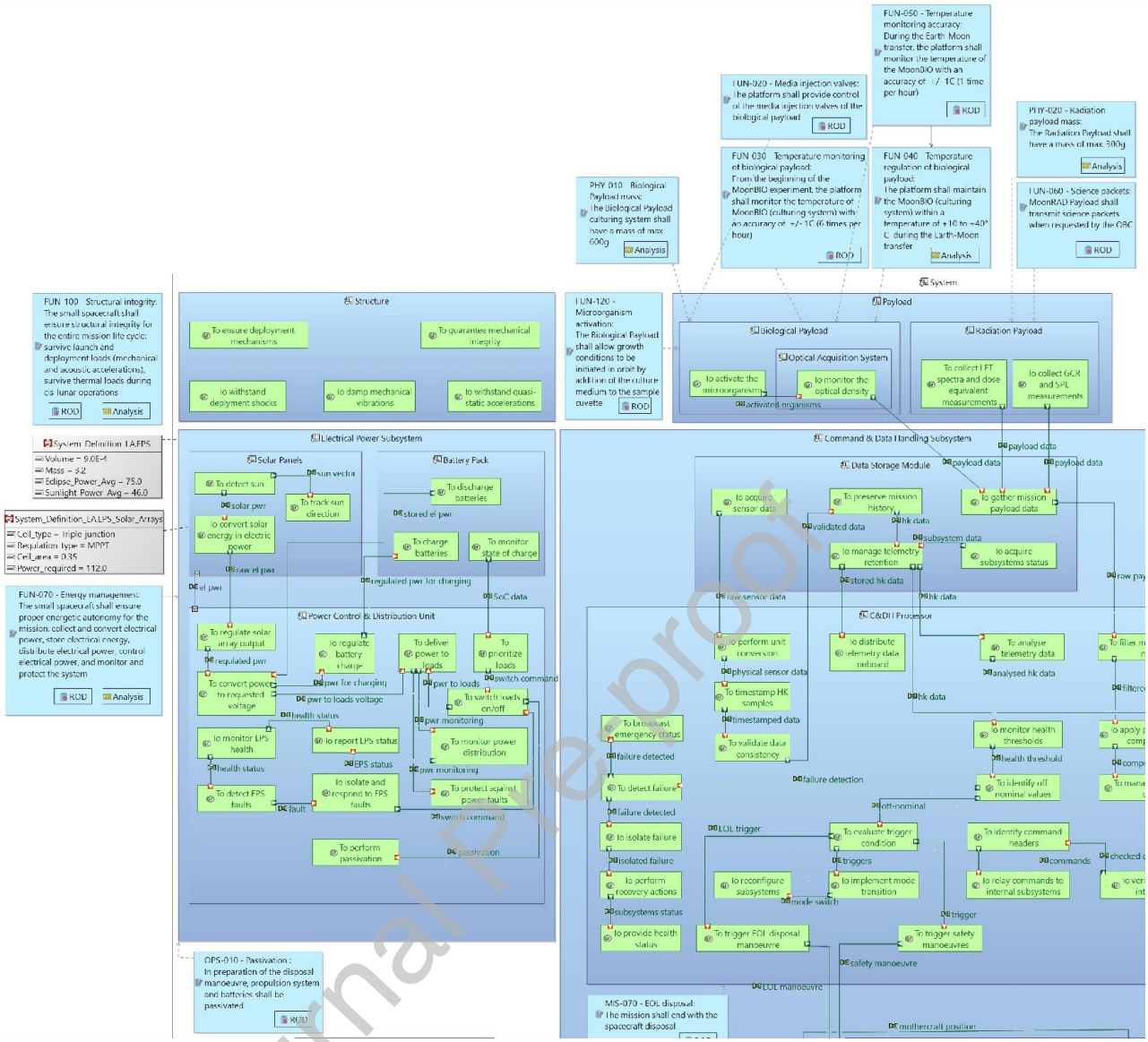


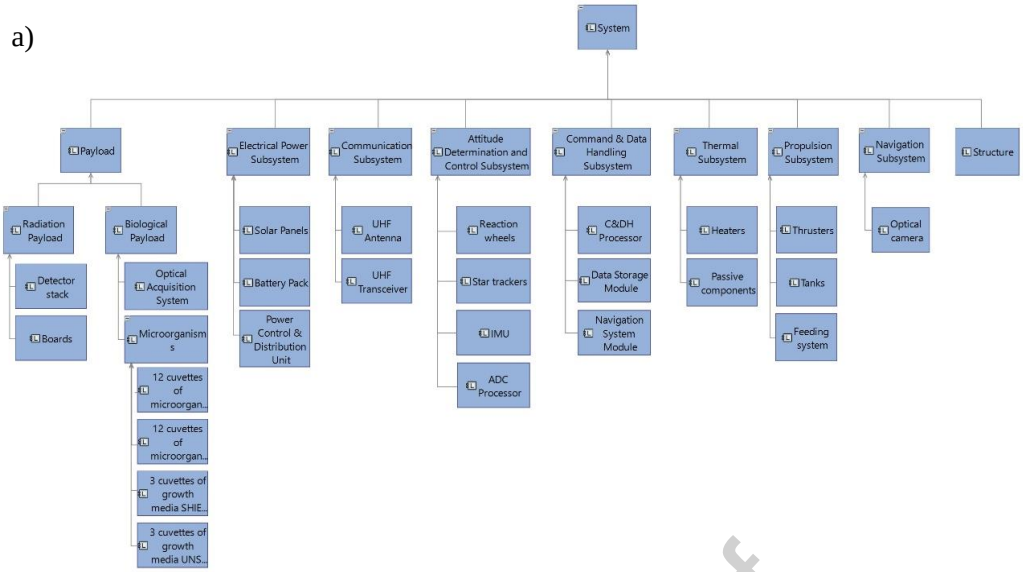
Figure 12 Logical Architecture Blank (LAB) diagram – EPS and Payload

Overall, the modelling sequence established continuous information from the problem space to the solution space. This traceability structure provided the architectural basis for the subsequent exchange and management of requirements, engineering properties and product information across the toolchain. For instance, explicit relationships provide the basis for identifying which functions, components, requirements, interfaces, and engineering parameters may be affected when a design change is introduced during a concurrent session.

