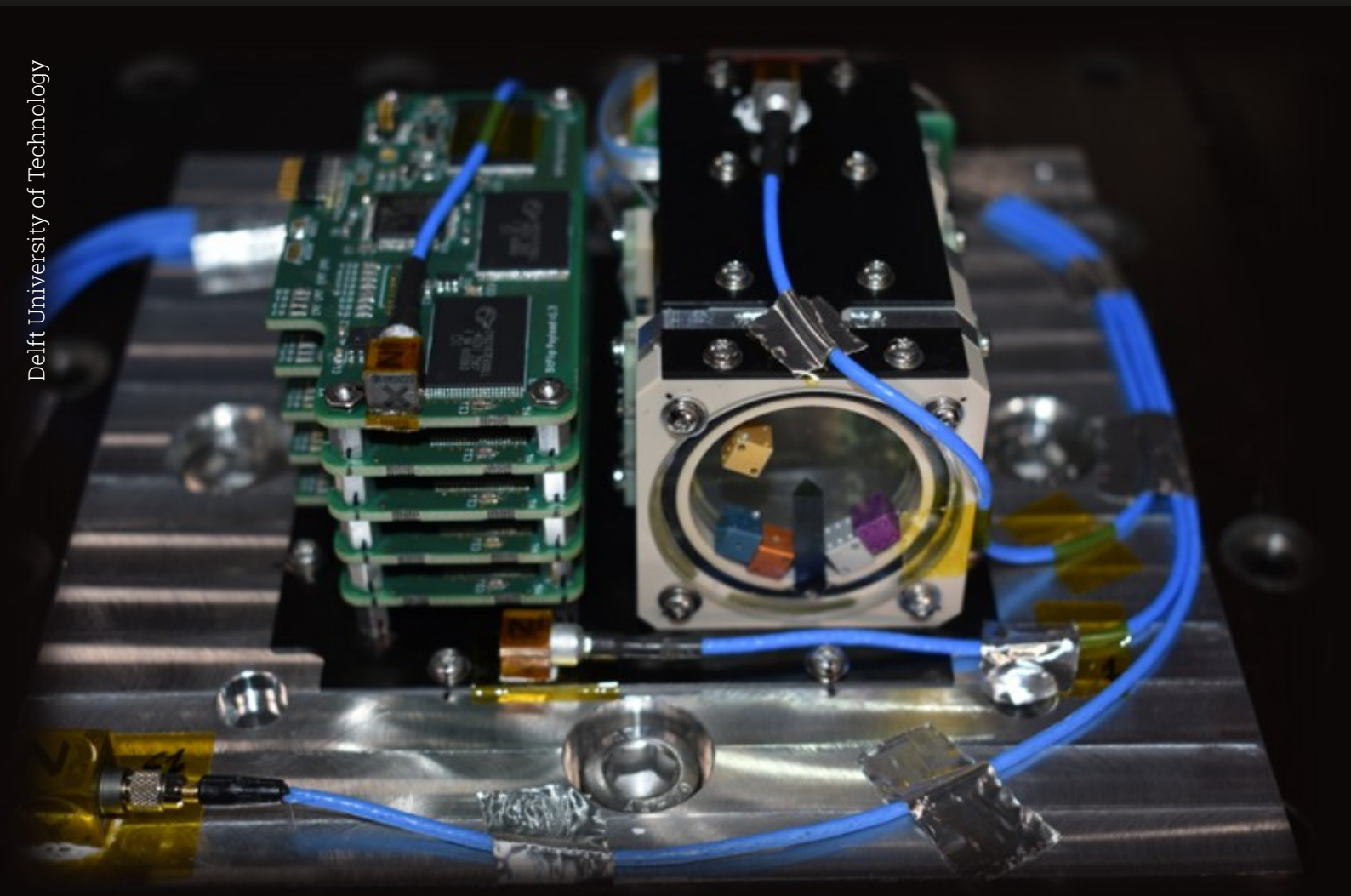
Implementation and Application of MBSE in Post-CDR Satellite Projects

AE5822: Thesis Space

Nicolas Oidtmann



Delft University of Technology

TU Delft

Implementation and Application of MBSE in Post-CDR Satellite Projects

by

Nicolas Oidtmann
Student Number: 6067611

Supervisor:	J. Vennekens
Supervisor:	I. Akay
Project Duration:	June, 2025 - May, 2026
Faculty:	Faculty of Aerospace Engineering

Cover: Picture of Da Vinci Satellite Payload Module by the Da Vinci Satellite team; Website: `https://davincisatellite.nl/`

TU Delft

Contents

List of Figures

List of Tables

1 Introduction

1.1. Background and Motivation

In its Agenda 2025 Document, the European Space Agency (ESA) highlights the increasing trend towards more advanced space missions, developed by cross-disciplinary project teams, that address some of today's most pressing global issues. To effectively manage the growing scope and complexity of such missions, creating digital continuity throughout the project life cycle is identified as a key enabler for efficient project and engineering management. [18]

One approach that supports this goal is Model-Based Systems Engineering (MBSE). MBSE promises to contribute to faster, less error-prone, and more cost-effective design, development, deployment, and verification of space systems [20]. Over the last few years, the shift from document-centric to model-centric engineering has been driven by increasing system complexity and the need for coordinated integration of information across the whole life cycle of a mission [4]. In this context, MBSE constitutes a response to the limitations of traditional, document-centric approaches and enables the creation and maintenance of a digital thread throughout the project life cycle [4].

The application of MBSE for performing systems engineering activities that dominate the early stages of a space project's life cycle has been widely researched and adopted in recent years. In contrast, systems engineering activities related to later project phases, such as Assembly, Integration and Testing (AIT) and Verification and Validation (V&V), have not received comparable attention [57].

1.2. Problem Statement

There is a growing interest within the space domain to leverage MBSE beyond the initial design stages, particularly in the later phases of projects where the benefits of model-based approaches can provide significant value. However, extending MBSE for projects in the AIT / V&V phase presents additional challenges as MBSE implementation remains fragmented and inconsistent across tools, methods, and organizations [4]. This lack of a structured approach to align methods and tools explicitly for the systems engineering processes related to AIT and V&V presents a significant challenge for the adoption of MBSE over the full project life cycle.

As AIT and V&V activities dominate the later stages of space projects and are characterized by complex, document-centric processes, the absence of a structured MBSE approach tailored to these activities represents a critical gap. Addressing this gap is essential to enable the extension of digital engineering practices into the post-CDR phase and to fully realize the benefits of MBSE across the entire system life cycle.

1.3. Objectives of the Thesis

This thesis investigates how MBSE can be effectively applied in the later stages of the project life cycle, using the DaVinci Satellite (DVS) student team as the primary application case. As a multidisciplinary,

student-led satellite project, currently in the post-CDR phase, DVS provides a representative example of the challenges and opportunities associated with applying MBSE beyond the early design stages. Over the course of the research, this thesis will explore how a tailored MBSE methodology can support the planning, execution, and evaluation of assembly, integration, and verification activities to make use of the benefits of a digitalized way of working through the application of MBSE. The thesis aims to demonstrate the feasibility of the developed approach in a representative project context to showcase the ability to extend MBSE into the AIV domain, enabling end-to-end traceability through the creation of a digital thread.

1.4. Thesis Structure

This thesis introduces a model-based methodology to perform Assembly, Integration, and Verification (AIV) of Space Systems within later project phases. This document provides an overview of the research objectives and questions of the thesis project, the development process and application of the methodology, and an evaluation of the approach. This section serves as an overview to the reader and describes the structure of the document. Next to the document, the models created during the thesis research are also made available.

Chapter 2 reviews relevant literature on Systems Engineering, life cycle models, AIV practices, and MBSE, establishing that AIV activities dominate the post-CDR phase and are characterized by complex, document-centric working processes. Based on this analysis, a research gap is identified regarding the application of MBSE to cater to these later life cycle stages, particularly in the absence of a dedicated methodology for AIV work. The reviewed literature on MBSE provides guidance on how the development of the MBSE AIV methodology is conducted.

Building on the literature review, Chapter 3 describes the research approach taken in this project, including the definition of research objective and questions.

Chapter 4 describes the core contribution of this thesis: the development of a tailored MBSE AIV methodology for application in post-CDR satellite projects. The chapter is structured in two main parts. First, Section 4.1 derives the methodological foundation by translating literature insights into stakeholder needs, use cases, and a set of requirements that the methodology must fulfill. These elements define the problem space and establish the criteria for a successful approach. The main contribution is then presented in Section 4.2, where the MBSE AIV methodology is designed to address these requirements. The methodology is structured into key components, including an ontology defining the core AIV concepts, a process describing AIV planning and execution, and supporting modeling methods and framework elements that enable consistent application. In addition, the selection of modeling tool and language is addressed. The methodology and its implementation in SysML v1 is demonstrated, showing the methodology's applicability to a small demonstration example and providing a concrete realization of the proposed approach.

Chapter 5 evaluates the developed methodology through a structured verification and validation approach. Verification assesses the internal consistency of the methodology, its compliance with the defined requirements, and its alignment with relevant standards such as ISO/IEC/IEEE 24641 and ECSS documentation requirements. Validation is performed through application to a representative CubeSat payload case study, demonstrating the feasibility and usefulness of the approach in a realistic project context.

Finally, Chapter 6 discusses the key results, benefits, and limitations of the proposed methodology, and reflects on its implications for the application of MBSE in post-CDR satellite projects. The chapter concludes by answering the research questions and outlining directions for future work. Together, these chapters provide a coherent argument that MBSE can be successfully extended beyond early design phases and applied to support AIV activities in later stages of the system life cycle.

2

Literature Review

2.1. Literature Review Structure

This section gives an overview of the main structure and flow of the literature review chapter by motivating the sub-chapters based on the research objective and research questions.

This thesis aims to investigate the implementation and use of Model-Based Systems Engineering in satellite projects after Critical Design Review (CDR). The first section of the literature review defines terms which are relevant to understand the topic of this thesis.

The next section provides an understanding of the concept of Systems Engineering and Project Life Cycle Models, by reviewing general and space-industry specific sources, including standards, handbooks, and academic publications on the topics. The term "post-CDR" is explained in detail, by comparing different life cycle models. The section looks in detail into how the work of a systems engineer changes after CDR. As a result, the section establishes that a significant amount of the Systems Engineering effort after CDR is spent on performing tasks related to Assembly, Integration, and Verification (AIV) of the Product. This directly motivates the literature research on the topic of AIV.

The following section dives into detail on aspects related to satellite AIV and its relation to Systems Engineering and the life cycle models. A special focus is put on the typical documentation generated during the testing phase of product development. This serves to provide an understanding of the typical information and artifacts that need to be managed by the Systems Engineering and AIV team, regardless of whether the work is following a traditional, or a model-based approach.

Lastly, literature on MBSE is presented and summarized. The section starts with a discussion of the definition, motivation, and potential benefits that MBSE can provide. Afterwards, the constituents of an MBSE Methodology are introduced, as well as their relation to each other. After the components of an MBSE methodology are established in theory, some common MBSE methodologies are presented, including methodologies developed for the development of space systems.

2.2. Systems Engineering

This thesis aims to investigate the application of MBSE for the later stages of the project life cycle, i.e. "post-CDR". This chapter will define the term Systems Engineering, as well as present and compare different life cycle models from literature and the related Systems Engineering activities per life cycle stage.

To understand the scope of this thesis and how it relates to the overall context of a space project, subsection 2.2.2 looks at the role of a systems engineer over different life cycle models found in literature and provides a big picture overview of the responsibilities of a Systems Engineer over the course of a (space) project. Generic life cycle models and life cycle model for space projects will be compared, to provide an understanding of the meaning of the term "post-CDR" in the context of space missions.

In subsection 2.2.3 the relation between engineering work that is performed in the post-CDR life cycle

stage and other life cycle stages is presented following the generic ISO and INCOSE life cycle models and the space mission specific life cycle models of ECSS and NASA.

Finally, subsection 2.2.4 will provide a detailed overview of the major deliverables and reviews that frame the work of Systems Engineers and the whole project in the post-CDR phase.

2.2.1. Definition and Benefits of Systems Engineering

Many definitions of Systems Engineering can be found in literature, which present large overlaps. The following section will discuss some definitions and highlight commonalities and differences.

The International Council on Systems Engineering (INCOSE) defines Systems Engineering as a transdisciplinary and integrative approach to enable the successful realization, use, and retirement of engineered systems, using systems principles and concepts, and scientific, technological, and management methods [34]. While INCOSE makes use of this definition, it acknowledges the existence and validity of other definitions, highlighting common characteristics mentioned in the different definitions. Some of the commonalities are the interdisciplinary and iterative nature of Systems Engineering, the focus on sociotechnical aspects and the goal to maintain a holistic perspective. It is simultaneously a perspective, a process, and a profession. [63]

The European Cooperation for Space Standardization (ECSS) defines Systems Engineering as an interdisciplinary approach governing the total technical effort required to transform requirements into a system solution [12]. This definition is adopted by the ECSS from the IEEE P1220 - Standard for Systems Engineering [32]. Similarly to the previously mentioned definitions, the IEEE considers both technical aspects and managerial aspects part of systems engineering work, covering the complete system life cycle and striving for a balanced overall system solution. Under systems engineering management, the IEEE understands planning of technical efforts and controlling the technical baseline for the project. This controlling task includes among others, Technical Review planning, Risk Management, and Interface Control. [32]

The definition of systems engineering provided by the ISO standard *ISO/IEC/IEEE 15288:2023* extends the definition of the IEEE by stating that systems engineering is a "trans-disciplinary and integrative approach to enable successful realization, use, and retirement of engineered systems using systems principles and concepts and scientific, technological and management methods" [37]. That standard has been adjusted from its original form to align with the *IEEE P1220* standard and the IEEE was involved in these activities [8]. The definition in the ISO standard is based on and references the INCOSE definition presented above.

The next definitions come from Sanford Friedenthal, Oliver Alt, Tim Weilkiens and Jon Holt. Sanford Friedenthal is an independent consultant and industry leader in Model-Based Systems Engineering. As a Lockheed Martin Fellow, he led the company's initiative to institutionalize Model-Based Systems Development (MBSD) and advanced engineering practices across the company, providing direct MBSE support to numerous programs. He has applied systems engineering across the full system life cycle and served as a systems engineering department manager. Furthermore, he was a key contributor to the development of the Object-Oriented Systems Engineering Method and also led the industry team responsible for creating SysML (v1) and guiding its adoption by the Object Management Group. He is recognized as an INCOSE Fellow for his contributions in the field of MBSE and Systems Engineering, expertise in MBSE methods, extensive publications, and international teaching engagements in graduate systems engineering programs. [25] Dr.-Ing. Oliver Alt is a trainer and consultant at LieberLieber Software GmbH for MBSE. His credibility in the field of Model-Based Systems Engineering (MBSE) is underpinned by his previous work as a systems engineer at Continental Teves, where he developed a SysML-based MBSE approach that has been applied for several years. He also wrote and collaborated several books on the application of MBSE with SysML [59]. Tim Weilkiens is a board member at the German consulting firm oose, an experienced MBSE coach, and a key contributor to the SysML standard as a co-author of the initial specification and co-chair of the SysML v2 finalization task force. His active involvement in OMG and INCOSE, combined with industry coaching across various domains, makes him a credible source in the field of MBSE [47]. Prof. Jon Holt, INCOSE Fellow and systems engineering professor at Cranfield University, is internationally recognized as a leading expert in MBSE, with over 30 years of practical experience and authorship of 17 books on the subject. His deep involvement in

MBSE education and implementation across industries, along with his recognition as one of INCOSE's 25 most influential systems engineers, underscores his authority in the field [29].

In his book *A practical guide to SysML* Friedenthal, Moore and Steiner define Systems Engineering as a multidisciplinary approach with the goal to arrive at a balanced system solution that answers to a set of diverse stakeholder needs. A systems engineer employs both technical and managerial processes to balance development cost, schedule, technical performance and risk to ensure the success of the project. The technical work of a systems engineer includes analysis, system design and specification, integration, and verification. The managerial aspects of systems engineering encompass planning the technical effort, monitoring technical performance, managing risk, and controlling the system technical baseline. [25]

For Oliver Alt, Systems Engineering represents all development activities that are required to develop a system. Systems Engineering, according to Alt, is a set of techniques and methods which aim to reduce misunderstanding between people involved in the development process. The second, related goal of applying systems engineering is to clearly define system boundaries, subsystems, and system contexts. This definition puts similar emphasis on the importance of ensuring effective communication within a project as the other ones. However, it does not explicitly include the management aspects of systems engineering which are mentioned in other definitions. Furthermore, Alt states system requirements, system architecture, and system behavior as the three building blocks of systems engineering, which are iteratively defined for different abstraction levels of the system. [2]

Tim Weilkiens definition of systems engineering mostly agrees with the one from INCOSE and Friedenthal, explicitly stating verification and validation as part of systems engineering and summarizes systems engineering as an interdisciplinary approach that integrates all other relevant engineering disciplines and accounts for technical and economic aspects of a given project. [67]

According to Jon Holt, Systems Engineering is concerned with solving problems while handling the project's complexity and ensuring effective communication over the whole project life cycle. In addition to this one, a second, more concise definition is offered stating that Systems Engineering is in essence the implementation of common sense into any engineering endeavor. [30]

This last definition hints towards the need for systems engineering as stated in literature. According to Holt, there are three common causes for issues that may arise in the development process of any complex technical system, which can lead to failure of the overall project if not considered. These are:

- a lack of management of the complexity of the project,
- a lack of communication or inefficient communication, and
- a lack of understanding.

These problems may occur at any stage of the project and will amplify each other and will likely lead to an unsuccessful system / project if left unchecked. This is labeled as the "vicious triangle of evil in engineering" [31]. Systems Engineering aims to provide a methodology to cope with these threats and minimize the programmatic and technical risks that could lead to project failure in all project phases. [30, 31]

Another reason for the necessity of systems engineering in engineering projects is the typical distribution of life cycle costs over time, as shown in Figure 2.1. The figure shows that a large share of the total project costs are committed already early on in the design process with decisions made in these project phases, while the actual money that is spent in an early project stage remains relatively low. This means that errors that are only found in production or operation stages are much more expensive to fix than errors that are found earlier. Systems Engineering enables a more rigorous concept exploration, by ensuring a solid information baseline and logical trade-off processes. This reduces the risk of hasty decisions and helps to avoid mistakes. [63]

Friedenthal and the ESA Agenda 2025 state coordination and technical effort management of cross-disciplinary teams as an additional driver to implement system engineering in projects [25, 21].

The structure of the Systems Engineering Process can be visualized and described with different, so called Life Cycle Model Approaches, according to the INCOSE Handbook and the ISO 14748-2. These can

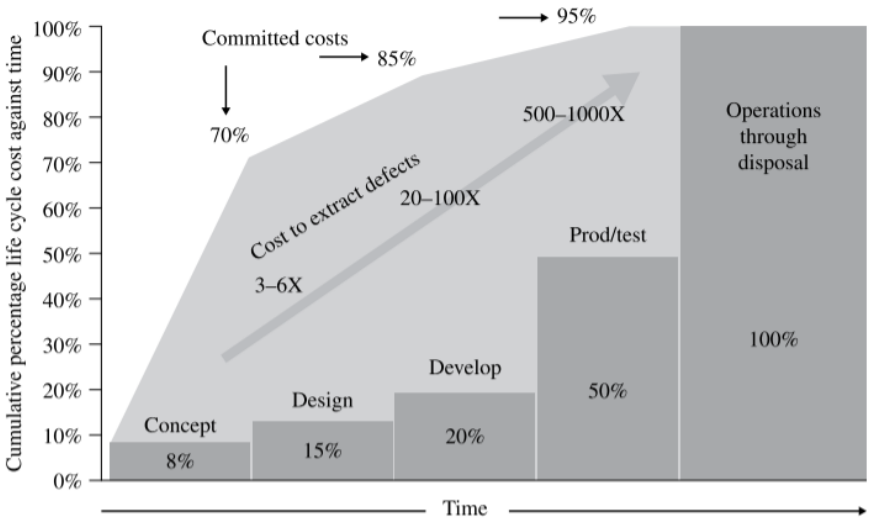


Figure 2.1: Committed Life Cycle Cost against Time [63]

be categorized in Sequential approaches, Incremental Approaches, and Evolutionary Approaches. [63, 40]

The Vee Model is an example for a model that describes a sequential approach to Systems Engineering. It was first described by Forsberg and Mooz, 1991, in [23]. The process model is visualized in Figure 2.2. The Vee model consists of two main parts, the left arm and the right arm of the Vee, the first one corresponding to the specification of the system, while the latter corresponds to the assembly, integration, verification, and validation of the system. The Vee model highlights the necessity to conduct validation against stakeholder needs and planning for verification already early on in the life cycle. A typical project would start at the top left of the Vee with the stakeholder needs and requirements definition, then move along the arm of the Vee to generate the system architecture and more detailed specifications of the system elements. These elements are then implemented at the bottom of the V. The right side of the Vee represents the sequential integration and verification of system elements, first on lower components and subsystem levels, and eventually for the whole system, finishing with validation of the final system at the top right side of the Vee. [63]

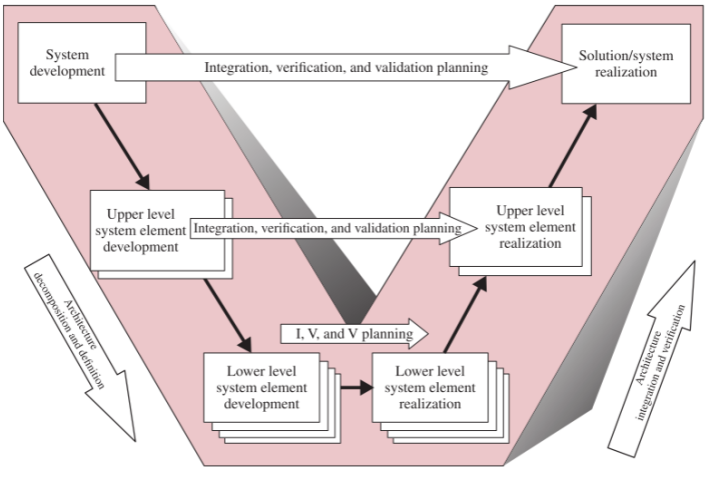


Figure 2.2: Vee model visualization, adapted from [63]

An example for an incremental approach is the spiral model, shown in Figure 2.3. Each rotation of the spiral results in the generation of a prototype of the final system. For some systems, these prototypes could possibly already be brought to market. While the Vee model highlights the need to

conduct verification and validation over all phases of the project, the spiral model adopts a risk-based philosophy. At any time of the process, the system elements and the interfaces between these elements must be known, requiring a well defined system architecture as a baseline. All decisions, trade-offs, and analyses must be driven primarily with the objective to reduce the overall risk of the project and consider the complete system life cycle. For each design cycle, or rotation, the goals and constraints of the stakeholders must be defined and considered. Each design cycle investigates different system and process alternatives, as well as manages risks and uncertainties. In order to proceed from one design cycle to the next one, all stakeholders must agree on a plan to continue. [28]

Gujarathi et al. state the increased focus on risk identification and mitigation as one of the main benefits of the spiral model. They mention the interfaces as one of the major sources for risks within complex projects and highlight the ability of the spiral model to manage those risks. Similarly to the Vee model, the spiral model also envisions stakeholder integration into the process for purposes and alignment on the project plans. The spiral model can be considered a relatively flexible life cycle model approach, as it allows to integrate different life cycle models for different rotations around the spiral. Lastly, they mention that spiral development aligns well with modern trends in product development, such as digitalization, Model-Based Systems Engineering, and agile hardware prototyping. [28]

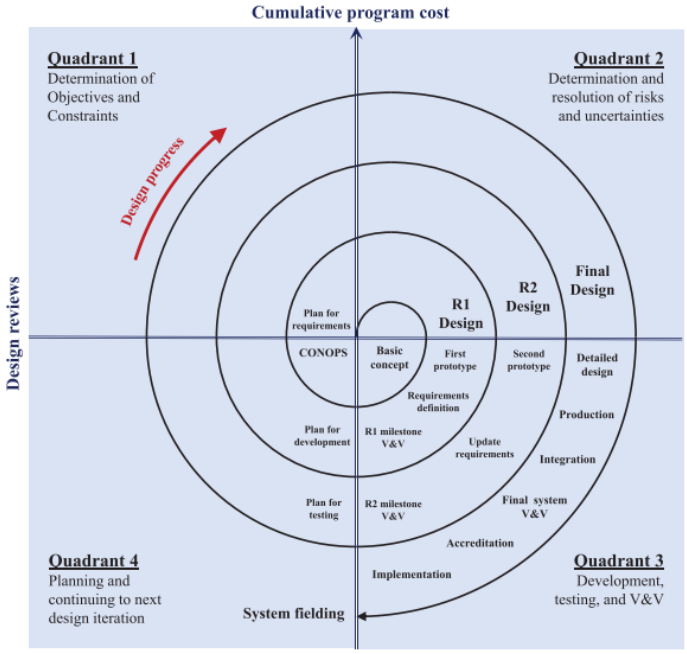


Figure 2.3: Spiral model visualization, adapted from [28]

2.2.2. SE Life Cycle Models

To understand the evolution of engineering, and especially systems engineering tasks over the project life cycle, three life cycle models from literature are compared, the generic life cycle model described in multiple ISO standards, the ECSS life cycle model used in ESA projects, and the NASA life cycle model. The sources used to create the description for the different life cycles are presented in an overview in Table 2.1.

Table 2.1: Overview of Literature Sources on Life Cycle Processes and Systems Engineering

Model/Standard	Source	Focus and Contribution
Generic ISO Life Cycle Model	ISO 24748-1:2024 [39]	*Guidelines for life cycle management:* defines life cycle concepts and stages, with the purpose to ensuring consistency in terminology and application across domains. Includes pointers to other ISO standards.
	ISO 24748-2:2018 [40]	*Guidelines for Application of ISO/IEC/IEEE 15288* aligns with ISO 24748-1 to provide a detailed explanation and tailoring of life cycle stages for Systems Engineering.
	ISO 24748-4:2016 [41]	*Systems engineering planning:* describes the process of planning and management for the Systems Engineering domain aligned with ISO 15288, 24748-1, and 24748-2.
	ISO 15288:2023 [37]	*System life cycle processes:* provides a framework of processes using a systems engineering approach.
	ISO 24641:2023 [38]	*Methods and tools for model-based systems and software engineering:* supports systems engineering and software engineering processes during digital transformation, from both process and cognitive modeling perspectives.
	INCOSE SE Handbook [63]	Codifies “good practices” in SE, consistent with ISO 15288 and SEBoK, including life cycle process integration.
ISO Life Cycle for Space Projects	ISO 14300-1:2023 [36]	*Structuring of a Space Project:* defines requirements for structuring and managing space projects within a standardized programme management framework. Focus on balancing performance, costs, and schedules with risks across all phases of a space project’s life cycle and provides a space project specific description of life cycle stages, consistent with the generic ISO life cycle model.
ESA/ECSS Life Cycle Model	ECSS-M-ST-10C Rev. 1 [14]	Space project planning and implementation: life cycle structure tailored to space projects.
	ECSS-M-30-01A [13]	Review processes: management and conduct of formal reviews across life cycle phases.
	ECSS-E-ST-10C Rev. 1 [12]	System Engineering general requirements: aligns engineering activities with ECSS life cycle model.
NASA Life Cycle Model	NASA SE Handbook [50]	Overview of NASA’s life cycle and SE approach: combines programmatic and technical perspectives. Emphasizes tailoring, integration, and iterative nature of SE processes through the life cycle.

Generic ISO Life Cycle Model

According to ISO 15288 [37], the life cycle of a project is the "evolution of a system, product, service, project, or other human-made entity from conception through retirement". For each of these human-made entities, the progress along the life cycle can be described using a life cycle model. The ISO standard 24748-1 [39] describes a generic system life cycle model. The purpose of defining such a life cycle model is to provide a starting point for planning efforts to realize, operate or support the System of Interest by segmenting the complete life cycle of the system into so called life cycle stages. The generic life cycle stages from [39] are:

1. Concept
2. Development
3. Production
4. Utilization
5. Support
6. Retirement

This segmentation is often used to insert decision-making gates in the development process to reduce risk. Different processes, procedures, methods, and tools might be appropriate for different life cycle

stages. With the segmentation, these aspects can be selected and reevaluated for each stage. The life cycle stages as per ISO 24748-1 are shown in Figure 2.4 and are explained further in the following.

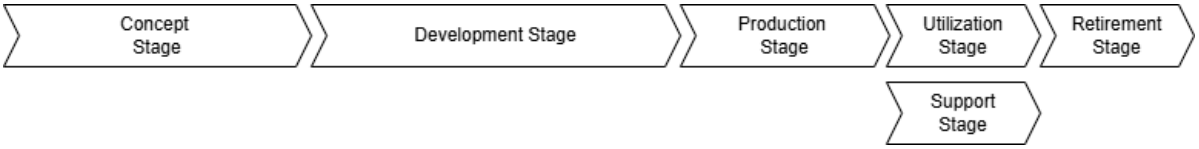


Figure 2.4: ISO Life Cycle Model

Concept Stage
The concept stage is the initial phase of the system life cycle and begins with the definition of needs towards the end product, which could be either a completely new system-of-interest (SoI) or a modification to an existing one. This stage is dedicated to early exploration and planning. It involves identifying potential business opportunities or mission requirements and assessing their feasibility from technical, economic, strategic, and market perspectives.

During this stage, multiple alternative concepts are developed and evaluated making use of trade-off studies, risk assessments, cost and schedule estimations, and early prototypes where necessary. These concepts may be entirely new or derived from existing systems. The role of enabling systems across the entire life cycle has to be considered in the solution evaluation process.

The concept stage also results in a clearer understanding of stakeholder needs and preliminary requirements, along with at least one feasible early-stage system architecture and operational concept. It establishes the foundation for all subsequent stages. At the end of the concept stage, a decision is made whether to proceed with system development.

Key outcomes of the concept stage include:

- Identification of new system concepts. These concepts might offer benefits such as new capabilities, improved performance, or reduced costs to stakeholders.
- Feasibility assessment of the concepts, including initial system architecture and requirements for crucial enabling systems in later life cycle stages.
- Preliminary high level requirements.
- Cost estimates and defined outcomes, and entry/exit criteria for all subsequent life cycle stages.
- Risk management strategies applicable throughout the life cycle.

Development Stage
The development stage focuses on transforming stakeholder requirements and system concepts into a completely defined and feasible system-of-interest that meets the stakeholder requirements. This includes specifying, designing, integrating, and evaluating hardware, software, human, process, and facility components. At this stage, either a prototype or an operational system (potentially being adapted) is developed to support future utilization.

The development process takes into account constraints and requirements from future life cycle stages regarding production, utilization, support, and retirement. Technical and stakeholder feedback is gathered through reviews to guide design and integration decisions.

Key outcomes of the development stage include:

- Complete architecture models and system design for the SoI.
- A structured SoI composed of system elements and interfaces.
- Verification and validation (V&V) results with supporting evidence to demonstrate compliance with requirements.
- Planning and preparation for operational deployment, including training and operational procedures.

- Confirmation that the SoI is producible, operable, supportable, and ready for retirement as needed.
- Refined requirements for enabling systems and traceability tools for managing system requirements and development artifacts.
- Availability of a prototype or an operable SoI.
- Updated cost estimates and refined outcomes for the subsequent stages.

Production Stage
The production stage is where the finalized system is manufactured, assembled, and tested for deployment. For complex systems, like satellites, this could mean the production of the final flight model and performing acceptance testing to confirm readiness.

Depending on the project, there may be some overlap between development and production activities. During this stage, the system is produced in its operational form, support tools are finalized, and everything is prepared for system delivery. The output is a system ready for deployment, along with refined plans and cost estimates for operational use and eventual retirement.

Key outcomes of the production stage include:

- The production process is qualified, ensuring consistent delivery of systems that meet all specifications.
- All necessary resources, i.e. materials, services, supporting systems and system elements, are available and known to support the production of the SoI.
- A fully operable system is produced, and the acquirer formally accepts the verified and validated system.
- Completed systems are packaged and delivered to the acquirer.

Utilization Stage
In the utilization stage, the system is put into operation in its intended environment. This is where the system starts delivering its designed services or capabilities to users. The stage begins with installation and transition into active use, which may include tasks like operator training.

The primary focus during utilization is to ensure the system performs reliably and continues to meet user expectations. Feedback from this stage can lead to improvements or enhancements to the system. It's also common to monitor performance data and usage patterns to inform future development or updates.

Key outcomes of the utilization stage include:

- Training of personnel for operations
- SoI is installed in operational environment and capable of being operated
- New opportunities for SoI enhancements are identified through stakeholder feedback.

Support Stage
Running in parallel with the utilization stage, the support stage ensures that the system continues to function effectively over time. It includes activities such as routine maintenance, troubleshooting, software updates, and logistical support. Depending on the system, the support stage can be crucial for sustaining service availability and user satisfaction.

Support also involves correcting any deficiencies that may arise, as well as updating documentation and procedures based on operational experience. Well-planned support strategies can significantly extend the life of a system and reduce total ownership costs.

Key outcomes of the support stage include:

- Trained personnel to perform supporting activities
- Organizational and support system interfaces are provided that allow to resolve problems and perform corrective actions.

- Problems and deficiencies are identified and corrected.

Retirement Stage
Eventually, every system reaches the end of its useful life. The retirement stage involves the planned and responsible removal of the system from its operational environment, as well as discontinuation of related operational and support services. This includes decommissioning activities, such as dismantling equipment, archiving data, and restoring the operational environment, if necessary.

Planning for retirement often begins well before the actual end of service, especially for systems with environmental, safety, or legal implications. The focus is on ensuring a smooth and secure retirement, minimizing disruptions, and documenting lessons learned for future projects.

Key outcomes of the retirement stage include:

- Enabling systems and services to remove the SoI from its operational environment are available.
- The operational environment is returned to its original or an agreed state
- SoI is removed from its operational environment and all related enabling systems and services are discontinued.

ISO Life Cycle Processes
ISO 15288 defines Systems Engineering processes that can be applied in each stage of the life cycle. The processes are further organized into the categories: Agreement processes, Organizational project-enabling processes, Technical management processes, and technical processes. Table 2.2 provides an overview of the life cycle processes found in ISO 15288. The life cycle processes define more detailed the systems engineering work that is performed, while the life cycle model provides an overview of the purpose and the main outcomes for each stage / phase in the life cycle. This means that the framework provided in the ISO 24748-1 is only loosely linked to the exact definition of systems engineering processes in ISO 15288. The selection of which life cycle processes will be applied during which life cycle stages has to be made individually for each project and usually will be agreed upon between the project management and the customer [37]. This selection and agreement of life cycle processes is guided by the specific purpose and expected outcomes for each life cycle stage and the life cycle stage entry and exit criteria, keeping in mind the overall risk posture for the project.

Table 2.2: ISO/IEC/IEEE 15288 System Life Cycle Processes Overview

Category	Process
Agreement Processes	Acquisition Supply
Organizational Project-Enabling Processes	Life Cycle Model Management Infrastructure Management Portfolio Management Human Resource Management Quality Management Knowledge Management
Technical Management Processes	Project Planning Project Assessment and Control Decision Management Risk Management Configuration Management Information Management Measurement Quality Assurance
Technical Processes	Business or Mission Analysis Stakeholder Needs and Requirements Definition System Requirements Definition Architecture Definition Design Definition System Analysis Implementation Integration Verification Transition Validation Operation Maintenance Disposal

The agreement processes, organizational project-enabling processes and technical management processes are applied throughout all life cycle stages. For technical processes, the ISO 24748-2 [40] and the INCOSE Systems Engineering Handbook [63] both provide some general guidelines which processes might be applied during the different stages, while still emphasizing the importance of tailoring the selection to the specific project needs. The technical processes that are used during system development relate closely to the typical work packages of a systems engineer and can be further categorized in the following manner [40, 63]:

Understand Needs:

- Business or Mission Analysis
- Stakeholder Needs and Requirements Definition
- System Requirements Definition

Define the Solution:

- Architecture Definition
- Design Definition

Realize the Solution:

- Implementation
- Integration
- Verification
- Transition
- Validation

The System Analysis Process runs concurrently to all other processes during the development of the system, while Operation, Maintenance and Disposal Processes are invoked after the system has been developed. This categorization is consistent with the typical Vee model presented in subsection 2.2.1. The left side of the Vee corresponds to the processes in the categories *Understand Needs* and *Define the Solution*. The bottom of the Vee corresponds to the Implementation process and the right side of the Vee corresponds to the Integration, Verification, Validation and Transition Processes. To understand how these life cycle processes are applied throughout the life cycle of space projects specifically, first, the individual technical processes will be described briefly, and then the life cycle models of typical ESA and NASA missions will be discussed with the goal of mapping the technical processes from ISO 15288 to the ESA and NASA life cycle models.

ECSS Life Cycle Model

The ECSS life cycle model consists of 7 major phases, as shown in Figure 2.5. Phase 0 is also referred to as Pre-Phase A. After each phase, a review is held to evaluate the project status and confirm the readiness to move into the next project phase. Each of the phases presented in the following are further described in [13, 14, 12].

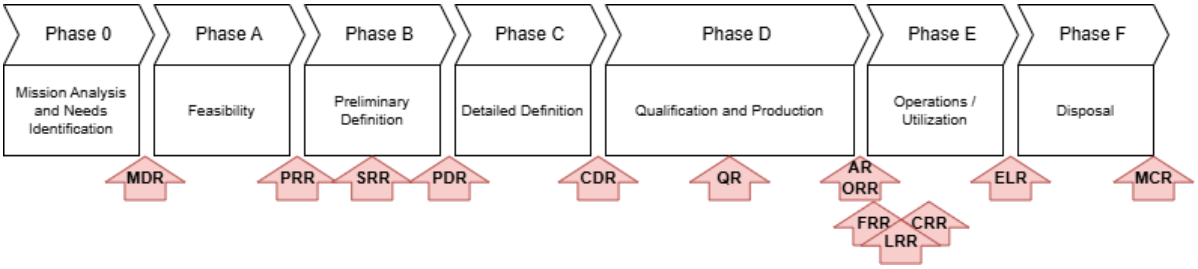


Figure 2.5: ECSS Life Cycle Model

Phase 0

This phase marks the starting point of an ESA project life cycle and is dedicated to identifying and articulating the mission's fundamental needs, objectives, and constraints. The primary goal of this phase is to determine the overall feasibility of the mission concept and lay the foundation for further technical and programmatic development.

During Phase 0, a comprehensive analysis is carried out to translate user needs into high-level mission requirements. This involves collaborating with the customer to understand the intended mission outcomes and required performance, as well as to identify environmental, technological, and operational constraints for the mission.

A key task in this phase is the generation of possible mission concepts that could fulfill the customer needs. These concepts are evaluated with regards to technical feasibility, alignment with mission objectives, and compatibility with existing technologies or technologies currently under development. Alongside the technical considerations, preliminary assessments of programmatic aspects such as mission cost, development timelines, and risk factors are also performed.

From a systems engineering (SE) perspective, this phase includes the derivation of mission-level requirements from user needs, the generation of candidate system concepts, and the preparation of documentation for the Mission Definition Review (MDR). The MDR is a decision gate to assess whether the mission is sufficiently defined and ready to proceed to Phase A. It typically involves reviewing the mission statement, validating the preliminary technical requirements, and confirming the programmatic viability of the proposed concepts.

Phase A
The overarching goal of this phase is to identify, compare and validate multiple technically and programmatically viable solutions that can fulfill the mission requirements and serve as a baseline for more detailed development in Phase B.

This phase includes evaluation of candidate concepts through detailed analysis of their technical and life cycle implications. System architectures and operational approaches are defined and compared against the mission objectives.

As part of the systems engineering effort, customer needs and mission requirements undergo further refinement and validation. This ensures that the technical requirements derived from Phase 0 are consistent with the actual expectations and constraints of the stakeholders. The system functionality is then analyzed and decomposed into a functional tree, which supports early system architecture and interfaces definition and lays the foundation for subsequent system design.

In parallel, preliminary plans are developed for how the project will be managed in the following phases. This includes the preparation of the management plan, the systems engineering plan, and the product assurance plan, as well as a preliminary verification approach.

The end of Phase A is marked by the Preliminary Requirements Review (PRR). At this stage, a preferred system and operations concept is selected and baselined, and the initial verification strategy is reviewed. Critical technical risks and items are identified and tracked, along with a revised assessment of overall programmatic risk. The PRR confirms that the selected concept is both technically feasible and programmatically viable, and that the project can proceed into the next phase.

Phase B
Phase B focuses on developing a solid and detailed planning for the full-scale development of the system in the next phase. Building on the concept and baseline selected at the end of Phase A, this phase has the goal to confirm that the chosen system concept is technically and programmatically sound. The goal is to finalize key planning elements and prepare all aspects of the project for detailed design and development.

A major activity in Phase B is the finalization of core planning documents. The project management, systems engineering, and product assurance plans are completed, setting out how the rest of the project will be organized and controlled. Alongside this, key project structures such as the master schedule, cost baseline, work breakdown structure, organizational breakdown, and specification tree are established.