4.3.3 Consistency during Architectural Change

To preserve consistency across tools, the updated product hierarchy and associated requirements were synchronised with the Altium RSP. A dedicated Python "bridge" script was developed to interface Capella and Altium simultaneously and verify alignment of the Product Tree (Figure 13).

a)



b)



Journal Pre-proof

Figure 13 Product Tree view in a) Capella and b) Altium RSP

The script automatically detects structural inconsistencies and updates the RSP component tree when discrepancies occur, restoring conformity between platforms. The execution workflow of the bridge script is structured into three sequential phases: authentication, data comparison, and automated synchronization. Upon launch, the bridge script initializes a connection to the RSP platform. As shown in Listing 1, the script prompts the user to enter the specific RSP project identifier (e.g., Project ID: 30) and to select the system architecture file exported from Capella.

```
1        Authenticating Valispace...
2
3        You are currently using the provided Token
4        Connected to https://...
5        Enter Valispace Project ID: 30
6        Select Capella Excel export file...
```

Listing 1 Authentication and project connection

Once the Capella export is loaded, the script processes the file to extract component names and parent-child relationships. Then, the script retrieves the current product tree from the RSP tool and performs a comparison against the Capella structure. The algorithm evaluates three distinct types of discrepancies: missing components in the RSP, hierarchy mismatches and RSP components that are no longer part of the Capella architecture.

The script then compiles a pre-execution synchronization report. For the purpose of the study, and as part of the assessment process, later described in Section 4.4.1, a relevant design change was introduced, namely the choice of an active Thermal Control Subsystem (TCS). Therefore, as illustrated in Listing 2, during the evaluated reference test run, the script analysed 56 components from Capella and verified that 55 were already present in the RSP portal. It isolated exactly one new component requiring creation (e.g., Heaters), belonging to the parent *Thermal_Subsystem*. To prevent unauthorized or unintended database modifications, execution proceeds only after explicit user confirmation.

```
1      ============================================================
2      SYNCHRONIZATION REPORT
3      ============================================================
4      Capella Components: 56
5      Components already present and valid: 55
6
7      TO BE CREATED IN VALISPACE (1):
8       + 'Heaters' (Parent: 'Thermal_Subsystem')
9
10     PARENTS TO BE UPDATED (0):
11      (None)
12
13     COMPONENTS PRESENT ONLY IN VALISPACE (0):
14      (None)
15
16     Proceed with execution? (y/n): y
17     Created: 'Heaters'
18
19     Sync completed!
20
21     Created (1):
22      • Heaters (Parent: Thermal_Subsystem)
23
24     Reparented (0):
25      None
```

Listing 2 Synchronization report generated during execution.

Journal Pre-proof

In the selected TCS scenario, the bridge therefore supports the architectural propagation step of the concurrent iteration by aligning the updated product structure across Capella and RSP.

4.3.4 Operative Modes and Engineering Data Exchange

Architectural changes may also affect operational assumptions and quantitative engineering assessments, which must therefore be reconsidered within the same concurrent iteration. Although the following activities are described sequentially for clarity, they are coupled within the concurrent workflow: architectural updates, operating modes, engineering parameters, budgets, and requirement assessments may be iterated in parallel as new information becomes available from the involved disciplines.

A key modelling feature exploited in Capella was the Modes and States Machine (MSM), used to define system operational modes and their transition triggers, as per Figure 14.

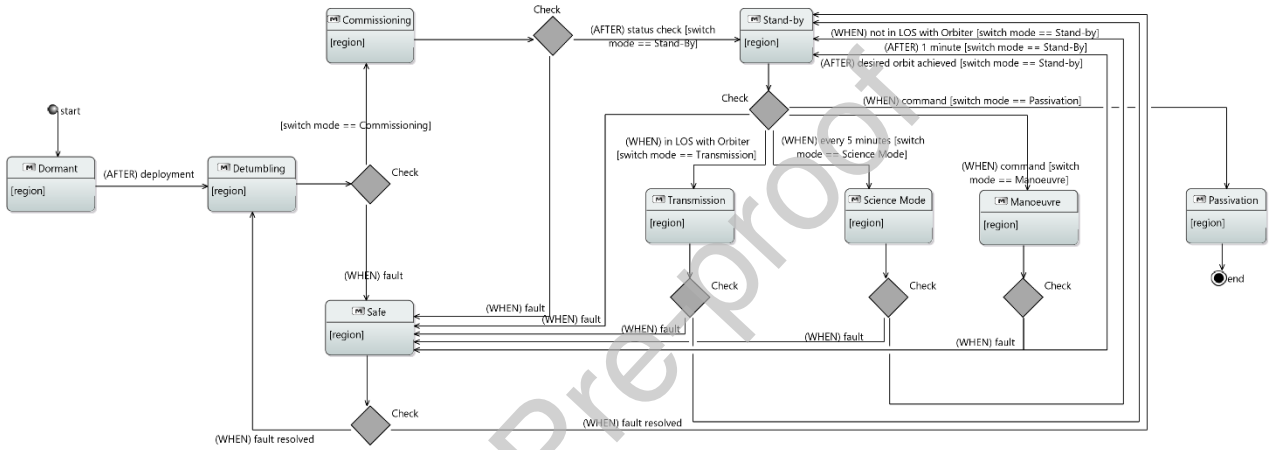


Figure 14 Operative Modes in Capella

These modes were exported to Altium as Modelists and assigned to the System component, as shown in Figure 15. This integration enables the use of Altium's integrated budget estimation capabilities, making it possible to perform dynamic, mode-dependent budget calculations.

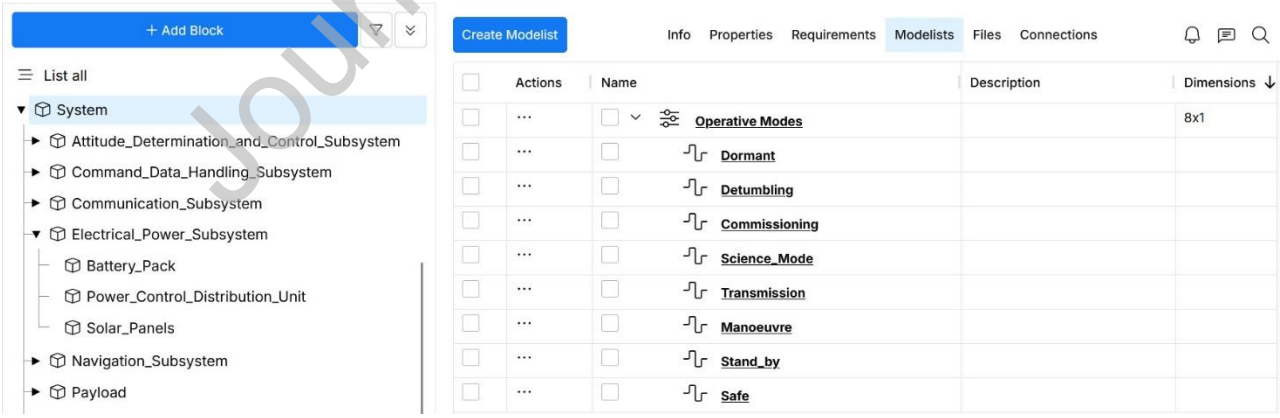


Figure 15 Operative Modes in Altium RSP

Furthermore, through the synchronisation of PVMT properties defined in Capella, a consistent dataset of component-level engineering parameters (e.g., mass, power, operational constraints) was maintained across tools.

Journal Pre-proof

As shown in Figure 16, parameters allocated in Capella to the Electrical Power Subsystem, such as mass, volume and eclipse/sunlight required power, were imported directly into the RSP as Properties. Similarly, mechanical and launch environmental constraints assigned to the Launcher component were synchronized into the platform.