From a technical perspective, Phase B involves confirming the feasibility of the system concept and selecting the preferred technical solutions. A preliminary design definition is created to describe the system and its elements and interfaces. This includes defining external interfaces with other systems or stakeholders and beginning development on any critical technologies that are necessary for the mission. Long-lead items are identified and procurement may already begin during this phase.

Verification planning is also part of Phase B. A clear approach to system verification is developed, including the overall model philosophy that will guide how prototypes, test models, and flight models are used throughout the project. Business and technical agreements with suppliers may also be set up at this point.

Two key milestones are part of Phase B. The first is the System Requirements Review (SRR), which ensures that all system-level requirements are well defined and aligned with mission objectives. This review leads to the release of the final technical requirements specifications. The second milestone is the Preliminary Design Review (PDR). The successful completion of the PDR indicates that the system design is ready to move into detailed development in Phase C.

By the end of Phase B, final versions of the planning documents, including the Systems Engineering Management Plan, product breakdown structures, and the verification plan are released. The systems engineering activities ensure that the chosen preliminary design is verified to meet technical requirements and that there is a clear and consistent development and verification approach in place for the rest of the life cycle.

Phase C
Phase C is dedicated to completing the detailed design of the system and preparing for its full-scale

manufacturing and qualification in Phase D. It marks the transition from high-level architectural and functional definitions to more detailed technical specifications and design definition. The activities in this phase are guided by the model philosophy and verification strategy established in the earlier phases.

A key objective of Phase C is to finalize the detailed system design. This involves defining all subsystems, components, and interfaces in detail. Internal and external interfaces are reviewed and finalized to ensure that the system can be integrated.

In parallel with the design effort, development testing is carried out on critical system elements and components. This testing is supported by the creation of engineering models that are used to verify design assumptions and reduce risk. The results of these tests provide important input to the qualification and validation of the system.

During this phase, planning for assembly, integration, and testing (AIT) is also finalized. This includes defining how subsystems will be assembled, how the integrated system will be tested, and the procedures that will be followed for manufacturing and testing. Additionally, a preliminary version of the user manual is prepared, describing how the system will be operated.

The final milestone of Phase C is the Critical Design Review (CDR). This review serves as a formal approval of the complete system design. It also includes an evaluation of the AIT plans and the preliminary user manual. The CDR confirms that the design is approved, that all interface definitions are complete and compatible, and that the system is ready to move forward into production and qualification in Phase D. It also assesses whether the verification and validation progress is on track with the project timeline.

From a systems engineering perspective, Phase C ensures that the system design is fully defined and that all technical requirements are addressed in a traceable and verifiable manner. The successful completion of this phase means that the complete system design is in alignment with mission goals and needs and the AIT plan and the planned V&V activities will provide the necessary proof that will lead to complete Qualification and Acceptance of the System.

Phase D
Phase D focuses on completing the system's qualification, producing the flight hardware and software, and preparing the system for operational use. This phase moves the project from design to fully built and tested components, ensuring everything is ready for delivery and launch.

The main technical goal is to finalize all verification activities, including qualification testing to demonstrate that the system meets all applicable requirements. Manufacturing, assembly, and integration of flight units and ground support equipment are carried out during this phase. Interoperability between space and ground segments is also tested to confirm readiness for mission operations.

The Qualification Review (QR) confirms that the system has passed all required tests, and that verification records are complete and accurate to prove final qualification of the system. The Acceptance Review (AR) ensures the final product is free of workmanship errors, all deliverables are in place, and the as-built system matches the as-designed system description. The Operational Readiness Review (ORR) is conducted at the end of Phase D to confirm the readiness of the operations team and the compatibility of operational procedures with the flight system. This review marks the transition to Operations

From a systems engineering perspective, this phase finalizes the development and conducts qualification and acceptance testing of the system, while keeping full traceability between test results, design definition, and requirements on all levels. Furthermore, the preparations for transition into operational phase of the project are performed.

Phase E
Phase E focuses on executing the mission through launch, in-orbit verification, nominal and extended operations, and preparations for eventual end-of-life activities. This phase begins with placing the system into its operational environment and continues through the full utilization of the space and ground segments, ensuring mission objectives are achieved.

The main technical goal is to perform the Launch and Early Orbit Phase (LEOP), complete on-orbit verification and commissioning, and carry out routine mission operations. During this phase, the

system's performance in space is confirmed, and operational procedures are refined and executed.

The Flight Readiness Review (FRR) confirms the readiness of all flight and ground segments, including communication, tracking, and safety systems, for launch. The Commissioning Result Review (CRR) validates that the system meets its in-orbit performance requirements and confirms readiness for routine operations, often marking the formal handover to the satellite operators. At the end of this phase, the End-of-Life Review (ELR) verifies that the mission has completed its service and that the system is in a safe configuration for disposal.

From a systems engineering perspective, this phase involves supporting the launch campaign, assisting the operations team, participating in all major reviews, and providing technical support for any anomaly investigation and resolution throughout the operational period as needed.

Phase F
Phase F focuses on the safe and controlled disposal of all space and ground assets at the end of the mission. This includes the execution of the previously defined disposal plan, ensuring compliance with applicable regulations and minimizing long-term risks to the space environment.

The phase concludes with the Mission Close-Out Review (MCR), which confirms that all disposal activities have been properly completed and that the system has been safely decommissioned.

From a systems engineering perspective, this phase involves supporting the organization responsible for disposal and supporting the completion of the MCR.

Mapping of ISO 15288 Processes to ESA Life Cycle Phases

Based on the description of the ECSS life cycle model a mapping is shown in Table 2.3. This mapping is an indication of which technical processes from ISO 15288 are most relevant during the different ECSS project phases. The ISO processes regarding technical management, organizational project-enabling, and agreement processes are relevant during all project phases. Processes regarding the needs and requirements definition and refinement are most important in the early stages, while architecture and design definition processes are most dominant in phases A, B and C. System analysis is conducted in all project stages, at different levels of detail, to support all other processes. As mentioned above, the purpose of all reviews at the end of each project phase is to validate that the deliverables align with the actual stakeholder needs, so validation is always performed in all phases. The verification process starts already in Phase A. As can be seen in *Table A-1: Systems engineering deliverable documents* of the ECSS-E-ST-10C Rev.1 [12], the verification plan is an expected deliverable for the Preliminary Requirements Review (PRR) at the end of Phase A. While most of the implementation and manufacturing of system components is started in Phase C and completed in Phase D, development of long-lead components can already be started in Phase B. Even though previous phases are already concerned with verification of models of the system and its components, as well as planning for the final verification and validation approach, the majority of the work to actually perform verification and validation is conducted during Phase D.

Table 2.3: ECSS Phases to ISO 15288 technical processes mapping

ISO 15288 Technical Process	Phase 0	Phase A	Phase B	Phase C	Phase D	Phase E	Phase F
Stakeholder Needs and Requirements Definition	✓	✓	✓				
System Requirements Definition	✓	✓	✓				
System Architecture Definition		✓	✓	✓			
Design Definition		✓	✓	✓			
System Analysis	✓	✓	✓	✓	✓	✓	✓
Implementation			✓	✓	✓		
Integration				✓	✓		
Verification		✓	✓	✓	✓	✓	
Validation	✓	✓	✓	✓	✓	✓	
Transition					✓	✓	
Operation						✓	
Maintenance						✓	
Disposal							✓

NASA Life Cycle Model

The life cycle model for NASA missions, on a high level, is very similar to the ECSS one, with some differences that can be seen when looking at it in a bit more detail.

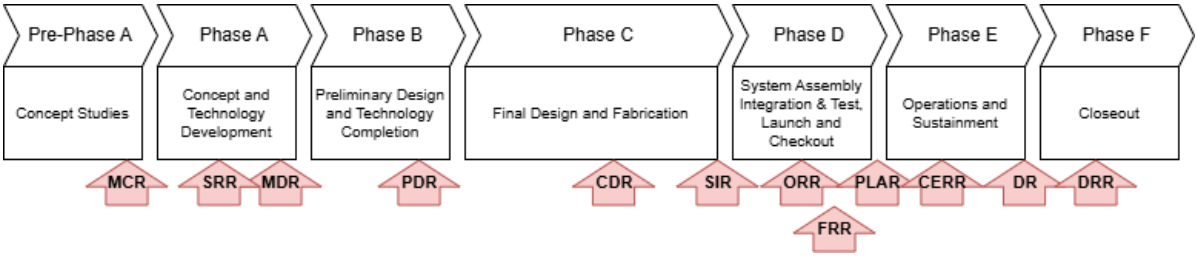


Figure 2.6: NASA Life Cycle Model

Like the ECSS model, the NASA life cycle model consists of 7 phases which follow the same naming pattern, as shown in Figure 2.6. One notable difference is that the Key Decision Points to progress from one phase to the next are not automatically connected to a review, but are separate milestones in the life cycle (not shown in Figure 2.6). The purpose of the different project phases are so similar to the ECSS description, that they will not be presented in detail in this thesis. The added value of considering the NASA life cycle model in the literature review is given by the fact that similarly to the life cycle processes in the ISO 15288 and the INCOSE Systems Engineering Handbook, the NASA Systems Engineering Handbook also describes Technical Development Processes. These are organized into the categories of *System Design Processes* which corresponds to activities on the left side of the Systems Engineering Vee Model and *Product Realization* corresponding to the right side of the Vee. The processes are:

- System Design Process
  - Stakeholder Expectation Definition
  - Requirement Definition
  - Logical Decomposition
  - Design Solution Definition
- Product Realization
  - Product Implementation
  - Project Integration
  - Product Verification
  - Product Validation
  - Product Transition

Additional processes are part of the *Crosscutting Technical Management* category which applies to all project phases, similar to the technical management processes from ISO 15288. The NASA Systems Engineering Handbook states that all Technical Development Processes should be applied during pre-phase A, phase A, and phase B, for each system level, constituting one full Vee model cycle. Phase C and Phase D together constitute a final Vee model cycle, where Phase C includes all activities from the left side, as well as final implementation, while Phase D includes all activities from the right side of the Vee model and finishes with the start of the preparations for operations. The CDR marks an important milestone for the system, which occurs during Phase C, in order to confirm that the design of the system conforms to requirements and stakeholder needs. A CDR should always be conducted before any manufacturing or other implementation activities are started. On system level, the CDR marks the transition from final design activities to full fabrication of the system.

Comparison

Figure 2.7 shows a side-by-side comparison of the presented life cycle models from ISO, ECSS and NASA. To understand the scope of this thesis, it is important to define clearly what is meant by the term "post-CDR" in the context of space missions. In both the ECSS and the NASA life cycle, the CDR marks the final review of the system design. Passing CDR results in confirmation that the system design will be able to meet system requirements and stakeholder needs. After CDR, the development activities focus on fabrication, integration, verification, and validation of the system.

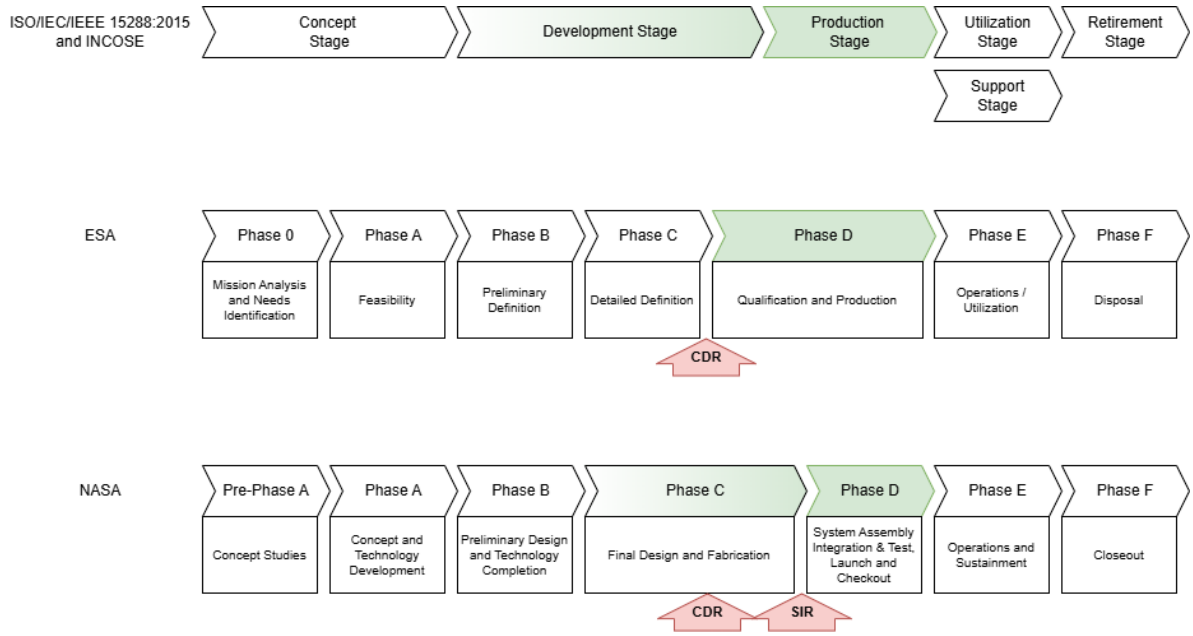


Figure 2.7: Life Cycle Model Comparisons

Requirements Volatility over Project Life cycle
Requirements volatility refers to the percentage of requirements that are added, deleted, or modified over time. This includes requirements creep (i.e. increase in number or scope of requirements) and requirements churn (instability due to frequent modifications). Such volatility is an inherent characteristic of complex system development projects, driven by evolving stakeholder needs and increasing system understanding over time [55].

Requirements volatility is strongly dependent on the system life cycle phase. In early project phases, a relatively high level of volatility is expected and can even be considered beneficial, as requirements are refined and system knowledge matures. As the project progresses toward later phases, the overall level of volatility typically decreases, reflecting an increasing stabilization of the system baseline definition and requirements. However, studies show that volatility does not decrease monotonically, but can exhibit peaks at transitions between life cycle phases, where new insights, inconsistencies, or integration challenges are uncovered [55].

While requirement changes in early phases can often be accommodated with limited effort, changes introduced in later life cycle phases have significantly higher consequences. These changes may require rework and thereby increasing both cost and schedule pressure [55]. This effect is consistent with classical systems engineering observations that the cost and effort associated with changes increase substantially as the system progresses through its life cycle, as presented earlier.

Furthermore, requirements volatility has been shown to directly impact systems engineering effort by increasing the functional size of the system and inducing rework across engineering artefacts. Even seemingly minor changes can propagate through multiple system elements due to interdependencies, leading to cumulative effects on project performance. As a result, requirements volatility is recognized as a major driver of project risk and a key factor influencing cost, schedule, and quality. [55]

This research on requirements volatility conducted by Mauricio Peña and Ricardo Valerdi [55] shows the importance of maintaining end-to-end traceability and managing system complexity are essential throughout the entire project life cycle.

The increasing complexity of modern space systems and their tightly coupled architectures further increases the effects of requirements volatility. Model-Based Systems Engineering (MBSE) approaches have been proposed to address these challenges by enabling traceability, consistency, and impact analysis across system artefacts [4]. This is particularly relevant in later lifecycle phases, where requirement changes propagate across design and verification elements and can lead to significant rework.

2.2.3. Systems Engineering Activities after Critical Design Review

The different ISO life cycle processes and NASA development processes that are applied after CDR in space projects provide an understanding of the role of systems engineering in these project stages. As mentioned above, the main systems engineering activities that are still part of system development, but are conducted after CDR are focusing on system integration, verification, validation, and transition. [63, 13, 50]

In real life projects, the CDR usually happens after implementation of some of the system elements has already started [17, 69]. The end of the system development process is marked by the transition of the final product to the end-user or operator. In the NASA life cycle model, successful transition is marked by passing Operational Readiness Review (ORR). [50]

The INCOSE Systems Engineering Handbook provides an overview of all ISO life cycle processes (see Table 2.2) and their interrelation with each other in the form of a large N2-Chart, which can be seen in Figure C.1, in the annex, which has been modified from the diagram found in the INCOSE Systems Engineering Handbook [64]. The relevant technical processes for post-CDR are indicated in red as belonging to the "right side of the V". Each life cycle process is abbreviated with an acronym, the legend explaining which acronym corresponds with which life cycle process can be found in Appendix C or in the INCOSE Systems Engineering Handbook [64], which also includes a short description of the entities flowing as inputs and outputs between the processes in Appendix E of the Handbook.

Implementation

Purpose of the implementation process is to create the system elements according to their specifications. This process starts with creating and defining the procedures, tools, and equipment to be used for implementation of the system elements. The systems engineers collaborate with stakeholders and domain expert engineers to document any constraints imposed on the system, that originate from the concrete implementation strategy, as well as create implementation plans to ensure access to required resources for implementation. Software and hardware is inspected and verified during implementation to confirm correct functionality. An important decision related to the systems engineering work is whether a system element shall be produced / coded in-house, bought from an external supplier, or reused from previous projects. [63, 64, 50]

If the decision is to buy or reuse, the system element needs to be inspected carefully against the specifications, and any required adaptions are done. For components that are produced in-house, the development should be supported from a systems engineering perspective by keeping track of implementation progress, especially for elements on the critical path in the implementation schedule. [63, 64]

The input and output relations to other ISO life cycle processes are shown in Table 2.4. The artifacts are listed, with the source and target processes indicated by their acronyms in brackets.

Table 2.4: Inputs and Outputs for the Implementation Process, from [64]

Implementation Inputs	Implementation Outputs
Life cycle concepts (BMA, SNRD)	Implementation strategy/approach (PP)
System viewpoints, views and models (SRD, SAD, DD)	System element (INT)
System architecture description (SAD)	System element description (INT)
System architecture rationale (SAD)	Training materials (TRAN)
System design description (DD)	Constraints on solution (DM, BMA, SNRD, SRD, SAD, DD, SA)
System design rationale (DD)	Requirements imposed on enabling systems (ENAB)
System interface definition (SAD, DD, INT)	Traceability mapping (ENAB)
System design characteristics (DD)	Implementation report (PAC)
System element description (DD)	Implementation records/artifacts (KM)

Integration

After implementation of the system elements from their specifications, the integration process focuses on building up the SoI by combining its elements into one system. The system elements can be software, hardware, or operational components. A major part of the work of a systems engineer during integration is to make sure that the interfaces between system elements are correctly implemented and conforming to interface requirements. This process runs in parallel to the verification process, to check interfaces, functionality and critical characteristics before and during system integration. [64]

Typically, systems engineers are also tasked with tracking conformance to requirements during integration and addressing any non-conformance that might come up. It is important that bidirectional traceability is maintained during integration between the integrated system, the design definition, the system requirements and the integration plans and procedures. [64, 50]

The input and output relations to other ISO life cycle processes are shown in Table 2.5. The artifacts are listed, with the source and target processes indicated by their acronyms in brackets.

Table 2.5: Inputs and Outputs for the Integration Process, from [64]

Integration Inputs	Integration Outputs
Life cycle concepts (BMA, SNRD)	Integration strategy/approach (PP)
System architecture description (SAD)	Integration procedure (PAC)
System architecture rationale (SAD)	Integrated system or system element (VER)
System design description (DD)	System interface definition (INT, TRAN)
System design rationale (DD)	Constraints on solution (DM, BMA, SNRD, SRD, SAD, DD, SA)
System interface definition (SAD, DD, INT)	Requirements imposed on enabling systems (ENAB)
System element (IMPL)	Traceability mapping (ENAB)
System element description (DD)	Integration report (PAC)
Accepted system or system element (ACQ)	Integration records/artifacts (KM)
Reused system or system element (KM)	

Verification

Purpose of verification is to generate evidence that the system conforms to its requirements and characteristics [37]. The outputs of the verification process, according to [64] prove:

- that the system that was verified "has been made right", i.e. according to specifications [50, 64, 37],
- that no errors, defects, or faults are exhibited when operating the system.
- that the selected verification strategy, method, and procedures are suitable to detect any anomaly in the system, should an anomaly occur.

Systems Engineers prepare for the verification process by identifying the verification scope, defining the requirements to be verified during different verification activities, and supporting verification planning and helping define the overall verification strategy. Each verification activity can consist of multiple verification actions, while each verification action needs to specify at least one verification method and associated success criteria. The ECSS specifies the following verification methods [9]:

- Test
- Analysis
- Review of Design
- Inspection

Depending on the life cycle stage, the verification method can differ for a given requirement. Space systems go through different stages of verification, which adds another dimension to the verification strategy. Typical verification stages include qualification, acceptance, pre-launch, and in-orbit / commissioning verification. Depending on the verification stage, different test levels can be applied, e.g. qualification levels and acceptance levels in a satellite mission. [9]

The input and output relations to other ISO life cycle processes are shown in Table 2.6. The artifacts are listed, with the source and target processes indicated by their acronyms in brackets.

Table 2.6: Inputs and Outputs for the Verification Process, from [64]

Verification Inputs	Verification Outputs
Life cycle concepts (BMA, SNRD)	Verification strategy/approach (PP)
Verification criteria (SNRD, SRD, VER)	Verification criteria (VER)
System requirements (SRD)	Verification procedure (PAC)
System architecture description (SAD)	Verified system requirements (SRD)
System architecture rationale (SAD)	Verified systems architecture and design (SAD, DD)
System design description (DD)	Verified system (TRAN, VAL)
System design rationale (DD)	Other verified artifacts (PAC)
System interface definition (SAD, DD)	Constraints on solution (DM, BMA, SNRD, SRD, SAD, DD, SA)
Integrated system or system element (INT)	Requirements imposed on enabling systems (ENAB)
Integration report (INT)	Traceability mapping (ENAB)
	Verification report (PAC, TRAN, VAL)
	Verification records/artifacts (KM)

Validation

In contrast to verification, the purpose of validation is to provide objective evidence that the system, when used under operational conditions, fulfills its intended use, mission objectives, and stakeholder needs and requirements [37]. Validation ensures that "the right system has been made," i.e., that the delivered system performs as expected in its actual operational environment and delivers the outcomes as desired by its stakeholders [64].

Similar to verification, the validation process begins with establishing a clear validation scope, identifying the stakeholder needs and operational objectives to be validated in different validation activities. Any additional constraints or objectives from the validation strategy that should be reflected within the set of stakeholder requirements need to be identified and captured. Planning for validation includes defining the steps to be performed for validation, the required enabling systems, ensuring access to the operational environments, or suitable analogues, and selecting appropriate validation methods along with associated success criteria. [50, 63, 64]

Each validation action typically corresponds to a specific system usage scenario and is realized through the execution of one or more validation procedures. As in verification, these procedures must specify the validation method, the conditions under which validation is successful, and the expected outcomes. However, validation differs from verification in that validation always involves checking against stakeholder needs. Validation activities are typically conducted at key milestones of the system life cycle. [50, 63, 64]

Traditionally, the results of validation actions are captured in a validation report. Any identified defects, deviations from expected behavior, or operational incidents must be tracked and resolved. Additionally, traceability must be maintained between validated system elements and the associated validation artifacts, such as procedures, results, and stakeholder requirements/needs, to ensure full coverage across the system life cycle [37, 63, 64].

The input and output relations to other ISO life cycle processes are shown in Table 2.7. The artifacts are listed, with the source and target processes indicated by their acronyms in brackets.

Table 2.7: Inputs and Outputs for the Validation Process, from [64]

Validation Inputs	Validation Outputs
Life cycle concepts (BMA, SNRD)	Validation strategy/approach (PP)
Validation criteria (SNRD, VAL)	Validation criteria (VAL)
Stakeholder needs and requirements definition (SNRD)	Validation procedure (PAC)
System architecture description (SAD)	Validated stakeholder needs and requirements (SNRD)
System architecture rationale (SAD)	Validated system architecture and design (SAD, DD)
System design description (DD)	Validated system (SUP, TRAN)
System design rationale (DD)	Other validated artifacts (PAC)
System interface definition (SAD, DD)	Constraints on solution (DM, BMA, SNRD, SRD, SAD, DD, SA)
Verified system (VER)	Requirements imposed on enabling systems (ENAB)
Verification report (VER)	Traceability mapping (ENAB)
Installed system (TRAN)	Validation report (PAC, TRAN)
Transition report (TRAN)	Validation records/artifacts (KM)

Transition

Transition is the last step before the end system is being operated in its operational environment or a lower level end product is integrated into a higher level system. All remaining necessary preparations for operations or higher level integration are made during the transition process. At system level, the transition process can include a handover from the developer to the end-user, e.g. from the satellite developer to the satellite operator in the case of a satellite mission. Key documentation that is finalized during the transition process is the operations manual. When the transition process is applied to lower system levels, i.e. for subsystems and components, the transition for the subsystem effectively runs in parallel with the preparations for the integration process at system level. [50]

The input and output relations to other ISO life cycle processes are shown in Table 2.8. The artifacts are listed, with their source or target processes indicated by their acronyms in brackets.

Table 2.8: Inputs and Outputs for the Transition Process

Transition Inputs	Transition Outputs
Life cycle concepts (BMA, SNRD)	Transition strategy/approach (PP)
System interface definition (SAD, DD)	Installation procedure (PAC)
Verified system (VER)	Installed system (SUP, VAL, OPER)
Verification report (VER)	Trained personnel (OPER, MAINT, DISP)
Validated system (VAL)	Constraints on solution (DM, BMA, SNRD, SRD, SAD, DD, SA)
Validation report (VAL)	Requirements imposed on enabling systems (ENAB)
Training materials (IMPL)	Traceability mapping (ENAB)
	Transition report (PAC, VAL, OPER)
	Transition records/artifacts (KM)

NASA Product Realization Processes Input-Output Relationships

Similarly to the input output relations presented in the INCOSE Handbook, the NASA Systems Engineering Handbook also discusses the topic. Figure 2.6 shows a simplified flowchart like diagram that indicates input output relations between the five technical processes related to realization of the product. It shows clearly the purpose of the transition process to make a validated (sub)system available for integration into the next system level. To this end, the transition process for example provides enabling products for the next iteration of product realization processes.

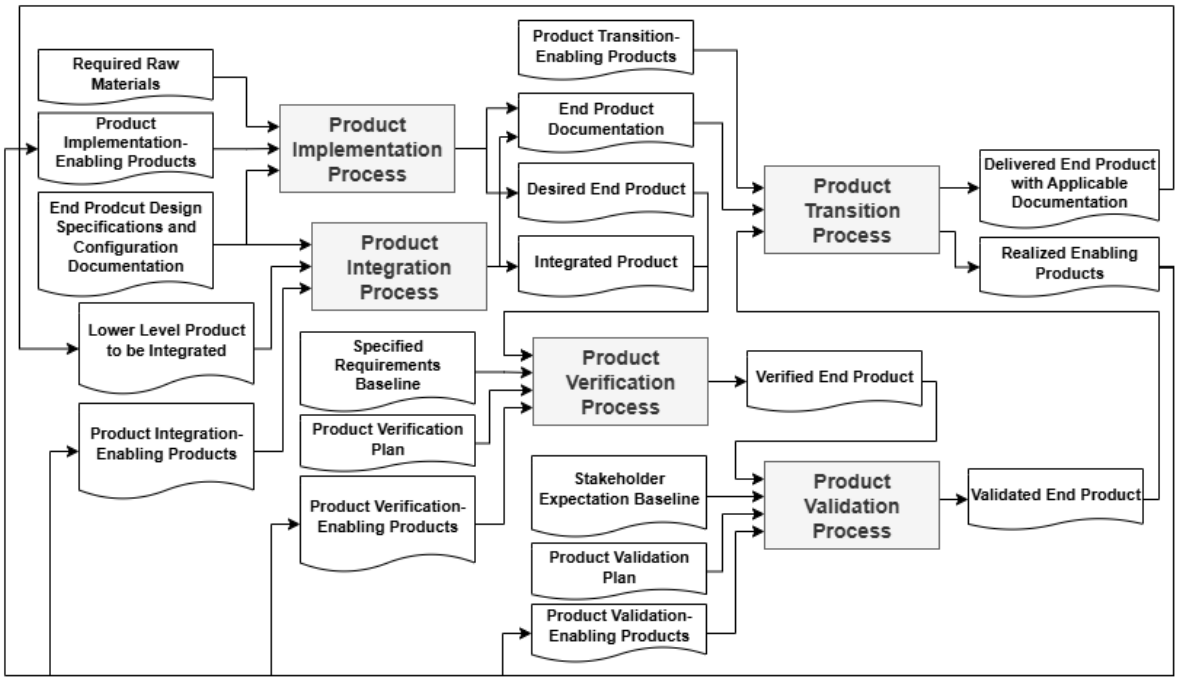


Figure 2.8: NASA Product Realization Processes, from [50], adapted by the author

2.2.4. Reviews after CDR

After passing CDR, the systems engineering work in space projects will focus on preparing for the next reviews. In the ESA life cycle, the next reviews are the Qualification Review (QR), the Acceptance Review (AR), and the Operational Readiness Review (ORR). In the NASA life cycle, the reviews following the CDR are the Production Readiness Review (PRR), System Integration Review (SIR), System Acceptance Review (SAR) and Operational Readiness Review (ORR). The purpose of the Reviews specified in the ECSS and NASA literature is summarized in Table 2.9. The main difference between the reviews specified in ECSS and NASA standards is that the ECSS specifies a Qualification Review, while at NASA, there is a System Integration Review instead. The System Integration Reviews' purpose is to confirm that the system is ready to undergo integration. Such a review is not explicitly mentioned in the ECSS life cycle process, but partially included in the CDR. The ECSS introduces the Qualification Review after qualification testing is completed. A dedicated review for that purpose is not included in the NASA life cycle by default, but covered during the Acceptance Review, which exists in both the ECSS and the NASA life cycle. The Reviews from Table 2.9 focus primarily on the Assembly, Integration, Verification, and Validation efforts of the project, which is consistent with the previously discussed project life cycle models used at NASA and ESA.

Table 2.9: ECSS and NASA Reviews description

ECSS		NASA	
Review	**Purpose**	**Review**	**Purpose**
CDR	• Assess the qualification and validation plans for phase D • Confirm compatibility with external interfaces • Release final design • Release manufacturing, assembly, integration, and test planning	CDR	• Product baseline, production plans, and verification plans are approved • Authorization for system qualification testing and integration
		SIR	• Integration Plans and procedures are on track for completion • Integration workflow is clearly defined and documented. • All interfaces have been tested to ensure that integration can proceed successfully
QR	• Confirm that verification process has demonstrated that the design meets applicable requirements.		
AR	• Confirm that product is free of workmanship errors and ready for operational use • Verify the as-built product against as-designed product • Authorize delivery of the product	SAR	• Tests show that system will perform in the expected operational environment • System meets acceptance criteria • Lessons learned for organizational improvement and operations are captured • All test results are available, showing that Verification and Validation is passed
ORR	• Verify readiness of the operational procedures and compatibility with the flight system • Verify readiness of operations team • Accept and release ground segment for operations	ORR	• Confirm that system is ready to assume normal operations • All supporting systems and personnel is ready for operations • Operations plans, procedures and schedules are consisten with mission objectives.

ECSS and NASA defines the set of documents that are required for entering the Qualification Review. These documents give an indication on what information is expected to be presented and discussed during the reviews. For the ECSS, all the required documents are indicated in Table A-1 of the ECSS-E-ST-10C Rev.1 [12], and for NASA reviews all documents are listed in Table 5-1 of the NPR 7123.1D NASA Procedural Requirements Document [51]. Both the ESA and NASA standards show that each review after CDR is concerned with the same type of documents, just at different stages of completion. For the purpose of this thesis, the focus will lie mostly on the reviews that are directly following the CDR, meaning the Qualification Review in the ECSS and the System Integration Review in the NASA standards. In both cases, important documents to be delivered include the Verification Plan, Test Specifications, Procedures and Reports, and Documents for Verification Control. These will be discussed further in subsection 2.3.2.

2.3. Assembly, Integration, Verification, and Validation

In space projects, the tasks related to verification, validation and testing are usually associated with the role of an AIV Engineer. Even though the name *AIV Engineering* could be misinterpreted as being involved only during the later phases of a project, when the system is assembled and tested, in reality AIV Engineers are involved during all phases of the project life cycle. During the earlier stages, the most amount of work is spent on planning and optimizing the V&V approach and planning and making use of early (digital or physical) prototypes to conduct early verification of the system. Once the system design is matured, the project is moving into the verification and testing phase in order to provide proof that the developed system complies with requirements and is able to fulfill the customer needs. [69, 17]

2.3.1. V&V common principles

There are common AIV principles stated in literature, that describe the working processes for AIV engineering, which are applied during the whole project life cycle. These common principles and processes are explained in the next section.

Book - Guide to Verification and Validation [69]

In their Book *Guide to Verification and Validation*, Wolfgang et al. define several stages to categorize the verification and validation efforts. These stages are the Planning, Definition, Execution Stage, Reporting, and Approval. They need to be considered throughout the project life cycle by the AIV Engineers.

The first step of **Planning Stage** for V&V is to establish the success criteria for the verification and validation efforts. The success criteria can originate for example from stakeholder needs and requirements, organizational standards, qualification, acceptance and certification requirements and the design outputs. Another related and important aspect of V&V success criteria is to ensure that needs statements and requirements are verifiable.

After the success criteria are established, the overall V&V strategy is determined or updated. A key aspect in the strategy is to balance the insights achieved through testing with cost-effectiveness of the overall approach. The authors suggest a risk-driven approach to deciding the depth and scope of the V&V activities, taking into account possible impacts and uncertainties on product performance, project schedule, and overall project cost. Furthermore, the V&V strategy needs to be aligned with any external parties, like suppliers and facility providers. Other activities include setting up and updating the verification and validation matrices, which provide means to control the status of verification and close-out of requirements and to select the verification and validation methods to be used for each requirement. The management of V&V attributes is a crucial part of monitoring the V&V efforts of a project. The authors recommend a digital tool that enables automation and mention MBSE as one option for such a tool. According to the authors, parameters to track per requirement usually include the following

- Need/requirement ID and description
- Success criteria
- Strategy reference
- Verification / Validation Method used
- Verification activity ID
- Event ID (for project schedule)
- Hardware maturity (e.g., development model, prototype, production)
- Preliminary or final activity status
- Integration level (in system architecture)
- Responsible organization/institution
- Priority from a stakeholder perspective (high, medium, low)
- Criticality (essential or not for the system to achieve its main purpose)
- Safety/security risk association
- Date and time when the activity is performed
- Serial numbers or other identifiers of any hardware used.

The **Definition Stage** of V&V ensures that each system element, subsystem, and interface of the System of Interest (SOI) will be properly verified and validated before being integrated into higher-level assemblies.

The authors recommend conducting V&V activities at each hierarchical level of the system architecture prior to integration. This is especially critical for verifying interfaces, which are common sources of integration failure. Model-Based Systems Engineering (MBSE) tools can support this effort by enabling early modeling of interface behavior, and by revealing hidden interfaces and interactions.

Each V&V activity must be formally documented. These Verification Activity Definitions form the foundation for the development of corresponding verification or validation procedures. Each activity definition includes key attributes such as the verification method, success criteria, responsible organization. Verification activities are derived from the requirements that are to be verified or validated. The description of the V&V activity also specifies the equipment and facilities used, data collection and evidence recording protocols, and the roles and responsibilities of all participants. This forms the basis for development of the V&V procedures. The procedures themselves clearly define steps and actions, often in the form of checklists and may invoke other pre-existing procedures or routines (e.g., equipment power-up or configuration steps).

Each procedure is structured to gather the data necessary to demonstrate that the defined success criteria have been achieved. Once developed, procedures are reviewed and, in more complex or safety-critical situations, dry runs may be performed. Only after successful review and validation are the procedures approved, baselined, and the V&V activity is performed.

The **Execution Stage** marks a crucial stage in the engineering development life cycle, as it involves performing the actual verification and validation, following the previously defined procedures.

During this stage the project is faced with the practical realities of engineering, where design flaws and other discrepancies and issues are expected to occur eventually. These deviations can range from minor issues to significant failures in functionality or performance. When encountered, such discrepancies must be documented, investigated, and resolved through corrective actions or compensatory measures.

Formal change impact analyses become essential when discrepancies arise that require design changes. These analyses assess how modifications might affect verified or validated requirements and whether re-verification or re-validation is necessary. For systems under strict configuration management, any changes must be systematically tracked and justified.

Furthermore, when a system does not fully meet its success criteria, options include correcting the design, modifying the requirement, or requesting a waiver or concession. These formal processes typically involve internal quality assurance teams and may also require customer or regulatory approval, particularly for safety-critical systems. The success of these requests depends on transparency, early communication, and thorough impact assessment.

Ultimately, the execution stage is as much about governance as it is about engineering. The ability to manage imperfections while maintaining stakeholder trust is an indicator of successful engineering programs.

The **Reporting Stage** focuses on documenting the outcomes of all V&V activities. After executing the procedures, results and evidence are recorded to establish a traceable record for each activity. This documentation is crucial to confirm that the system-of-interest meets its requirements and stakeholder needs, while also preventing costly re-testing later in the life cycle.

A key concept in this stage is the *chain of evidence of compliance*. For high-risk or safety-related needs, extensive documentation may be necessary, often guided by predefined compliance report templates.

Compliance matrices are widely used to consolidate and visualize verification results. These matrices link each requirement to its verification method, success criteria, evidence, and status. They provide a comprehensive view of requirement coverage and are typically submitted as part of the final V&V deliverables for certification or acceptance.

During the **Approval Stage**, the final acceptance of the System of Interest is issued after successfully completing all verification and validation activities with no remaining open issues. Acceptance requires documented evidence that the SOI meets all necessary requirements, with any exceptions and deviations formally agreed upon by the approval authority.

Late changes, waivers, or non-compliances can complicate approval and should be minimized and well-documented with impact assessments. Consistent systems engineering practices and open communication with customers or regulators help ensure a smooth approval process. Ultimately, final acceptance confirms that the SOI meets its intended needs.

ECSS Verification Process [9]
The ECSS-E-ST-10-02C Standard on Verification specifies the high level process and activities as shown in Figure 2.9. The overall purpose of this process is to demonstrate the qualification of the product, to meet its requirements, that the product is free of workmanship defects, and able to fulfill its mission objectives.

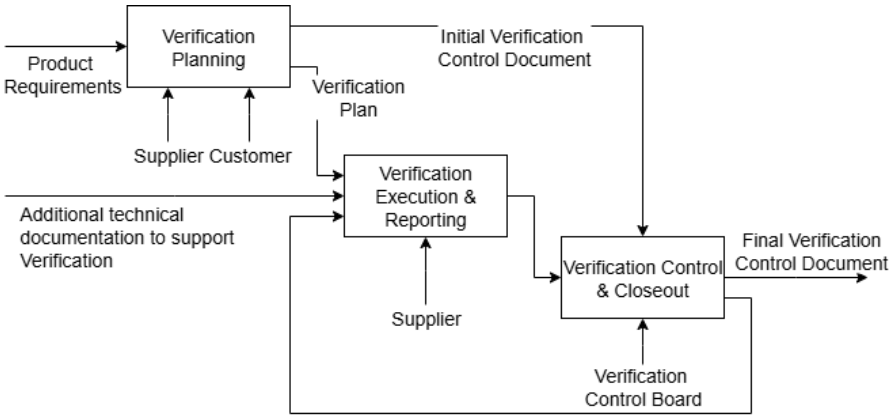


Figure 2.9: Verification Process and Activities [9]

Verification Planning includes specification of the overall verification approach, verification tools, methods, environment and facilities, and the corresponding costs and schedule estimates. The planning process follows a *What - How - When* approach, starting with defining what products and requirements are to be verified, then specifying how to verify them including which verification methods to use and identifying the necessary Ground Support Equipment. Lastly, an overall timeline of all verification activities is created and maintained. Additionally, the overall model philosophy is defined. This specifies the number of additional models to be developed, next to the final flight model, for the system and for system elements, as well as their purpose.

For each verification activity the following information needs to be specified:

- Verification Method: Test, Analysis, Review of Design, and/or Inspection
- Verification Level: corresponds to the system hierarchy level, e.g. system, subsystem, equipment.
- Verification Stage: life cycle stage in which the verification activity will take place. This can be for example qualification, acceptance, or in-orbit.

Verification Execution and Reporting happens continuously while the verification activities are performed. Usually this happens incrementally, starting at lower system levels following a bottom-up approach. A more detailed overview of the usual documents for verification as per ECSS is given in subsection 2.3.2.

Verification Control and Closeout The overall progress of the verification and validation activities needs to be controlled and a holistic overview is kept using Verification Control Documents.

Verification, Validation, and Testing (VVT) Process [17]
The authors state the importance of effective Verification, Validation, and Testing (VVT) strategy to ensuring the technical success of any development program. By initiating VVT activities early and integrating them throughout the system life cycle, programmatic risks are reduced and costly rework is minimized. Well-planned VVT helps avoid redundant testing, and thus lowering costs and conserving resources. Additionally, continuous attention to VVT enables better control over quality and alignment with project objectives. To that end, Engel presents the VVT methodology which is shown in Figure 2.10 and presented in the following.