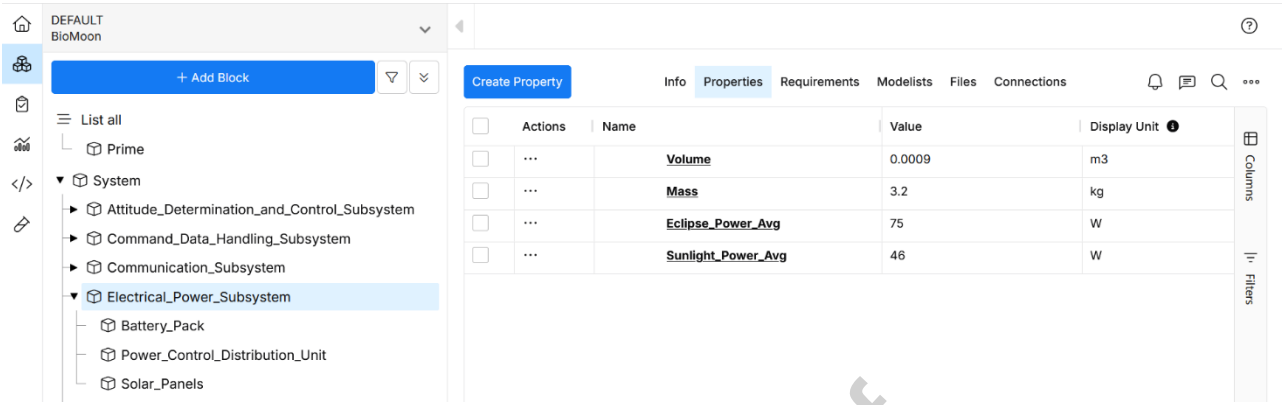


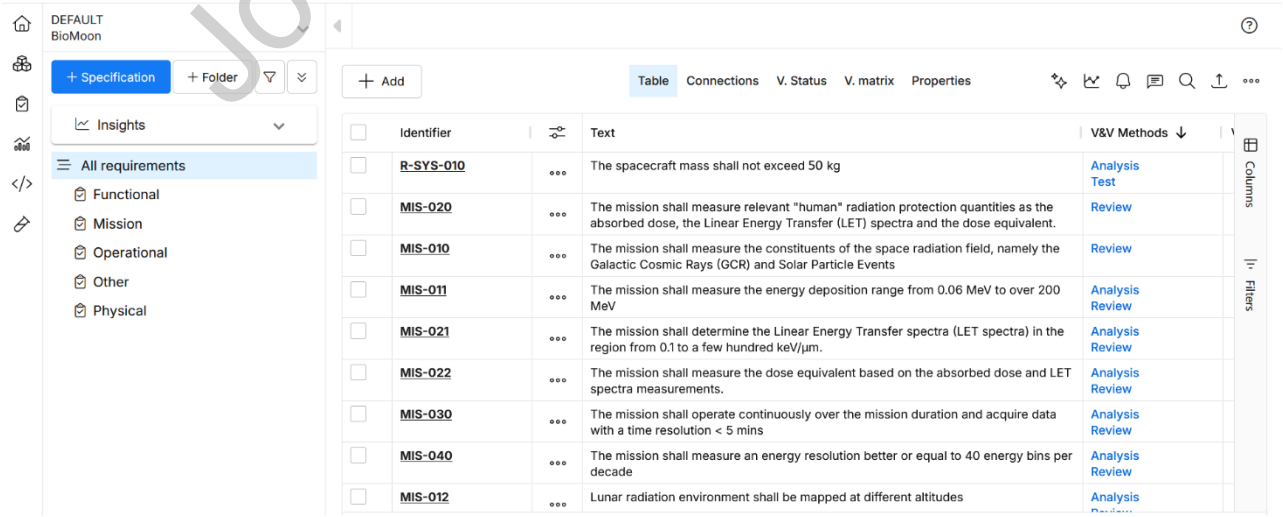
Figure 16 EPS Properties

This combined mapping of operational modes and PVMT attributes supports controlled alignment of the engineering information exchanged between the architectural and engineering environments, helping to keep system budgets consistent with the Capella architectural baseline. In this way, discipline-specific analyses can be updated in parallel while remaining consistent with the evolving system definition, reducing the need for repeated manual reconciliation between domains.

4.3.5 Verification & Validation

Following the propagation of the updated architectural and engineering information, the concurrent workflow proceeds with the reassessment of the affected analyses and requirements.

Verification & Validation (V&V) are integrated as a continuous process throughout the modelling workflow. Requirements defined through the *ViVA* viewpoint in Capella are directly associated with the corresponding model elements across the different abstraction levels, preserving their architectural context and supporting traceability from stakeholder needs and mission objectives to system functions and components, as shown in Figure 11 and Figure 12. The requirements are then transferred to Altium RSP, where their attributes and associated verification information can be formally managed, as shown in Figure 17.



Journal Pre-proof

Figure 17 Requirements in Altium

The connection between requirements and engineering data enables verification activities to be progressively performed as the system definition evolves. In the *BioMoon* case study, quantitative engineering assessments including power, energy, and link budgets were implemented directly within the RSP Scripting environment using Python-based analytical models, as illustrated in Figure 18. The input parameters used by these analyses are derived from the engineering properties associated with the corresponding architectural elements and transferred through the model-based workflow.

The resulting analytical outputs can subsequently be compared with the applicable requirement thresholds and design constraints, providing quantitative evidence to support compliance assessment. In this way, the implemented workflow establishes a traceable information path from the requirement and its associated architectural element to the engineering parameters used for the analysis and, finally, to the corresponding verification result. This mechanism is particularly relevant when a design change occurs, since the affected requirements, architectural elements, and engineering parameters can be identified and reassessed within the same engineering workflow.



Figure 18 Power & Energy budget script snapshot

As a representative example, the Power & Energy budget analytical model was used to verify the requirement related to spacecraft's power generation adequacy, after the introduction of the new active thermal control subsystem. The requirement is linked to the relevant system elements and engineering parameters, while the budget calculation provides the quantitative result used to assess compliance, as shown in Figure 19.

Identifier	Title	Text	Rationale	Type	V&V M...	V&V Activities	V&V Status	Applicable Bl...
EPS-010	Solar Panel Power...	The solar arrays shall generate an average power of at least 120.00 watts	Ensure a positive energy balance during all nominal mission phases - design choice	Performance	Analysis	Power & Energy	Verified 1/1	Solar_Panels

Figure 19 Example of requirement verification through the Power & Energy budget activity in RSP

4.4. Key results: Benefits and Limitations

Journal Pre-proof

The application of the CE-MBSE methodology to the *BioMoon* case study provided both qualitative and quantitative evidence regarding its contribution to space mission and system design. The results highlight methodological strengths, though some practical limitations still exist, which are discussed below.

4.4.1 Workflow Efficiency Assessment

To provide a quantitative indication of workflow efficiency, the *Baseline Restoration Time* (BRT) was introduced. The metric represents the cumulative execution time associated with the activities required to restore a consistent engineering baseline following a design change, considering the most relevant affected information and engineering environments.

In the present assessment, BRT is evaluated as cumulative activity time rather than uninterrupted calendar elapsed time. This choice reflects the structure of the available historical project records, which provide execution times for individual engineering and consolidation activities. Thus,

$$BRT = \sum_{i=1}^{n} t_i$$

where t_i represents the execution time associated with the i-th baseline restoration activity.

The relative improvement associated with the CE-MBSE workflow is calculated as

$$R_{BRT} = \frac{BRT_{reference} - BRT_{CE-MBSE}}{BRT_{reference}} 100$$

where $BRT_{reference}$ represents the baseline restoration time observed for the conventional workflow and $BRT_{CE-MBSE}$ the corresponding value obtained using the proposed methodology. R_{BRT} therefore expresses the relative reduction in information propagation and reconciliation time, excluding the time required to make the engineering decision itself.

The reference workflow was defined as a conventional sequential Systems Engineering process, consistent with V-model practice. To ensure a meaningful comparison, the reference workflow was not assumed to be tool free. It represents a conventional tool-supported Systems Engineering process in which architecture, requirements, engineering parameters, and product information are managed through specialised environments but remain coordinated through separate documents and project-level configuration activities. The CE-MBSE workflow addresses the same engineering activities while introducing an integrated architectural backbone, explicit model relationships, and controlled cross-tool information exchanges.

Activity	**Reference Workflow**	**CE-MBSE Workflow**
Design change introduction	Change formally introduced through project change/configuration process	Change introduced in the shared system model
Impact identification	Affected artefacts and disciplines identified through project coordination and review of separate sources	Impacted functions, components, requirements and parameters identified through explicit model relationships
Engineering impact assessment	Mass, power, energy and other relevant budgets updated in discipline-specific tools/repositories	Engineering properties propagated from the model to RSP and assessed through integrated budgets even though expert-driven discussion remains

Journal Pre-proof

Activity	Reference Workflow	CE-MBSE Workflow
Operational impact assessment	Operative modes and operational constraints reviewed and updated in dedicated artefacts/tools	Operative modes updated in Capella and propagated to RSP as Modelists even though engineering assessment and discussion remain expert-driven
Requirements impact assessment	Affected requirements identified and updated in the requirements management environment	Model links and *ViVA* support identification and allocation of affected requirements, which are formally managed in RSP
Architecture and product tree update	Architecture and related design documentation updated in the corresponding engineering environment	Architecture is updated in Capella and the corresponding product structure is aligned with RSP through the dedicated bridge
Compliance assessment	Updated budgets and requirements are reviewed to determine whether the modified design remains compliant	Updated engineering values and requirement thresholds are assessed using the integrated engineering workflow
Consistency check and baseline consolidation	Updated artefacts are cross-checked and consolidated through the project configuration management process	Model relationships and bridge-supported checks support consistency assessment before the selected information is released as the updated engineering baseline

Table 5 Engineering activities performed when a relevant design change is introduced

For both reference and CE-MBSE workflows, the start of the measurement was defined as the introduction of a design change, namely the adoption of an active Thermal Control Subsystem (TCS), whereas the end condition corresponded to the establishment of a new consistent engineering baseline across all affected artefacts. A baseline was considered restored when all the elements affected by the selected change had been updated, no known inconsistency remained among the considered environments, and the updated information was available for subsequent engineering activities.

The reference execution times were derived from historical data collected from a previous lunar space project developed by the authors and their research team according to traditional approaches. For activities supported by historical records, the duration was taken from the documented execution time associated with the corresponding engineering or consolidation activity. Where sufficiently detailed timing information was not available, the duration was retrospectively estimated from the available project documentation and the engineering experience of the team. The CE-MBSE execution times were instead directly recorded for each corresponding baseline-restoration activity during the BioMoon TCS change scenario. In the previous lunar project, the selection of an active TCS similarly triggered a multidisciplinary reassessment. This same type of design change was therefore selected for the BioMoon case study to provide a more comparable basis for evaluating the CE-MBSE workflow.

Although the two projects differ in their specific mission configuration, both workflows were assessed using the same categories of baseline-restoration activities reported in Table 5 and equivalent completion criteria. BRT was evaluated as cumulative engineering execution time, excluding waiting time between activities.

Change type	Reference BRT [min]	Baseline source	CE-MBSE BRT [min]	BRT reduction [%]
Impact identification	30	*Estimated* – based on previous project experience and available documentation	20	33.3
Engineering impact assessment	120	*Historical records* – time needed to perform new budget iteration together with inputs from relevant specific discipline experts	90	25.0
Operational impact assessment	45	*Historical records* – time needed to the Systems Engineer to iterate with Power expert and Operations Team	30	33.3
Requirements impact assessment	30	*Estimated* – based on comparable requirement management activities	15	50.0
Architecture & Product tree update	45	*Historical records* – time needed to update the architecture in draw.io and modify the relevant system documents where it was reported	10	77.8
Compliance assessment	15	*Estimated* – based on previous projects experience concerning compliance assessment activities	10	33.3
Baseline review and consolidation	120	*Historical records* – time needed to the Systems Engineer to review all the final documents and cross check information for consistency	70	41.7
-	405	-	245	39.5%

Table 6 Baseline Restoration Time assessment for the introduction of a new subsystem

As shown in Table 6, for a selected scenario where the TCS was added during the *BioMoon* design phase, the cumulative Baseline Restoration Time decreased from 405[1] min for the reference workflow to 245 min for the CE-MBSE-supported workflow, corresponding to a nominal reduction of 39.5% for the selected scenario. The reduction was not uniformly distributed across the considered activities. The largest relative decrease was observed for the architecture and product-tree update, whose execution time decreased from 45 min to 10 min. This result is primarily associated with the use of Capella as the architectural backbone and with the automated alignment

[1] Of the total 405-min reference BRT, 330 min (81.5%) were derived from historical project records, while 75 min (18.5%) were based on retrospective engineering estimates.

Journal Pre-proof

of the corresponding product structure in RSP through the dedicated bridge.

Requirement impact assessment also showed a relevant reduction, from 30 min to 15 min, as explicit model relationships and *ViVA*-supported links reduced the effort required to identify and allocate affected requirements. Conversely, engineering impact assessment showed a more limited reduction, from 120 min to 90 min. This activity remains largely dependent on engineering reasoning, discipline-specific analysis, and multidisciplinary discussion, which are supported and enhanced but not replaced by the CE-MBSE environment.

A similar behaviour was observed for operational and compliance assessments. The integrated representation of modes, parameters, and requirement thresholds reduced information-retrieval and propagation effort, while interpretation of the results and engineering decision-making remained expert-driven. Finally, baseline review and consolidation decreased from 120 min to 70 min, mainly because a smaller number of independently maintained artefacts had to be reconciled after the change.