The guiding philosophy behind VVT emphasizes the principle to conduct verification early and continuously in the project. This mindset ensures that potential issues are identified and addressed before they escalate. A successful VVT process harmonizes technical considerations with programmatic

constraints, promoting efficient resource use and risk management. Ultimately, this approach supports cost savings, maintains project schedules, strengthens component and subsystem integration through robust interface validation, and fosters stakeholder confidence by validating true needs early enough to enable meaningful course correction.

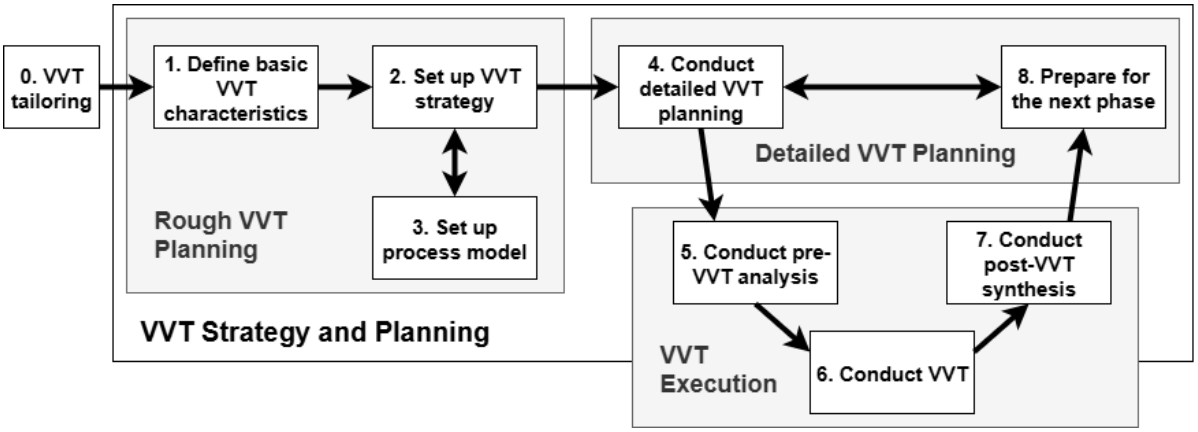


Figure 2.10: Verification, Validation, and Testing (VVT) Methodology for Strategy and Planning, from [17]

VVT Strategy and Planning Steps

The VVT process begins with *VVT Tailoring*, where project and enterprise-specific factors are assessed to adapt the VVT methodology to the organizational context, using lessons learned from past projects. Next is *Rough VVT Planning*, which establishes a high-level VVT strategy early in the project by aligning with business objectives, identifying potential risks, and addressing programmatic constraints. This stage involves three key sub-steps: defining basic VVT characteristics to guide the strategy, setting up the VVT strategy by selecting appropriate activities and verification methods, and establishing a VVT process model that uses cost, time, and risk analyses to evaluate alternative strategies.

Following this, *Detailed VVT Planning* is conducted at the beginning of each lifecycle phase, refining the overall plan and specifying the activities, tools, levels of effort, and the required formality for execution. The *VVT Execution* phase implements the strategy across lifecycle stages through a structured process: conducting a pre-VVT analysis to incorporate recent developments, executing the planned VVT activities, and performing a post-VVT synthesis. Finally, the generated results are used as a basis to inform and improve future VVT planning. This iterative approach ensures that VVT remains aligned with evolving project needs and constraints throughout the system life cycle.

2.3.2. AIV Documentation

The following section discusses different documentation that is related to AIV and testing. First, the relevant standards from the ECSS will be reviewed, followed by the IEEE-829-2008 [33].

Figure 2.5 shows all required documents that need to be delivered for the Qualification Review. As mentioned in the previous chapter, the Qualification Review focuses primarily on the verification efforts conducted, and is the next major review in the ECSS life cycle after CDR. As a result, there are multiple documents involved in Qualification Review that directly relate to the verification, validation and testing activities. These documents are highlighted in blue in Figure 2.5.

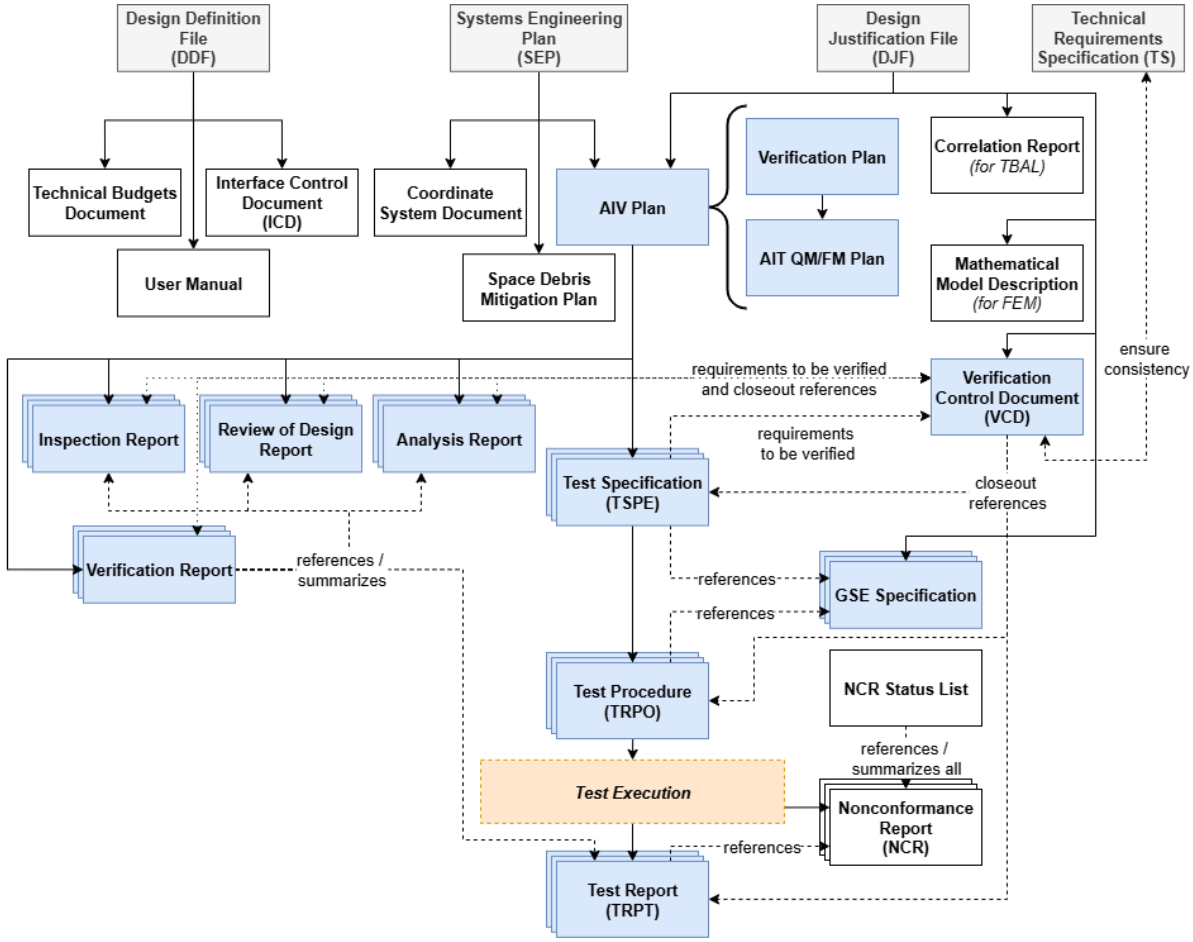


Figure 2.11: ECSS - Documents required for Qualification Review (QR)

All documents shown in Figure 2.5 are explained further in Appendix A in section A.1, including also the reference to the specific ECSS standard and the corresponding DRD.

The main documents related to testing and test execution are the AIV Plan, the Test Specification, the Test Procedure, the Test Report, the GSE specification, and the Verification Control Document.

The AIV Plan provides the highest level overview over the verification approach for the project and how compliance with requirements will be demonstrated overall. It specifies the model philosophy to be used and explains the means to control verification and validation status throughout the project, i.e. explains how the project specific VCD shall be used. The AIV Plan furthermore describes the timeline and planning for all verification activities of the project, including logistics and scheduling. Additionally, it includes requirements on all documentation that will be created during verification and testing, and it provides an overview of organizational roles and responsibilities. The AIV Plan can either be one document, or it can be split into a Verification Plan and an AIT Plan, which together contain the same information. [9]

Based on the AIV Plan and for each planned test, a Test Specification is created that further describes the activity. Note that this is only done for test and not for the verification methods Inspection, Review of Design, or Analysis. The Test Specification defines the detailed requirements for the test activity. This includes the purpose and overall approach for the test, all requirements that are to be verified, the configuration of the item under test and any required Ground Support Equipment, instrumentation, and applicable constraints on testing facilities. The required test conditions and a high level sequence of steps performed during the test is defined, as well as other organizational aspects like the responsibilities of different engineers involved, and scheduling of the test activity, in accordance with the AIV Plan. The Test Specification serves as a baseline to define the detailed test procedures. [10]

The Test Procedure provides detailed step-by-step instructions on how to execute the test activity. Overlap between the Test Specification and Test Procedure should be avoided. Within the step-by-step procedure, pass / fail criteria are included along the way, with the purpose of verifying single requirements during the test. The Test Procedure is created in a way such that it can be filled out while performing the test, by the AIV or the Quality Assurance Engineers, to become the "as-run" test procedure. This "as-run" test procedure serves as a detailed record of everything that happened during the test, including all observations made. Any deviations from the "as-written" test procedure need to be clearly indicated and any deviation from expected results or any mishaps that occur are captured in a Nonconformance Report (NCR) such that these aspects can be tracked until resolution. [10, 15]

After the test is conducted, the results are captured in the Test Report and a final verdict is made on the closeout status of all requirements that were planned to be verified during the test. Any anomalies that occurred during testing are mentioned in the test report and the Test Specification, Test Procedure and any NCRs are referenced. Once the verification closeout status is determined, it can be updated in the VCD. The VCD provides an overview over all requirements and their status of verification. It is created in conjunction with the AIV Plan and continuously updated to reflect the progress of verification for the whole project. VCDs are usually used in the ECSS life cycle phases C and later. [9]

The IEEE 829-2008 [33] Standard for Software and System Test Documentation, also specifies a similar set of documents related to test planning execution and reporting. The documents and their relationships are shown in Figure 2.12 and a table explaining all documents shown in the figure can be found in section A.2. As in the ECSS, the IEEE 829 also specifies one Master Test Plan which includes the overall test flow, organizational aspects like scheduling, prioritization of tests, roles and responsibilities, and requirements on how to conduct documentation.

For each test level there is a dedicated Test Plan created, which defines the scope of the tests on that level, the items required for the test and their configurations, what requirements and features are supposed to be tested in the form of a traceability matrix, the concrete deliverables to be produced, pass/fail criteria per test, and management and quality assurance aspects. These plans need to be kept consistent with the Master Test Plan.

The tests that appear in the Test Plan can be further detailed by a Test Design document. The Test Design describes for a set of related Test Cases, a more elaborate relation between them. This extra layer of test planning can be used to specify structural and sequencing relations between Test Cases and might include elaboration of technical details. In the context of satellite testing, a test design might be used to plan an environmental test campaign which includes multiple test cases with different test setups.

Each Test Case defines a single test objective and how it will be evaluated. This includes inputs and expected outputs of the test and a detailed description of the test environment and hardware and software setup for the test. A Test Procedure provides the step-by-step description of how to execute one or multiple test cases, this also covers all steps necessary to perform logging and measurements of needed results. The results of a test are captured in Test Logs and Anomaly Reports. For each test level and for the overall testing efforts, the results are eventually summarized.

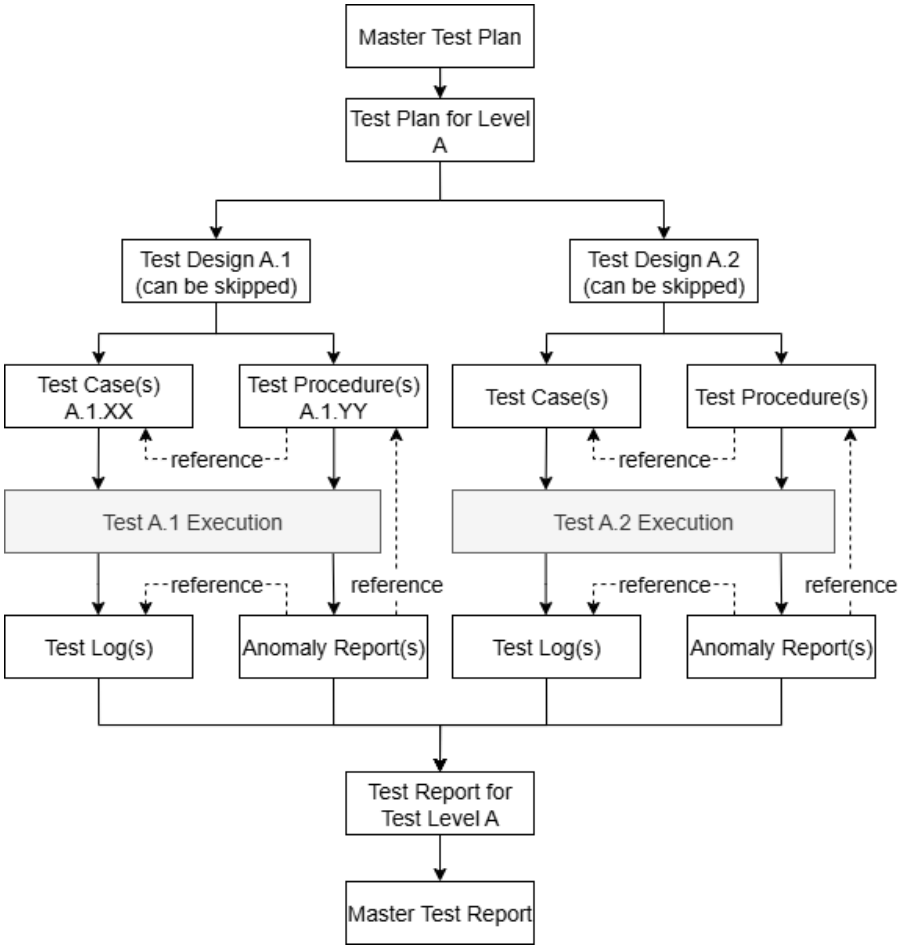


Figure 2.12: Software and System Test Documentation overview, adapted from [33]

2.4. Model-Based Systems Engineering

2.4.1. Definition and Benefits of MBSE

In the traditional systems engineering approach, information is usually captured in a set of documents. In contrast to that, model-based systems engineering (MBSE) manages all relevant information in a central, digital system model. The creation of this digital model is equivalent to developing the system itself [2]. Alt defines the terms abstraction, view, and model as central concepts to explain MBSE. This concept will be explained briefly below.

The concept of abstraction means to Alt the targeted action of omitting irrelevant information. In the context of model-based development, abstraction allows to focus on the essential characteristics of a system. [2]

A view contains only certain information that is relevant for a specific observer, who aims to consult the model to gain insights into a distinct aspect of the system. In other words, the observer looks at the system model from their own perspective and the model view uses the concept of abstraction to provide only the relevant information to the observer. [2]

Lastly, Alt defines the MBSE model as an abstract and formal description of reality, that can be used to automatically generate other artifacts. The information within an MBSE model can be interpreted and processed by a computer. [2]

INCOSE defines MBSE as a specific kind of systems engineering, which formalizes the use of models over the course of a complete development process to perform systems engineering tasks [35, 25]. Weilkiens adds to this that MBSE models represent the system and its environment using a modeling language that covers systems engineering concepts and agrees with Alt, that applying MBSE means

that information is documented in a way that is interpretable by computers [68, 66]. In comparison to the definition of MBSE provided by Alt, the definitions by INCOSE, Friedenthal, and Weilkiens include the fact that MBSE can be applied throughout all project life cycle phases directly in the definition of the term MBSE. Alt also acknowledges the need to consider verification and validation activities in his publications, with the small distinction that this is not explicitly stated in his definitions of systems engineering and MBSE.

Delligatti highlights that in both the traditional, document-based systems engineering approach and the model-based approach, the systems engineers perform the same life cycle activities and produce the same set of deliverables. The main difference between the approaches lies in the way these deliverables are documented. While applying MBSE, the deliverables are not created as stand-alone artifacts, but are integrated into a coherent and consistent system model, using a dedicated modeling tool. Documents and other artefacts are then generated from the MBSE model automatically. [7]

In this thesis, the definition of MBSE according to INCOSE, Friedenthal, and Weilkiens will be used, defining MBSE as the "formalized application of modeling to support system requirements, design, analysis, verification, and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases". Application of MBSE produces a coherent system model that constitutes the engineering baseline [34, 25, 66]. This definition makes it clear that applying MBSE means to do Systems Engineering tasks in a specific way. The underlying tasks and responsibilities of a Systems Engineer do not change, whether they use MBSE or a traditional approach to Systems Engineering.

Benefits of MBSE

As described above, applying MBSE differs from traditional Systems Engineering approaches in the sense that it places integrated models of the System of Interest at the center of the development process [7]. When correctly applied, MBSE can provide key benefits during the development of systems as compared to document-based systems engineering approaches. These benefits, as described by Friedenthal, Delligatti, and INCOSE, are discussed in this section.

Improved Product Quality Through Rigorously Enforced Modeling Process

MBSE improves the quality of the developed system by enforcing a more disciplined and integrated design approach. With a central system model capturing all design elements, design decisions can be explicitly documented and linked, ensuring consistency and traceability of all model elements across the entire system model. This significantly improves the ability to manage the increasing complexity of technical systems [24]. Applying MBSE allows for a more consistent capture of information within the team, leading to less ambiguous, more precise system model that can be evaluated for consistency, correctness, and completeness [63].

Enhanced Communication Through a Shared System Model

Instead of disconnected documents, MBSE provides a shared, centralized system model. A common, consistent source of information provides a baseline for efficient and effective communication and provides a shared understanding of the system between stakeholders. This shared information baseline reduces misunderstandings, aligns expectations, and supports more informed decision-making throughout the life cycle process. [24, 63]

Reduced Development Risk

During the development of the system, the MBSE model can be checked continuously, from the earliest stages of the life cycle process. This additional insight, though continuous model review, analysis, and simulation, can reduce the risk of expensive reworks later on. Design alternatives can be evaluated systematically to guide the design and arrive at a balanced overall solution. [24]

Increased Productivity Through Automation and Reuse

MBSE can boost efficiency of engineers by automating the generation of documentation from the system model, enabling reuse of model elements and entire aspects of models, and enabling faster, more thorough impact analysis. Instead of manually updating multiple documents, engineers make changes

in the model, which are then propagated throughout all derived system artifacts. This saves time, by reducing manual effort, and reduces errors, which in turn speeds up the development. [24]

Improved Knowledge Capture and Transfer

MBSE enables and requires the explicit capture of domain knowledge within the system model, which can then be used, shared, and understood across teams and projects. This can reduce the dependence on individual experts, by making critical system knowledge accessible and persistent. This has the potential to enhance learning, onboarding, training, and long-term knowledge preservation within a project or organization. [24, 63]

Constituents of an MBSE Methodology

There are different explanations found in literature what constitutes an MBSE methodology. These sources will be presented, compared, and finally combined to derive an understanding of important elements for an MBSE methodology which will be used for this thesis.

Delligatti defines three pillars of MBSE. These pillars are the modeling language, the modeling method, and the modeling tool. All three aspects must be coordinated with each other and with the specific MBSE project and the environment in which this project takes place. [7]

The modeling language is a standardized notation for system elements and relations between them. The language is used to standardize communication within the MBSE project. Examples of modeling languages are Systems Modeling Language (SysML) [53], Unified Modeling Language (UML) [54], Business Process Model and Notation (BPMN) [52]. The modeling method defines a specific way of modeling. The modeling tasks, their sequence, and how the modeling tasks are to be solved using the modeling language and the modeling tool are defined in the method. A modeling tool is a software tool that allows you to work with one or more modeling languages and supports the creation of MBSE models. [7]

According to Walden et al. in the INCOSE Systems Engineering Handbook, an MBSE methodology describes how MBSE is used to capture the necessary information within the system's model and its associated artifacts. The methodology needs to be compliant with the particular needs of the project and organization. To ensure that this is the case, an appropriate life cycle model needs to be defined, including a tailored set of activities and work products which align with the project scope and modeling objectives. Furthermore, an MBSE methodology needs to specify the appropriate tools to create and manage models and any other related data. [63]

In his paper *Survey of Model-Based Systems Engineering (MBSE) Methodologies*, Estefan uses the definition of the term methodology as defined in [46] as a collection of processes, methods, and tools, to explain and compare different MBSE methodologies and how they support systems engineering by using a model-based approach [22]. In contrast to Delligatti, Estefan emphasizes the hierarchical relationship between MBSE processes and methods. In this context, a process is a sequence of tasks that need to be fulfilled to perform systems engineering work, describing what needs to be done, but not how it is supposed to be done. The individual process steps are carried out with the help of methods, whereby a method describes how each task should be performed. He adds that a method can itself be viewed as a process at a lower level of abstraction, since it is also made up of individual tasks that describe what needs to be done and can have another lower level method explaining how each task should be performed [22]. Tools facilitate the execution of methods and processes and are usually software applications. Furthermore, external factors such as social, cultural, or organizational aspects must also be taken into account, as these can significantly influence the execution of methods [22]. The relation between process, method, tools, and environment are illustrated in Figure 2.13.

Next to this process view on MBSE, Estefan also mentions the importance of a so-called Information View or Information Model, using the definition of the term by Baker et al. [3]. It should be noted that Baker et al. use the term Model Driven System Design (MDSD) instead of Model-Based Systems Engineering (MBSE), while Estefan considers the two terms as largely synonymous [22]. This Information Model shows what information needs to be present in a model (or in documents, in case of a traditional approach to Systems Engineering) and defines the directional relation between these kinds of information [22]. The contents of this Information Model can differ depending on the stage in the life cycle of the project, as the kind of information that is relevant also differs over the life cycle of a project [3].

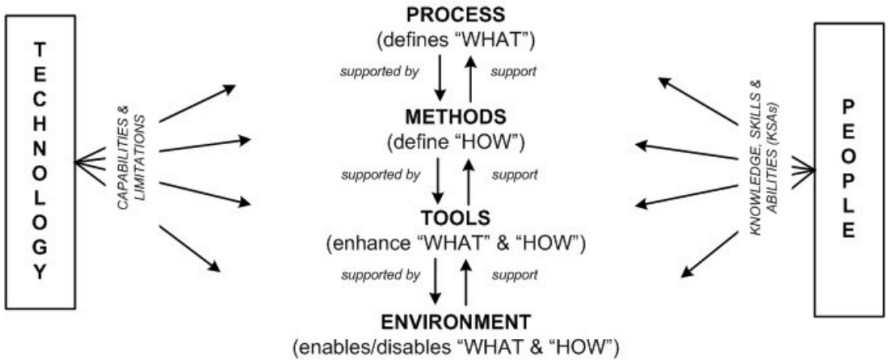


Figure 2.13: Concepts of a MBSE Methodology, according to Estefan [22]

This aspect of different kinds of information and the relationship between them represents a second important aspect of an MBSE methodology, distinct from the process view on systems engineering and MBSE. Many different terms can be found in literature to refer to this part of an MBSE methodology, some of them are: Schema, Information Model, Conceptual Model, Metamodel, and Ontology. While some of the quoted sources use these terms interchangeably or in different ways, there are theoretical differences between the terms, which are shortly explained, following the definition of [6, 27] [1]

Ontology refers to a structured and formal representation of the key concepts and knowledge of a certain domain. The term originates from philosophy. Usually, the goal of an ontology is to represent a conceptualization of a domain (or a universe of discourse) completely and accurately. In [27], there is a further distinction between foundational ontologies that define very general, high-level concepts that apply across all possible domains, and domain ontologies that define all concepts specifically for a particular domain. A conceptual model is equal to a reference ontology, which is a sub-type of domain ontology that is created using highly expressive languages to approximate the theoretical ideal domain ontology as accurately as possible. The counterpart to this reference ontology would be a lightweight ontology which should be derived from the reference ontology with the goal to create desirable computational properties, and are created using languages like OWL and LINGO. [27]

Metamodel is a description of a language's abstract syntax. This means a metamodel defines a set of elements of a language, as well as the rules for combining these elements into valid models in the language. Furthermore, a metamodel is a specification of the conceptual model that underlies a modeling language, meaning that it describes what the elements of the language represent in the real world. The difference between a metamodel and an ontology lies their purpose. While an ontology aims to describe a conceptualization of a domain, a metamodel aims to define the concepts of a language used to create models of things in the real world. [27]

Information Model and *Schema* refer to an organization of data as a blueprint of a database architecture [6].

There are multiple reasons why thinking about Ontologies and Metamodels is important when discussing the implementation and application of MBSE. According to Friedenthal, two criteria to judge the quality of an MBSE model are wether the model is "well-formed", and wether modeling conventions are documented and used consistently [25]. Defining well-formedness rules for a language is part of the languages syntax and thus can be done with a metamodel [27]. An ontology can be used as a basis to define and enforce modeling conventions, which then contributes to creating better MBSE models [25]. Additionally, ontologies and metamodels provide a set of standard elements to be used during modelling and thus create structure and consistency in the model and its connection to associated products [30].

David Long argues that Metamodels are necessary when digitizing systems engineering as a discipline, because they capture knowledge about a domain explicitly, which would usually exist only implicitly in the minds of the domain experts. If this knowledge is instead captured explicitly and in a computer-

[1] David Longs elaborates his views in a presentation held at the 2021 annual INCOSE international workshop:
VitechCorp. (2020, December 3). Schema and Metamodels and Ontologies, Oh My! with David Long [Video]. YouTube. https://www.youtube.com/watch?v=47qCVelLiUE

understandable way, it forms the basis to connecting Systems Engineering with other engineering domains through semantically meaningful and aligned metamodels. [6]

The ISO/IEC/IEEE 24641 [38] uses very similar definitions of the terms Ontology and Metamodel as [27] to define the main building blocks of a Model-Based Systems and Software Engineering (MBSSE) Methodology. The standard extends the definition of MBSE by INCOSE to include software engineering. MBSE can be considered a subset of MBSSE, which means the standard is relevant and applicable in this case.

First, defining an accurate ontology of the relevant domain(s) serves to establish the main concepts for the MBSSE approach. Second, one should define the languages that are to be used for modeling and define or adapt the underlying meta-models to the specific needs of the MBSSE context. The standard underlines the importance of the above mentioned duality of an MBS(S)E methodology by Estefan, including a conceptual view of the domain (i.e. ontological view) and a process view, as well as the importance of relating the two views to each other. The concept view defines key concepts of the domain, which are of interest to the stakeholders, and the relation between the concepts, and the process view describes how those key concepts are modeled to accomplish stakeholder objectives. [38]

The relations between Process, Method, Ontology, Meta-Model, Modeling Language, Modeling Tool, etc. are visualized in Figure 2.14. One aspect that is missing in the visualization, but nevertheless described in the standard, is the direct relation between metamodel and modeling language. The metamodel specifies the abstract syntax and semantics of the model elements which are used for modeling the system, as a way to customize (or "profile") the modeling language to specific stakeholder needs.

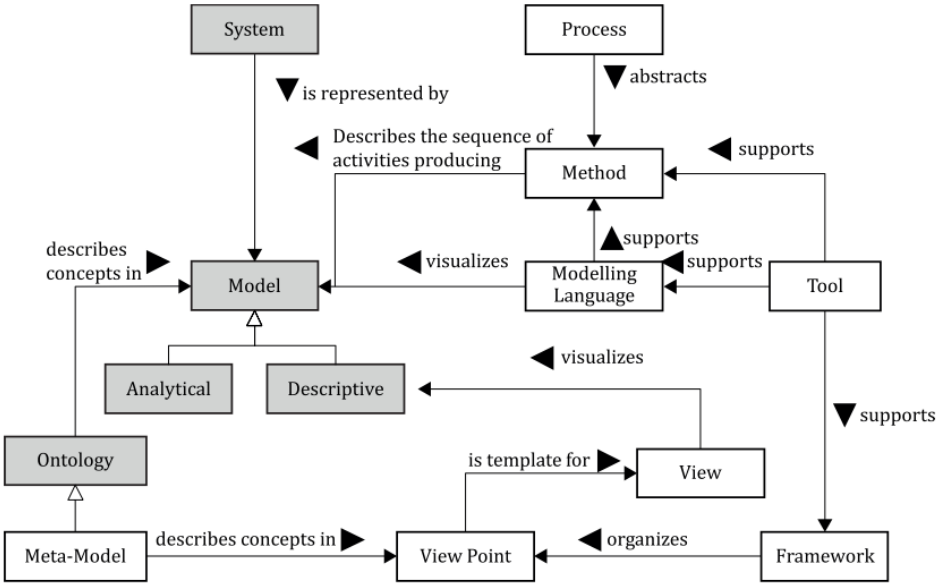


Figure 2.14: Relationship between relevant terms in an MBS(S)E methodology, according to ISO 24641 [38].

Logan, et al. [45] present this connection between the concept view with the process view on a practical example, covering a subset of the complete life cycle process described in ISO 15288 [37].

The standard also mentions that the selection of the used modeling tools is critical and introduces requirements that should be met by an MBS(S)E modeling tool [38]. These are:

1. Allow customization of the tool to fit specific needs of the methodology. an example for a modeling language which has this capability is SysML v1, with its profiling mechanism, allowing modelers to define their own stereotypes [25, 53].
2. Allow for verification of the model,
3. Allow to execute models, such that logical errors can be mitigated.

For the purpose of this thesis, the following terminology will be used to talk about and compare MBSE methodologies, based on [25, 38, 27, 22]:

- **Ontology**: Explains abstract SE and domain engineering concepts and their relations, capturing the underlying understanding and interpretation of systems engineering adopted by the MBSE methodology.
- **Metamodel**: Shows how abstract SE concepts from the Ontology are mapped to the conceptual elements of the modeling language. The Metamodel describes the abstract syntax of the used modeling language. By mapping the Ontology elements onto a more concrete metamodel of the modeling language, two things are achieved. Firstly, since the metamodel is the definition of the modeling language, the mapping to the domain ontology connects the existing model elements to real life concepts. Secondly, defining a metamodel separate from an ontology can allow to explicitly exclude elements from the domain from the modeling language definition.
- **Process**: Sequence of work packages / tasks to be performed to do the SE work at a high level. Similarly to the Ontology, this captures the understanding of the systems engineering process which underlies the methodology description.
- **Methods**: Detailed descriptions of how the modeling tool, language, including the metamodel customizations should be used to create the actual model of the system, i.e. to perform tasks in the Process.
- **Framework**: Shows a kind of skeletal structure of an MBSE model that defines suggested artifacts, views, and viewpoints. The MBSE framework organizes model information into viewpoints and views, relates model views to method and metamodel elements and forms the basis for model templates and model organization.
- **Tools**: Tools used to apply the MBSE methodology according to the Process and Method, allowing the user to create a model using the modeling language.
- **Language**: Language(s) used while applying the MBSE methodology according to the Process and Method. The model elements are described in the Metamodel.
- **Environment**: Any other, external factors which might influence the successful application of the MBSE methodology.

2.4.2. Systems Modeling Language

The Systems Modeling Language version 1 (SysML v1) is used as a modeling language in the frame of this thesis. SysML is a graphical modeling language, developed by the Object Management Group (OMG) for specification, analysis, design, and verification of complex, multidisciplinary systems [53]. With SysML it is possible to express aspects of systems engineering, like requirements, product structures, and product behavior in a formal and unified way [31].

The Unified Modeling Language (UML) was created for software development in 1997 and forms the basis for SysML v1. As shown in Figure 2.15, SysML v1 is based on a part of UML and extends it with additional concepts for Systems Engineering. Because of this, SysML v1 can be seen as an extension of UML [31].

Since UML is used for modeling object oriented software, also SysML v1 shows aspects of the object oriented approach. UML defines classes and instances of these classes, called objects. Additionally, UML Parts exists, which are special kind of instances owned by a class, and which can only exist within the context of the owning class. In SysML v1, the SysML Block are an extension of UML Classes, instance specifications in SysML v1 extend UML Objects, and Part Properties extend the concept of UML Parts.

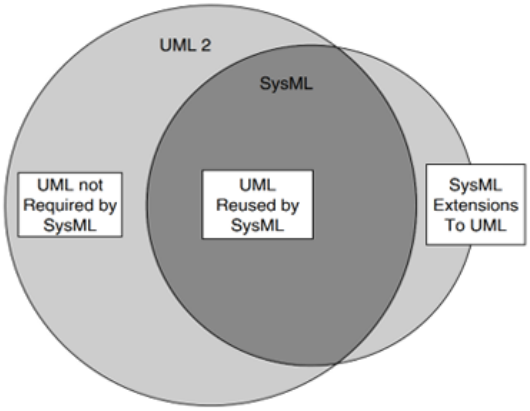


Figure 2.15: Relationship between UML and SysML v1 [31]

The SysML v1 diagrams present a specific view on the model and constitute the interface between the model user and the model [2]. As shown in Figure 2.16, the diagrams were either directly taken from UML, or modified from it, or newly introduced for SysML v1. [2]

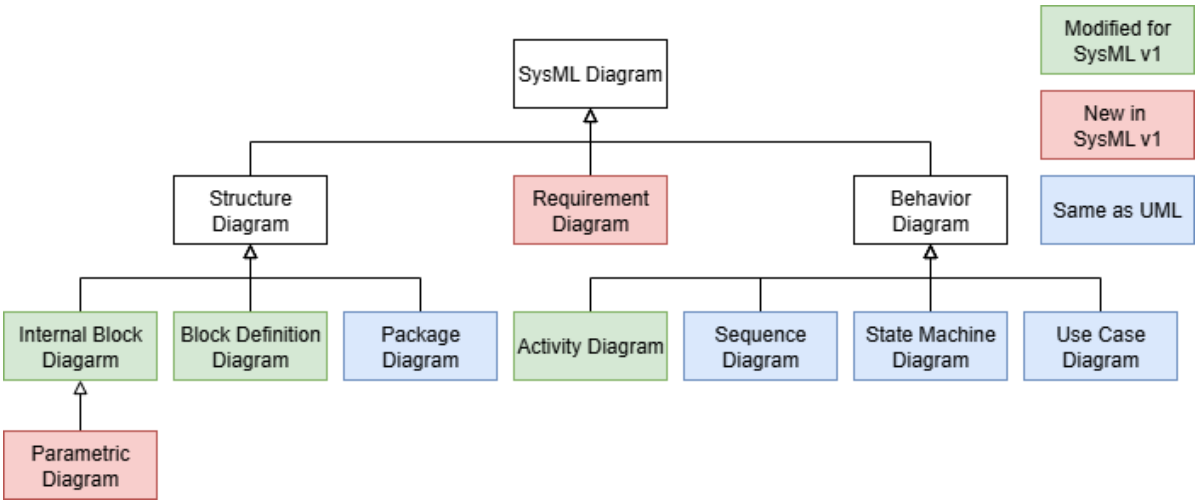


Figure 2.16: Relationship between UML and SysML v1 [31]

New concepts in SysML v1, in addition to requirement diagrams and parametric diagrams, include tools for modeling flows of matter, data, or energy between ports, as well as the concept of allocation [31, 24]. SysML v1 is not intended to replace existing, specialized tools within the development process, but rather to complement them. The goal is to define a common foundation from which detailed development can be carried out in specialized tools. Using SysML v1, domain-specific and cross-domain information can be linked based on this shared foundation, enabling a consistent and traceable system development process [2].

In the following, some important concepts of SysML v1 will be explained.

SysML v1 Block

A SysML Block is a general modeling element in SysML that is used to represent architectural elements. Such elements may include complete systems or system components, such as hardware and software. Furthermore, a Block can also represent an entity that is transported between two structural elements.

A Block can have multiple features that define its structure or behavior. Structural features are defined using *Part Properties*, which represent the elements of which the Block is composed. Part Properties are instances of other Blocks and inherit their characteristics. In general, Blocks and Properties follow a

pattern in which Blocks define a structural element, while Properties represent the usage of this element within a specific context, for example within a higher-level structure.

In addition to Part Properties, *Reference Properties* can be defined, which refer to elements that exist outside the owning instance of a Block. Unlike Part Properties, Reference Properties are not deleted when the owning instance that contains them is deleted.

Value Properties are used to model quantifiable characteristics of a Block. Analogous to Part Properties and Blocks, which define structure, a Value Property is based on a *Value Type*, which defines the set of possible values. For example, a Value Property named "mass" can be assigned the Value Type "kilogram" and take real, non-negative values [24].

A *Block Definition Diagram (BDD)* shows Blocks, their features, and the relationships between Blocks, for example to define structural hierarchies [24, 53].

An *Internal Block Diagram (IBD)* describes the internal relationships between Part Properties and Reference Properties using *Connectors*. A Connector is a UML modeling element that connects two Properties and allows them to interact, without specifying how this interaction is implemented [24].

SysML v1 Activity

Next to structural components, a SysML Block can also have a modeled behavior. One option to describe such behavior is through the use of Activities displayed in activity diagrams. A SysML v1 Activity specifies the transformation of a set of inputs to a set of outputs. This is achieved by a sequence of actions owned by the activity. Actions can themselves call other activities. This mechanism allows to create an interconnected functional breakdown of the system behavior in the model.

Figure 2.17 shows an example activity diagram for a SysML v1 Activity. This Activity contains a SysML v1 Action, which is itself referencing / calling another activity. This is indicated in the name of the element with the name followed by a double colon, and the name of the called behavior. It should be noted that this representation is common also for other model elements that reference, instantiate, or call another model element. One example is the inputs and outputs of the activity, which are named input / output and are typed by the model elements TypeIn / TypeOut. The inputs and outputs are modeled using parameters owned by the activity, and they are represented in the diagram by Activity Parameter Nodes, shown at the borders of the diagram.

The action in the example diagram is placed within a so-called swimlane. This creates a "hidden" SysML v1 relationship between the element in the swim lane and the element referenced at the top of the swim lane. In the later presented examples, this swim lane representation will be used to indicate the allocation between actions and part properties. Since the Action within the swimlane is calling another activity from the model, the parameters of that Activity are represented as pins of the Action. Conceptually, the pins have a similar job to the Activity Parameter Nodes to represent the parameters of the Activity.

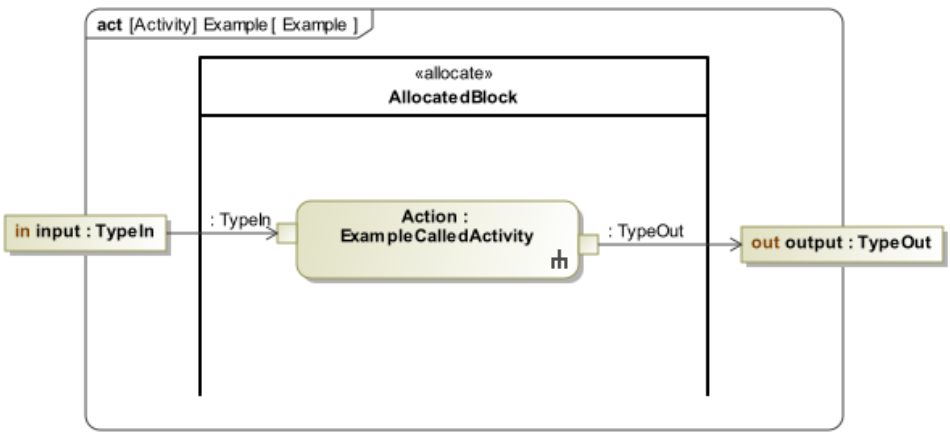


Figure 2.17: Example Activity Diagram

SysML Relationships

SysML provides a variety of relationships that can be modeled between elements. Some are adopted from UML, while others are newly introduced or extended in SysML. The following presents a selection of relationships relevant to this work as seen in Figure 2.18).

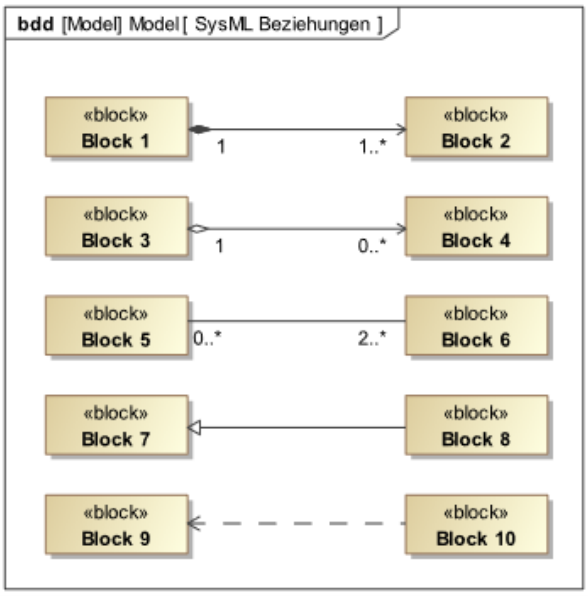


Figure 2.18: Relations in SysML v1 [31]

The first three relationships are *Associations*, with the first two being special types:

- **Composition** is used to represent hierarchical relationships between Blocks. It models a "consists of" relationship, defining the parts of a Block. Composition ensures that parts are instantiated as Part Properties of the parent Block and that their existence depends on the parent. It is denoted by a filled (black) diamond at the beginning of the line [31].
- **Aggregation** is another form of "consists of" relationship. In this case, the relationship creates Reference Properties for the contained Blocks, and the parts are not deleted when the parent Block is deleted. Aggregation is represented by an open (white) diamond [31].
- A general **Association** can be either unidirectional or bidirectional. Associations represent dependencies between Blocks and result in Reference Properties. The ends of Associations can have a *multiplicity*, specifying how many elements are referenced. For example, a Composition between two Blocks can indicate that one instance consists of one or more instances of another Block [31].