The BRT assessment should nevertheless be interpreted as a preliminary case-based comparison rather than as a statistical benchmark, as the analysis considers a single representative design-change scenario. Moreover, the reference workflow was reconstructed from a previous lunar project rather than reproduced experimentally on the same *BioMoon* configuration. The metric also represents cumulative execution time and should not be interpreted as an equivalent reduction in the duration of a complete concurrent session, project phase, or development programme. In addition, the comparison assumes that the CE-MBSE environment is already available. The initial effort required to construct the Capella model, configure RSP, develop *ViVA* and the Python interfaces, train users, and maintain the toolchain is outside the scope of the BRT assessment. The reported reduction therefore represents the recurring efficiency potential associated with change propagation and baseline restoration rather than a complete *return-on-investment* evaluation.

4.4.2 Observed Benefits

Beyond the quantitative BRT assessment, the *BioMoon* application provided evidence of additional methodological benefits.

A first benefit concerns the explicit traceability across abstraction levels. By structuring system knowledge within an integrated model, the methodology enabled explicit links between stakeholder needs, mission objectives, system functions and physical architectures. This continuity ensures that design decisions remain consistently aligned with initial mission drivers throughout the development process. In particular, the direct association between requirements and system elements facilitates impact analysis in case of design changes, reducing the risk of inconsistencies and requirement misinterpretation.

A second benefit concerns the coordination of multidisciplinary activities within recurring concurrent iterations. By providing a shared architectural reference and controlled access to updated engineering information, the approach allowed discipline experts to assess interconnected changes in parallel while maintaining visibility of their cross-domain consequences. This reduced the need for repeated sequential handovers and supported earlier identification of inconsistencies, interface mismatches, and design gaps.

Another important result is the integration of simulation-supported validation within the design loop. By linking the system model with external simulation environments, the methodology enabled early verification of system behaviour against mission requirements and constraints. As a result, decision-making is supported by both qualitative and quantitative analysis, strengthening the

reliability of trade-offs and design choices.

Additionally, the methodology demonstrates a strong capability in ensuring digital continuity across tools and lifecycle phases. The synchronisation between a modelling tool, with reference to Capella as an example, with an engineering database, in this instance Altium RSP, reduces information fragmentation, a well-known limitation of concurrent engineering environments, and contributes to the establishment of authoritative source of truth. Beyond the specific case study, this aspect is particularly valuable in multi-partner or distributed projects, where heterogeneous toolchains and workflows are typically involved.

Overall, these results indicate that the CE-MBSE approach supports a more coherent, transparent, and knowledge-driven design process, improving both the quality of system definition and the efficiency of design iterations.

4.4.3 Limitations

Despite the observed benefits, the application of the methodology also revealed some limitations to be carefully considered and mitigated.

A first challenge concerns the complexity introduced by the tool integration and synchronisation mechanisms. While multiple specialised tools provide flexibility, they require dedicated interfaces, such as Python bridges, adding technical overhead and a potential source of errors. Maintaining alignment between modelling and engineering data remains a critical issue, especially in highly iterative environments. This limitation can be mitigated by increasing the level of standardisation of data exchange processes, for instance through the adoption of common data models and model governance rules.

A second limitation relates to the effort and expertise required for MBSE adoption. Developing structured system models demands both methodological knowledge and modelling experience, representing a relevant entry barrier. In early phases, this effort may also appear disproportionate to immediate design outputs, reducing its perceived value. This aspect can be addressed through progressive adoption strategies, starting from high-level modelling and incrementally increasing model fidelity. The use of predefined templates, viewpoints, and modelling guidelines can further reduce the learning curve and improve usability, facilitating broader adoption across multidisciplinary teams.

Finally, while effective for CubeSat-scale missions, the scalability to larger complex systems remains to be fully explored. In a more complex context, such as System-of-Systems, an increased number of stakeholders and partners involved, as well as elements and interfaces, may significantly challenge model management, data synchronisation, and collaboration, requiring further methodological tuning and automation. The present bridges do not yet address simultaneous conflicting updates, partial synchronization failures, or automated multi-user conflict resolution.

5. Conclusions

This paper presented a structured Concurrent Model-Based Systems Engineering (CE-MBSE) methodology aimed at improving the design of complex space missions by combining the collaborative nature of concurrent engineering with the rigour and traceability of model-based approaches. The proposed framework supports a continuous and coherent development process across progressively increasing levels of system maturity, enabling both rapid design iterations and structured system definition. The application of the methodology to the *BioMoon* lunar CubeSat

Journal Pre-proof

mission demonstrated its capability to maintain consistency across abstraction levels, enhance multidisciplinary coordination, and integrate verification activities early in the design process. In particular, the use of a shared system model, coupled with a synchronised engineering database, proved effective in reducing information fragmentation and supporting informed decision-making in a concurrent environment.

The quantitative assessment further indicates that the proposed CE-MBSE workflow can reduce the effort associated with change propagation and baseline restoration. In the selected design-change scenario, the cumulative Baseline Restoration Time decreased from 405 min in the reference workflow to 245 min, corresponding to a nominal reduction of 39.5%. The largest gains were observed in activities related to architectural update, information propagation, and cross-tool reconciliation, whereas engineering reassessment remained largely dependent on expert judgement. This suggests that the main efficiency contribution of the methodology lies in reducing coordination and information management overhead rather than replacing discipline-specific engineering activities. Although this result is based on a single representative change scenario and should not be interpreted as a general benchmark, it provides preliminary evidence that structured model relationships and controlled tool integration can improve the efficiency of multidisciplinary baseline restoration in concurrent design.