In addition to "consists of" relationships, inheritance can be modeled using **Generalization**. This relationship is indicated by a hollow triangular arrowhead. It implies that a derived Block inherits all features of a base Block and may define additional features, or redefine inherited ones [31].

Another relationship is the **Dependency**, which indicates that a dependency exists between two model elements. It is the weakest relationship, as it only indicates the existence of a dependency without specifying its nature.

For working with requirements in SysML, specialized Dependencies exist:

- **`«derive»`**: indicates that one element is derived from another element, typically through analysis or transformation
- **`«refine»`**: indicates that one element describes another element in more detail
- **`«allocate»`**: indicates that an element is assigned to another element for implementation, execution, or responsibility, e.g. a behavior is allocated to a structural element.

- **«trace»**: indicates a general dependency between elements to support traceability without implying a specific semantic meaning
- **«satisfy»**: indicates that a model element fulfills (or must fulfill) a linked requirement
- **«verify»**: indicates that a model element confirms that a linked requirement is met
- **«deriveReqt»**: indicates that a requirement is derived from another requirement

These relationships are represented by stereotypes placed above the dependency arrow [31, 53]. The relation between these relationships, i.e. how they are defined and derived from each other is shown in Figure 2.19. Generalization, Association, Dependency, Abstraction, Derive, and Refine relations originate from UML. Trace, Allocate, Satisfy, Verify, and DeriveReqt are SysML-specific extensions [53].

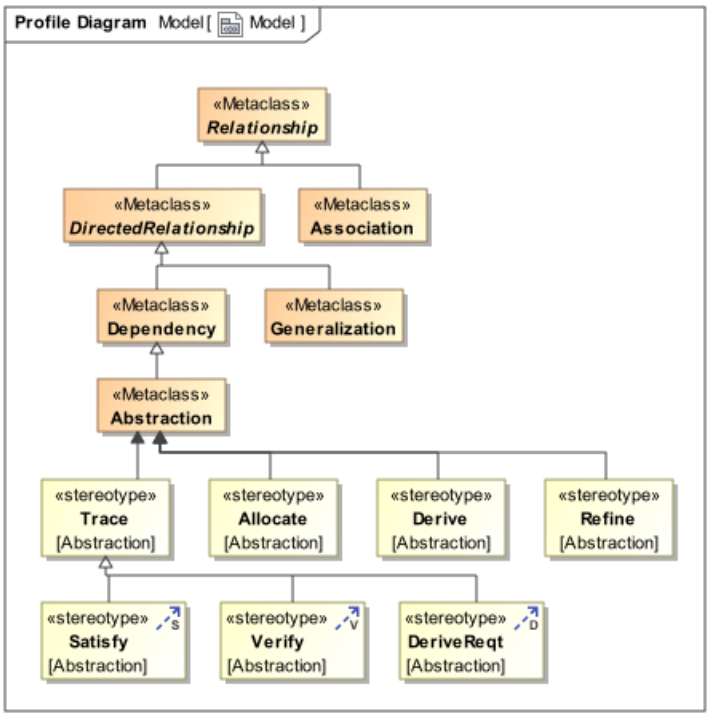


Figure 2.19: Relations in SysML v1 [31]

2.4.3. Methodologies found in Literature

OOSEM method [25]

OOSEM provides an integrated framework that combines object-oriented techniques, model-based design, and traditional top-down systems engineering practices. Initially developed around UML modeling, OOSEM was realigned with SysML in 2006 and has since been widely promoted as an example of MBSE best practice. A detailed description is available through INCOSE. OOSEM has also been aligned with the standard ISO/IEC/IEEE 15288, linking the methodology to internationally recognized systems engineering life cycle processes. This alignment helps identify the sequence of processes required to develop and deliver the main system artifacts. Within this context, systems engineering processes are commonly organized into groups such as agreement, organizational project-enabling, technical management, and technical processes. This makes OOSEM relevant not only as a modeling approach, but also as a process-oriented methodology for guiding system development across the life cycle.

Friedenthal presents a simplified MBSE method for applying SysML in [25]. A high level description of the method is shown in Figure 2.20.

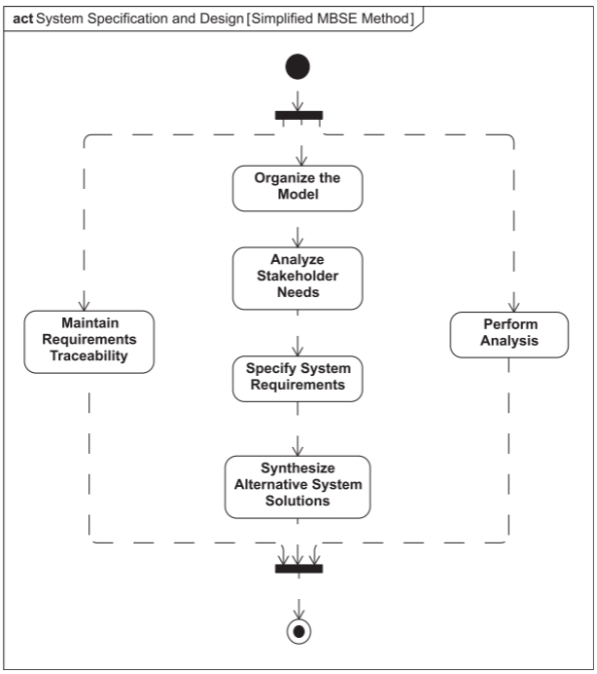


Figure 2.20: OOSEM method [25]

MagicGrid [1]

MagicGrid provides a SysML-based framework that helps guide engineers through the structured development of system models by organizing the modeling activities within a matrix-style model template. The framework divides the system development into three distinct domains: the Problem domain, the Solution domain, and the Implementation domain.

Each domain is represented as a row in the matrix, with the Problem domain further split into a black-box and white-box view, reflecting different levels of abstraction. The Solution domain is broken into multiple rows to represent varying levels of detail in the system's architecture. In contrast, the Implementation domain is less detailed and falls largely outside the scope of MBSE, except for the specification of implementation requirements.

The columns of the MagicGrid matrix correspond to the four fundamental types of diagrams of SysML, also referred to as the Four Pillars of SysML: Requirements, Structure, Behavior, and Parametric Diagrams. A combination of row (domain) and column (Pillar of SysML) represents a certain view onto the model. These views help engineers document and understand the system from different perspectives.

ESA MBSE Solution (formerly ESA SysML Solution) [26]

The ESA MBSE Solution is a tailored approach to applying MBSE, developed specifically for space missions and in compliance with standards provided by the European Cooperation for Space Standardization (ECSS) and systems engineering practices at the European Space Agency (ESA). Similarly to the MagicGrid framework, the ESA SysML Solution can also be organized in a grid-like overview, separating Mission Specification, System of Interest Specification (Problem Space), and Functional Design and Physical Design (Solution Space) into rows. The columns categorize the different views on the model per row into breakdown views, architecture views, states and modes views, and scenarios views. In parallel to these different horizontal layers, the methodology also provides concepts to model aspects of Verification and Validation, Stakeholders, External Systems, Exchange Items, and Textual Requirements as cross-cutting elements in the methodology.

Concurrent Engineering with CDP4-COMET [58]

CDP4-COMET is an open-source software tool designed to support Model-Based Systems Engineering

(MBSE) through the Concurrent Design method. It enables multidisciplinary teams to collaboratively develop integrated system models, allowing stakeholders to modify design parameters and assess the system's performance against its requirements. By organizing the system into a modular structure, CDP4-COMET helps teams analyze key design challenges and trade-offs early in the process. The model serves as a single point of truth, ensuring all team members have access to the up-to-date information they require for their domain of expertise. Systems Engineers can generate typical Systems Engineering deliverables, like technical budget reports, directly from the model. The tool supports both co-located and geographically distributed teams, ensuring clear role assignments and access controls to the central data repository.

Arcadia/Capella [61]

ARCADIA provides a method dedicated to systems and architecture engineering, supported by the Capella modeling tool. The framework defines a structured process that helps stakeholders understand the customer's real needs, define and share the product architecture, validate the design early, and manage Integration, Validation, Verification, and Qualification.

ARCADIA is applied to complex systems, hardware, and software architectures, particularly those requiring reconciliation of constraints such as cost, performance, safety, and security.

The methodology divides system development into five core engineering activities:

- Operational Analysis (OA)
- System Need Analysis (SA)
- Logical Architecture Design (LA)
- Physical Architecture Design (PA)
- Building Strategy Definition (BS)

Each activity contributes to a detailed system model, with each engineering activity represented as a distinct perspective of the model, ensuring all aspects of the system are accounted for and aligned.

Arcadia/Capella offers good support for architecture-driven IVV planning and traceability, particularly through its explicit modeling of operational needs, functional chains, scenarios, logical and physical architectures, interfaces, and allocation relationships. These capabilities make it well suited for deriving verification concerns from the system architecture and for supporting integration reasoning across functions, components, and interfaces.

However, the main emphasis of Arcadia/Capella remains on architecture definition, justification, and consistency across engineering perspectives. The methodology provides less direct support for the detailed post-CDR AIV execution workflows, such as explicit evidence management, criterion-based verdict assignment, procedure execution, verification close-out, non-conformance handling, and ECSS-aligned AIV reporting. Furthermore, Arcadia/Capella provides a tool-specific modeling process, which would require a separate mapping between the proposed AIV ontology and the Arcadia/Capella metamodel. For this reason, it is considered as relevant related work and a potential future research direction to align the Arcadia/Capella approach with the developed methodology discussed in this thesis.

Data Driven Systems Engineering (DDSE) with ValiSpace [44]

Data-Driven Systems Engineering moves away from document-centric approaches by focusing on working with data points within a single source of truth application, such as Valispace.

In this approach, all data is stored, accessed, and managed in a central database, eliminating the need for duplication and increasing data consistency. Engineers can reference data points in real-time from anywhere, ensuring consistent, up-to-date information across the entire development process.

DDSE enables decisions and trade-offs to be made with the latest project data, which is automatically propagated throughout the system when altered. Reports in DDSE link directly to live data, always reflecting the current state of development and easily shareable with stakeholders in real-time.

2.4.4. Application of MBSE for Verification, Validation, and Testing

There are a few sources in literature that discuss the application of MBSE for Verification, Validation, and Testing.

OOSEM methodology [25]

Verification is briefly mentioned in the OOSEM methodology, by Friedenthal, discussing an implementation using SysML, which is compatible with the rest of the methodology. The verification scope is modeled using a SysML block, and the verification context is captured in a block definition diagram, defining different context elements, like the System under Test, the Test Operator and supporting systems. The test case itself is captured as an activity and specified in an activity diagram, owned by the context block. This activity diagram shows a high-level test procedure or test sequence, as well as inputs and outputs of the involved SysML actions. It is mentioned that the test case and scope should be traced in the model to use cases and requirements. [25]

The process of how to model the test cases themselves is clearly defined, but only shown on a very small example case. Other aspects of the systems engineering and AIT engineering work, like verification planning and control are not considered explicitly.

V&V Strategy Pattern [56, 42]

In their work, Alejandro Salado, et al. present the use of network graphs to model verification strategies of complex systems. They introduce an approach to capture verification activities and system parameters as nodes of a mathematical graph, with their interrelation represented as the edges of the graph. This mathematical formulation of the verification strategy allows to identify various fundamental and recurring patterns in the verification strategy of engineered systems. Based on these patterns, engineers are enabled to reason about critical aspects of the work of systems engineering and AIV engineering from another perspective, with the goal to reduce the effect of cognitive biases and other subconscious effects that might lead the adoption of sub-optimal verification approaches.

In their paper [43], Kannan and Salado discuss a mathematical formalization of Verification and Validation into a theoretical framework using formal logic. Even though this publication is not discussing a methodology for application of MBSE, it provides a sharpened interpretation of different conceptual devices often used in verification and validation. The main definitions adopted for the purpose of this thesis are the definitions of verification activity, verification criterion, and verification evidence. The presented interpretation of V&V activities as knowledge-building activities is adopted in this thesis. This basically means that every activity, whether it is a test, analysis, etc. aims to contribute to gain knowledge about the system of interest.

Verification criteria define the conditions under which a requirement is considered satisfied. Rather than only prescribing verification methods, verification criteria specify measurable or observable conditions, which helps limiting the otherwise infinite set of possible observations that could be gathered to make claims about the satisfaction of a requirement. In the context of space systems engineering, these criteria translate a requirements statement into a clearly evaluable criterion within the context of a verification activity. A verification activity is an event through which an agent acquires knowledge about verification evidence. Verification evidence represents the data or information obtained from performing the activity. The acquisition of this evidence enables the agent to form a justified belief about whether an associated criterion is satisfied.

Verification and Validation for MagicGrid [48]

Morkevicius et al. developed an approach to extend the existing MagicGrid framework [1] to enable capturing of Verification and Validation concepts in the model. The approach is using pure SysML v1, similarly to the approach of Friedenthal. Test context and scope are defined in a block definition diagram and test cases are captured as behavior of the context block in SysML. The difference to Friedenthal is that in the MagicGrid approach, the test case is modelled using a sequence diagram rather than an activity diagram to show interactions between test context elements. This presents a clear way how the traceability to requirements is captured in the model and how the result of the modeled test is captured. The result is captured in a dedicated value property of the context block, and can be related to the system behavior through parametric relations. The authors further describe the value of creating

dedicated user interfaces inside the used modeling tool Cameo Systems Modeler, to allow for easier simulation of the test procedure and the modeled system behavior. [48]

The publication focuses on introducing Verification and Validation aspects into MagicGrid, to enable early V&V with MagicGrid and Cameo Systems Modeler. The presented approach does not describe how to exploit the model to address important information like requirements coverage or compliance status. There is no description how to handle, how to relate the information in the model to actual test procedures or other practical matters, like dedicated views showing the overall AIV flow for the system.

ESA MBSE Solution [19]
The ESA MBSE Solution provides a methodology to model verification and validation using MBSE. The main concepts related to V&V in the ESA methodology are the Verification Scenario Scenario and Verification Activity. A Verification Scenario defines a series of steps to be executed together or sequentially verify one or more requirements. A verification activity describes one step within a verification scenario and has the purpose of demonstrating the verification of a requirement.

A verification scenario is further modeled in SysML (v1) either by means of an activity diagram or a sequence diagram. Each verification scenario can verify a number of requirement and each requirement can be verified by multiple verification scenarios, which is indicated using a "verify" relationship. A verification scenario can involve model elements that represent products (including subsystems and the system of interest) and external systems. These systems exhibit behavior during the verification activity which is specified in the model using verification activities that are allocated to the system that performs them. The ESA MBSE solution also describes a metamodel and a framework definition with views and viewpoints, which is consistent with the ISO standards described constituents of an MBSE methodology, but it is lacking a complete process and methods definition. The ESA MBSE solution is one of the few MBSE approaches that explicitly contain AIV related aspects as concrete model elements, while also involving a significant share of the constituents of an MBSE methodology, as described in the ISO standards.

Paper: PERFORMING VERIFICATION AND VALIDATION ACTIVITIES IN A MODEL-BASED ENVIRONMENT [49]
The paper by Rebecca Mulholland, et. al., published in October 2025, describes a methodology for performing verification and validation in a model-based environment. In contrast to the other approaches found in literature, this work defines a metamodel, its implementation in SysML v1, and process model that provides guidance for the application and tailoring of the approach. The authors cover needs and requirements verification as well as the integration, verification and validation of designed systems. While this publication certainly covers much more than any other reviewed sources in terms of the work of AIV engineers, and maturity of MBSE methodology definition, the authors mention only informal linking and traceability between activities and test setups. Since the applied terminology and metamodel definitions presented in the paper are directly derived from INCOSE sources, they are not immediately transferable to the space domain. For full application for performing AIV work in a space systems development project, a formal traceability between verification activities and setups is needed and the terms verification activity, event, and evaluation verdict need to be aligned with ECSS standards.

The authors argue the importance that the model-based approach is able to establish a digital thread from requirements to objective evidence that is collected in the verification activities. The stated, potential benefits of the model-based approach are enhanced traceability between V&V attributes and system needs and requirements, improved test coverage across the design, and support for the reuse of verification activities throughout the life cycle. In addition, this paper introduces the idea of integrating aspects of V&V planning, management, and execution results handling within a single environment, from which consolidated verification plans and procedures can be generated in the future.

2.5. Chapter Summary

This chapter reviews existing literature to position the thesis within the topics of systems engineering, space project life cycle management, AIV, and MBSE. First, systems engineering is introduced as an interdisciplinary and integrative approach for managing technical complexity, coordinating multidisciplinary work, and maintaining consistency across the complete system life cycle. By comparing

generic ISO life cycle models with ECSS and NASA space project life cycles, the chapter establishes that the Critical Design Review marks a transition from detailed design toward integration, verification, validation, and transition activities. In this project stage, typical activities are grouped under the label of Assembly, Integration, and Verification activities.

Although verification planning starts much earlier, the post-CDR phase is characterized by the practical execution and control of AIV work, including integration, testing, verification close-out, non-conformance handling, and preparation for qualification, acceptance, and operational readiness reviews. The review of AIV literature and standards shows that AIV is not only a test development and execution activity, but a life cycle-spanning engineering discipline concerned with planning, defining, executing, and reporting verification and validation activities. As a discipline, AIV sits in the interface between technical work, quality assurance, and project management. The interpretation of the AIV process as a knowledge-building process, as discussed by Kannan and Salado in [43] is adopted for this thesis.

ECSS and IEEE documentation standards further clarify the key artefacts that are generated and managed during AIV, including AIV plans, verification plans, test specifications, test procedures, test reports, non-conformance reports, and verification control documents. Finally, the chapter reviews MBSE as a model-centered approach to performing systems engineering and discusses its expected benefits, including improved traceability, consistency, communication, automation, and knowledge capture.

The analysis of MBSE methodology constituents, following ISO standards structure, shows that a complete methodology requires not only a modeling language and tool, but also an ontology, metamodel, process, methods, framework, and consideration of the organizational environment. Existing MBSE methodologies and AIV-oriented approaches provide useful concepts for modeling requirements, architectures, test contexts, verification activities, and traceability, but they only partially address the practical needs of post-CDR AIV in space projects.

In particular, the literature shows a gap in integrated methodological support for ECSS-aligned AIV planning, execution, evidence management, verification control, procedure handling, and non-conformance tracking within a coherent MBSE framework. This gap motivates the development of the post-CDR MBSE AIV methodology presented in the following chapters.

3

Research Design

3.1. Synthesis of Literature Review

The first part of the literature review, subsection 2.2.2, establishes Systems Engineering as a critical discipline for the development of highly complex systems, common for the space domain. Multiple sources are presented, including INCOSE publications, ISO standards, ESA and ECSS resources and NASA sources, that connect the common life cycle models to the activities performed by systems engineers during the different life cycle stages. It is established that product realization processes (Implementation, Integration, Verification, Validation, and Transition, as described in the INCOSE Systems Engineering Handbook [63]) form the most prominent type of systems engineering work in later stages of the life cycle. Table 2.3 shows the relevance of these systems engineering processes in relation to the ECSS life cycle model.

Further study of these INCOSE systems engineering processes in subsection 2.2.3 details the work for systems engineers during later life cycle stages. The processes of Implementation, Integration, Verification, Validation, and Transition are not isolated activities, but form a highly interdependent and iterative set of processes with extensive input–output relationships, as highlighted by the NASA and INCOSE sources [63, 50]. The complexity of this combination of processes creates the need to coordinate project management, systems engineering, and domain engineering activities carefully. This tightly coupled set of product realization processes aligns closely with the discipline commonly referred to in the space domain as Assembly, Integration, and Verification or AIV. While terminology differs slightly across sources such as ESA, NASA, and INCOSE, the underlying activities described in the literature are conceptually equivalent.

The subsection 2.3.1 introduces the common principles shared among various literature sources, with an extended focus on the description of the work process. The section demonstrates that the AIV domain is marked by technical / engineering work but also management and programmatic tasks. The presented process models for this work highlight the need for AIV engineers to align the technical challenges, defects, and other problems discovered during the execution of verification activities, with the programmatic constraints of a project [63, 50, 69]. Such programmatic constraints are for example scheduling and timeline management, managing stakeholders, keeping project information and documentation consistent, and preparing for formal reviews and decision gates. Traditionally, the complexity of the tasks managed by AIV engineers has required a high degree of documentation to be created and maintained manually. The typical AIV related documents are discussed in subsection 2.3.2. The section further underscores the complexity that is introduced into a project, just by the need to manage all this documentation. In addition, this section provides a bounded scope for the work of AIV engineers, differentiating the work of an AIV engineer from the work of Systems Engineering in early life cycle stages. In this context, AIV engineering is distinguished from early-phase systems engineering by its primary focus on the realization, integration, verification, and validation of an already defined system, rather than on requirements definition, concept development, and system architecture design [9, 12].

Given the inherently complex and document-centric AIV processes, and the potential upsides of MBSE mentioned in the introduction, MBSE related literature is reviewed. This aims to investigate the extent to which existing approaches in literature address the work related to AIV. section 2.4 of the literature review first introduces the basics of MBSE and its benefits cited in literature. Furthermore, the section provides an overview of the best practices on how to structure an MBSE approach following the ISO 24641 standard [38]. In the last part of the section, a selection of existing MBSE approaches are presented with an additional focus on approaches that include aspects of verification and validation work.

3.2. Research Gap and Objective

While the literature provides extensive descriptions of Systems Engineering processes in post-CDR stages and highlights the complexity of AIV activities, a clear gap emerges when comparing these insights with existing MBSE methodologies. Current MBSE approaches predominantly focus on early life cycle phases, such as requirements definition and system architecture design, and provide limited support for the explicit modeling of AIV-specific concepts such as verification activities, test setups, verification control, and verification evidence management.

As a result, the later life cycle stages remain largely document-centric, despite their high complexity, strong interdependencies, and critical role in ensuring system compliance. This lack of methodological support for applying MBSE in post-CDR phases limits the realization of a consistent digital thread across the whole system life cycle and reduces the potential benefits of MBSE in the real world.

Therefore, there is a need to investigate how the application of MBSE can be extended or tailored to effectively support Systems Engineering and AIV activities in post-CDR project stages. This leads directly to the research objective of the thesis to develop an MBSE Methodology to support Systems Engineering and Assembly, Integration and Verification in post-CDR project stages. To show that the approach is suitable for practical application, the second objective is to implement the approach in part of a real-world project.

3.3. Research Questions

Based on the identified gap between the complexity of AIV processes and the limited support provided by existing MBSE methodologies for post-CDR stages and the defined research objective, three research questions are formulated. For each research question, a hypothesis is formulated and later evaluated to provide answers to the questions and guide the research activities in the process. The research questions and hypotheses are presented in Table 3.1.

Table 3.1: Research Questions

ID	Research Topic / Research Question
RO	Develop an MBSE Methodology for Systems Engineering and Assembly, Integration and Verification in post-CDR project stages and implement it in part of a project.
RQ-1	How and to what extent do current MBSE approaches in literature address the post-CDR stage of projects?
HYP-1	Most MBSE approaches described in literature are focused on SE activities until CDR, while most of the SE concepts for post-CDR described in relevant SE standards and literature, such as AIV planning, verification control, and test specifications, are not covered in MBSE Methodologies found in literature.
RQ-2	How can an MBSE methodology be defined and implemented to effectively support Systems Engineering and AIV activities in post-CDR project stages?
HYP-2	It is possible to define and implement an MBSE methodology tailored to post-CDR project needs that supports Systems Engineering and AIV activities by focusing on requirements traceability, explicit modeling of verification activities, collected evidence, setups, and verification control.
RQ-3	What benefits and limitations can be observed when applying MBSE to the post-CDR phase of a project such as the Da Vinci Satellite?
HYP-3	Applying the MBSE approach results in improved consistency and traceability of requirements to other information, better planning and overview over the AIV process, and increased stakeholder confidence in the system definition and verification control, compared to a conventional document-centric approach.

The three research questions are addressed through a sequential research approach, where the outcome of each step informs the subsequent one.

RQ-1 is addressed through an analysis of existing MBSE methodologies identified in the literature review. These methodologies are evaluated with respect to their coverage of post-CDR Systems Engineering and AIV-related concepts.

RQ-2 builds directly on the findings of RQ-1. Based on the identified gaps and the previously established understanding of post-CDR Systems Engineering and AIV processes, a tailored MBSE methodology is developed. This includes the definition of an ontology, modeling approach, and supporting framework that explicitly incorporates AIV-related concepts and enables end-to-end traceability in the AIV process. The scope of the methodology is defined by selecting the elements that are most relevant for post-CDR stages. The suitability of this scope is used to evaluate HYP-2.

RQ-3 is addressed through the application of the developed MBSE methodology to a case study, namely the Da Vinci Satellite project. The implementation focuses on selected AIV processes and artifacts, allowing for an assessment of the practical applicability of the approach. The evaluation compares the model-based approach to the existing document-based practices in terms of traceability, consistency, and transparency of AIV planning and verification control. Observed benefits and limitations are used to evaluate HYP-3.

3.4. Chapter Summary

This chapter defines the research design of this thesis project by synthesizing the findings of the literature review into a concrete research gap, research objective, research questions, and hypotheses. The synthesis shows that post-CDR systems engineering work is dominated by highly interdependent product realization processes, especially implementation, integration, verification, validation, and transition. In the space domain, these activities closely correspond to the discipline of Assembly, Integration, and Verification. The literature further shows that AIV is not only a technical testing activity, but also includes planning, coordination, documentation, verification control, review preparation, and non-conformance handling.

The chapter identifies a gap between the complexity of post-CDR AIV work and the support provided by existing MBSE methodologies. Current MBSE approaches mainly focus on earlier life cycle activities such as requirements definition, system architecture, and design, while AIV-specific concepts are only partially addressed. This motivates the research objective: to develop and implement an MBSE methodology that supports Systems Engineering and AIV activities in post-CDR project stages.

Based on this objective, three research questions are formulated. RQ-1 investigates how existing MBSE approaches address post-CDR project stages. RQ-2 focuses on how an MBSE methodology can be defined and implemented to support post-CDR Systems Engineering and AIV work. RQ-3 evaluates the observed benefits and limitations of applying the developed approach in the context of the Da Vinci Satellite project. Together, these questions structure the research approach from literature analysis, to methodology development, to practical case study application and evaluation.

4

Development of the MBSE AIV Methodology

4.1. Requirements Elicitation

4.1.1. Stakeholder Analysis and Stakeholder Needs Definition

In this section the stakeholders, stakeholder needs, and requirements for the MBSE AIV Methodology will be presented. They will serve as a baseline for the Methodology development and the verification and validation approach of the methodology. First, the main stakeholders and their needs will be derived from the literature study section. Then, the stakeholder needs will be mapped to the main use cases that the methodology must provide. The Use Cases will provide a rough description on how the different stakeholders could make use of the MBSE AIV methodology to perform (parts of) the work they are usually expected to perform.

Furthermore, the stakeholder needs are grouped into four categories. These categories are based on the three high-level processes related to the AIV discipline mentioned in [9], which are Verification Planning, Verification Execution and Reporting, and Verification Control and Closeout. Additionally, there is an added category capturing the specific needs to ensure traceability and documentation consistency which can be understood as a cross-cutting category.

The stakeholder needs identified in this work originate from key roles involved in Assembly, Integration, and Verification (AIV) activities within a space programme as stated in the relevant ECSS standards. The *Systems & AIV Engineer* is responsible for planning and coordinating verification activities, ensuring consistency between requirements, system design, and verification execution.

The *Subsystem Engineer / Subject Matter Expert (SME) / Test Operator* provides detailed technical expertise for specific subsystems and is typically responsible for defining, executing, and assessing subsystem-level tests and analyses.

The *Project Manager* oversees overall project progress, schedule, and resource allocation, requiring insight into the status of AIV activities and verification progress. The *Product Assurance Engineer* ensures compliance with applicable standards, such as ECSS, and is responsible for configuration control, anomaly management, and verification evidence integrity.

Finally, *ESA / Customer / Reviewer* stakeholders represent the external authorities responsible for conducting formal project reviews (e.g., Qualification Review), assessing compliance with contractual requirements and approving achieved verification and qualification status.

Table 4.1: Stakeholder Needs for MBSE Methodology - AIV control and closeout

Need ID	Need Title	Need Description	Rationale	related Stakeholder
SN-1	Anomalies and NCRs	Needs to ensure full traceability of anomalies and non-conformances from detection to closure.	E.g. raised NCRs in ECSS terms, and tracking their closeout.	Product Assurance Engineer
SN-2	Assessment at Milestone	Needs to conduct reviews to assess status of qualification efforts at a specific point in time.	Information needs to be provided as a data package delivery, with the information being frozen at a specific point in time.	ESA / Customer / Reviewer
SN-3	Compliance with Standards	Needs to be able to show compliance with applicable standards.	For example, show compliance to ECSS standards for satellite testing	Product Assurance Engineer
SN-4	Configuration Manage-ment	Needs to conduct configuration management for all configuration items.	"According to ECSS-Q-ST-10C-Rev.1 Annex B. This includes unique identifiers for all hardware and software used, and verification evidence generated. "	Product Assurance Engineer
SN-5	Impact Analysis	Needs to support impact and decision analysis for changes in requirements, design definition, and V&V planning and execution.	Derived from responsibilities of Systems Engineering discipline as stated in ECSS-E-ST-10C and in the INCOSE SE Handbook, during all phases of a project.	Systems & AIV Engineer
SN-6	Qualification Status Tracking	Needs to track qualification status for all configuration items.	According to ECSS-Q-ST-10C-Rev.1 Annex B.	Product Assurance Engineer
SN-7	Readiness for Reviews	Needs to create and maintain an overview of overall status of verification (e.g. VCD-like views).	Responsibility of Systems Engineering discipline during Phase D, directly stated in ECSS-E-ST-10C. This includes ensuring consistency between different documents.	Systems & AIV Engineer
SN-8	Requirement Close-Out Status	Needs to assess requirements close-out status and coverage, including references to verification results and reports.	This need is directly linked to the VCD(s) that need to be delivered for QR and mandated by ECSS.	Systems & AIV Engineer
SN-9	Review artefacts for QR	Needs to review all verification evidence and reports to approve achieved qualification status.	To be able to conduct Qualification Review	ESA / Customer / Reviewer
SN-10	Verification Status Tracking	Needs to be able to track verification status over the lifecycle.	support tracking of overall progress and status of the project in Phase D.	Project Manager

Table 4.2: Stakeholder Needs for MBSE Methodology - AIV execution

Need ID	Need Title	Need Description	Rationale	related Stakeholder
SN-11	Capture analysis, review, and inspection activities	Needs include analyses, review of design, and inspection activities in the overall AIV process and handle their results throughout the lifecycle.	Analysis, inspection, and review of design is a crucial part of AIV activities and has to be also considered in order to be compatible with ECSS and to be able to show realistic requirement coverage, as some requirements cannot be verified by test.	Systems & AIV Engineer; Product Assurance Engineer; Subsystem Engineer / Subject Matter Expert (SME) / Test Operator
SN-12	Capture Verification Evidence	Needs to capture and manage verification evidence and related reports, including test results and raw measurements.	As stated in ECSS-Q-ST-10C-Rev.1.	Product Assurance Engineer
SN-13	AIV execution according to approved procedures	Needs to ensure that all AIV activities are executed in accordance with approved and released procedures.	Creating procedures and making sure they are followed is a typical product assurance task.	Product Assurance Engineer

Table 4.3: Stakeholder Needs for MBSE Methodology - AIV planning

Need ID	Need title	Need Description	Rationale	related Stakeholder
SN-14	Integration Planning	Needs to define and manage integration dependencies, integration scenarios, integration workflow and sequence.	Core activity during system integration process in INCOSE & NASA SE HB.	Systems & AIV Engineer
SN-15	AIV Process Planning	Needs to define and update AIV Planning and the sequence of AIV activities	Relates to SE Lifecycle processes of control for AIV related processes.	Systems & AIV Engineer; Project Manager
SN-16	Define Test Specifications	Needs to define test setups, test sequences, required GSE, and capture related information.	Important for all testing related aspects.	Systems & AIV Engineer
SN-17	Define Verification Criteria	Need to define clear criteria per verification activity and requirement to assess success of the activity and requirement closeout.		Systems & AIV Engineer

Continued on next page

Need ID	Need title	Need Description	Rationale	related Stakeholder
SN-18	Performance Parameters	Needs to keep track of parameters related to testing, incl. performance budgets, calibrations, and measurement uncertainties.	Insight from interview with TNO experts on Satellite AIV. Budgeting performance parameters for tests, e.g. measurement accuracy of equipment is a very important task for AIV Engineers.	Systems & AIV Engineer
SN-19	Reference GSE specification	Needs to reference GSE specifications when defining verification activities and test setups.	Same as in a document-based approach, GSE specification is referenced when defining verification activities (e.g. test setups).	Systems & AIV Engineer; Subsystem Engineer / Subject Matter Expert (SME) / Test Operator
SN-20	Verification Methods	Needs to define verification methods for all requirements.	Part of planning verification activities according to INCOSE & NASA SE HB.	Systems & AIV Engineer

Table 4.4: Stakeholder Needs for MBSE Methodology - Traceability

Need ID	Need title	Need Description	Rationale	related Stakeholder
SN-21	Documentation of Configurations and Models	Needs to ensure proper documentation of model philosophy for equipment, subsystems, and system in accordance with the AIV plan.	Responsibility of Systems Engineering discipline during Phase D, directly stated in ECSS-E-ST-10C.	Systems & AIV Engineer
SN-22	Interface Definitions	Needs to get access to up-to-date interface definitions of System elements, GSE, tools, facilities, etc. and show compatibility between different elements.	Required input for planning and executing AIV activities.	Systems & AIV Engineer
SN-23	Traceability	Needs to maintain full traceability between requirements, design definition, verification activities, and verification & testing results throughout lifecycle.	Derived from responsibilities of Systems Engineering discipline as stated in ECSS-E-ST-10C, true for all phases of a project.	Systems & AIV Engineer

Continued on next page

Need ID	Need title	Need Description	Rationale	related Stakeholder
SN-24	Verification Activities Traceability	Needs to link requirements to verification activities and corresponding verification criteria, test specifications, and procedures.	Part of planning the verification approach for the project, maintaining traceability from requirements to verification results, to be able to show proof of successful verification.	Systems & AIV Engineer

4.1.2. Use Cases of the MBSE AIV Methodology

The MBSE AIV methodology addresses a set of core use cases that support the planning, execution, and assessment of Assembly, Integration, and Verification activities throughout the system lifecycle. These use cases capture the main interactions between stakeholders and the model-based methodology, and they operationalize the stakeholder needs identified in the previous section. Each use case contributes to structuring verification activities, maintaining traceability between requirements and verification evidence, and supporting monitoring and review processes. An overview of the identified use cases, including their descriptions, associated stakeholder needs, and involved stakeholders, is provided in Table 4.5. This set of use cases has been derived from the stakeholder needs and is mainly based on the presented literature research.

Table 4.5: Overview of Use Cases for the MBSE AIV Methodology

Name	Description	Refines	Related Stakeholders
Define and Prepare Verification Activities	Define and model verification activities required to demonstrate compliance of system requirements. This includes mapping verification activities to the requirements under verification, defining verification criteria, specifying test setups, personnel, performance parameters, required inputs (e.g., tools, GSE, consumables), and expected outputs (e.g., AIV artefacts and verification evidence). Verification procedures are then constructed, exported, and finalized for approval and execution.	• SN-16 • SN-18 • SN-19 • SN-17 • SN-20 • SN-13	Subsystem Engineer / Subject Matter Expert (SME) / Test Operator Product Assurance Engineer Systems & AIV Engineer
Maintain Digital Thread from Requirements to Verification Evidence	Maintain end-to-end traceability between requirements, verification activities, verification criteria, and verification evidence. For each requirement, verification activities and derived verification criteria are linked and their compliance status tracked. Verification criteria are associated with evidence produced during AIV activities and assigned verdicts. These verdicts are used to update requirement compliance and closeout status throughout the verification process and during review milestones.	• SN-23 • SN-5 • SN-24 • SN-12 • SN-11	Systems & AIV Engineer Subsystem Engineer / Subject Matter Expert (SME) / Test Operator Product Assurance Engineer
Track Overall Verification Status	Provide an integrated overview of the verification status of system requirements and verification activities. This includes monitoring requirement close-out status, qualification status, and the progress of verification activities, enabling stakeholders to assess verification completeness and readiness for project reviews.	• SN-8 • SN-6 • SN-10	Product Assurance Engineer Project Manager Systems & AIV Engineer
Define Model Philosophy Consistent with the AIV Plan	Keep an overview over the adopted model philosophy and differences in configuration between different models of the same hardware element. A model philosophy that is in line with the overall AIV Plan ensures that the required hardware models are available when needed to be integrated into higher level assemblies or test setups.	• SN-21 • SN-22 • SN-4	Product Assurance Engineer Systems & AIV Engineer

Continued on next page

Name	Description	Refines	Related Stakeholders
Develop and Maintain the AIV Plan	Develop and maintain a model-based representation of the AIV plan, capturing dependencies between AIV activities and defining their logical execution sequence. Inputs and outputs of AIV activities are linked to represent integration and verification dependencies, providing a structured basis for planning and optimization of the overall AIV strategy.	• SN-14 • SN-15	Systems & AIV Engineer Project Manager
Prepare for Qualification Review	Support the preparation of project review artefacts by generating documentation and evidence packages from the MBSE model. This includes compiling verification results, verification evidence, and traceability information required to demonstrate readiness for qualification reviews.	• SN-7 • SN-9 • SN-3	Product Assurance Engineer Systems & AIV Engineer ESA / Customer / Reviewer
Handle Anomalies and NCRs	Support the documentation and tracking of anomalies and non-conformance reports (NCRs) identified during AIV activities. While anomaly handling is not the primary focus of the methodology, the model can capture links between anomalies, affected system elements, and verification activities.	• SN-1	Product Assurance Engineer

4.1.3. MBSE AIV Methodology Requirements

The stakeholder needs presented in the previous section were translated into a set of requirements for the MBSE AIV methodology. These requirements define what the methodology should enable in order to support Systems Engineering and AIV activities in post-CDR project stages. They address traceability between requirements and verification activities, modeling of AIV elements and interfaces, AIV planning and flow modeling, verification control, evidence management, procedure generation, non-conformance handling, and the definition of the methodology itself in terms of metamodel, process, modeling methods, views, and tool implementation.

Table 4.6 provides an overview of the resulting methodology requirements. Each requirement is identified by an ID, given a short name and requirement statement, and traced back to the stakeholder needs from which it was derived. The rationale column explains why the requirement is relevant for a post-CDR MBSE AIV methodology. Together, these requirements form the basis for the methodology design and are later used to evaluate whether the developed approach satisfies its intended purpose.

Table 4.6: Requirements for the MBSE AIV Methodology

ID	Name	Text	Derived From	Rationale
R-2	Requirements to Verification Activities Traceability	The MBSE AIV Methodology shall allow traceability from Requirements to Verification Activities.	• SN-5 • SN-23 • SN-11	Needed to indicate Verification Activities are used to verify which Requirements.
R-3	Elements and Interfaces	The MBSE AIV Methodology shall allow modeling and reuse of AIV elements such as System Models, GSE, and facilities and their interfaces.	• SN-22 • SN-21 • SN-5	Expert interviews revealed that unclear interface definitions are often a source of mistakes.
R-4	Automated Interface Consistency Check	The MBSE AIV Methodology shall be able to highlight incompatible interfaces during preparation of a test activity.	• SN-5	When modeling interfaces of test setups, automated interface consistency checks can help to ensure correct definition.
R-5	AIV Flow Modeling	The MBSE AIV Methodology shall enable the user to model the flow of AIV Events and Activities.	• SN-15 • SN-14	An AIV flow is needed to properly plan the AIV effort for e.g. a system or subsystem.

Continued on next page

ID	Name	Text	Derived From	Rationale
R-6	AIV Flow Dependencies	The MBSE AIV Methodology shall allow the user to model can and Activities in the overall AIV Flow.	• SN-5 • SN-15 • SN-14	Dependencies between AIV activities are relevant in the planning of AIV activities and for impact analysis.
R-7	Verification Control Document Views	The MBSE AIV Methodology shall allow the user to generate at least one Verification Control Document View from the Model with up to date data.	• SN-12 • SN-10 • SN-9 • SN-8 • SN-7	A Verification Control Document is often used in space missions to document and communicate Verification status and is a central deliverable for projects.
R-8	Verification Control Parameters	The MBSE AIV Methodology shall allow the user to define the following verification indicators: Requirement Closeout and Status of Compliance for each Requirement; Verification Method, Stage, Level, and Kind for each Verification Activity; Verdict for each Verification Criterion.	• SN-20 • SN-8	According to ECSS-E-ST-10-02C Rev.1
R-9	Verification Results Tracking	The MBSE AIV Methodology shall allow the user to define and trace the results of verification activities to the relevant criteria that are evaluated based on this evidence.	• SN-24 • SN-12 • SN-10 • SN-8 • SN-5	Results of verification activities need to be connected to criteria and to the activities that create them to be able to reason about requirement closeout and verification activity success.
R-10	Reference Classical Documentation	The MBSE AIV Methodology shall allow the user to reference classical documents (meaning pdf, docx, and excel) to model elements.	• SN-23 • SN-19	Referencing classical documentation is realistically needed even if the project itself is fully model-based.
R-11	Standards Alignment	The MBSE AIV Methodology shall conform to ECSS wording and highlight any deviations clearly.	• SN-3	Using similar and consistent wording helps adopting a model-based approach and reduces confusion.
R-12	Test Specification Modeling	The MBSE AIV Methodology shall allow the user to create test specifications according to ECSS-E-ST-10-03-Rev.1 Space Engineering: Testing, Annex B.	• SN-18 • SN-16 • SN-17 • SN-3	Creating test specifications is an important part of AIV planning.
R-16	Generating test procedures	The MBSE AIV Methodology shall allow the user to generate test procedures from the model for easy use in a lab or cleanroom.	• SN-13	Printed out procedures are preferred by engineers working in the cleanroom compared to working on a computer. Some cleanrooms don't allow to take dedicated computers into cleanrooms.
R-18	Unique Identification	The MBSE AIV Methodology shall ensure that Requirements, Verification Activities, Elements, Artefacts, and Criteria all have unique identifiers.	• SN-6 • SN-4	Numbered IDs on model elements help with tracking and identification also outside of the modeling environment.
R-19	NCR Processing	The MBSE AIV Methodology shall allow the user to add Non-Conformances to the Model to track and process anomalies.	• SN-1	Handling Nonconformances is an important part of the product / quality assurance work.
R-21	Methodology Metamodel	The MBSE AIV Methodology shall describe the underlying Metamodel that is used for the methodology.	• SN-3	Part of an MBSE methodology according to ISO 24641
R-22	Methodology Process	The MBSE AIV Methodology shall describe a set of processes that are covered by it.	• SN-3	Part of an MBSE methodology according to ISO 24641

Continued on next page

ID	Name	Text	Derived From	Rationale
R-23	Methodology Modeling Methods	The MBSE AIV Methodology shall describe a set of concrete modeling methods that are foreseen to perform the set of processes.	• SN-3	Part of an MBSE methodology according to ISO 24641
R-24	Methodology Views and Viewpoints	The MBSE AIV methodology shall define viewpoints and corresponding views that support the processes and methods.	• SN-3	Part of an MBSE methodology according to ISO 24641
R-25	Metamodel implementation	The metamodel shall be implemented in at least one existing modeling language used for MBSE.	• SN-3	Part of an MBSE methodology according to ISO 24641
R-26	Modeling tool selection	The methodology design shall include a preliminary tool selection considering the aspects: Customizability, Report Generation & visualization capabilities, Industrial Relevance, and Academic Availability.	• SN-3	Part of an MBSE methodology according to ISO 24641

4.2. Methodology Design

As mentioned in the literature study, an MBSE Methodology consists of an ontology, a metamodel, a process and methods description, an integrating framework, a modeling tool, and modeling language.

The following sections will present each of these aspects for the MBSE AIV Methodology.

4.2.1. Ontology

For the scope of this thesis it was considered sufficient to not differentiate between the metamodel and the Ontology. As stated before, the difference between a metamodel and an ontology is that the ontology describes a certain domain (in this case the SE and AIV domain) while a metamodel is a conceptual description of a modeling language. This following section aims to introduce a concrete view of the SE and AIV domain, with a relevant set of concepts that will guide further understanding and scoping of the methodology, as well as describe the used model elements. For simplicity, the term Ontology will be used, but the reader is cautioned to be aware of the theoretical differences between the terms metamodel and ontology. Each element of the ontology will be found as a model element, implemented in the modeling language, in the concrete examples discussed later in the thesis.

The following requirements are imposed on the definition of the ontology to promote a consistent definition of the conceptual foundation of the methodology. These are presented in Table 4.7. These requirements will serve as a guideline for the definition of the ontology, and also represent an input for the verification of the ontology later.

Table 4.7: Ontology definition rules used for the MBSE AIV methodology

ID	Ontology Definition Rule
O-1	The ontology shall consist only of ontology elements and relations between them. No additional modelling constructs shall be introduced at the conceptual level.
O-2	Each relation shall connect exactly two ontology elements. Relations shall be strictly binary.
O-3	Each relation shall have a unique and descriptive name that clearly expresses the semantic meaning of the relationship between the two ontology elements.
O-4	For each relation, both connected ontology elements shall be assigned a role name, defining how each element participates in the relation.
O-5	Each relation shall define multiplicities for both roles, specifying the allowed number of participating instances (e.g. 0..1, 1, 0..*).
O-6	Specialization relationships between ontology elements shall be expressed using generalization (inheritance), indicating that a specialized element is a specific kind of a more general element.
O-7	The ontology shall be defined such that each element and relation can be directly mapped to SysML v1 constructs.

The notation in Figure 4.3 has been simplified as compared to the full ontology that can be found in the annex. Both versions are different views on the fully modeled Ontology, which was modeled in SysML. Other options would have been Object-Role-Modeling [60] or the Web Ontology Language [62]. While these would have allowed for more semantically rigorous modeling, this would likely have hindered understanding, while SysML v1 is also suitable. The full Ontology and its implementation are shown in Appendix B.

The notation used for ontology modeling is explained with Figure 4.1 and Figure 4.2. The ontology only consists of ontology elements and relations between them. A relation always connects two ontology elements with each other. The relation has a name and specifies the roles that the two ontology Elements take in relation to each other, as well as the allowed multiplicity. In the example shown in Figure 4.1, *LibraryMember* and *Item* are the elements which are related to each other by the borrows relationship. That relationship indicates that a *LibraryMember* can borrow an *Item* and the *Item* would take the role of *borrowedItem* for the *LibraryMember* while the *LibraryMember* takes the role of *borrower* for the *Item*. Multiplicities are also indicated in the relation. *0..1* means that an *Item* can be borrowed by none or exactly one *LibraryMember* and the *0..** indicates that a *LibraryMember* can borrow zero or more *Item*. Lastly, the concept of Specialization / Generalization is indicated by the hollow arrow. This is equivalent to an inheritance and in this case means that *Journal*, *Book*, and *DVD* are all a kind of *Item* that can be borrowed. Figure 4.2 shows the ontology elements and associated roles of other elements, if the relation is not shown in the diagram. Even if the relation is not shown, the underlying model still contains all information of the roles and multiplicities, but it is represented differently.

In the later depictions of ontologies, often the Roles are omitted in the text to preserve readability, but a complete description including roles can be found in the annex.

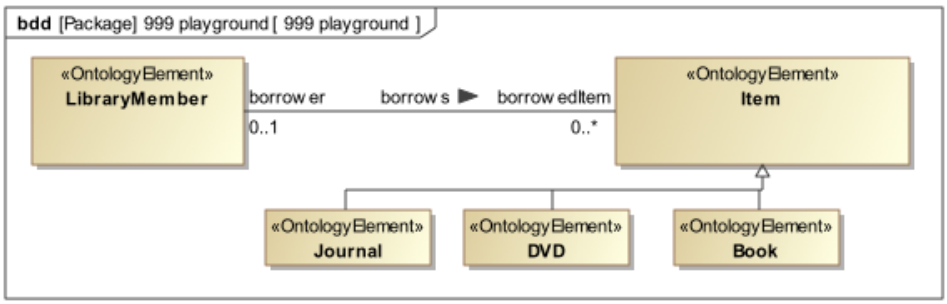


Figure 4.1: Ontology notation first example

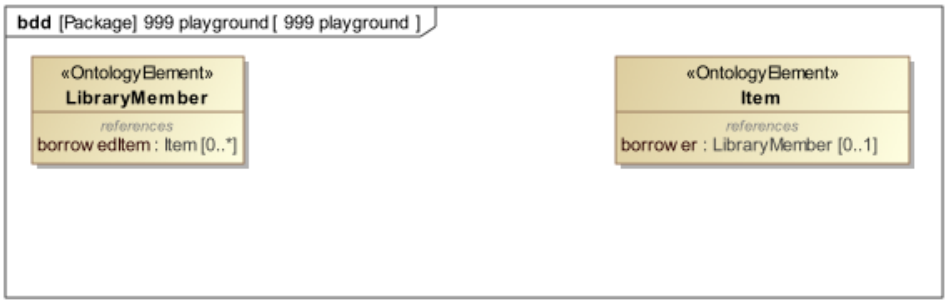


Figure 4.2: Ontology notation second example

The main elements in the Ontology that will be described here are:

- Requirement
- AIV Activity
- AIV Criterion
- AIV Evidence
- AIV Step
- AIV Entity
- AIV Setup
- AIV Event
- AIV Flow

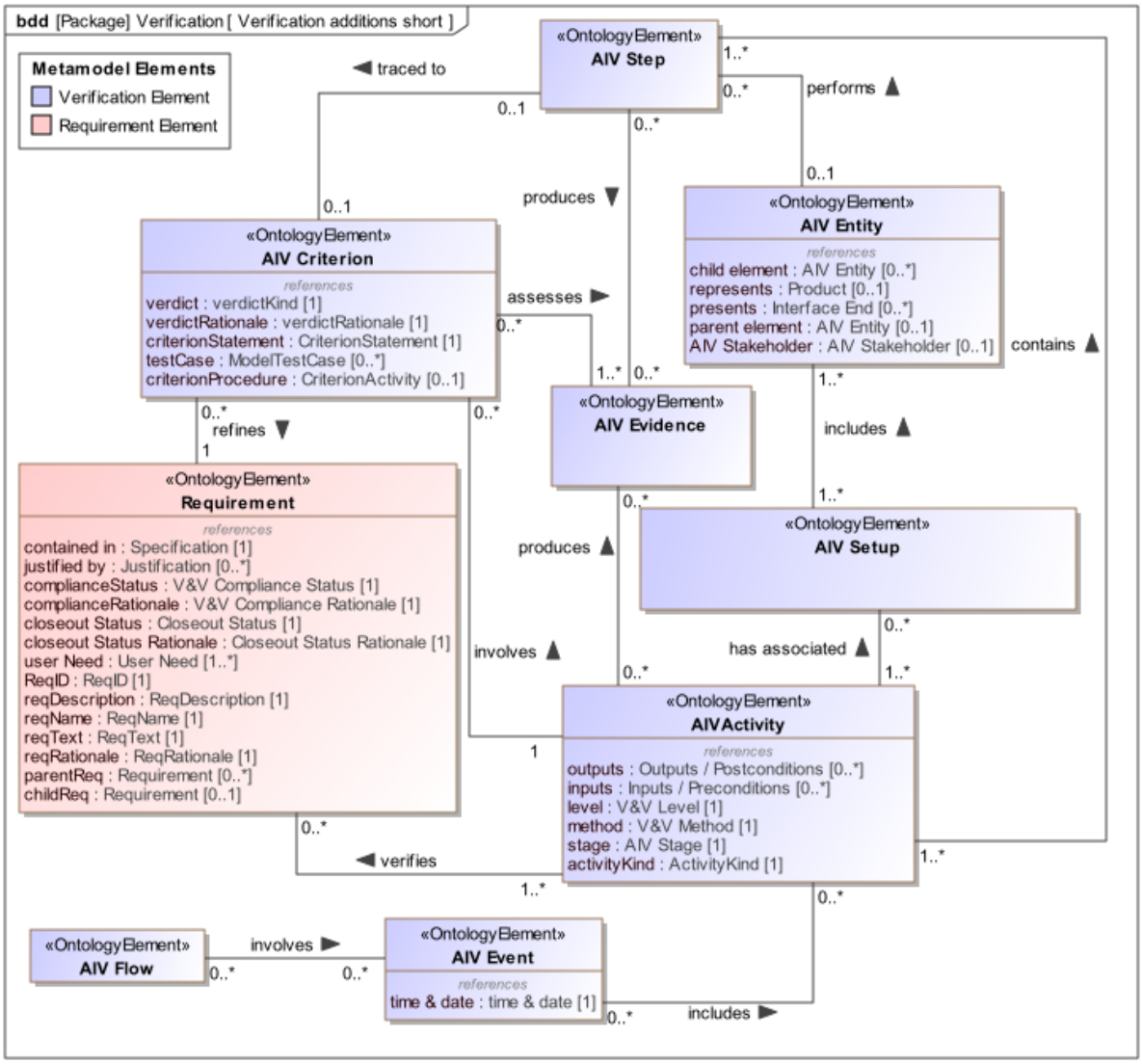


Figure 4.3: Simplified MBSE AIV Ontology

Requirement

Requirements are very important in a satellite project life cycle and especially during the AIV phases of a project. There are a few conceptual elements and properties that are important to capture in the ontology that relate to a requirement in practice. The following selection of concepts is not necessarily complete and might differ in actual projects. A requirement usually has a unique identifier / requirement ID which is often used to reference the requirement in a document based approach. The requirement text contains the textual shall statement of the requirement. Additionally the requirement has a name (sometimes also called title) and a rationale giving additional detail. Another option to provide further details is to attach a justification to the requirement. This concept is directly taken from the ESA MBSE Solution [19]. For the purpose of verification control, the ECSS-E-ST-10-02C [9] calls for a status of compliance (possible values: Compliant, Non-compliant, Partially compliant, Unknown, Not applicable), a close-out status (possible values: Open, Closed) and a reason / rationale for the close-out status. Sometimes the status of compliance can be split into one status per major project phase. In later chapters of the thesis this will be shown. Traceability between requirements is also important and has to be captured by the model. This is indicated in the ontology via the parent requirement and child requirement roles. According to the ECSS standard also the parameters verification level, verification stage, and verification method are supposed to be related to a requirement. The following will present that these parameters are related to AIV Activities instead. Another important related elements to be

explained for the purpose of this thesis is the AIV Criterion.

AIV Activity

One of the major objectives of this thesis is to explicitly model the AIV aspects of a satellite development project. In the later stages of the project it is crucial to capture the approach to perform assembly, integration, verification and validation. As presented in subsection 2.3.1, the AIV process is broken down into singular AIV activities. While many sources in literature utilize a slightly more restrictive terminology of verification and validation activities, this thesis generalizes the scope to also include assembly and integration activities. The expanded term AIV Activity indicates that also in assembly and integration activities, certain criteria are checked and certain evidence is created that can be evaluated to inform the compliance status of a requirement.

An AIV Activity *verifies* zero to many requirements, which is indicated by verify relations and a requirement is verified by at least one AIV Activity. An AIV Activity has exactly one of each of the parameters: verification level, verification method, verification stage, and activity kind.

The *verification level* indicates the level in the system architecture breakdown and is highly project specific because projects might adopt a specific terminology. In a satellite development project these levels might be mission level, system level, subsystem level, or component level. A verification activity is conducted on exactly one level. The level of an integration activity is the level that corresponds to the final product. For example system integration means integrating subsystems into the complete system.

Possible values for the *verification method* are either test, analysis, review of design, or inspection, taken from ECSS [9]. An AIV Activity is associated with exactly one of these methods, so an activity can either be an analysis or a test, but not both.

Verification stage refers to the stage of the project, or its phase. This can be very project specific, but for projects that make use of the ECSS terminology this could be Phase 0, A, B, C, D, E, or F, as presented in subsection 2.2.2.

The *activity kind* describes what kind of activity is meant and indicates the overall purpose of the activity. Possible values are assembly, integration, verification, or validation. Also here, real projects might adopt slightly different terminology.

The goal of an AIV Activity is to evaluate a certain set of criteria based on evidence that is produced by performing the activity. This logic closely follows the terminology presented in [43]. More generally, an AIV Activity can have inputs that are required to perform the activity and outputs that are produced by the activity.

It should be noted that an AIV Activity often has a procedure that represents or describes it, especially if the AIV Activity is an assembly or integration activity, or has the test method. The main purpose of a procedure is to show a sequence of steps (=AIV Steps) that describe how the activity should be performed. Because of this, it is considered as a view of the model rather than a model element itself and thus not present in the ontology.

AIV Criterion

An AIV criterion is a refinement of a requirement which is specific to one AIV activity. The criterion describes what exactly is checked or evaluated in order to verify the requirement. Similarly to a requirement, a criterion has an ID, a name, a statement (= textual description), and a rationale. Furthermore it has a verdict parameter that can have the values pass, fail, error, or inconclusive. A criterion must be defined such that it can be evaluated and the evaluation can be expressed as one of the four values of the verdict parameter. In ECSS terminology, in the context of the ECSS-E-ST-10-03-Rev.1 [10], the term pass/fail criterion is used.

The important conceptual distinction between a requirement and a criterion is that a criterion is created for one specific AIV Activity, while a requirement can be verified by multiple AIV Activities. One abstracted example for this distinction from the space domain is the following [1]:

A requirement states:

[1]The example focuses on the concept and does not show properly defined and worded requirements

- *The satellite shall survive the launch environment of a specific rocket*.

The requirement might be verified by means of Analysis and Test. One criterion refining this requirement could be that the AIV Activity *Vibration Analysis* shall show:

- *All structural elements shall exhibit positive margins of safety under the specified quasi-static and dynamic loads derived from the launcher environment, with applied simulation margins.*

Other criteria may be related to the AIV activity *Random Vibration Test on Shaker* and read:

- *One vibration sensor shall confirm that the experienced vibration levels induced by the shaker conform with the specified random vibration profile.*
- *The satellite shall exhibit no structural damage, no loose components, and no shift of witness marks after exposure to the specified random vibration profile.*
- *The satellite shall remain fully functional after exposure to the specified random vibration profile.*

All criteria relate to the same requirement, but there are two AIV Activities and each needs to have its own criteria. Each AIV activity therefore requires its own set of criteria that translate the same requirement into method-specific, observable, or computable conditions. While the requirement expresses the intended capability of the system in a solution-independent manner, the criteria operationalize the requirement in the context of a particular verification approach. Consequently, the satisfaction of a requirement is not assessed directly, but rather inferred from the successful evaluation of all associated AIV criteria. When all relevant criteria have a pass verdict the requirement can be considered verified. Conversely, a fail, error, or inconclusive verdict in any criterion indicates that the requirement verification might be incomplete or unsuccessful and requires further analysis, testing, or corrective action.

The relation that an *AIV Criterion assesses AIV Evidence* expresses that a criterion is not evaluated in isolation, but always based on concrete evidence produced by an AIV activity. In other words, the criterion defines what to check, while the evidence provides what is checked. For example, in the random vibration test, sensor data, pictures of the hardware inspection, confirmation that nothing is rattling, and pictures of witness marks constitute the AIV evidence. Evidence relevant for evaluating a criterion is therefore related in the model. This relation captures the step of transforming raw test or analysis results into a singular verdict value (pass, fail, inconclusive, error) and supports the reasoning and traceability for the value.

AIV Evidence

AIV Evidence as an Ontology Element represents all the artifacts that are produced by AIV Activities that help to assess an AIV Criterion. In practice this can be a lot of things. Some examples that were mentioned above are raw sensor data or pictures taken during performance of the activity. The main purpose of modeling the evidence before the actual activity is performed is to create a common understanding about the answers to the following questions:

- What evidence / raw data will be generated during the planned activity?
- How does this raw data relate to the defined criteria?
- Is the expected data sufficient to evaluate all criteria in question?
- Does everyone understand the rules by which the evidence is used to evaluate the criteria?

When describing an AIV activity in more detail, it is broken down into singular *AIV Steps*. In this lower level of detail, the AIV Evidence is produced by an AIV Step within the higher level AIV Activity.

AIV Step

The AIV Step describes one step within an AIV Activity, i.e. steps that are performed in a procedure that represents the AIV Activity. It is performed by some AIV Entity and can be related to an AIV Criterion. To stay with the previous example, during the Random Vibration Test, there could be a step called *Check and take pictures of witness marks*.

AIV Entity

An AIV Entity is a concept that represents anything physical. It can be the unit under test, e.g. satellite qualification model, ground support equipment, a test sensor, tools, or facilities. It can also represent a role that a person plays during an activity, e.g. the role of test operator or quality assurance engineer. An entity can perform AIV Steps within an AIV Activity, and it can be included in one or many setups. Part of the work of planning an AIV Activity is to assign responsibilities to different stakeholders. This could indicate what person / organization is responsible for providing a test facility, the test operator role, the quality assurance engineer role, or the unit under test.

AIV Setup

An AIV Setup describes the complete set of AIV Entities, their configuration and interfaces that are needed to perform the AIV Activity. It represents the physical context in which an AIV Activity is executed. In particular, the modeling of interfaces between AIV Entities is relevant to ensure compatibility and prevent surprises during the execution of the activity. While an AIV Activity defines what needs to be done, the AIV Setup defines which items, ground support equipment, tools, and interfaces are involved.

AIV Event

The ontology element AIV Event represents a planned event during which one or more AIV Activities are performed. An event typically has a time and a date associated with it for planning purposes. An example for an AIV Event could be an environmental qualification test campaign of a satellite, which would include vibration tests and thermal vacuum tests as activities.

In addition to scheduling information, an AIV Event provides a structuring mechanism to group AIV Activities that are executed within a common context or setting, such as a specific facility or test campaign. Such compartmentalization can help with splitting up the work into smaller packages and putting structure to a collaborative workflow.

AIV Flow

The AIV Flow is an overview over different AIV Events, showing the logical sequence of events and how they relate in terms of inputs and outputs. Dependencies between AIV Events can be captured in the AIV Flow, for example if an analysis needs to be completed in preparation for a test campaign. An example from the space systems engineering context is an environmental test campaign, which usually requires multiple analyses to be performed for example to justify sensor placements. Another example is testing of a subsystem assembly before it is integrated into the overall system. The AIV Flow and the AIV Events represent the interface between classical AIV and Systems Engineering and project management responsibilities and work packages.

4.2.2. Ontology Implementation

The above described ontology is implemented into SysML v1 using the stereotyping mechanism of the modeling language and further augmented by customization elements which allow to modify and streamline the modeling experience further. A partial implementation is shown in Figure 4.4, showing the stereotypes that implement the core ontology elements. All stereotypes are organized within the *AIV-profile*. This allows to easily apply the profile, including all its stereotypes and other customizations in any project. When applying the profile in other projects, it is advisable to carefully review all profile elements and customizations and tailor them to the specific needs, conventions, and terminology that is used.

A certain degree of customization can also be seen in Figure 4.4, on the stereotype implementing the ontology element Requirement. The stereotype is called *DVSRequirement* because this is in line with the application example presented later. The full implementation mapping of all ontology elements and relations is presented in Appendix B.

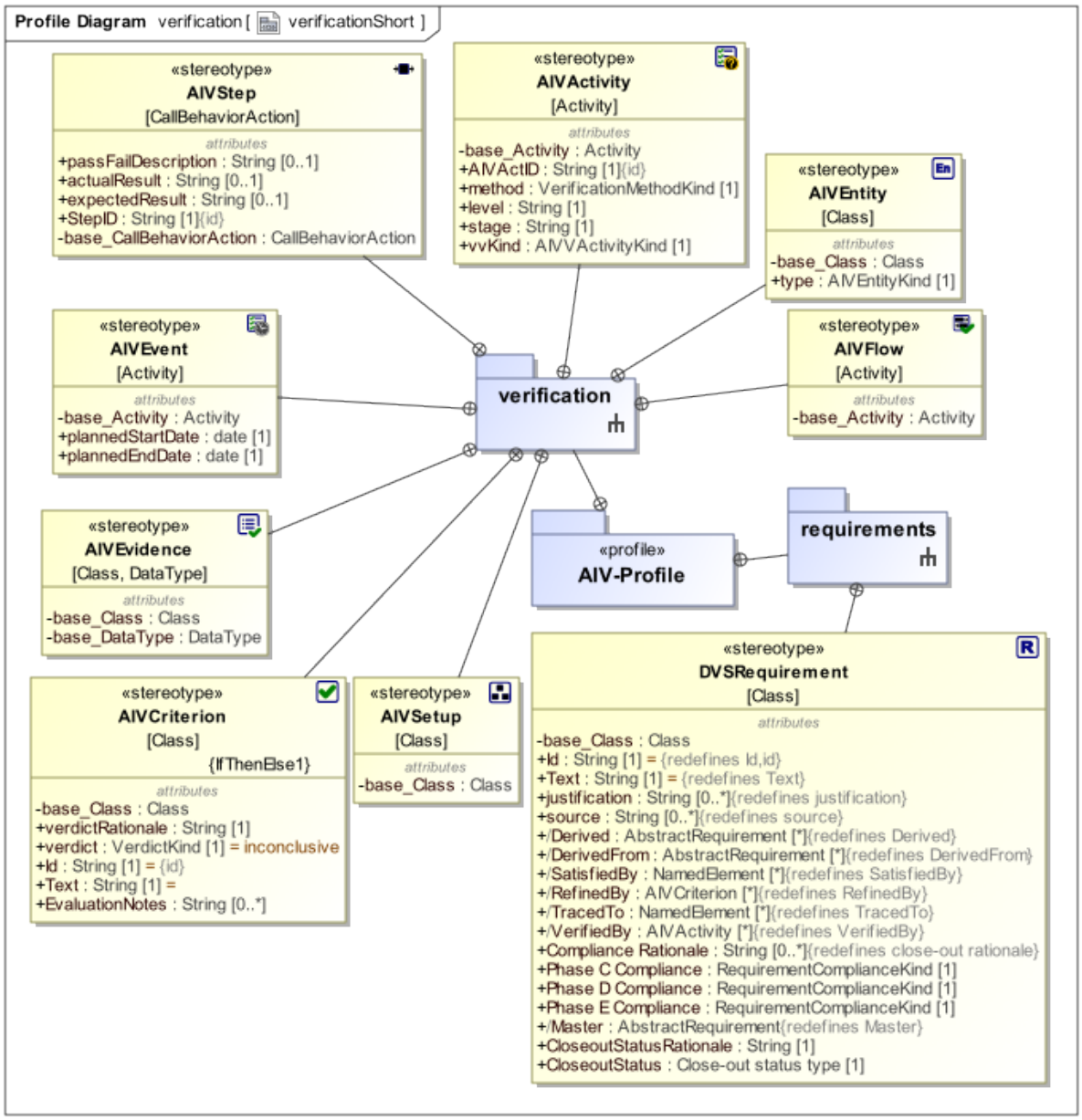


Figure 4.4: Implementation of the Ontology - abbreviated

4.2.3. Process

This section will describe the high level process that is considered for the application of the MBSE AIV Methodology, focusing on the overall objectives of the process steps and the main inputs and outputs. The following section will then dive into detail for each process step and describe how the modeling framework and the methods facilitate the process, making use of a small example.

The process considered in this methodology is closely aligned with the typical process applied by AIV engineers in projects, which was presented in the literature review chapter. The process definition is heavily inspired by the process shown in Figure 2.10, as presented by Avner Engel in [17].

It is important to keep in mind that this process, even though it is represented here as a linear progression, is highly iterative in nature. The point of showing this process model is to facilitate an overview over the different steps and serve as a starting point for the MBSE method steps, which explain how to perform them in the modeling environment.

The process is organized in these main steps:

0. Tailoring of the MBSE AIV Methodology
1. Rough AIV Planning
2. Detailed per AIV Activity Planning
3. AIV Activity Execution & Postprocessing
4. AIV Process & Quality Control

The overall process is shown in Figure 4.5.

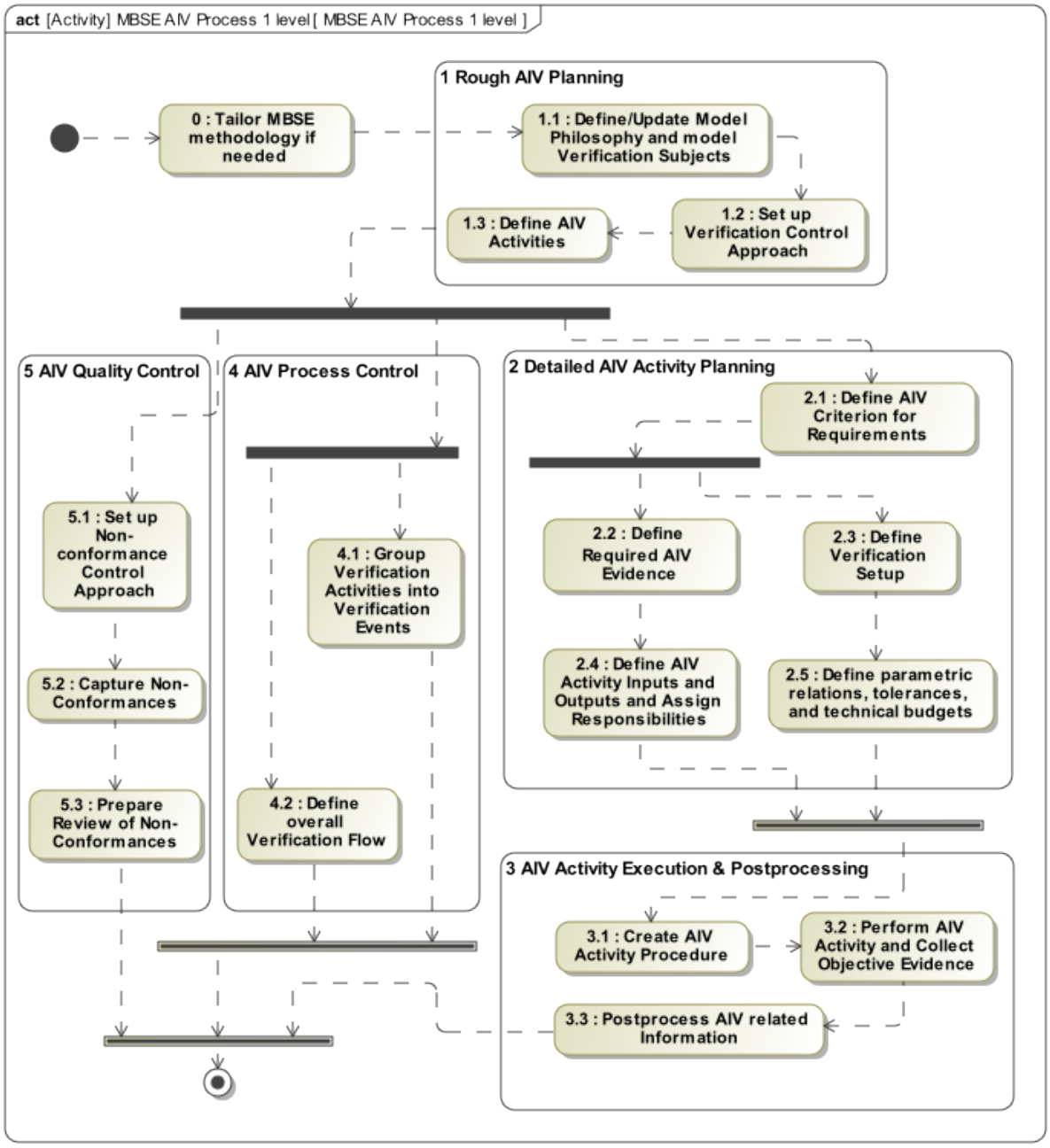


Figure 4.5: AIV MBSE Process modeled in SysML

0. Tailoring of the MBSE AIV Methodology

The very first step of the process consists of tailoring the methodology to the specific project context in which it is applied. All elements of the methodology may be adapted where necessary to ensure

suitability with project constraints, objectives, and organizational practices. This tailoring step not only enables alignment with the project environment, but may also foster deeper understanding of the methodology. Such an understanding is required, because MBSE is generally not an out-of-the-box solution, but rather an approach that requires conscious and informed application by the user.

1. Rough AIV Planning
The purpose of Rough AIV Planning is to establish an initial, high-level definition of how verification will be approached for the system under consideration and its subsystems. In this step, the focus is on structuring the verification approach rather than defining detailed implementations. The key elements in this step include the definition of an initial model philosophy, which determines which verification subjects exist, and the establishment of a verification control approach, to specify how compliance of requirements is tracked and assessed throughout the project. Based on that, a first set of AIV Activities is defined, to indicate which activities will be performed. For each AIV Activity the method, stage, level, and the kind of activity is described. The outcome of this process step is a structured overview of the verification approach per requirement in a view similar to a Verification Control Document, and the applied model philosophy, meaning how many versions of hardware models will be used.

The subprocesses involved in Rough AIV Planning are:

1.1 Define or update model philosophy and verification subjects

1.2 Define verification control approach

1.3 Define AIV Activities

The main outputs of this process step are:

- Model philosophy for AIV processes
- Defined verification control approach
- Initial set of AIV Activities, including their verification level, method, stage, and kind
- Mapping between requirements and AIV Activities

2. Detailed AIV Activity Planning
The purpose of Detailed AIV Activity Planning is to refine the previously defined AIV Activities into executable and verifiable elements by specifying how each activity will be performed and how compliance will be assessed. While Rough AIV Planning establishes *what* activities need to be performed, this step focuses on defining *how* these activities are realized in terms of specific criteria, generated evidence, and required setup.

For each requirement that is verified by an AIV Activity, AIV Criteria are defined which translate the requirement into observable or measurable conditions that can be evaluated within the scope of the activity. Based on the criteria, the required AIV Evidence is specified, ensuring that all necessary data and artifacts are planned prior to execution and aligned with the criteria. Furthermore, the AIV Setup is defined, capturing all required AIV Entities, their configuration, and interfaces needed to perform the activity, as well as the roles needed to be performed by humans to complete the activity.

All inputs and outputs of the activity are identified, and responsibilities are assigned to relevant stakeholders. Where applicable, parametric relations are considered, for example to ensure that measurement tolerances and error budgets are considered in the planning and within acceptable margins.

The outcome of this process step is a complete and consistent definition of each AIV Activity, including all elements required for execution and evaluation. This forms the basis for generating procedures and performing the activities in a controlled and traceable manner.

The subprocesses involved in Detailed AIV Activity Planning are:

2.1 Define AIV Criteria for Requirements

2.2 Define Required AIV Evidence

2.3 Define AIV Setup

2.4 Define AIV Activity Inputs, Outputs, and Responsibilities

2.5 Define parametric relations, tolerances, and technical budgets

The main outputs of this process step are:

- Defined AIV Criteria for each requirement–activity combination
- Defined AIV Evidence required to evaluate each criterion
- Defined AIV Setups, including entities, configurations, and interfaces
- Defined inputs and outputs of AIV Activities and assigned responsibilities
- Defined parametric relations, tolerances, and technical budgets where needed
- Fully specified AIV Activities ready for creation of step-by-step procedures and execution

3. AIV Activity Execution and Postprocessing

The purpose of AIV Activity Execution and Postprocessing is to perform the previously defined AIV Activities in accordance with their specifications and to transform the generated raw data into structured information that can be used for assessing requirement compliance and closeout. In contrast to the preceding steps, this one represents the transition from planning and defining specifications to actual execution and evaluation.

During this phase, the AIV activity is executed based on predefined procedures, which describe the sequence of AIV Steps to be performed in detail. As part of the execution, AIV Evidence is generated in the form of raw data, measurements, observations, or other artefacts. This evidence is collected in a predefined, controlled, and traceable manner.

Following the execution, the generated evidence is post-processed and all defined criteria are evaluated. Based on the processed evidence, verdicts are assigned to the AIV Criteria, indicating whether they are satisfied. The verdicts then contribute to the overall assessment of requirement compliance and finally closeout. The overall outcome of this process step is a set of evaluated AIV Criteria with associated and traceable evidence.

The subprocesses involved in AIV Activity Execution and Postprocessing are:

3.1 Define AIV Activity Procedure

3.2 Perform AIV Activity and collect evidence

3.3 Postprocess AIV-related information

The main outputs of this process step are:

- Defined AIV Activity procedures describing detailed execution steps
- Collected AIV Evidence in the form of raw data generated during activity execution
- Assigned verdicts for AIV Criteria based on post-processed AIV Evidence
- Updated verification status of requirements based on criterion results

4. AIV Process Control

The purpose of AIV Process Control is to structure and coordinate the execution of AIV Activities at a higher level with the aim to ensure that the overall verification process progresses and that a logical sequence of steps is kept. While the previous step focuses on individual AIV Activities in isolation, this step focuses on organizing them into a sensible process that aligns with project timelines and milestones.

In this step, individual AIV Activities are grouped into AIV Events, which represent planned occasions such as test campaigns during which multiple activities might be performed. All the activities and events can be combined into an AIV Flow that captures the dependencies between different events.

The subprocesses involved in AIV Process Control are:

4.1 Group AIV Activities into AIV Events

4.2 Define overall AIV Flow

The main outputs of this process step are:

- Defined AIV Events grouping related AIV Activities
- Defined AIV Flow capturing sequence and dependencies between events and activities

5. AIV Quality Control
The purpose of AIV Quality Control is to ensure that deviations from expected outcomes are systematically captured, assessed, and resolved. While previous steps focus on planning, executing, post-processing, and coordination of AIV Activities, this step addresses the management of non-conformances that might arise.

In this step, the approach on how to handle non-conformances is established, defining how they are documented and reviewed. When deviations are observed during the execution of AIV Activities or evaluation of AIV Criteria, non-conformances are captured and traced to relevant artifacts. The non-conformance is classified as major or minor depending on the severity. To assess its impact, the non-conformance is linked with all affected elements, including requirements, AIV Activities, AIV Entities (e.g. hardware or software under test), and, where applicable, specific AIV Steps within a procedure. Each non-conformance includes information about its cause and a description of proposed or implemented corrective actions. A disposition is assigned to indicate how the non-conformance is handled.

After that, a review of the non-conformance is prepared by gathering all relevant information and identifying the relevant stakeholders to be involved in the approval and closeout process. The goal of the approval and closeout process is to identify the real impact, root cause, and proposed resolution for the non-conformance. The review process involves one or more designated stakeholders as approvers, who are responsible for assessment and issuing formal approval.

The subprocesses involved in AIV Quality Control are:

5.1 Set up non-conformance control approach

5.2 Capture non-conformances

5.3 Prepare review of non-conformances

The main outputs of this process step are:

- Defined non-conformance control approach
- Captured Non-Conformances
- Traceable links to affected requirements, activities, evidence, entities, and procedure steps
- Documented causes, corrective actions, and dispositions
- Prepared deliverables for non-conformance reviews

4.2.4. Modeling Tool and Language selection

The modeling tool selection and language selection are presented here in a single section, since they practically go together, meaning that MBSE tools usually have a certain language associated with them.

Purpose of the MBSE Tool Trade-off
The objective of the tool trade-off is to select a modeling environment that supports the development and application of an MBSE AIV Methodology. In this context, the tool must enable:

- Requirements engineering
- Verification modeling
- End-to-end traceability
- AIV process modeling
- Automated documentation generation

In addition, the selected tool must support methodology implementation through mechanisms such as custom stereotypes, profiles, and model extensions. The trade-off therefore not only demonstrates methodological rigor by considering alternatives, but also justifies the selected modeling environment.

Candidate MBSE Tools

Table 4.8 shows the MBSE tools considered for this trade-off.

Table 4.8: Candidate MBSE tools considered in the trade-off

Tool	Developer	Modeling Language
Magic Systems of Systems Engineer (Cameo)	Dassault Systèmes	SysML v1
Capella	Eclipse Foundation	Arcadia
Enterprise Architect	Sparx Systems	SysML v1
SysIDE	Sensmetry	SysML v2
Eclipse Papyrus	Eclipse Foundation	SysML v1

These tools represent both commercial and open-source approaches and are widely adopted in both industry and academia. Two tools that were explicitly excluded were CDP4-COMET and Valispace, because they do not primarily function as full MBSE modeling environments. Instead, they focus on data management, concurrent engineering, and parameter tracking rather than providing comprehensive support for system modeling across requirements, structure, behavior, and verification. As a result, they do not sufficiently support the definition and application of a model-based AIV methodology within a unified modeling framework and were therefore not considered further in the trade-off.

Evaluation Criteria

The evaluation criteria are derived from the specific needs of an MBSE-based AIV methodology.

Extensibility and Methodology Customization - 40 % This thesis project aims to develop a methodology to apply MBSE for AIV related work within a satellite project. This requires the ability to implement custom modeling concepts and ideally a high degree of flexibility to tailor the modeling experience according to the specific needs and requirements identified above. This criterion represents the customizability of the considered tools to add custom model elements, model checking rules, and other user interface and user experience adjustments.