From a Concurrent Engineering perspective, this result highlights the role of the proposed approach in maintaining alignment among parallel multidisciplinary activities as the design evolves, by supporting continuous cross-domain coordination and controlled information propagation within recurring design iterations.

At the same time, the study highlighted important challenges. The integration of heterogeneous tools requires dedicated synchronisation mechanisms, increasing the overall system complexity, while the adoption of MBSE introduces an initial effort in terms of modelling and expertise. These aspects underline the need for a careful balance between modelling depth and process efficiency, especially in early design phases.

Future work will focus on increasing the level of automation within the toolchain, improving interoperability with simulation environments, and extending the methodology to larger and more complex mission scenarios. Additionally, the integration of advanced capabilities, such as real-time simulation and AI-driven support tools, will be investigated to further enhance decision-making and system validation processes.

Overall, the CE-MBSE approach represents a promising step toward a more integrated, scalable, and knowledge-driven design paradigm for next-generation space missions.

Journal Pre-proof

Appendix A. Acronyms and Abbreviations

The following abbreviations are used in this manuscript:

API	Application Programming Interface
ARCADIA	ARChitecture Analysis and Design Integrated Approach
BRT	Baseline Restoration Time
CDF	Concurrent Design Facility
CE	Concurrent Engineering
CEC	Concurrent Engineering Center
CE-MBSE	Concurrent Model-Based Systems Engineering
COMET	COncurrent Model-Based Engineering Tool

ConOps	Concept of Operations
ECSS	European Cooperation for Space Standardization
EPS	Electrical Power Subsystem
ESA	European Space Agency
ESTEC	European Space Research and Technology Centre
ESTRACK	European Space Tracking
INCOSE	International Council on Systems Engineering
JPL	Jet Propulsion Laboratory
LAB	Logical Architecture Blank
M&S	Modelling & Simulation
MBSE	Model-Based Systems Engineering
MBSSE	Model-Based methodology to support Space Systems Engineering
MCC	Mission Control Centre
MOSAiC	Modeling and Simulation Architecture for integrated Complex systems
MSM	Modes and States Machine
NASA	National Aeronautics and Space Administration
OAB	Operational Architecture Blank
OCB	Operational Capabilities Blank
OCDT	Open Concurrent Design Tool
OOSEM	Object-Oriented Systems Engineering Method
PVMT	Property Value Management Tool
RSP	Requirements & System Portal
SAB	System Architecture Blank
SE	Systems Engineering
SoI	System of Interest
SysML	Systems Modeling Language
TCS	Thermal Control Subsystem
UHF	Ultra High Frequency
UML	Unified Modeling Language
V&V	Verification & Validation
ViVA	Verification & Validation Assessment

References

[1] International Council on Systems Engineering (INCOSE), "Systems engineering definition.," www.incose.org/about-systems-engineering/systemand- se-definitions/systems-engineering-definition.

[2] International Council on Systems Engineering (INCOSE), *INCOSE Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*. 2015.

[3] European Cooperation for Space Standardization, "ECSS-M-ST-10c rev.1: Project planning and implementation.," 2009, *ESA Requirements and Standards Division*.

[4] M. Bandecchi, B. Melton, and F. Ongaro, "Concurrent Engineering Applied to Space Mission Assessment and Design," 1999.

[5] D. Knoll, C. Fortin, and A. Golkar, "Review of Concurrent Engineering Design practice in the space sector: state of the art and future perspectives," in *2018 IEEE International Systems Engineering Symposium (ISSE)*, 2018, pp. 1–6. doi: 10.1109/SysEng.2018.8544387.

[6] J. Stjepandic, W. Verhagen, and N. Wognum, "CE Challenges – Work to Do," in *22st ISPE Inc. International Conference on Concurrent Engineering*, Jan. 2015. doi: 10.3233/978-1-61499-544-9-627.

[7] R. J. Winner, J. P. Pennel, H. E. Bertrand, and M. M. Slusarczuk, "The role of concurrent engineering in weapons system acquisition," 1988.

Journal Pre-proof

[8] J. L. Smith, *Concurrent Engineering in the Jet Propulsion Laboratory Project Design Center*, vol. 107. SAE Transactions, 1994.

[9] S. Friedenthal, D. Dori, and Y. Mordecai, "SEBoK: What is a Model? ." Accessed: May 06, 2026. [Online]. Available: https://sebokwiki.org/wiki/What_is_a_Model%3F

[10] J. A. Estefan, "Survey of Model-Based Systems Engineering (MBSE) Methodologies," 2008.

[11] T. Weilkiens, *Systems Engineering with SysML/UML*. Elsevier, 2007. doi: 10.1016/B978-0-12-374274-2.X0001-6.

[12] M. Hause, "The OMG Modelling Language (SYSML)," in *Fifth European Systems Engineering Conference*, International Council of System Engineering (INCOSE), May 2006, p. 14.

[13] "OMG Systems Modeling Language (SysML) v2 Release." Accessed: Aug. 31, 2026. [Online]. Available: https://github.com/Systems-Modeling/SysML-v2-Release

[14] R. B. Buruso, S. Himmiche, T. Tejumola, Y. Alemayehu, and B. Rose, "Model-Based Systems Engineering in Space Applications: A Comprehensive Literature Review," Sep. 01, 2026, *John Wiley and Sons Inc*. doi: 10.1002/sys.70051.

[15] E. Brusa, A. Calà, and D. Ferretto, *Systems Engineering and Its Application to Industrial Product Development*, vol. 134. Cham: Springer International Publishing, 2018. doi: 10.1007/978-3-319-71837-8.