AIV-related concepts modeling - 30 % The criterion describes the capability of the tool to represent AIV-related concepts, including tasks to model requirements, behavior and interactions between systems, structure and interface modeling, and parametric / mathematical equations modeling. All these tasks are relevant for performing AIV planning and execution of AIV Activities.

Reporting and Documentation - 10 % Part of the tasks of the model users of the MBSE AIV Methodology will be to generate documentation from the model, for example to prepare for lab tests or activities in a cleanroom, where bringing a document might be preferred to bringing a laptop. Support for automated generation of engineering documentation is considered a desired feature for the modeling tool selection.

Industrial Adoption - 10 % This criterion describes the relevance of the tool within the aerospace and space sectors. The more widely used a tool is, the better chances there are that the MBSE AIV Methodology can be integrated into an overall MBSE approach in real life. A high score in this category contributes to the credibility of the methodology, the willingness of industry to adopt the methodology, and the required investments to setup the tool in an organization.

Learning Curve and Accessibility of Support Material - 10 % This criterion is another very important one to judge adoption of the approach, but from a model user perspective. The more intuitive the tool is, the easier it will be for new model users to get up to speed finding their way around the tool. Another important aspect that contributes to this is the availability of learning material, for example online materials like videos or forums that can be used for self-study.

Weighing Factors The weighting factors were selected to reflect the primary objective of this research, namely the development and implementation of a model-based AIV methodology. Extensibility is assigned the highest weight (0.4), as the ability to tailor the modeling environment through custom stereotypes, validation rules, and automation is considered essential for embedding AIV-specific concepts and workflows. AIV-related concepts modeling is weighted second highest (0.3), since the tool must natively support the representation of requirements, system behavior, interfaces, and parametric logic. Reporting, industrial adoption, and learning curve are each assigned a lower but equal weight (0.1), as they are important for practical application and usability, but do not directly determine whether the methodology can be implemented in the first place. This distribution ensures that the evaluation prioritizes methodological feasibility.

Evaluation Results
The tools are evaluated using a qualitative scoring scale from 1 to 5:

- 1 = poor
- 3 = moderate
- 5 = excellent

Cameo / Magic Systems of Systems Architect:

Extensibility and Customizability — Score: 5/5 Cameo provides a very strong extensibility through the SysML v1 profiles and stereotypes mechanism, enabling the creation of specific model elements. It furthermore supports scripting in multiple programming languages and more tool native support to implement model validation rules, design custom diagrams and model views, creating model templates, and creating custom plugins for the tool. Tool customizations also allow to hide major parts of the user interfaces complexity by defining custom modeling perspectives.

AIV-related concept modeling — Score: 5/5 Cameo offers a comprehensive implementation of SysML v1, allowing consistent modeling of requirements, structure, behavior, interfaces, and parametrics within the tool.

Reporting and Documentation — Score: 4/5 the tool includes the so-called Report Wizard functionality that enables automated generation of engineering documentation directly from the model. Users can create customizable report templates using the Velocity Template Language, allowing alignment with existing office based documentation. In a collaborative version of the tool, there is also the option to publish interactive document-like representations of models or parts of models in a browser-based environment.

Industrial Adoption — Score: 5/5 The tool is widely used in aerospace, defence, and space engineering, including ESA-related projects.

Learning Curve and Accessibility of Support Material — Score: 4/5 Cameo is a feature-rich tool, which results in a relatively steep learning curve, especially for users new to SysML or MBSE. However, this is mitigated by a large amount of available learning resources, including official documentation tutorials, community forums, and many videos on YouTube. The widespread use in academia and industry also increases chances that a new user has experienced users in their network / professional environment.

Capella:

Extensibility and Customizability — Score: 3/5 Capella provides extensibility primarily through its underlying Eclipse framework and add-ons. While it allows some tailoring of the modeling environment, it does not offer the same level of flexibility as SysML-based tools in defining custom stereotypes or deeply modifying the underlying metamodel. Implementing domain-specific constructs or enforcing methodology-specific constraints is therefore more limited. This makes it moderately suitable, but less ideal for developing a customized MBSE AIV methodology.

AIV-related concept modeling — Score: 4/5 Capella is based on the Arcadia method, which provides strong support for system architecture modeling, including functional and logical decomposition. However, it lacks native support for detailed parametric modeling, and process-like behavior modeling with explicit flow logic. Parametric modeling can be done using external tool solutions, like MATLAB, Python, or excel.

Reporting and Documentation — Score: 3/5 Capella provides basic reporting capabilities through Eclipse-based tools and add-ons, but these are less mature and flexible. While model information can be exported, the level of automation and control over document formatting is more limited. Therefore, it is considered moderately capable in this area.

Industrial Adoption — Score: 4/5 Capella is widely used in aerospace and defence. Its open-source nature makes it attractive for organizations seeking cost-effective MBSE solutions. However, its adoption is still more limited compared to SysML-based tools in certain domains, especially where strong standardization on SysML exists.

Learning Curve and Accessibility of Support Material — Score: 5/5 Capella is generally considered more intuitive than many SysML-based tools due to the structured Arcadia methodology and guided modeling approach. This makes it easier for new users to understand and apply, especially in early system design phases. Additionally, a wide range of tutorials, documentation, and community support is available, particularly due to its open-source ecosystem. This results in excellent accessibility for new users.

Enterprise Architect:

Extensibility and Customizability — Score: 5/5 Enterprise Architect provides customization capabilities through UML profiles, stereotypes, and add-ins, allowing users to extend the modeling language to some extent. It also supports scripting and automation via its API, which enables certain workflow customizations and integrations.

AIV-related concept modeling — Score: 5/5 Just as Cameo, Enterprise Architect supports SysML modeling, including requirements, structure, and behavioral diagrams, which enables representation of the main AIV-related concepts.

Reporting and Documentation — Score: 4/5 The tool provides built-in document generation capabilities, allowing users to create structured reports based on model content. It includes template-based reporting that can be customized, supporting common engineering documentation needs.

Industrial Adoption — Score: 3/5 Enterprise Architect is widely used across various engineering domains, including systems engineering, software engineering, and, to some extent, aerospace. Its relatively low cost and broad applicability contribute to its widespread adoption, particularly in small to medium-sized organizations. However, specifically in the space domain other tools are more dominant.

Learning Curve and Accessibility of Support Material — Score: 4/5 Enterprise Architect is generally considered accessible to new users due to its relatively straightforward interface and lower complexity compared to more feature-rich MBSE tools. A large amount of documentation, tutorials, and community support is available, which facilitates self-learning.

SysIDE:

Extensibility and Customizability — Score: 4/5 SysIDE, based on SysML v2, provides a flexible modeling environment with improved language semantics and a strong focus on modularity and reuse. SysML v2 introduces more formal and extensible language constructs compared to SysML v1, which supports the definition of domain-specific modeling patterns. However, as the ecosystem is still maturing, tooling support for deep customization (e.g., plugins, advanced UI tailoring, extensive scripting environments) is currently less developed than in established tools. Therefore, it offers strong

conceptual extensibility, but somewhat limited practical customization capabilities on tooling and user interface.

AIV-related concept modeling — Score: 4/5 SysML v2 provides enhanced capabilities for modeling structure, behavior, and constraints with improved consistency and precision compared to SysML v1. This makes it well-suited for representing AIV-related concepts, including system interactions and logical verification constructs. However, tooling and best practices are still evolving and established modeling patterns are less mature. As a result, it is highly capable in principle, but not yet fully optimized for applications in practice.

Reporting and Documentation — Score: 5/5 Through the SysML v2 API, model data can be accessed and integrated into highly customizable document generation workflows. With the systems-as-code approach, the data can structurally be accessed and integrated into easy to customize markdown document templates. The tool is suitable for workflows that depend heavily on automated generation of structured engineering documents. The integration into VS Code makes this even more user friendly, making document generation a part of a software pipeline.

Industrial Adoption — Score: 3/5 SysML v2 and tools such as SysIDE are still in an early adoption phase within industry. While there is strong interest and ongoing development, widespread use in aerospace and space engineering projects is currently limited. Organizations are still transitioning from SysML v1 and evaluating the benefits of the new standard. As such, industrial adoption is currently low but expected to grow heavily in the future.

Learning Curve and Accessibility of Support Material — Score: 3/5 SysML v2 introduces new concepts and syntax, which can present a learning challenge even for users familiar with SysML v1. At the same time, the language aims to improve clarity and consistency, which may benefit users in the long term. Learning materials, tutorials, and community support are still developing and less common than for established tools. This results in a moderate level of accessibility at the current stage.

Papyrus:

Extensibility and Customizability — Score: 4/5 Papyrus, as an Eclipse-based open-source modeling tool, provides strong extensibility through profiles, stereotypes, and integration with the Eclipse ecosystem. Users can develop custom plugins, viewpoints, and modeling extensions, allowing a high degree of tailoring to specific methodologies. However, achieving advanced customization requires significant effort and technical expertise compared to commercial tools. Therefore, it offers high extensibility in principle, but with a higher overhead.

AIV-related concept modeling — Score: 4/5 Papyrus supports SysML modeling, including requirements, structure, behavior, and parametrics, enabling representation of most AIV-related concepts. This allows modeling of verification activities, system interactions. Behavior simulation is not as easy to do as in other tools, and a native parametric solver is missing.

Reporting and Documentation — Score: 3/5 Papyrus provides reporting capabilities through Eclipse-based tools the same way as Capella.

Industrial Adoption — Score: 3/5 Papyrus is used in both academia and certain industrial contexts, particularly in organizations that favor open-source solutions. It has a presence in aerospace and defence, often in research or specialized environments. However, its adoption is less widespread compared to leading commercial MBSE tools in large-scale industrial projects.

Learning Curve and Accessibility of Support Material — Score: 3/5 Papyrus can present a moderate to steep learning curve due to its Eclipse-based environment and less guided user experience compared to Capella. While documentation and community resources are available, they are generally less

structured and extensive compared to other tools. Users may require more time to become proficient, especially when configuring advanced features.

Criteria	Weight	Cameo	Capella	EA	SysIDE	Papyrus
Extensibility	0,4	5	3	5	4	4
AIV-concepts modeling	0,3	5	4	5	4	4
Reporting	0,1	4	3	4	5	3
Industrial adoption	0,1	5	4	3	3	3
Learning curve	0,1	4	5	4	3	3
Total Score	1	4,8	3,6	4,6	3,9	3,7

Table 4.9: Evaluation matrix of MBSE tools including weighting factors and resulting scores

Discussion of Results
The results show that the tools separate quite clearly based on their ability to support the development and application of an MBSE AIV Methodology. Cameo and Enterprise Architect score highest due to their strong extensibility and solid SysML support, which play a dominant role in this trade-off. Capella performs well in terms of usability and accessibility, but is limited when it comes to the possibilities to perform detailed behavioral modeling and customization. Papyrus and SysIDE both show strong potential, especially in terms of openness and flexibility, but require more effort or still lack maturity in some areas. In particular, SysIDE and SysML v2 stand out conceptually, offering a cleaner and more rigorous modeling foundation, but are not yet at the same level of tooling maturity. They should be considered for a future implementation.

Tool Selection and Conclusion
Based on the weighted evaluation, Cameo (Magic Systems of Systems Architect) is selected as the modeling tool for this work. It provides the best balance between extensibility, AIV-related modeling capabilities, and industrial relevance, which are the key drivers for implementing the proposed MBSE AIV methodology. While SysML v2 and tools such as SysIDE show clear long-term advantages, especially in terms of language consistency and integration with modern engineering workflows, they are currently not mature enough for this application. Nevertheless, they represent a strong direction for future work and further evolution of the MBSE AIV Methodology.

4.2.5. Framework and Modeling Methods

This section provides an overview over the framework and the more detailed modeling steps to follow the process of the MBSE AIV Methodology using SysML v1 within the modeling tool. For each step, the relevant ontology elements are shown and appropriate model views and diagrams are proposed to support their representation.

The methodology is illustrated using a simplified example of toasting a slice of bread, which is used to demonstrate the application of the modeling concepts in a consistent and intuitive manner. In the frame of sis, the full and detailed method description, including a step-by-step explanation of how to create the example models is considered out of scope. Instead the examples have been chosen such that they can be reproduced by users with a basic understanding of the tool and the language.

The mapping from ontology elements to SysML v1 stereotypes is described where relevant throughout the section. For a full overview of the implementation mapping, the reader is referred to the corresponding section in Appendix B.

The presented methodology can be used as an extension of already existing methodologies, extending them by the core concepts of the AIV discipline. In the small example shown in the following, some model elements were taken from the ESA MBSE solution [19].

0. Tailoring of the MBSE AIV Methodology
Tailoring the MBSE AIV Methodology can happen at any level, meaning that if there is a good reason for it, users might want to, and are encouraged to, tailor any of the constituents to their needs. This can affect the ontology, process, methods, framework, modeling language, and modeling tool. The selection of the modeling tool and language was presented above. Changing other constituents of the

methodology can have many motivations, ranging from project-specific constraints and requirements, organizational standards, and personal preferences, to differences in system complexity, development phase, or available resources. Tailoring might be required to be compatible with existing processes or toolchains within an organization.

This MBSE Methodology can be taken as an addition to an already existing approach, that focuses on early-stage developments. Some candidate approaches were presented in the literature review chapter. In this case, a tailoring will almost certainly be necessary to integrate into an overarching methodology.

Tailoring should reflect the intended use of the methodology. For example, for early design stages, detailed AIV planning or execution phases might not require definition of step-by-step procedures but rather focusing on making the design testable and ensuring that interfaces are well defined in advance. It is advisable to perform any tailoring in a controlled and well-justified manner, while preserving the consistency of the approach.

1. Rough AIV Planning

1.1 Define or update model philosophy and verification subjects Figure 4.6 shows the extract of the ontology specifically relevant for this process step. In order to define the model philosophy, model elements of the type AIV Entity are created. Each model element can represent a product on any level of the system hierarchy. The represented product is modeled using the *Product* stereotype from the ESA MBSE solution implementation for SysML v1. The *represents* relation is modeled as a SysML Generalization. This means that the AIV Entity will inherit all properties of the represented Product model element by default. If necessary, certain properties can be redefined, deleted, or added to indicate the differences between for example an engineering model and a flight model.

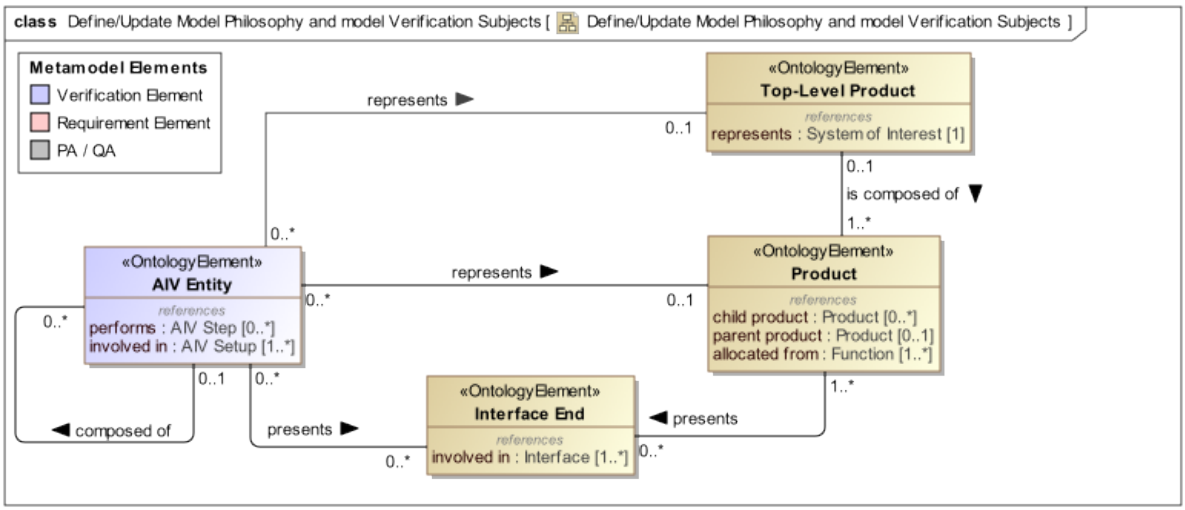
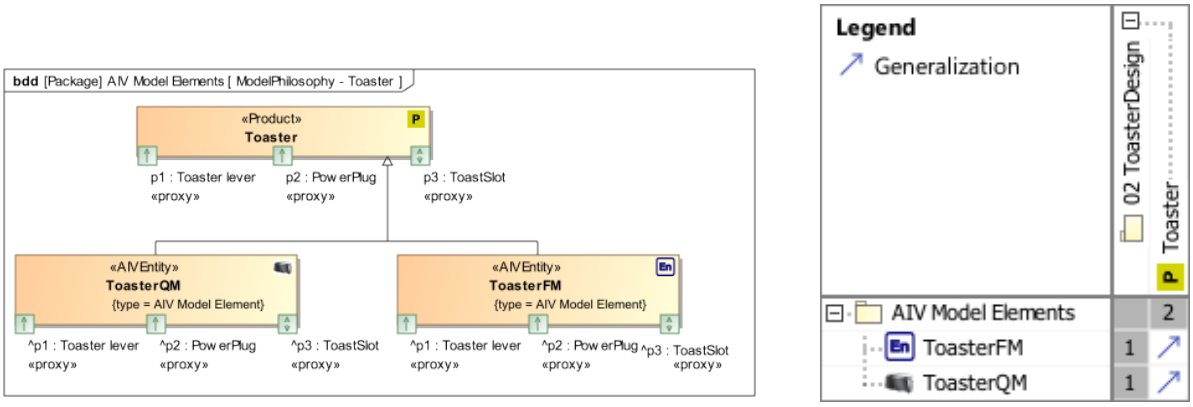


Figure 4.6: Ontology elements for step 1.1 Define or update model philosophy and verification subjects

Two different views are suggested to be used to perform the modeling steps. For creating a new AIV Entity and the SysML Generalization relations in the first place, a Block Definition Diagram can be used, as shown in Figure 4.7a. The Block Definition Diagram allows to refine the inherited properties if needed, for example adding testing interfaces and redefining other properties. If the project requires many AIV Entities, just using one Block Definition Diagram might get too crowded. To keep an overview over all AIV Entities and their references to the Product model elements, a Dependency Matrix is useful in addition, shown in Figure 4.7b. The examples show that the Product "Toaster" has two different model elements that will be used in its lifecycle, a "qualification model" and a "flight model". The terminology is adopted from the space domain to illustrate the underlying concept of distinct testing and operational product versions, and is used here in an abstracted sense for demonstration purposes.



(a) Block Definition Diagram

(b) Dependency Matrix for defining model philosophy

Figure 4.7: Example views for 1.1 Define or update model philosophy and verification subjects

1.2 Define verification control approach The approach to verification control is fundamentally centered around requirements. In the space domain, a common tool for verification control is the Verification Control Document (VCD), which is used to capture for each requirement information about how it is planned to be verified, the current status of compliance of the system to the requirement, and whether the requirement is considered open or closed, meaning that compliance is fully proven and accepted by all relevant stakeholders or not. This step focuses on creating a VCD-like model view. A full definition of verification control processes in a heavily regulated domain like space involves multiple organizational or even contractual agreements. Capturing such agreements is considered out of the scope of the thesis.

Figure 4.8 shows the ontology element Requirement and all parameters that might be populated in a Verification Control Document. Each requirement is uniquely identified by a Requirement ID and textually described with a Name and Requirement Text containing the formal "shall" statement. Additional context can be provided through a Rationale. A requirement might be derived from a captured User Need (stereotype from the ESA MBSE Solution). Requirements are typically organized within specification packages (stereotype from the ESA MBSE Solution). Hierarchical relationships between requirements are captured through parent and child relations between Requirements.

From a verification control perspective, each requirement is associated with one or more AIV Activities via a SysML Verify relation, indicating how the requirement is verified. The requirement may further be refined into one or more AIV Criteria through the refine relation. At this stage, the objective is not to fully establish all relations between requirement, AIV Activities, and criteria, but to define the structure of verification control. The VCD view can then be incrementally populated as the project progresses.

The outcome of verification is captured through the compliance status, indicating whether the requirement is compliant, non-compliant, partially compliant, unknown, or not applicable, and the compliance rationale, which provides justification for this assessment. In addition, the life cycle state of the requirement with respect to verification is represented by the closeout status, indicating whether verification activities related to the requirement are open or closed, and the corresponding closeout status rationale, documenting the reasoning behind this status. Properties with predefined possible values are typed by so-called enumerations, restricting model users to only input one specific value into the model. One example for such a property is the compliance status of a requirement. By representing verification control within the model, traceability between requirements, verification activities, and results can be maintained consistently and updated dynamically, overcoming limitations of static document-based approaches.

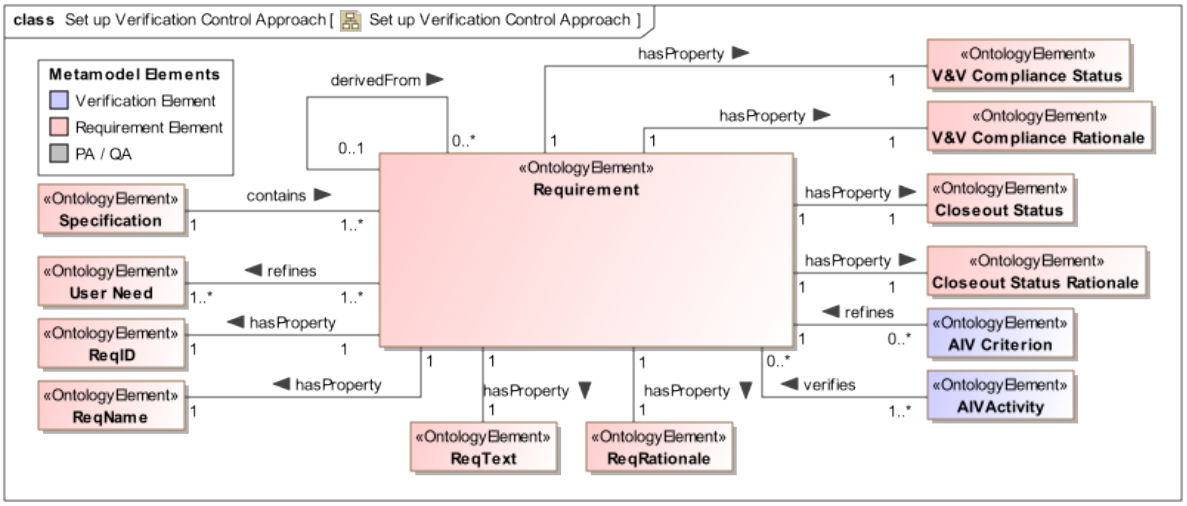


Figure 4.8: Ontology elements for step 1.2 Define verification control approach

An example VCD like view in the model is shown in Figure 4.9 for the toaster example, including 4 requirements which are derived from 2 User Needs, and already connected to AIV Activities and AIV Criteria. Another customization that was done for the example and subsequent examples that are presented in the thesis is that the compliance status is tracked per project phase, as indicated in the properties Phase C / D / E compliance. In a real life project, this VCD view will be much larger and might be partitioned into multiple views. Additional views might be helpful, for example a view that shows traceability between Requirements and User Needs as shown in Figure 4.10.

#	Id	Name	Text	Derived From	Verified By	VvStatements	Refined By	CritStatements	Phase C Complian...	Phase D Complian...	Phase E Complia...	Complia... Rationale	Closeout Status	Closeout Status Rationale
1	1	Toast Browning Level	The toaster shall produce a uniformly golden-brown toast surface within the levels 5 to 8 on the toastiness scale.	3 Golden-brown Toast	PID-1 ToasterQualificationRun(context ToasterSetup)	Test at System level in the Morning during Verification activity PID-1.	Crit-1 Toastiness of Toast within 5 to 8	Crit-1 evaluated as inconclusive	Not Applicable	Unknown	Unknown		OPEN	
2	2	Toasting Duration	The toaster shall complete the toasting process within at most 180 seconds for a standard slice of bread.	3 Golden-brown Toast	PID-1 ToasterQualificationRun(context ToasterSetup)	Test at System level in the Morning during Verification activity PID-1.	Crit-2 Toasting Duration	Crit-2 evaluated as inconclusive	Not Applicable	Unknown	Unknown		OPEN	
3	5	Toaster Mass	The toaster shall have a mass of less then 4 kg.	6 Toaster Weight	PID-2 WeighingToaste	Test at System level in the Morning during Verification activity PID-2.	Crit-4 ToasterWeightMeasurement	Crit-4 evaluated as inconclusive	Not Applicable	Unknown	Unknown		OPEN	
4	7	Toaster Mass Uncertainty	The final toaster mass shall be known with a maximum uncertainty of +/- 2 grams.	6 Toaster Weight	PID-2 WeighingToaste	Test at System level in the Morning during Verification activity PID-2.	Crit-5 ToasterWeighingUncertainty	Crit-5 evaluated as inconclusive	Not Applicable	Unknown	Unknown		OPEN	

Figure 4.9: VCD like view for step 1.2 Define verification control approach

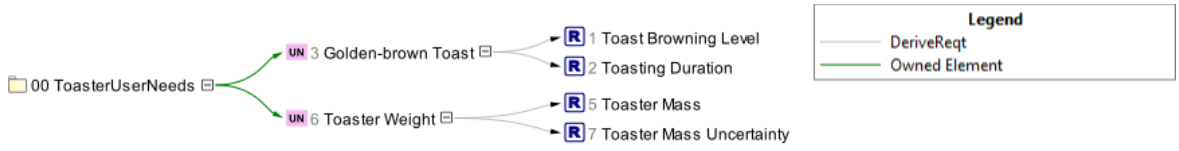


Figure 4.10: Additional views for requirements traceability

If requirements are initially captured outside the model, they can be imported into the modeling tool. The selected tool provides multiple mechanisms for this purpose, such as integration with spreadsheet-based sources or interfaces to requirements management tools (e.g. IBM DOORS). This allows to use existing requirement while also ensuring consistency between external repositories and the MBSE model.

1.3 Define AIV Activities To finish the rough AIV planning, AIV Activities are created in the model. At the rough AIV planning stage, each AIV Activity has to have a name and a unique identifier, as well as a set of properties that describe its key characteristics. These properties include level, stage, method, and activity kind.

The level property indicates the level of the system architecture at which the activity is performed. In a satellite context, this typically corresponds to levels such as component, subsystem, system, or mission level. It defines the scope of the activity.

The stage property represents the project phase during which the activity is conducted. This typically aligns with the project life cycle phases and provides temporal context.

The method property specifies how requirements are verified during an activity. Common values, as defined in ECSS standards, include test, analysis, inspection, and review of design. Each activity is associated with exactly one method, defining the primary means by which evidence is generated.

The activity kind property describes the general purpose of the activity within the AIV domain. Activities are distinguished between assembly, integration, verification, and validation.

To establish traceability to requirements, each AIV Activity must be linked to one or more Requirements via a SysML verify relation, indicating that the activity contributes to the verification of the requirement.

For organizational purposes, AIV Activities are organized in the model in an AIV Activity Specification, which is realized as a SysML Package.

These relations are shown in the extract of the ontology in Figure 4.11.

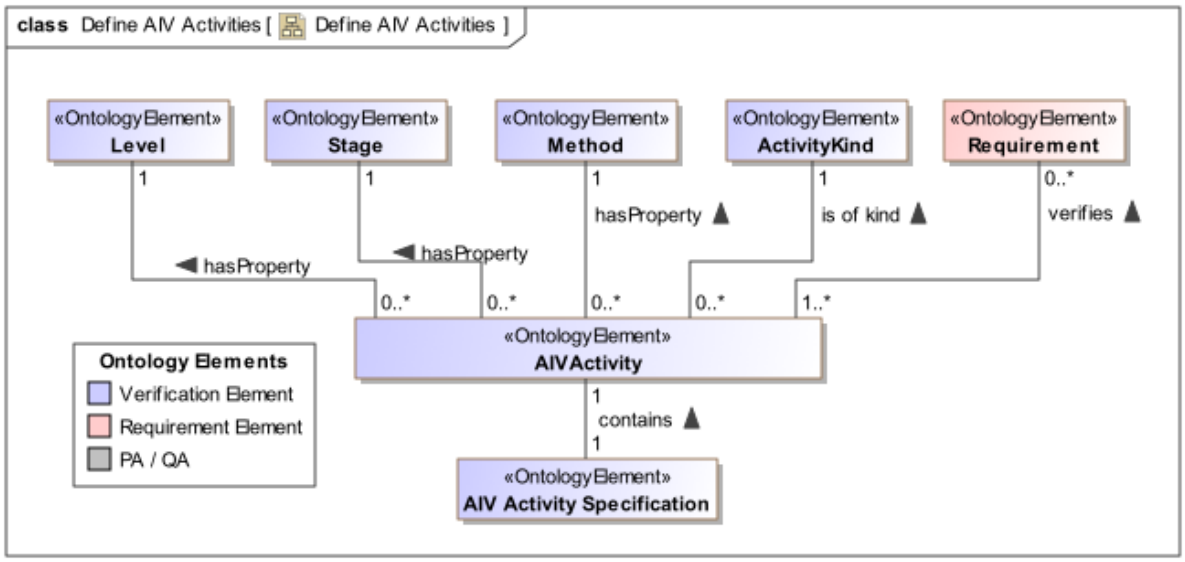


Figure 4.11: Ontology elements for step 1.3 Define AIV Activities

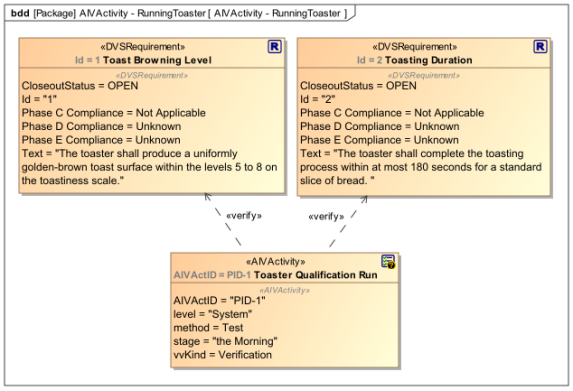
It is practical to work in a table view when creating multiple AIV Activities in the model at once. One example for such a table view is shown in Figure 4.12. In the example, the AIV Criteria are already populated, which is not necessarily the case at this stage in the process. Since the properties method and Vv kind are enumerations, they present drop-down menus for the model user. In addition to specifying the described properties, it is generally good practice to add a description of the activity in textual form. The modeling tool provides a documentation field for every model element for this purpose.

#	AIV Act ID	Name	Verifies	Related AIV Criteria	Level	Method	Stage	Vv Kind	Documentation
1		AIVActivity - RunningToaster							
2	PID-1	Toaster Qualification Run	1 Toast Browning Level 2 Toasting Duration	Crit-1 Toastiness of Toast within 5 to 8 Crit-2 Toasting Duration	System	Test	the Morning	Verification	The purpose of this AIV activity is to verify that the toaster under qualification meets its performance requirements with respect to toasting duration and achieved toastiness level. The procedure defines a controlled and repeatable method to operate the toaster, measure the toasting time, and assess the resulting toast quality using objective evidence. This activity contributes to the generation of verification evidence in the form of timing measurements, images, and evaluated toastiness levels, enabling an objective assessment of compliance with the defined AIV criteria. The structured execution of the procedure ensures consistency, traceability, and reproducibility of results within the verification campaign.
3		AIVActivity - WeighingToaster							
4	PID-2	Weighing Toaster	5 Toaster Mass 7 Toaster Mass Uncertainty	Crit-4 ToasterWeightMeasurement Crit-5 ToasterWeighingUncertainty	System	Test	the Morning	Verification	The purpose of the AIV Activity is to measure the mass of the toaster to confirm that it is within the allowed mass and that the mass is measured with sufficient accuracy.

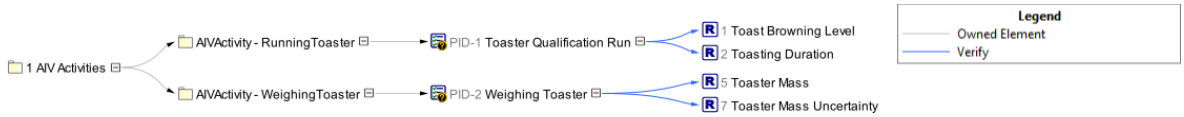
Figure 4.12: Table view to specify AIV Activities

Other model views can be used in order to define and review relations from AIV Activities to Requirements. For creating links quickly between two types of model elements, like creating verify links between AIV Activities and Requirements, a dependency matrix is a great tool. The dependency matrix for the toaster example is shown in Figure 4.13a. There are many other options to create these links more manually, one of them would be inside a dedicated Block Definition Diagram as shown in Figure 4.13b. Lastly, the traceability can be reviewed in a tree-like structure using a relation map, as shown in Figure 4.13c. Since this is a model-based way of working, it does not really matter that much, which view is used to create the elements or the links, as long as the underlying information is captured in the model according to the ontology.

(a) SysML Dependency Matrix



(b) SysML Block Definition Diagram



(c) SysML Relation Map

Figure 4.13: Model views to create and review traceability from AIV Activities to Requirements

2. Detailed AIV Activity Planning

2.1 Define AIV Criteria for Requirements In this step, AIV Criteria are defined to refine requirements into activity-specific, observable, or measurable conditions that can be evaluated during verification. Each criterion is associated with exactly one AIV Activity and translates the requirement into a form that is compatible with the selected verification method. The criteria define what needs to be checked and establish the basis for assigning verdicts. This ensures that requirements are not evaluated abstractly, but through clearly defined and assessable conditions in the context of the AIV Activity, that can be directly linked to generated evidence.

Figure 4.14 shows the ontology elements relevant for this step. An AIV Criterion is always defined for one specific AIV Activity. This is indicated by the involves relation in the ontology. This relation is implemented using a SysML trace relationship from the activity to the criterion. For the purpose of model organization, the AIV criteria for one AIV Activity are placed within the same package in the model containment tree. A criterion also refines a requirement, which is indicated by a refine relation between the criterion and the corresponding requirement. An AIV Criterion can have exactly one trace relation to an activity and exactly one refines relation to a requirement. To ensure this is the case, automatic MBSE validation rules can be defined that prevent linking the criterion to multiple of these elements. Another MBSE validation rule exists to ensure that the linked AIV Activity and the linked Requirement are themselves linked via a verifies relation. If a model user would create a situation where this is not the case, the model would generate an error message.

Next to these relations, the AIV Criterion is furthermore specified with a criterion statement and a verdict property, which is by default set to inconclusive. Once evidence is evaluated, the verdict of the criterion may be changed. Other properties of a criterion are its unique identifier, name, and optional notes providing details about the evaluation of the criterion.

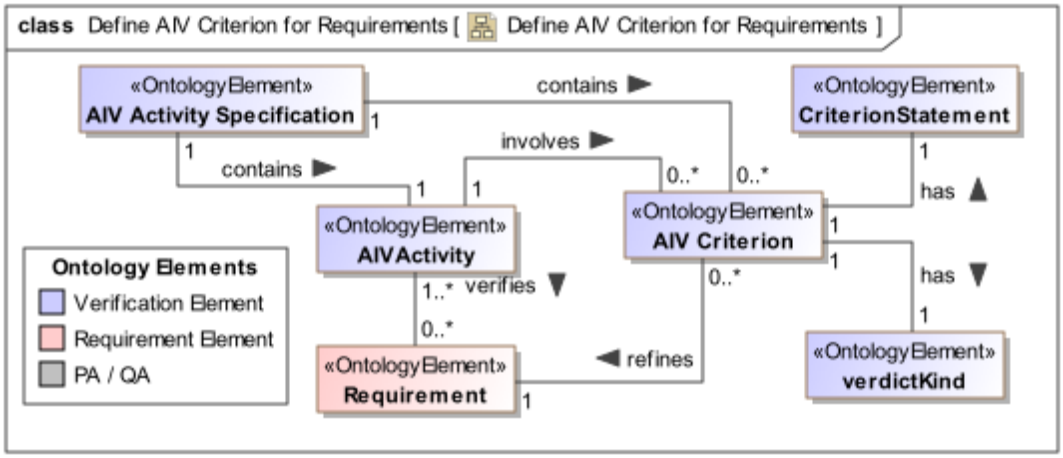


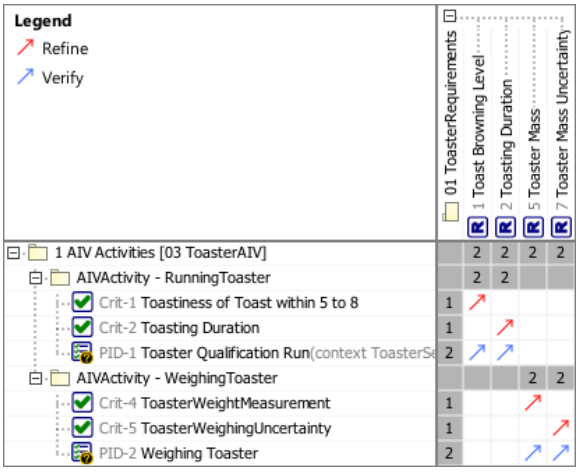
Figure 4.14: Ontology elements for step 2.1 Define AIV Criteria for Requirements

The properties of a criterion can be specified in a table view, like in Figure 4.15. Defining an AIV Activity fully will usually require to work on multiple process steps at the same time and if necessary to make adjustments to the table to include additional details. Adding a column to show related AIV Evidence is often helpful and will be shown in a later step.

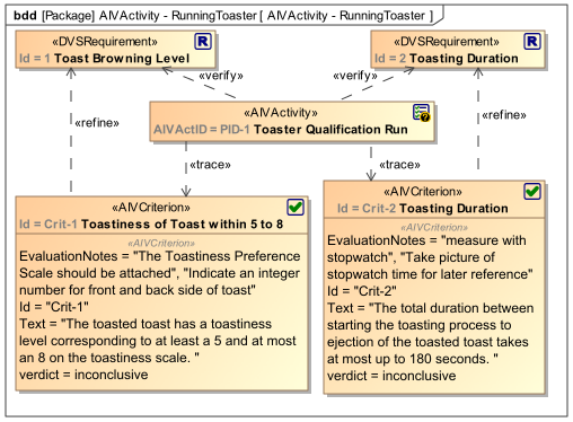
#	Traced From	Id	Name	Text	Evaluation Notes	Refines	Verdict	Verdict Rationale
1	PID-1 Toaster Qualification Run(context ToasterSetup)	Crit-1	Toastiness of Toast within 5 to 8	The toasted toast has a toastiness level corresponding to at least a 5 and at most an 8 on the toastiness scale.	The Toastiness Preference Scale should be attached Indicate an integer number for front and back side of toast	1 Toast Browning Level	inconclusive	
2	PID-1 Toaster Qualification Run(context ToasterSetup)	Crit-2	Toasting Duration	The total duration between starting the toasting process to ejection of the toasted toast takes at most up to 180 seconds.	measure with stopwatch Take picture of stopwatch time for later reference	2 Toasting Duration	inconclusive	

Figure 4.15: Table view for defining AIV Criteria for one AIV Activity

Similar to the examples in the previous steps, when modeling traceability between model elements, other views can be helpful to complement the table view. Figure 4.16a shows a dependency matrix that allows the model user to view, create, and delete the refine relations of criteria to requirements while also seeing the relations of the relevant AIV Activities. Figure 4.16b shows the same relations, but in a SysML Block Definition Diagram. Finally, the criteria can be included in the already existing relation map, indicating the digital thread from AIV Activities and Requirements to Criteria. This is shown in Figure 4.16c.



(a) SysML Dependency Matrix



(b) SysML Block Definition Diagram

(c) SysML Relation Map

Figure 4.16: Model views to create and review traceability from AIV Criteria to Activities and Requirements

2.2 Define Required AIV Evidence With the AIV Criteria being defined, the required AIV Evidence is specified to ensure that data is generated during activity execution to support evaluation. This includes identifying what types of data, measurements, observations, or artifacts need to be produced and how they relate to the criteria. The objective of this step is to establish a shared understanding of what verification evidence will be obtained and assessed, ensuring that all necessary information is planned for in advance and that criteria can be evaluated unambiguously.

Figure 4.17 shows the ontology elements that are relevant for this step, which is the AIV Evidence, the AIV Criterion, and the relation between them to indicate what evidence is needed to evaluate a criterion. In the model, this relationship is a SysML Trace relation. The model element AIV Evidence has a parameter to track the status of the modeled evidence. This parameter is implemented as an enumeration, with the possible values: planned, collected, and outdated.