[16] A. Gonzalez Fernandez, E. Maleki, N. Fischer, E. Honoré- Livermore, N. Christofi, and M. Verhoef, "The European Space Agency MBSE Methodology," *INCOSE International Symposium*, vol. 34, no. 1, pp. 2069–2086, Jul. 2024, doi: 10.1002/iis2.13256.

[17] J.-L. Voirin, *Model-Based System and Architecture Engineering with the Arcadia Method*. Elsevier, 2018. doi: 10.1016/C2016-0-00862-8.

[18] B. Cole *et al.*, "Domain-specific languages and diagram customization for a concurrent engineering environment," in *2013 IEEE Aerospace Conference*, IEEE, Mar. 2013, pp. 1–12. doi: 10.1109/AERO.2013.6497134.

[19] S. I. Infeld, D. Goggin, K. Vipavetz, and T. Grondin, "A SysML Model Template for NASA Concurrent Engineering Studies," in *2018 AIAA SPACE and Astronautics Forum and Exposition*, Reston, Virginia: American Institute of Aeronautics and Astronautics, Sep. 2018. doi: 10.2514/6.2018-5392.

[20] "CDF OCDT." Accessed: May 08, 2026. [Online]. Available: https://www.esa.int/Enabling_Support/Space_Engineering_Technology/CDF/OCDT_tutorial_now_available

[21] "COMET-CDP4." Accessed: May 08, 2026. [Online]. Available: https://www.stariongroup.eu/services-solutions/system-engineering/concurrent-design/cdp4-comet/

[22] F. Jakob, S. Mazzini, and A. Jung, "A SysML-Based Methodology in a Concurrent Satellite Design Process," in *Proceedings of Aerospace Technology Conference and Exposition* , Oct. 2011. doi: 10.4271/2011-01-2713.

[23] S. Campioli, F. Stesina, E. La Bella, S. Corpino, L. Niero, and C. My, "Concurrent Engineering to Enhance Autonomy for Deep-Space CubeSat Mission Design," *IFAC-PapersOnLine*, vol. 58, no. 16, pp. 163–168, 2024, doi: 10.1016/j.ifacol.2024.08.480.

[24] E. La Bella, S. Campioli, L. Iossa, S. Corpino, and F. Stesina, "Concurrent Model-Based Approach for CubeSat Mission Design," in *75th 75th International Astronautical Congress*, International Astronautical Federation (IAF), 2024. doi: 10.52202/078374-0035.

[25] S. Campioli, E. La Bella, A. D'Ortona, S. Corpino, and F. Stesina, "Concurrent Model-Based Approach for Project-based Education," in *11th International Systems & Concurrent Engineering for Space Applications Conference*, European Space Agency, 2024.

[26] F. Stesina and S. Corpino, "In the Loop Simulation to support the Cubesat projects in any phase of the product lifecycle," in *IFAC-PapersOnLine*, Elsevier B.V., Jul. 2024, pp. 89–94. doi: 10.1016/j.ifacol.2024.08.467.

[27] L. Li, S. Aslam, A. Wileman, and S. Perinpanayagam, "Digital Twin in Aerospace Industry: A Gentle Introduction," *IEEE Access*, vol. 10, pp. 9543–9562, 2022, doi: 10.1109/ACCESS.2021.3136458.

Journal Pre-proof

[28] S. Campioli, G. Luccisano, D. Ferretto, and F. Stesina, "Towards a Unified Modeling and Simulation Framework for Space Systems: Integrating Model-Based Systems Engineering with Open Source Multi-Domain Simulation Environments," *Aerospace*, vol. 12, no. 8, p. 745, Aug. 2025, doi: 10.3390/aerospace12080745.
[29] P. Nowodzienski and J. Navas, "From Model- based to Model and Simulation- based Systems Architectures—Achieving Quality Engineering through Descriptive and Analytical Models," *INSIGHT*, vol. 26, no. 1, pp. 40–50, Mar. 2023, doi: 10.1002/inst.12428.
[30] C. My, S. Corpino, S. Campioli, E. Honore-Livermore, and A. Deckert, "An MBSE Framework for Verification and Validation of Small Satellites Missions Projects," Politecnico di Torino, 2026.
[31] S. Campioli, C. Deiana, and S. Corpino, "Integrating Concurrent Engineering with Data-Driven and Model-Based Systems Engineering for End-to-End Small-Satellite Design," in *Proceedings of the 76th International Astronautical Congress*, International Astronautical Federation, 2025. doi: https://doi.org/10.52202/083091-0034.
[32] European Cooperation for Space Standardization, "ECSS-E-ST-10-06C Space engineering: Technical requirements specification," 2009.
[33] International Space Exploration Coordination Group (ISECG), "Global Exploration Roadmap," 2024.
[34] National Aeronautics and Space Administration (NASA), "Moon to Mars Architecture," https://www.nasa.gov/moontomarsarchitecture/.
[35] European Space Agency (ESA), "Earth orbit, Moon, Mars: ESA's ambitious roadmap," https://www.esa.int/Science_Exploration/Human_and_Robotic_Exploration/Earth_orbit_Moon_Mars_ESA_s_ambitious_roadmap.
[36] European Space Agency (ESA), "ESA-Space-Debris-Mitigation-Requirements-ESSB-ST-U-007-Issue1," 2023.

Declaration of interests - *" A Concurrent Model-Based Systems Engineering Methodology for SmallSat Lunar Mission Design"*

☒ The authors declare that they have **no known competing financial interests or personal relationships** that could have appeared to influence the work reported in this paper.

☐ The authors declare the following financial interests/personal relationships which may be considered as potential competing interests:

Journal Pre-proof