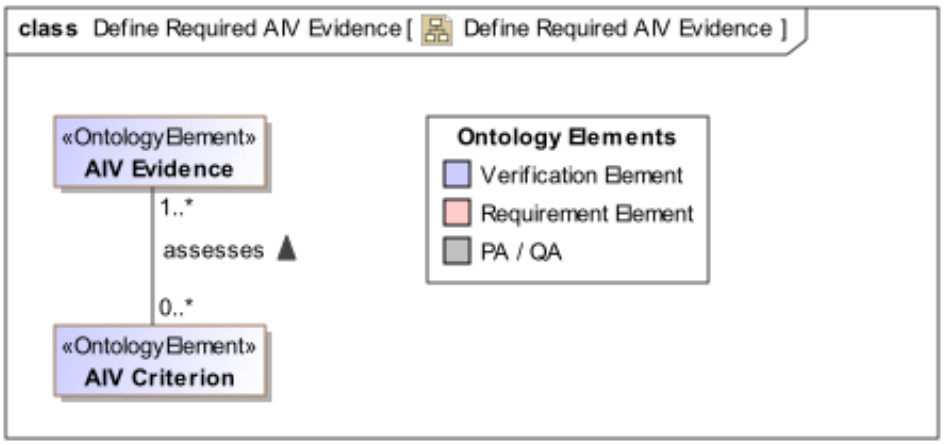


Figure 4.17: Ontology elements for step 2.2 Define Required AIV Evidence

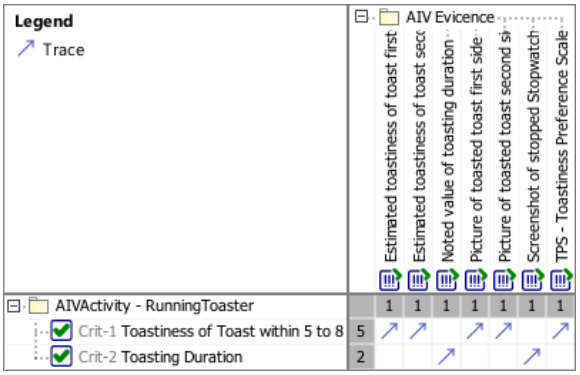
A lot of model views can be used to perform this step. The focus should be on keeping the overview

and creating a clear understanding of the link between all planned evidence and all criteria. Figure 4.18 shows the table that was already created to model AIV Criteria, now extended with the evidence. The traceability can also be viewed in a dependency matrix, like in Figure 4.19a, or in another dedicated table shown in Figure 4.19c.

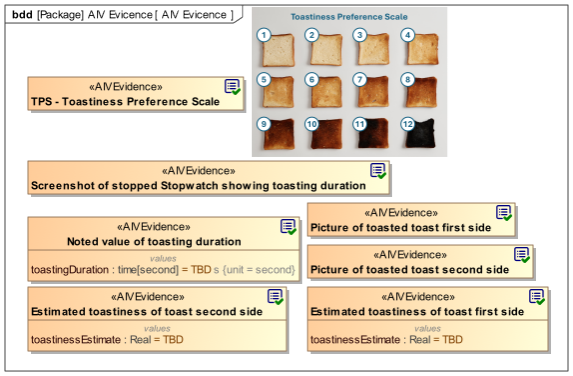
The example that is presented here aims to test the degree to which a bread is toasted. For this purpose, the toasted bread will be compared against a reference scale which can be seen in Figure 4.19b. This also illustrates that AIV Evidence does not necessarily only represent things that are generated during an AIV Activity, but can also be other resources that are needed to evaluate a criterion and to defend the verdict. The example AIV Activity to toast a slice of bread involves 2 criteria in total, one checking the toasting duration and the other one checking the degree to which the bread is toasted. For the latter criterion, a total of five AIV Evidence model elements are indicated. First the already referenced "Toastiness Preference Scale", which serves as a common scale to quantify the toastiness of each side of the toast. During the activity execution, the operator will compare the two sides and provide their own estimate according to the scale, and take a picture of each side as additional evidence. Such numerical inputs, like the number on the "Toastiness Preference Scale", can be captured directly in the model in value properties of the AIV Evidence model elements. External artefacts, like the reference scale, or more generally any documents pdf documents, can be attached to model elements directly. With a large number of attached files to the model, the application's performance might decrease, so another way would be to provide a link to a shared drive in the Documentation field. Finally, Figure 4.19d shows the digital thread for one AIV Activity extended to include the AIV Evidence model elements.

#	Id	Name	Text	Evaluation Notes	Refines	Verdict	Verdict Rationale	TracedTo
1	Crit-1	Toastiness of Toast within 5 to 8	The toasted toast has a toastiness level corresponding to at least a 5 and at most an 8 on the toastiness scale.	The Toastiness Preference Scale should be attached Indicate an integer number for front and back side of toast	1 Toast Browning Level	inconclusive		Picture of toasted toast first side Estimated toastiness of toast second side Picture of toasted toast second side TPS - Toastiness Preference Scale Estimated toastiness of toast first side
2	Crit-2	Toasting Duration	The total duration between starting the toasting process to ejection of the toasted toast takes at most up to 180 seconds.	measure with stopwatch Take picture of stopwatch time for later reference	2 Toasting Duration	inconclusive		Screenshot of stopped Stopwatch showing toasting duration Noted value of toasting duration

Figure 4.18: Table view of criteria extended with evidence column



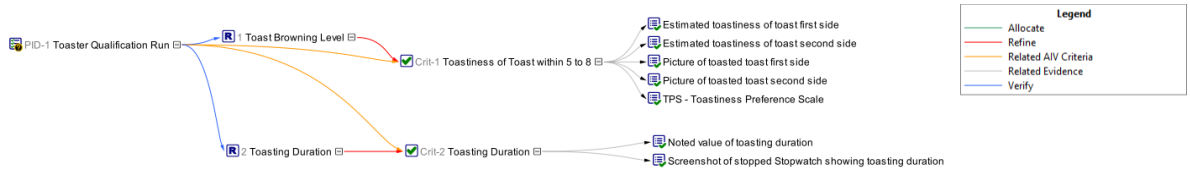
(a) SysML Dependency Matrix



(b) SysML Block Definition Diagram

#	Name	Status	Documentation	Owned Properties	Traced From
1	Estimated toastiness of toast first side	planned		status = planned toastinessEstimate : Real = TBD	Crit-1 Toastiness of Toast within 5 to 8
2	Estimated toastiness of toast second side	planned		status = planned toastinessEstimate : Real = TBD	Crit-1 Toastiness of Toast within 5 to 8
3	Picture of toasted toast first side	planned		status = planned	Crit-1 Toastiness of Toast within 5 to 8
4	Picture of toasted toast second side	planned		status = planned	Crit-1 Toastiness of Toast within 5 to 8
5	TPS - Toastiness Preference Scale	collected		toastiness_preference_scale.png status = collected	Crit-1 Toastiness of Toast within 5 to 8
6	Noted value of toasting duration	planned		status = planned toastingDuration : ISO80000-3 Space and Time::Quantities::time::time[second] = TBD s	Crit-2 Toasting Duration
7	Screenshot of stopped Stopwatch showing toasting duration	planned		status = planned	Crit-2 Toasting Duration

(c) SysML Generic Table showing the Evidence for all criteria involved in an AIV Activity



(d) SysML Relation Map

Figure 4.19: Model views to create and review traceability from AIV Criteria to Activities and Requirements

2.3 Define AIV Setup In this step, the AIV Setup is defined, describing the complete set of AIV Entities required to perform the activity, including the unit under test, tools, facilities, and roles. In addition to identifying these elements, their configuration and interfaces are modeled to ensure that all interactions necessary for execution are properly defined.

Figure 4.20 shows the ontology for defining a setup for an AIV Activity. This step may be skipped if the AIV Activity is not a test, and a setup is not applicable. To represent a test setup, the ontology element AIV setup is used. A setup includes or is composed of different AIV Entities, these can be facilities, tools, Ground Support Equipment (GSE), AIV Model Elements, or even roles that are played by people / stakeholders, like a test operator. The relation between an AIV Activity and a Setup is called an allocation.

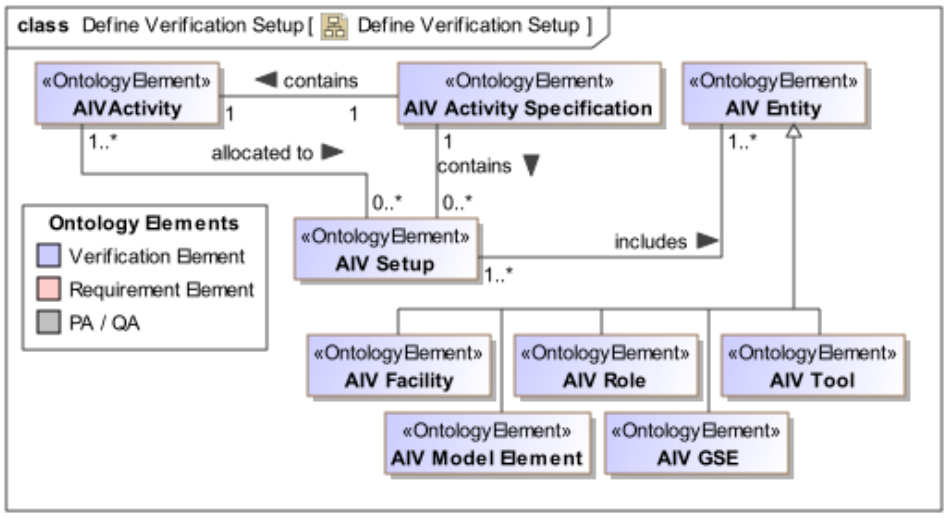


Figure 4.20: Ontology elements for step 2.3 Define AIV Setup

The main task in this process step is to model the structure of the AIV Setup. Figure 4.21 shows how such a setup is modeled in SysML, using the common way to model structure, using composition relationships. This creates SysML Part Properties owned by the model element Toaster Setup, which are typed by the AIV Entities. These part properties are shown in Figure 4.22, also showing their multiplicity, i.e. how many of these elements are included in the setup. For testing the toaster, the required setup includes a kitchen, a toaster (in this case the Toaster Qualification Model), one untoasted toast, wooden tweezers to handle toasted toast safely, a stopwatch to stop the time of the toasting process, and a camera to take pictures. The toaster owner is also part of the setup, as they will take on the role of operator of the toaster.

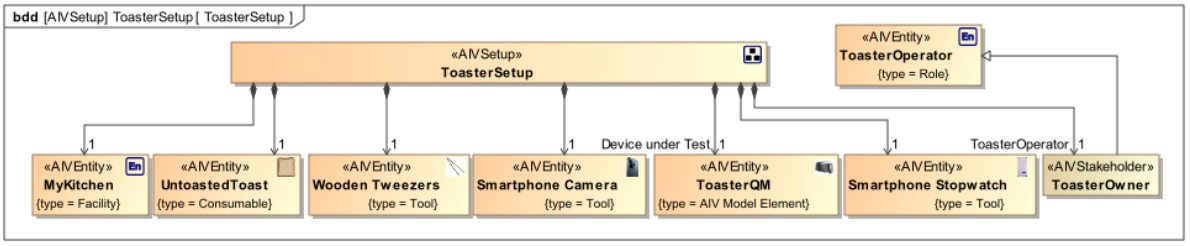


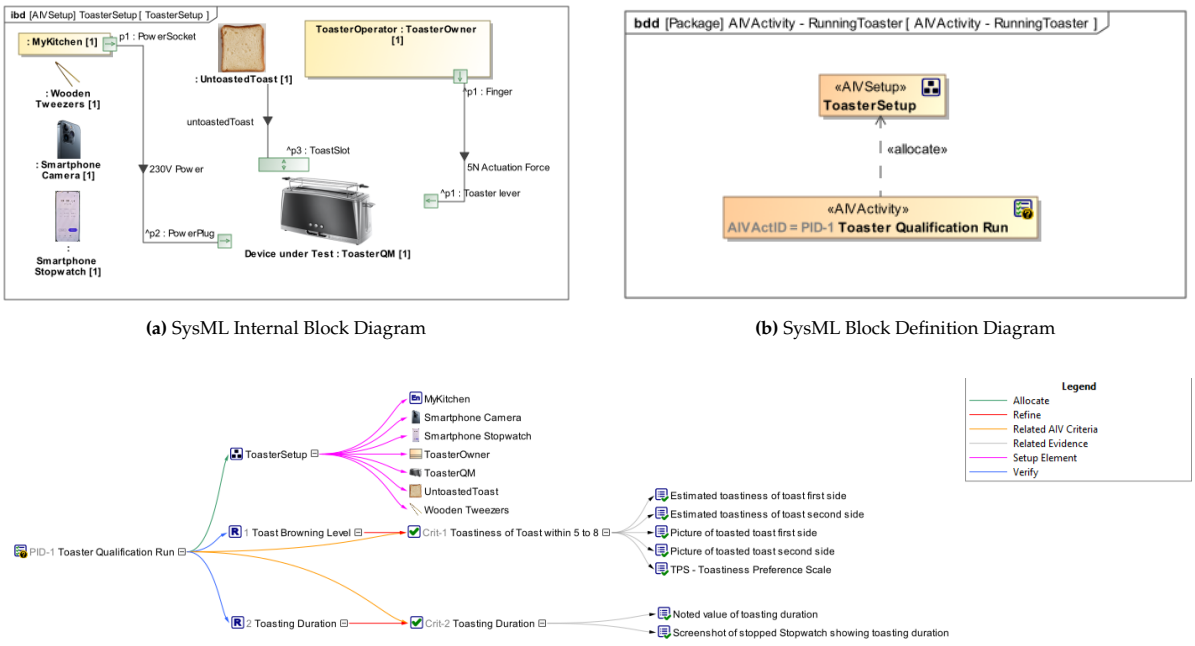
Figure 4.21: Block Definition Diagram to define AIV Setup

#	Name	Type	Multiplicity	AIVEntityType
1		MyKitchen	1	Facility
2	Device under Test	ToasterQM	1	AIV Model Element
3		UntoastedToast	1	Consumable
4		Wooden Tweezers	1	Tool
5		Smartphone Camera	1	Tool
6		Smartphone Stopwatch	1	Tool
7	ToasterOperator	ToasterOwner	1	Role

Figure 4.22: Block Definition Diagram to define AIV Setup

Once the structural breakdown of the setup is defined, the internal structure, including the interfaces

and connections, can be modeled. For the toaster example, this is shown in a SysML Internal Block Diagram in Figure 4.23a. The toaster's power plug has to be connected to the power socket of the kitchen, the untoasted toast needs to be able to go into the toaster's slot, and the toaster operator needs to be able to actuate the toaster lever. Custom images can optionally be used to give additional details to the model element, as shown in the example diagram. Next, the setup needs to be connected to an activity, using an allocate relation going from the activity to the setup, as shown in Figure 4.23b. Finally, the relation map showing the full digital thread is extended with the setup in Figure 4.23c.



(a) SysML Internal Block Diagram

(b) SysML Block Definition Diagram

(c) SysML Relation Map

Figure 4.23: Model views to define and review traceability of an AIV Setup

2.4 Define AIV Activity Inputs, Outputs, and Responsibilities This step focuses on defining the inputs required to perform the AIV Activity and the outputs that are produced as a result. In addition, responsibilities are assigned to relevant stakeholders, specifying who is responsible for providing required resources.

The ontology in Figure 4.24 shows that an AIV Activity has a set of concrete inputs and outputs, which are either typed by one AIV Entity or one AIV Evidence. Furthermore, the inputs and outputs are related to so called AIV Stakeholders, to indicate responsibility to provide them for the activity or handle the outputs of the activity. In SysML the inputs and outputs are modeled as parameters of the activity. Each parameter model element can have one type, which is either an AIV Entity or an AIV Evidence model element. Responsibility of a stakeholder is indicated by a SysML Trace relation.

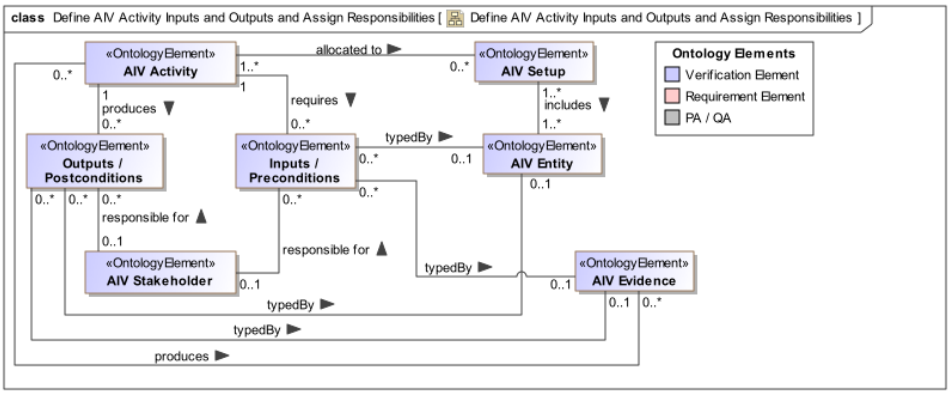
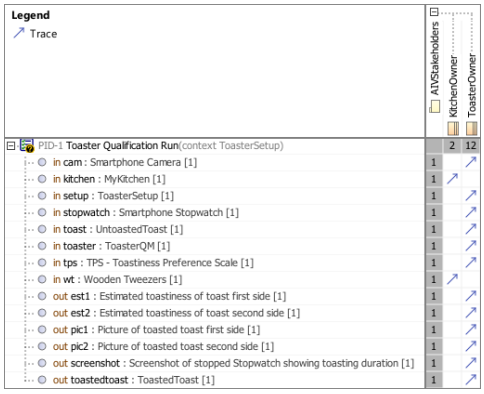


Figure 4.24: Ontology elements for step 2.4 Define AIV Activity Inputs, Outputs, and Responsibilities

Figure 4.25a shows all inputs and outputs modeled for the toaster test. The table also shows direction of the parameter (Input or Output), the multiplicity, and the responsible stakeholder. Figure 4.25b next to it shows a traceability matrix that can also be used to connect stakeholders to the activity parameters. The two stakeholders modeled in this example are the kitchen owner and the toaster owner. The owner of the kitchen is responsible for making the kitchen available and providing wooden tweezers, while the toaster owner is responsible for the rest of the inputs and outputs.

#	Name	Type	Direction	Multiplici...	Documentation	ResponsibleStakeholder
1	cam	Smartphone Camera	in	1		ToasterOwner
2	kitchen	MyKitchen	in	1		KitchenOwner
3	setup	ToasterSetup	in	1		ToasterOwner
4	stopwatch	Smartphone Stopwatch	in	1		ToasterOwner
5	toast	UntoastedToast	in	1		ToasterOwner
6	toaster	ToasterQM	in	1		ToasterOwner
7	tps	TPS - Toastiness Preference Scale	in	1		ToasterOwner
8	wt	Wooden Tweezers	in	1		KitchenOwner
9	est1	Estimated toastiness of toast first side	out	1		ToasterOwner
10	est2	Estimated toastiness of toast second side	out	1		ToasterOwner
11	pic1	Picture of toasted toast first side	out	1		ToasterOwner
12	pic2	Picture of toasted toast second side	out	1		ToasterOwner
13	screenshot	Screenshot of stopped Stopwatch showing toasting duration	out	1		ToasterOwner
14	toastedtoast	ToastedToast	out	1		ToasterOwner

(a) SysML Generic Table showing parameters of AIV Activity



(b) SysML Dependency Matrix

Figure 4.25: Model views to create AIV Activity inputs, outputs and responsibilities

2.5 Define parametric relations, tolerances, and technical budgets In this step, parametric relations and constraints are defined to support quantitative evaluation of AIV Criteria or to check that elements of the setup are suitable to perform the test. Figure 4.26 shows the corresponding ontology elements. An AIV Entity might have technical parameters, which may be constrained by AIV Criteria. These relations are implemented into the model using standard SysML elements, like value properties, connectors and constraint blocks.

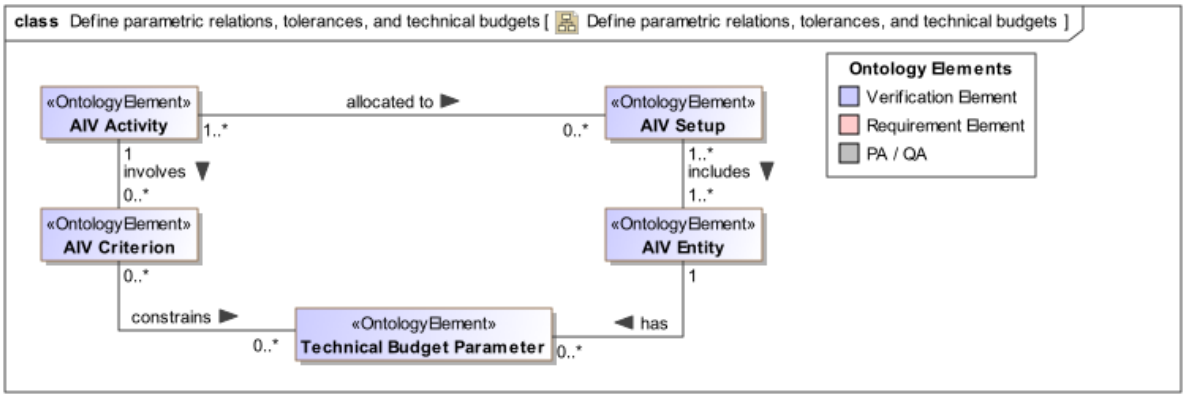
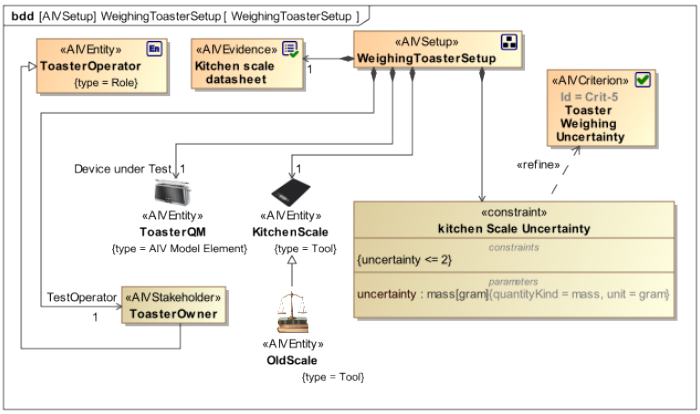


Figure 4.26: Ontology elements for step 2.5 Define parametric relations, tolerances, and technical budgets

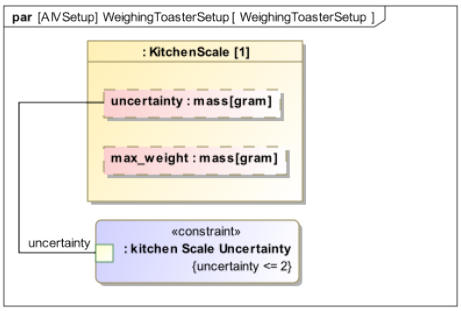
To enable parametric modeling in SysML, the AIV Setup is extended with a SysML Constraint Block. To illustrate the parametric modeling capabilities of the modeling tool, a different example AIV Activity is used. The goal of that activity is to measure the mass of the toaster. Two options are considered for the choice of what scale to use. One option is a typical kitchen scale, with a measurement uncertainty of 1 gram, and another option is an old fashioned scale with a measurement accuracy of 5 grams. The related requirements were presented in the VCD view in Figure 4.9. One of the requirements states that the mass shall be measured with an uncertainty of +/- 2 grams. Refining the requirement, it is then translated into an AIV Criterion for the scale uncertainty to be below or equal to 2 grams. The constraint block is used to translate the textual criterion statement into something that can be understood by the math engine of the modeling tool. In this case, it is a simple inequality, which could be easily evaluated without requiring parametric modeling, but evaluating measurement accuracies and error budgets of a certain test setup can become become very complex in practice.

Figure 4.27b shows the SysML parametric diagram that describes parametric relations within the test setup. The value property of the kitchen scale has to be connected to the single input parameter of the constraint element.

Once the parametric relations are defined, they can be evaluated. This is done in the model using an Instance Specification of the test setup. This instantiation mechanism is the same concept known from object oriented programming. Two different instances of the test setup are created, one using the kitchen scale and one using the old scale. The instances are automatically evaluated in the tool. The constraint evaluation results in a pass or a fail. This is shown in the first two rows of the instance table in Figure 4.27c. The tool colors the rows of the table according to the evaluation. As expected, if the normal scale is used, the constraint passes, and it fails with the old scale, because its uncertainty is too high. Additionally, the model user can create new instances and input custom values for the parameters to see their impact on constraint evaluation. This is shown in the last two rows of the table.



(a) Augmentations to AIV Setup for parametric analysis



(b) SysML Parametric Diagram

#	Name	Device under Test : ToasterQM	: KitchenScale	: kitchen Scale Uncertain...	:KitchenScale. max_weight : mass[gram]	:KitchenScale. uncertainty : mass[gram]
1	Kitchen Scale Setup	weighingToasterSetup.device under Test : 03 ToasterAIV::2 AIVEntities::AIV Model Elements:: ToasterQM	weighingToasterSetup.kitchenScale : 03 ToasterAIV::2 AIVEntities:: KitchenScale	pass	5000 g	1 g
2	Old Scale Setup	weighingToasterSetup1.device under Test : 03 ToasterAIV::2 AIVEntities::AIV Model Elements:: ToasterQM	weighingToasterSetup1.oldScale : 03 ToasterAIV::2 AIVEntities:: OldScale	fail	10000 g	5 g
3	Custom Scale Setup	weighingToasterSetup.device under Test1 : 03 ToasterAIV::2 AIVEntities::AIV Model Elements:: ToasterQM	weighingToasterSetup.kitchenScale1 : 03 ToasterAIV::2 AIVEntities:: KitchenScale	pass	1000 g	2 g
4	Custom Scale Setup1	weighingToasterSetup.device under Test2 : 03 ToasterAIV::2 AIVEntities::AIV Model Elements:: ToasterQM	weighingToasterSetup.kitchenScale2 : 03 ToasterAIV::2 AIVEntities:: KitchenScale	fail	1000 g	2.1 g

(c) SysML Instance Table

Figure 4.27: Model views to create and review traceability from AIV Criteria to Activities and Requirements

3. AIV Activity Execution and Postprocessing

3.1 Define AIV Activity Procedure In this step, a procedure is defined for each AIV Activity to describe how the activity is to be executed. The procedure captures the sequence of AIV Steps, including the actions to be performed, the involved AIV Entities, and the expected inputs and outputs. This provides a structured description of the activity execution. The procedure serves as a basis for performing the activity and supports traceability between planned actions, generated evidence, and the associated AIV Criteria. In addition to the already discussed ontology elements, the AIV Step is now introduced, as shown in Figure 4.28. AIV Steps are implemented as SysML Actions. Just like AIV Activities, they have inputs and outputs. Additionally, they can be related to an AIV Criterion and they are performed by an AIV Entity. In typical test procedures, the performing entity is often the test operator. The term procedure is used to refer to a view that presents multiple AIV Steps of one AIV Activity.

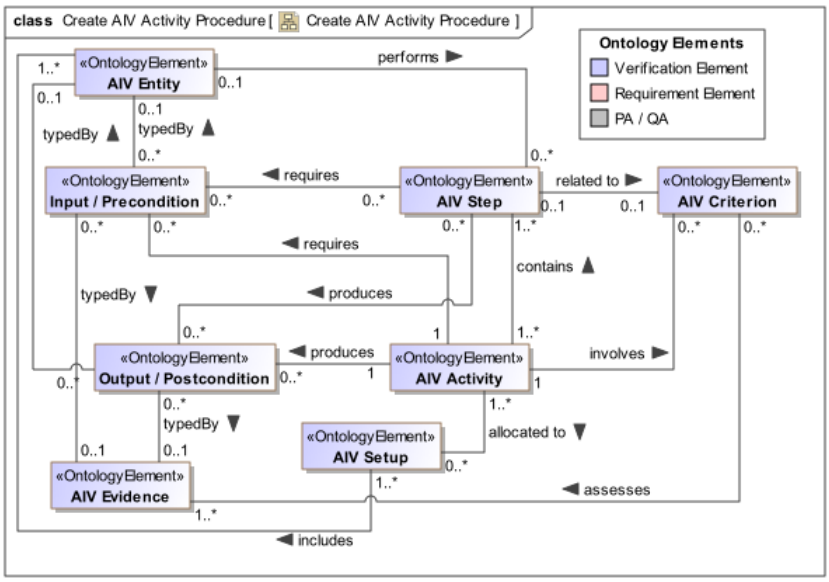


Figure 4.28: Ontology elements for step 3.1 Define AIV Activity Procedure

The previously modeled inputs and outputs of the AIV Activity and the structure of the AIV Setup can be leveraged when modeling the procedure for the AIV Activity. The model view shown in Figure 4.29 can be automatically generated and subsequently populated to define the step-by-step procedure. This already shows all inputs and outputs that need to be connected to different AIV Steps and helps to create a procedure that makes use of all defined inputs and provides all required outputs. Creating AIV Steps within the so-called swimlanes allocates them to the part properties of the AIV setup. In this procedure, all steps will be performed by the toaster operator, while some of the modeled behavior of the toaster model element is reused. A part of the modeled procedure is shown in Figure 4.30 and the full diagram is attached in the Annex. Next to the complete version of the diagram, the annex also contains a table view of the steps in order and a pdf document generated directly from the model, using the document generation capabilities of the tool. In addition to this, if the AIV Steps are modeled accurately, the procedure can be simulated using the tools simulation capabilities. The simulation can confirm that all input and output relations have been modeled correctly and that each step is performed in the correct sequence.

Figure 4.31 shows reveals the relations between one modeled AIV Steps and other model elements. The example step is called "determine toastiness level of first side of toast on the Toastiness Reference Scale and take picture". This step is related to two AIV Evidence elements, one Entity representing the Camera that is used to take the picture, to the Toastiness Reference Scale and to the relevant criterion Crit-1. Once the procedure is fully modeled and the procedure document is generated, reviewed, and approved, the actual test is executed and the required evidence generated.

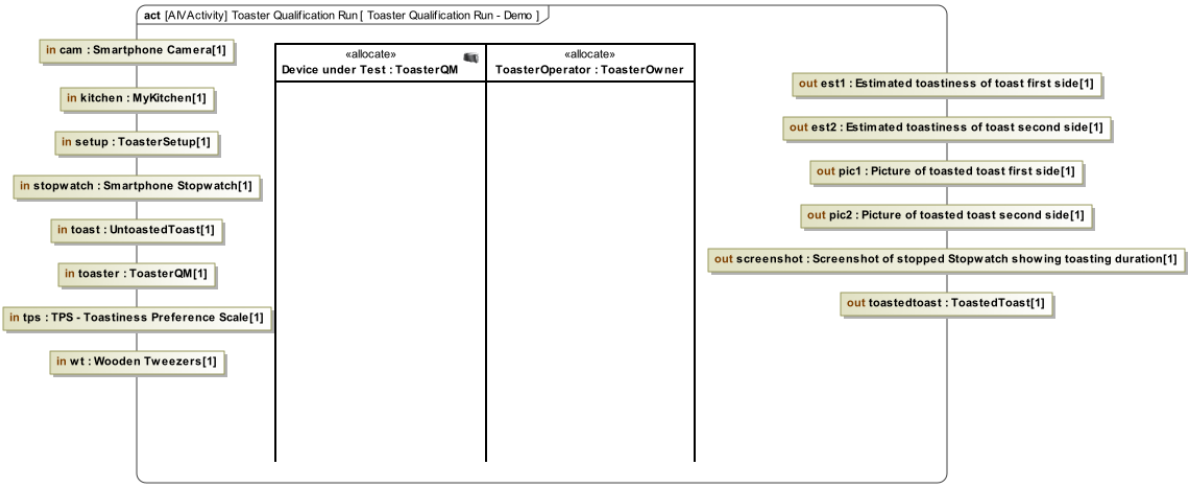


Figure 4.29: Model view automatically generated to start modeling the procedure.

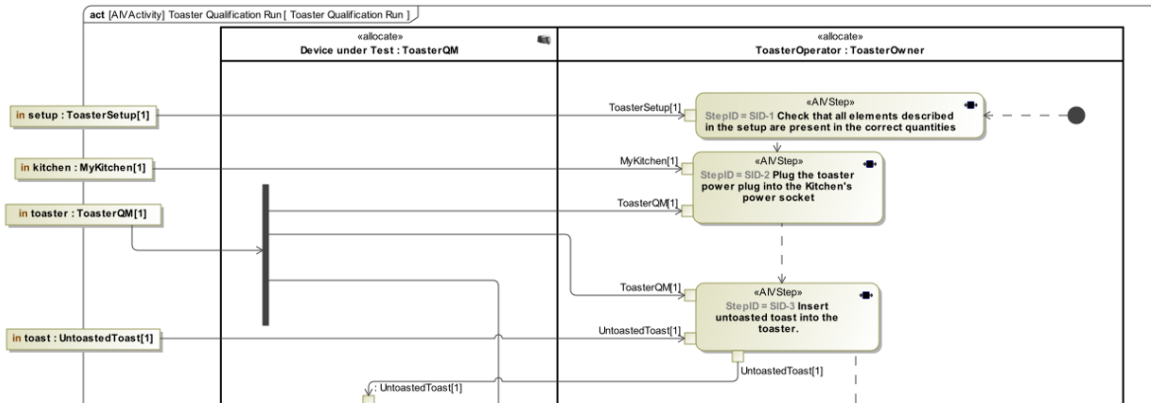


Figure 4.30: Ontology elements for step 3.1 Define AIV Activity Procedure

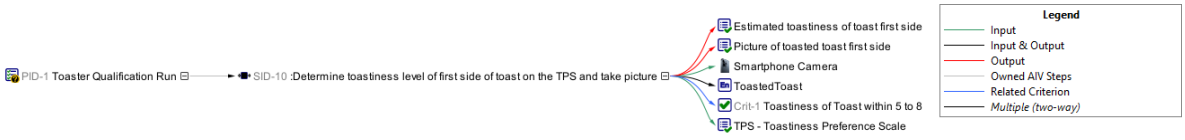


Figure 4.31: Ontology elements for step 3.1 Define AIV Activity Procedure

3.2 Perform AIV Activity and Collect Evidence Based on the defined procedure, the AIV Activity is executed. During execution, the specified AIV Steps are carried out, and AIV Evidence is generated in the form of measurements, observations, or documented artifacts. The next step after execution is to postprocess the results and update the model accordingly. This is described in the following paragraphs.

3.3 Postprocess AIV-related Information After execution, the collected AIV Evidence is postprocessed which includes capturing properties of the collected evidence formally in the model, evaluating the defined AIV Criteria and defining their verdict, and updating the VCD. No new ontology elements are required for this step.

The proposed way of working is to first capture the evidence in the model and then evaluate the criteria based on the as-run procedure document. As a last step, the Verification Control Document view can be updated. The following three tables show this progression for the toaster example, all tables are now extended by color coding to highlight the status of the displayed elements. Figure 4.32 shows all

evidence that is involved in the evaluation of the two criteria. The color coding indicates the status of the evidence. In this example, all evidence that was planned to be collected was actually collected. The table also shows that pictures and the as-run procedure have been captured in the model for later reference. This is important to preserve full traceability from the raw evidence to the criteria and finally requirements close out.

Figure 4.33 shows the two criteria, and the updated values for their verdict properties. A verdict rationale is added to each of the criteria to explain the decision to evaluate the verdicts as "pass".

Finally, the VCD view is updated, as shown in Figure 4.34. The two requirements that were supposed to be verified during the toaster run example activity can now be set to be compliant. This can be reasoned, since all criteria for each requirement are evaluated as a pass. The Phase D Compliance parameters for the two requirements are changed accordingly and a rationale is provided.

#	Name	Status	Documentati...	Owned Properties	Traced From
1	Estimated toastiness of toast first side	collected		status = collected toastinessEstimate : Real = 6	Crit-1 Toastiness of Toast within 5 to 8
2	Estimated toastiness of toast second side	collected		status = collected toastinessEstimate : Real = 6	Crit-1 Toastiness of Toast within 5 to 8
3	Picture of toasted toast first side	collected		toast_1st_side.jpg status = collected	Crit-1 Toastiness of Toast within 5 to 8
4	Picture of toasted toast second side	collected		toast_2nd_side.jpg status = collected	Crit-1 Toastiness of Toast within 5 to 8
5	TPS - Toastiness Preference Scale	collected		toastiness_preference_scale.png status = collected	Crit-1 Toastiness of Toast within 5 to 8
6	As-run procedure of toaster test	collected		As-run_toastingProcedure_10-03-2026.pdf status = collected	Crit-1 Toastiness of Toast within 5 to 8 Crit-2 Toasting Duration
7	Noted value of toasting duration	collected		status = collected toastingDuration : ISO80000-3 Space and Time::Quantities::time::time[second] = 132 s	Crit-2 Toasting Duration
8	Screenshot of stopped Stopwatch showing toasting duration	collected		toasting_duration_screenshot.png status = collected	Crit-2 Toasting Duration

Figure 4.32: Table showing post processed AIV Evidence model elements

#	Id	Name	Text	Evaluation Notes	Refines	relatedAIVSteps	Verdict	Verdict Rationale	TracedTo
1	Crit-1	Toastiness of Toast within 5 to 8	The toasted toast has a toastiness level corresponding to at least a 5 and at most an 8 on the toastiness scale.	The Toastiness Preference Scale should be attached Indicate an integer number for front and back side of toast	1 Toast Browning Level	SID-10 SID-12	pass	Pictures taken and notes during procedure indicate both sides of bread as 6/12 on TPS.	Picture of toasted toast first side Estimated toastiness of toast second side Picture of toasted toast second side TPS - Toastiness Preference Scale Estimated toastiness of toast first side
2	Crit-2	Toasting Duration	The total duration between starting the toasting process to ejection of the toasted toast takes at most up to 180 seconds.	measure with stopwatch Take picture of stopwatch time for later reference	2 Toasting Duration	SID-8	pass	Measured duration of 132 seconds, picture taken.	Screenshot of stopped Stopwatch showing toasting duration Noted value of toasting duration

Figure 4.33: Table showing post processed AIV Criterion model elements

#	Id	Name	Text	Derived From	Verified By	VvStatements	Refined By	CritStatements	Phase C Compliance	Phase D Compliance	Phase E Compliance	Compliance Rationale	Closeout Status	Closeout Status Rationale
1	1	Toast Browning Level	The toaster shall produce a uniformly golden-brown toast surface within the levels 5 to 8 on the toastiness scale.	3 Golden-brown Toast	PID-1 Toaster Qualification Run(context ToasterSetup)	Test at System level in the Morning during Verification activity PID-1.	Crit-1 Toastiness of Toast within 5 to 8	Crit-1 evaluated as pass	Not Applicable	Compliant	Not Applicable	Phase C and E not applicable Crit-1 passed.	OPEN	Pending formal review
2	2	Toasting Duration	The toaster shall complete the toasting process within at most 180 seconds for a standard slice of bread.	3 Golden-brown Toast	PID-1 Toaster Qualification Run(context ToasterSetup)	Test at System level in the Morning during Verification activity PID-1.	Crit-2 Toasting Duration	Crit-2 evaluated as pass	Not Applicable	Compliant	Not Applicable	Phase C and E not applicable Crit-2 passed.	CLOSED	Closed during formal review
3	5	Toaster Mass	The toaster shall have a mass of less then 4 kg.	6 Toaster Weight	PID-2 Weighing Toaster	Inspection at System level in the Morning during Verification activity PID-2.	Crit-4 Toaster Weight Measurement	Crit-4 evaluated as TBD	Not Applicable	Unknown	Not Applicable	Phase C and E not applicable Crit-4 yet evaluated.	OPEN	Pending formal review
4	7	Toaster Mass Uncertainty	The final toaster mass shall be known with a maximum uncertainty of +/- 2 grams.	6 Toaster Weight	PID-2 Weighing Toaster	Inspection at System level in the Morning during Verification activity PID-2.	Crit-5 Toaster Weighing Uncertainty	Crit-5 evaluated as TBD	Not Applicable	Unknown	Not Applicable	Phase C and E not applicable Crit-5 not yet evaluated.	OPEN	Pending formal review

Figure 4.34: Updated VCD after post processing results from AIV Activity execution

4. AIV Process Control

The ontology elements for the following two steps are shown combined in Figure 4.35. AIV Activities are included in / grouped in AIV Events. The events are included in an overall AIV Flow. An AIV Flow can be defined for a variety of scopes, for example to represent the sequence of AIV Events for a subsystem, for an overall system, or for a qualification model of a system. This is implemented with the typical behavior modeling approach using SysML Activities. The main objective is to indicate dependencies between activities and events, by modeling object and control flows between the model elements.

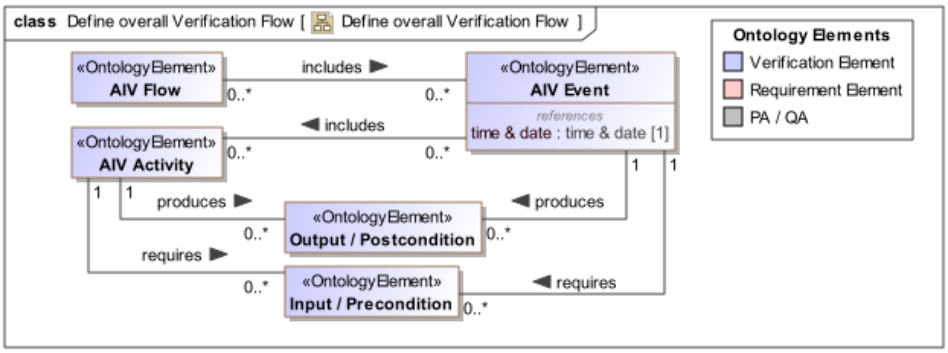
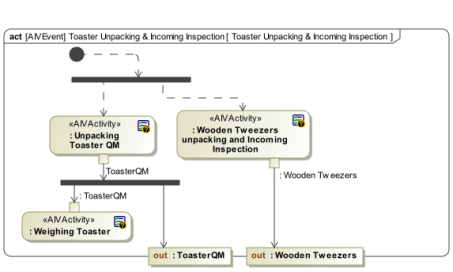
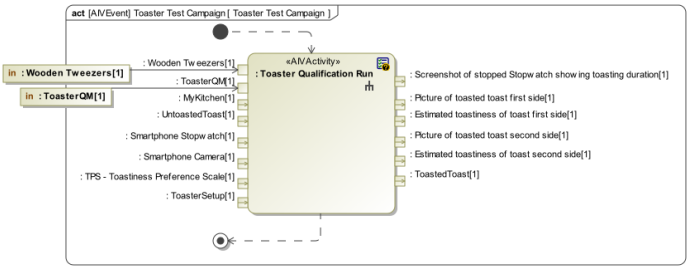


Figure 4.35: Ontology for steps 4.1 Group AIV Activities into AIV Events and 4.2 Define Overall AIV Flow

4.1 Group AIV Activities into AIV Events In this step, individual AIV Activities are grouped into AIV Events, representing planned occasions during which one or more activities are performed. Examples of such events include test campaigns or integration phases. Grouping activities into events enables coordinated planning of resources such as facilities, equipment, and personnel, and provides a structured context for execution.



(a) AIV Event for unpacking and inspection of elements



(b) AIV Event for the toaster qualification campaign

Figure 4.36: Diagrams showing AIV Events

4.2 Define Overall AIV Flow Based on the AIV Events, the AIV Flow is established, capturing the sequence and dependencies between events. It can provide visibility into how AIV progresses across different system levels and project phases, linking detailed activity execution to the overall project timeline. It should be noted that the MBSE modeling software is not a project management software and should not be used as such. Modeling the AIV Flow and AIV Events could serve as an interface to project managers, but the extra value of modeling in this step must be weighted per project against the expected benefits.

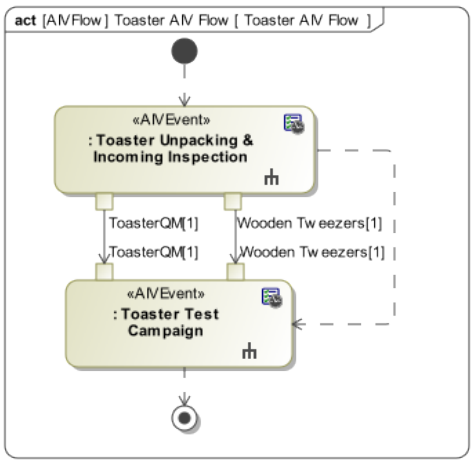


Figure 4.37: AIV Flow of AIV Events for the toaster example

5. AIV Quality Control

5.1 Set Up Non-Conformance Control Approach In this step, the overall approach for handling Non-Conformances (NC) is defined. This includes establishing the processes, roles, and responsibilities for identifying, documenting, assessing, and resolving Non-Conformances throughout their lifecycle. For this thesis, the ontology element for a Non-Conformance is shown in Figure 4.38.

Each Non-Conformance is characterized by a set of parameters that enable consistent documentation and traceability. These include a unique NCR-ID and NCR Title to identify the non-conformance, as well as the classification, which indicates its severity. The date of detection and the NCR initiator capture when and by whom the non-conformance was identified. The NCR cause provides the identified or suspected root cause, while the disposition defines the chosen or proposed resolution strategy. The NCR status reflects the current state within the non-conformance lifecycle, and the NCR approver represents the responsible authority for acceptance and closure. Finally, the corrective actions field documents the measures taken to resolve the non-conformance and prevent recurrence.

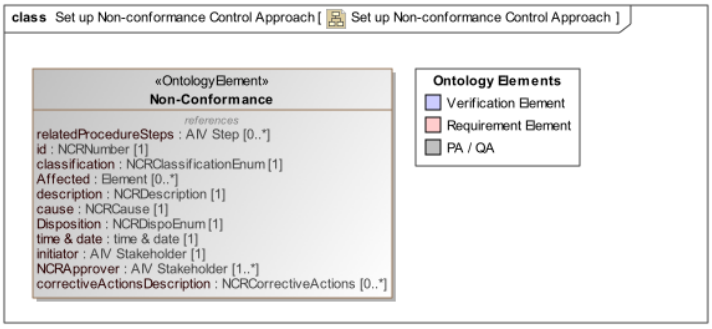


Figure 4.38: Ontology for step 5.1 Set Up Non-Conformance Control Approach, 5.2 Capture Non-Conformances, and 5.3 Prepare Review of Non-Conformances

In the modeling tool, this requires to define the model views that will be created for creating and tracking Non-Conformances throughout the AIV process. An example for such a table is shown in Figure 4.39. All properties of the Non-Conformance model element need to be defined in order to fully specify the Non-Conformance.

5.2 Capture Non-Conformances When deviations from expected results are observed during the evaluation of AIV Criteria, NCs are captured in the model. Each NC is uniquely identified and contains a description of the issue, its classification (e.g. major or minor), and a timestamp. The NCR is initiated by a responsible stakeholder and is linked to all affected elements, including requirements, AIV Activities,

AIV Entities, and, where applicable, AIV Steps. This ensures full traceability of the non-conformance and its impact on the system and verification process.

#	NCR-ID	NCR Title	Classification	Disposition	Date Of Detection	Body	Documentation	NCR Cause	NCR Initiator	Corrective Actions Description	NCR Approver	NCR Status	Annotated Element
1	NC-1	Toast burned during normal operation	minor	rework	3/19/26	One side of the toast scored a 9 on the Toastiness Preference Scale	During normal toaster operation on 19/03/2026, a slice of bread was found to be excessively browned, with one side scoring a 9 on the Toastiness Preference Scale …	Toaster heat setting too high for selected bread type	Nicolas Oidtmann	Adjust toaster setting to lower heat level Define standard toasting time per bread Redo the test to check toaster settings	Toaster Owner	In-progress	SID-10::Determine toastiness level of first side of toast on the TPS and take picture PID-1 Toaster Qualification Run(context ToasterSetup) Crit-1 Toastiness of Toast within 5 to 8 As-run procedure of toaster test Estimated toastiness of toast second side Picture of toasted toast second side TPS - Toastiness Preference Scale

Figure 4.39: NCR overview table in the model

In addition to capturing the parameters of the NC, it is also linked to other relevant model elements as shown in Figure 4.40. The annotated elements need to include the AIV Activity during which the NC was detected, the affected Criteria and Requirements, any AIV Evidence that is relevant for assessing the NCR, and the AIV Step that discovered the Non-Conformance.

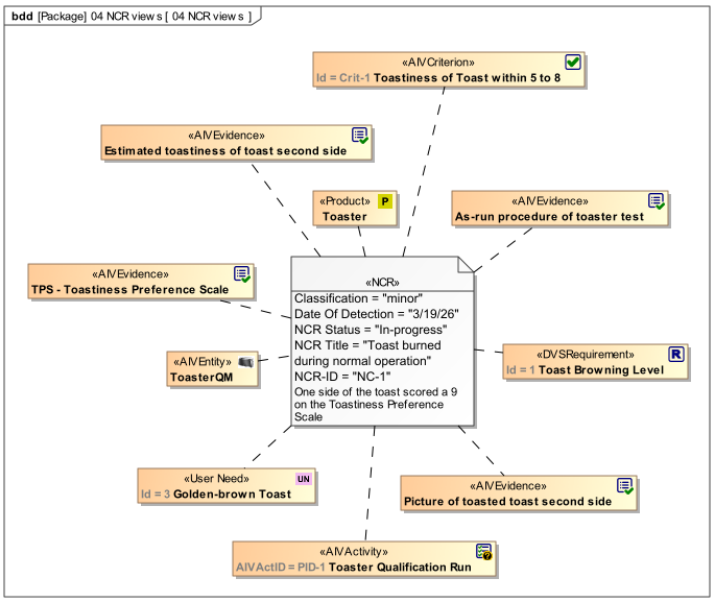


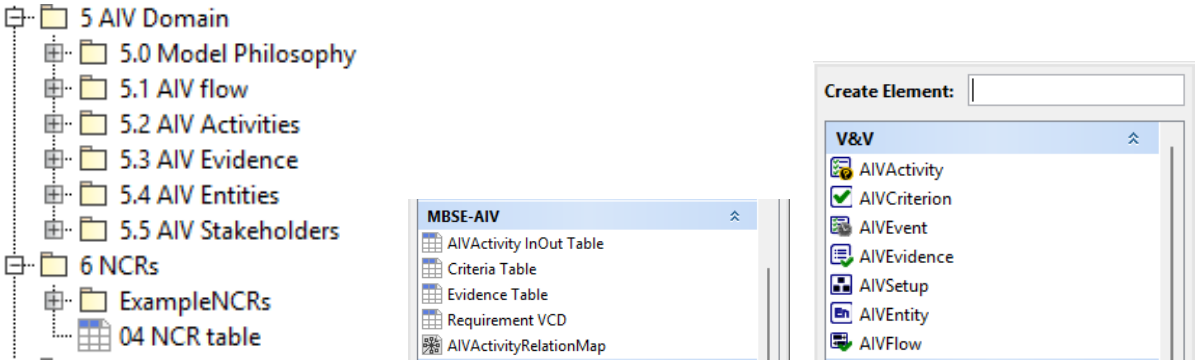
Figure 4.40: BDD showing all annotated elements of the example NC in the toaster model

5.3 Prepare Review of Non-Conformances In this step, the captured non-conformances are consolidated into structured review packages to support assessment and decision-making. Each review includes relevant information such as the description of the non-conformance, its cause, associated evidence, and proposed corrective actions. The review process involves designated stakeholders, who evaluate the non-conformance and approve its resolution and closeout. This might be a process involving external stakeholders, and requiring traditional documentation in the form of PDFs or similar. Information of the Non-Conformance can be exported and formatted into a generated document that can serve as a basis for such a formal process, though this is not shown for the discussed examples.

Supporting Model Organization and Framework

The model should be organized in a way that keeps the different elements organized and structured. Additionally, the model is customized to allow the user to only create the correct model elements within the AIV model. The quick-select menu of model elements is shown in Figure 4.41c. This tailoring of the tool guides the user towards using the correct model elements. The list of selectable elements is further customized depending on the parent element in the containment tree. The same concept is applied to the most common modeling views. The options for these are shown in Figure 4.41b. The views were already discussed in the previous sections. Making these customized views directly available in the tool saves time for the modeler. Lastly, the containment tree structure is an important practical

consideration, as it structures the MBSE model. The high-level structure that supports all modeling efforts presented here is shown in Figure 4.41a. A detailed view of the containment tree structure can be seen in Figure H.1. The model organization and the tool customizations support the modeler to follow the defined process more easily.



(a) Containment tree structure for AIV modeling

(b) Tool customization to quickly create model views

(c) Tool customization to quickly create model elements

Figure 4.41: Tool support for creating and organizing AIV model content

4.3. Chapter Summary

This chapter presents the development of the MBSE AIV methodology. It starts by deriving stakeholder needs from the literature review and from the expected responsibilities of key AIV stakeholders, including Systems and AIV Engineers, subsystem engineers, test operators, project managers, product assurance engineers, and customer or review authorities. These needs are grouped according to AIV planning, AIV execution, verification control and closeout, and traceability. Based on these needs, the chapter defines use cases and methodology requirements that establish what the MBSE AIV methodology must support.

The methodology is then designed according to the main constituents of an MBSE methodology: ontology, implementation in a modeling language, process, modeling methods, framework, and tool. The ontology defines the central AIV concepts used throughout the methodology, including Requirements, AIV Activities, AIV Criteria, AIV Evidence, AIV Steps, AIV Entities, AIV Setups, AIV Events, AIV Flows, and Non-Conformances. These concepts capture the digital thread from requirements to verification activities, criteria, evidence, verdicts, and requirement close-out status. The ontology is implemented in SysML v1 using stereotypes, while tool customizations provide additional assistance to the modeler.

The chapter further defines a process for applying the methodology. This process consists of tailoring the methodology, rough AIV planning, detailed AIV activity planning, AIV activity execution and postprocessing, AIV process control, and AIV quality control. Rough AIV planning establishes the model philosophy, verification control approach, and initial AIV activities. Detailed planning refines these activities into criteria, evidence, setups, inputs, outputs, responsibilities, and parametric relations. Execution and postprocessing support procedure definition, evidence collection, criterion evaluation, and verification status updates. Process control organizes activities into AIV events and flows, while quality control captures and tracks non-conformances.

A tool and language trade-off is performed to justify the implementation environment. Cameo/Magic Systems of Systems Architect with SysML v1 is selected because it provides strong support for customization, AIV-related modeling concepts, traceability, report generation, and industrial relevance. Finally, the chapter presents the modeling framework and methods using a simplified toaster example. The example demonstrates how the methodology is applied by running through the complete process.

5

Verification and Validation

This section presents the verification and validation of the MBSE AIV Methodology performed in the frame of the thesis. The section starts with providing an overall overview of the approach. Then, the verification is presented, followed by the validation of the methodology.

5.1. Approach to Verification and Validation

Verification and validation of an MBSE methodology follow the same underlying motivation as performing verification and validation for products. Verification is concerned with the question of *whether the methodology is built right*, while validation addresses whether *the right methodology is built*.

In the context of this thesis, verification focuses on assessing the internal correctness, consistency, and completeness of the developed MBSE AIV methodology with respect to its intended design and reference frameworks.

The verification approach is structured along four complementary perspectives. First, the internal consistency of the methodology is evaluated by analysing the coherence between the defined ontology, i.e. the conceptual representation of the AIV domain, and its implementation in SysML v1 using stereotypes. This includes verifying that all ontology elements and relationships are consistently reflected in the modelling environment.

Second, the methodology is evaluated against the previously defined requirements, which were derived from stakeholder needs and relevant standards. For each requirement, an assessment is made whether it is fulfilled, partially fulfilled, or not fulfilled, supported by a qualitative justification and evidence from the developed models.

Third, a compliance mapping against ISO/IEC/IEEE 24641 is performed, focusing on the process "Perform system verification and validation". This mapping assesses to what extent the methodology supports the preparation, execution, and result management of model-based verification and validation activities as defined in the standard. While this process-oriented reference framework enables assessment of modelling processes and methods, it does not explicitly address compliance with space industry-specific deliverables.

Therefore, as a fourth perspective, compliance with ECSS standards and their associated DRDs is evaluated. While ISO 24641 defines the processes for performing model-based verification and validation, ECSS standards define the required deliverables and their content. The presented MBSE AIV methodology aims to bridge both aspects by enabling process execution as well as the structured generation of ECSS-compliant documentation from the system model, thereby supporting a transition from document-centric to model-centric AIV practices.

Together, these verification activities provide a structured assessment of the extent to which the MBSE AIV methodology is internally consistent, requirements-compliant, and aligned with established model-based systems engineering and space systems engineering practices.

In contrast to verification, validation is concerned with the question of *whether the right methodology is built*. In this thesis, validation focuses on assessing the applicability, usefulness, and scalability of the MBSE AIV methodology in representative engineering contexts. To this end, the methodology is applied to two complementary case studies: a simplified toaster example which was already presented and a qualification campaign of the Da Vinci Satellite CubeSat payloads. The toaster case serves as a controlled environment in which the full AIV lifecycle, including planning, execution, result management, and non-conformance handling, can be demonstrated. In contrast, the CubeSat payload case represents a realistic and higher-complexity engineering context, with a focus on the process steps rough AIV planning and detailed AIV planning for environmental and functional testing of the payload module.

The validation approach combines a qualitative comparison of both case studies with an evaluation using the FEMPP framework, assessing aspects such as functionality, efficiency, maintainability, and practical applicability of the methodology.

5.2. Verification of the MBSE AIV Methodology

5.2.1. Methodology Internal Consistency

This chapter aims to show that the designed methodology is consistently defined from conceptual foundations shown in the ontology, to the actual representation of these concepts in the modeling language. First aspect to check is whether the ontology conforms to the notation rules that were introduced in subsection 4.2.1 in Table 4.7. These rules are again shown here:

Table 5.1: Ontology definition rules used for the MBSE AIV methodology

ID	Ontology Definition Rule
O-1	The ontology shall consist only of ontology elements and relations between them. No additional modeling elements shall be introduced at the conceptual level.
O-2	Each relation shall connect exactly two ontology elements. Relations shall be strictly binary.
O-3	Each relation shall have a unique and descriptive name that clearly expresses the semantic meaning of the relationship between the two ontology elements.
O-4	For each relation, both connected ontology elements shall be assigned a role name, defining how each element participates in the relation.
O-5	Each relation shall define multiplicities for both roles, specifying the allowed number of participating instances (e.g. 0..1, 1, 0..*).
O-6	Specialization relationships between ontology elements shall be expressed using generalization (inheritance), indicating that a specialized element is a specific kind of a more general element.
O-7	The ontology shall be defined such that each element and relation can be directly mapped to SysML v1 constructs.

The internal consistency of the MBSE AIV methodology is assessed by analyzing the coherence between the defined ontology and its implementation in SysML v1. The ontology provides the conceptual representation of the AIV domain, including its elements, attributes, and relations, while the SysML implementation makes these concepts available in the modeling tool through stereotypes and customization of the tool.

For each ontology element, a corresponding stereotype is defined, with its attributes implemented as stereotype properties / tag definitions. Similarly, all ontology relations are mapped to appropriate SysML constructs, such as associations, dependencies, or allocations. Whenever a mapping was not possible, e.g. if the relation of the ontology is mapping onto a more complex SysML v1 structure, this is indicated and explained in textual form. This shows that the semantic structure defined at the conceptual level is consistently preserved within the modeling environment. A complete mapping of ontology elements, attributes, and relations to their SysML implementation is provided in Appendix B. There are a total of 48 ontology elements, mapped to their SysML v1 implementation. For some of the ontology elements, Stereotypes from the ESA MBSE Solution are used, but for the majority, new, custom stereotypes were introduced. Furthermore, there are 62 relations defined in the ontology. For each relation, the implementation is shown and if necessary explained. Table 5.2 shows the evaluation of

the ontology definition rules. Rule 7 is not fully satisfied, because some ontology elements rely on text based explanation in the mapping.

Table 5.2: Evaluation of ontology definition rules

ID	Evaluation	Evidence	Rationale
O-1	pass	Ontology diagram in Appendix B and all ontology figures in subsection 4.2.5.	The ontology only contains ontology elements, associations and generalization relations. This can be viewed in Appendix B or directly in the model.
O-2	pass	Ontology relations table in Appendix B	All relations are connected to exactly one type in the ontology relations table
O-3	pass	Ontology relations table in Appendix B	All relations have a name, shown in the ontology relations table
O-4	pass	Ontology relations table in Appendix B	All relations have one role name for each connected ontology elements in the ontology relations table
O-5	pass	Ontology relations table in Appendix B	All relations in the ontology relations table have a clear multiplicity defined for each of the roles.
O-6	pass	Ontology diagram in Appendix B	The diagram shows the only Specialization (inheritance) between AIV Entity, and the different kinds of AIV Entities.
O-7	partial pass	Ontology tables in Appendix B	Some of the elements and relations have not been mapped 1 to 1 to SysML constructs, but rely on additional textual explanation. The mapping is still complete, but relying on textual explanation cannot be considered a 100 % unambiguous mapping.

5.2.2. Evaluation against Requirements

This section evaluates the developed MBSE AIV Methodology against the methodology requirements defined in Table 4.6. The purpose of this evaluation is to assess whether the methodology developed in section 4.2 satisfies the requirements that were derived from the stakeholder needs and use cases. The evaluation therefore provides a requirements compliance mapping for the methodology itself.

Each requirement is assessed using three possible evaluation outcomes: *pass*, *partial pass*, or *fail*. A requirement is evaluated as *pass* when the developed methodology provides explicit support for the required capability and this support is demonstrated. A requirement is evaluated as *partial pass* when the methodology addresses the requirement conceptually or partially, but does not provide complete implementation support, automated generation, or full coverage of all expected details. A requirement would be evaluated as *fail* if the methodology does not address the required capability.

The evaluation is summarized in Table 5.3. For each requirement, the table provides the assessment result, the main evidence used for the assessment, and a short explanation. The evidence primarily refers to the model views, process steps, ontology implementation, and generated artifacts presented in the methodology chapter and annexes.

Overall, the evaluation shows that the majority of requirements are fulfilled by the developed methodology. Most capabilities are assessed as fulfilled. However, one deviation is identified for R-12, which is assessed as a *partial pass*. While the methodology supports the modeling of the main technical information required for an ECSS test specification document as described in the corresponding DRD, it does not yet fully cover all organizational and personnel-related aspects of the test specification, and the current implementation does not directly generate a complete test specification document from the model. Furthermore, R-4 is assessed as *partial pass* based on the available capabilities of the selected modeling tool and the adopted interface modeling approach, but the interface consistency check itself is not explicitly demonstrated in the report. This should be considered a limitation of the presented evidence rather than a limitation of the methodology concept.

Table 5.3: Evaluation of MBSE AIV Methodology Requirements

ID	Evaluation	Evidence	Explanation
R-2	pass	Shown in Figure 4.13.	Traceability between Requirements and AIV Activities is established using the SysML Verify relation. The Process Step 1.2 Define AIV Activities supports creating this traceability.
R-3	pass	Interface modeling shown in Figure 4.23. Reuse indicated in Figure 4.27.	AIV Entities and interfaces are modeled in the process steps 1.1, 2.3, and 2.5. The provided examples show interface modeling and reuse of model elements in different setups.
R-4	partial pass	Leveraging tool capabilities	The modeling tool supports interface consistency checks out of the box. The capability was not demonstrated in this report. The way in which interfaces are modeled for this methodology is taken directly from the ESA MBSE solution.
R-5	pass	AIV Flow shown in activity diagrams in Figure 4.36 and Figure 4.37.	This functionality is supported by process step 4.1 and 4.2.
R-6	pass	AIV Flow with dependencies shown in activity diagrams in Figure 4.36 and Figure 4.37.	This functionality is supported by process step 4.1 and 4.2.
R-7	pass	VCD view shown in Figure 4.9	Creating VCD-like views is supported by process step 1.2. Modeling tool has the capability to export any tables into Excel, with minimal required post-processing in excel to create typical VCD views.
R-8	pass	Shown in Figure 4.9, Figure 4.12, and Figure 4.15	Compliance status and closeout status are parameters of the custom requirement stereotype, method, stage, and level are parameters of the AIV Activity stereotype, and verdict is a parameter of the AIV Criterion stereotype.
R-9	pass	Shown in Figure 4.19	Generated AIV Evidence is linked to criteria via trace relations.
R-10	pass	Linking external documents to model elements shown e.g. in Figure 4.32	External documents can be linked to any model element, either via the attached file mechanism (shown in the report) or by adding a hyperlink to a shared drive.
R-11	pass	Throughout section 4.2	Terminology aligns with ECSS wording throughout the methodology.
R-12	partial pass	Covered in steps 1 and 2 of the modeling methodology in section 4.2	The methodology describes how to include all technical aspects of a test specification according to ECSS, but some aspects, e.g. regarding the modeling of responsibilities of involved personnel, have been omitted. The current implementation does not allow to directly generate a test specification document from the model.
R-16	pass	Shown in Figure 4.30 and more detailed in Appendix D and Appendix E	Test procedures can be generated, if the model is created according to the modeling methods described. Some manual post-processing can be necessary.
R-18	pass	Throughout section 4.2	The model-based environment ensures that all model elements have a unique identifier. For real projects, the ID conventions should be modified per project needs during the tailoring step.
R-19	pass	Shown in Figure 4.39 and Figure 4.40	Modeling non-conformances is is described in process steps 5.1 to 5.3.
R-21	pass	Throughout section 4.2 and in Appendix B	The ontology is presented in this report.
R-22	pass	Throughout section 4.2	The process is presented in this report.

Continued on next page

ID	Evaluation	Evidence	Explanation
R-23	pass	Throughout section 4.2	The modeling methods are presented in this report.
R-24	pass	Throughout section 4.2	Views such as the VCD-like table view are presented.
R-25	pass	Shown in Appendix B and throughout section 4.2	The ontology is implemented using SysML v1 stereotypes in Cameo. A detailed mapping is presented in the annex.
R-26	pass	Presented in subsection 4.2.4	Tool selection is performed and described in this report.

5.2.3. ISO/IEC/IEEE 24641 [38] Compliance Mapping

In accordance with ISO/IEC/IEEE 24641, conformance is assessed with respect to the outcomes of the selected process "Perform system verification and validation". Full conformance to the standard is not claimed; instead, the evaluation demonstrates that the developed MBSE AIV methodology achieves the intended outcomes of the selected process within its defined scope.

The ISO standard provides a framework of systems engineering processes and requirements on how MBSE solutions can support these processes. The purpose of the process "Perform system verification and validation" is to demonstrate, based on system models and throughout the system lifecycle, that the system fulfils its requirements, meets stakeholder needs, and achieves its intended operational purpose. This aligns with the objective of the presented MBSE AIV methodology, which enables structured definition and execution of AIV activities, as well as traceable verification and validation of system requirements. The standard defines a lot more processes, which fall outside the scope of the MBSE AIV Methodology. Each process has a set of relevant inputs, required outputs, and tasks that break down the process. Each task has a set of requirements that address the methods employed to achieve the tasks goals, and the capabilities of the modeling tool(s) used to support work on the tasks.

For the process "Perform system verification and validation", the following inputs are defined in the ISO Standard:

1. System models and discipline-specific models
2. Criteria for decision making
3. System requirements & traceability
4. Stakeholder requirements

These inputs correlate quite nicely with the assumed starting point to start applying the MBSE AIV Methodology. The most important inputs are the system requirements and stakeholder requirements. The standard understands the term system models as MBSE models / digital representations of a system. In this regard, the framing of the MBSE AIV Methodology is a bit different, because it does not assume that a project has been applying MBSE from the start. The difference between verifying and validating digital models, or physical systems, is in the end captured by the difference in verification method, i.e. the difference between an analysis and a test. While ISO/IEC/IEEE 24641 assumes a model-centric approach in which the system model itself is the primary subject of verification and validation, the MBSE AIV methodology presented in this thesis focuses on the verification and validation of the physical system through model-based definition and management of AIV activities. In this context, the model serves as a structured representation to define, plan, and trace verification and validation activities, but the MBSE AIV Methodology does not strictly require a fully developed digital system model at the start of its application. A possibility to conceptually align the ISO standards scope and the MBSE AIV Methodology better is to understand a system model as either a digital or physical model.

For the process "Perform system verification and validation", the following outputs are defined in the ISO Standard:

1. Test cases and verification procedures are developed
2. System model is verified
3. System model is validated

These outputs are closely aligned with the goals of the MBSE AIV Methodology. The development of test cases and verification procedures is directly supported through the definition of AIV Activities and the generation of structured AIV Procedures from the model. Furthermore, the methodology enables the verification of system requirements by establishing traceability between requirements, AIV criteria, verification activities, and the resulting evidence and verdicts.

Validation of the system model with respect to stakeholder needs is supported indirectly through the traceability from stakeholder needs to system requirements and further to verification results. Additionally, validation events and activities can be modeled similarly to verification activities, though this is not shown in the examples.

To achieve these outcomes, the ISO process "Perform system verification and validation" is structured into three main tasks:

1. Prepare model-based verification and validation
2. Perform model-based verification and validation
3. Manage results

The methodology supports the preparation of model-based verification and validation through the presented modeling processes *1. Rough AIV Planning* and *2. Detailed AIV Planning*, including the definition of verification strategies, AIV Activities, inputs, outputs, required resources and setups. The execution of verification and validation is achieved through AIV Activities and Procedures, which define the steps to be performed. Finally, result management is supported through the modeling of AIV Evidence, verification verdicts, and Non-Conformances, enabling the recording, analysis, and reporting of verification results and anomalies. These tasks are covered by the presented modeling processes *2. Detailed AIV Planning*, *3. AIV Activity Execution and Postprocessing*, and *5. AIV Quality Control*.

Overall, the developed MBSE AIV methodology demonstrates alignment with the outcomes and tasks of the ISO/IEC/IEEE 24641 process "Perform system verification and validation". While not all tasks and tool capabilities defined in the standard are explicitly implemented, the methodology provides a coherent and model-based realization of the intended process outcomes.

The following presents a table of methods and tool capabilities listed in the standard that are covered and not covered by the MBSE AIV Methodology.

Table 5.4: Coverage of ISO/IEC/IEEE 24641 methods and tool capabilities by the MBSE AIV methodology

ISO Method (M) / Tool Capability (TC)	**Coverage**	**Rationale**
(M) Generation of verification and validation plans	Covered	Supported through rough and detailed AIV planning, including definition of activities, criteria, and setup.
(M) Setting up required (simulation) tool environment	Not covered	Tool environment setup is outside the scope of the methodology and assumed to be project-specific. Approach can possibly be used to specify simulation setups, but this was not further investigated in the thesis.
(M) Collecting and digitalizing required data	Covered	Requirements, criteria, and AIV-related information are represented and structured within the SysML model.

Continued on next page

ISO Method (M) / Tool Capability (TC)	Coverage	Rationale
(M) Preparing models and mock-ups for V&V	Partially covered	System elements, GSE, and test setups are modeled; however, detailed simulation models and User Interface mock-ups for validation are not explicitly addressed.
(M) Updating requirements management database for traceability	Covered	Full traceability is enabled between requirements, criteria, activities, and results within the model.
(TC) Formal modeling capabilities	Covered	The methodology is implemented using SysML v1, providing a formal modeling structure.
(TC) Generation of system verification plans	Covered	Verification planning artefacts are generated from the model (e.g. AIV planning, VCD views, procedures).
(TC) User interface mock-ups for demonstrations or simulations	Not covered	UI mock-ups are not covered by the methodology.
(TC) Implementation of advanced V&V techniques (model checking, runtime verification, formal proof)	Not covered	The methodology focuses on AIV activities and does not include advanced formal verification techniques.
(M) Execution and simulation of models	Covered	Simulation of system models is possible with small extensions to the presented approach to include system models with explicitly modeled behavior.
(M) Performing defined V&V techniques according to plans	Covered	This is covered by the explained process steps for creating procedures and describing how to post process results.
(TC) Support for model execution and simulation	Partially covered	Model execution and simulation are supported by the tool, but not extensively shown in the report.
(M) Recording verification results and anomalies	Covered	Verification results and anomalies are explicitly modeled.
(M) Generation of reports	Covered	The tool capability was demonstrated to generate procedure documents, this can be extended to other types of documentation.
(M) Managing verified model baseline	Partially covered	Model versioning is supported by the tool (e.g. Cameo), but not explicitly defined within the methodology.
(TC) Storing results	Covered	Results are stored within the model as evidence and linked artifacts.
(TC) Document generation capability	Covered	Supported through document generation from the model.

Continued on next page

ISO Method (M) / Tool Capability (TC)	Coverage	Rationale
(TC) Connection to configuration management (CM) tools	Not covered	Integration with external CM tools is outside the scope of the methodology.

5.2.4. ECSS DRDs Compliance Assessment

In addition to the process-oriented compliance assessment against ISO/IEC/IEEE 24641, the developed MBSE AIV methodology is evaluated with respect to its ability to produce documentation that conforms to ECSS standards. In the space engineering domain, ECSS standards define not only processes, but also the required deliverables and their expected content in the form of Document Requirement Definitions (DRDs). Ensuring that model-based approaches can support the generation of such deliverables is therefore a key aspect of practical applicability and industrial relevance.

This section presents a mapping of selected ECSS DRDs to the elements of the MBSE AIV methodology. The considered DRDs include the Verification Plan (VP), Verification Control Document (VCD), Test Specification (TSPE), Test Procedure (TPRO), and the Non-Conformance Report (NCR) status list. Together, these documents cover a large part of the AIV lifecycle, from planning to execution and anomaly handling.

For each DRD, the prescribed structure and content are mapped to corresponding model elements, i.e. SysML stereotypes and their associated properties, as defined in the methodology. The focus of this mapping is on showing that the required information can be represented within the model, rather than recreating the documents themselves in full detail.

The objective of this mapping is to demonstrate that the required information content of each document can be represented within the system model. By establishing this correspondence, it is shown that ECSS-compliant documentation can be supported directly from the model and, where required, generated in a consistent and traceable manner.

The following DRDs are considered in this analysis:

1. Verification Plan: ECSS-E-ST-10-02C Rev.1 Annex A Verification Plan (VP) - DRD
2. Verification Control Document: ECSS-E-ST-10-02C Rev.1 Annex B Verification Control Document (VCD) - DRD
3. Test Specification: ECSS-E-ST-10-03C Rev.1 Annex B Test Specification (TSPE) - DRD
4. Test Procedure: ECSS-E-ST-10-03C Rev.1 Annex C Test Procedure (TPRO) - DRD
5. NCR Status: ECSS-Q-ST-10-09C Rev.1 Annex B NCR Status List - DRD

The tables show a complete description of the sections of the DRDs and indicate the model elements that represent the relevant information that is expected in those sections of the documents. The paragraphs below discuss the mapping.

All DRDs call for an Introduction, a list of Applicable and Reference Documents, and an explanation of Definitions and abbreviations. This has not been explicitly implemented in the presented methodology, but can be done with relatively small refinements. It was already presented that any files can be attached in the model and linked to model elements. The tool provides functionalities to create a glossary of terms that can be added to any document as needed. These two aspects have already been shown in literature and could be added to the presented methodology.

Verification Plan

The verification plan serves the purpose to of describing the overall verification approach of a project on a high level. It includes the system that shall be tested, the model philosophy, method, levels and stages of the verification, and the sequence of verification activities to be conducted. It also includes a description of the required verification tools.

The information that is requested by the verification plan DRD is populated in the model during the steps *1 Rough AIV Planning* and *4 AIV Process Control* of the MBSE AIV Methodology. The model philosophy is modeled using AIVEntity model elements. The relations between the AIVEntities and the AIVActivities can be extracted by querying the model to reveal the information which activities involve which AIV model elements, e.g. qualification model, flight model, etc. and the required verification tools and facilities.

The programmatic aspects and especially the dependencies between AIV Activities are captured in the AIV Flow element and the AIV Events.

The mapping of model elements to the chapters of the DRD is shown in Figure H.2.

Verification Control Document
The Verification Control Document is included as a model view in the methodology. This implementation of the document as a model view is done in the process step *1.2 Define verification control approach*. The VCD as a model view includes the requirements and the required model element properties, as well as the criteria and verification activities related to the requirements in the VCD.

The mapping of model elements to the chapters of the DRD is shown in Figure H.3.

Test Specification
A test specification describes all aspects needed for planning, conducting, post-processing, and reporting a test activity. In the process step *3 Detailed AIV Activity Planning* the relevant information is modeled, that is needed to populate a test specification document, and which is more specialized than what is already included in the Verification Plan. All aspects of the DRD are covered by the different model elements, with the exception that a summary and the timeline of the activity is not explicitly modeled. The collected evidence is modeled before test execution, which is a prerequisite to be compliant with the DRD.

The mapping of model elements to the chapters of the DRD is shown in Figure H.4.

Test Procedure
A generated test procedure can be found in the annex. The procedures that are generated include the modeled test setup, with the Item under test, the required ground support equipment and instrumentation. The generated procedure allows to perform a test according to the previously modeled step-by-step instructions, which are connected to the criteria and the equipment and the facilities used in each step. All required inputs and all expected outputs of the test are listed and can be used to check if the pre-conditions and post-conditions of a test activity are met. Responsibilities of stakeholders are indicated to some extent. For full coverage of this aspect of the DRD, manual additions are required to include the names of people in the document.

The mapping of model elements to the chapters of the DRD is shown in Figure H.5.

NCR Status List
The MBSE AIV methodology allows users to capture information related to Non-Conformances in the model, which can be used to show an NCR status list. The Non-Conformances are handled within the *5 AIV Quality Control* process steps. The relevant model element is the Non-Conformance, implemented using a customized SysML Comment Stereotype with properties in accordance with the relevant parameters.

Overall, it can be observed that the model elements defined in the MBSE AIV methodology map well to the different sections of the considered DRDs. Most chapters of the documents can be directly related to one or more stereotypes and their associated properties, which indicates that the required information is already present in the model in a structured way. This shows that the methodology does not rely on additional, external documentation to capture essential AIV information, but instead embeds it within the model itself. As a consequence, the model provides a basis to decrease the effort to create and maintain ECSS-compliant documentation, while at the same time improving consistency and traceability across the different documents.

The mapping of model elements to the chapters of the DRD is shown in Figure H.6.

5.3. Validation of the MBSE AIV Methodology

5.3.1. Overview of Case Studies

The MBSE AIV Methodology was applied to three examples, each with a distinct purpose to showcase the usability and applicability of the methodology across different levels of system complexity and AIV contexts. The examples chosen for the thesis are summarized in Table 5.5. The second and third examples are part of the same model and involve the same technical systems.

Table 5.5: Overview of validation examples for the MBSE AIV Methodology

Example	Purpose	Scope	Role in Validation	Relevant System
Toaster Model	Demonstrate methodology in a controlled, low-complexity environment	Full AIV lifecycle (criteria, activities, execution, evidence, NCR handling)	Show completeness and internal consistency of the methodology and its ontology	Simple electromechanical system (Toaster)
DVS Payload Module Transportation Procedures	Apply methodology to realistic environmental and handling case.	Rough and detailed AIV planning and procedure generation	Demonstrate applicability of the methodology for assembly and verification activities.	DVS CubeSat payload module Transportation Setup
DVS Dice Payload Functional Testing	Showcase methodology for functional verification of a complex subsystem	Rough and detailed AIV planning and procedure generation with focus on criteria and activities for full and reduced functional testing	Demonstrate scalability and applicability to realistic subsystem-level verification	DVS CubeSat Dice Payload

5.3.2. The CubeSat Payload Application

The case study chosen for validating the methodology presented in this thesis is the Da Vinci Satellite Student CubeSat mission [5]. This project is a fully student-led satellite project that aims to develop a 2U CubeSat with two educational payloads. The mission aims to inspire and enthuse high school and primary school students about space missions and technology. At the time of writing the thesis, the project is part of the ESA Fly Your Satellite! programme. In the frame of this programme, the student team got the chance to conduct a full environmental test campaign of the payload module including the two payloads and a baseplate.

To prepare for the environmental test campaign, the procedures for transporting the payload and for the full functional tests and reduced functional tests were created following the MBSE AIV Methodology. In the following, the process steps will be presented for the transportation and functional testing examples.

0. Tailoring of the MBSE AIV Methodology

Only small tailoring to the requirement stereotype was required to apply the methodology in this case. First, the ID of the imported requirements were saved to the source field, rather than the ID field of the stereotype. This was done because the standard ID field would have required creating a custom numbering scheme, while the IDs of the requirements used in the project follow different schemes. The other options would have been to adapt the numbering scheme and make it compatible, which would be the recommended approach for future applications. The second is the use of compliance status attributes for phase C, D, and E, for each requirement.

One other obvious tailoring activity is needed to align the procedure templates that are used for automatic document generation to the project-specific documentation templates. This involves adapting the VTL templates used in the Report Wizard of the modeling tool.

1. Rough AIV Planning

The model philosophy is defined by creating AIV Entities that represent all AIV model elements. In this case, there are qualification models and flight models of the payloads and the baseplate. The AIV Entities have a generalization relationship to the product specification, meaning that they inherit all

interfaces, part properties, etc. from the product specification. This relationship and the different model elements are shown in Figure 5.1.

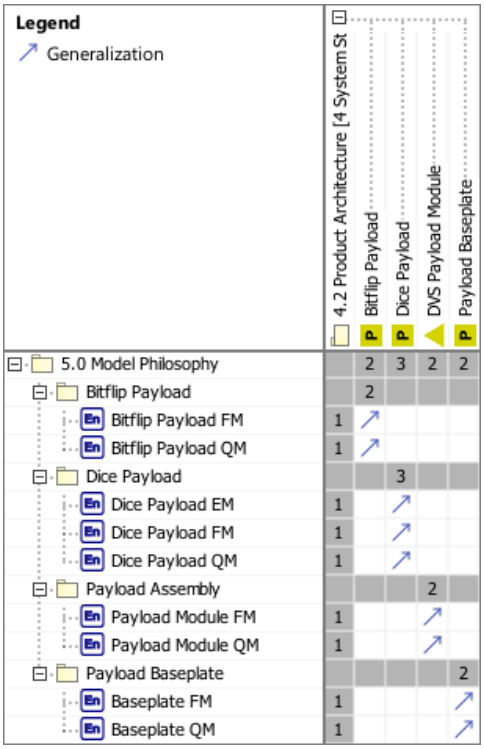


Figure 5.1: Model Philosophy defined for the example scope of the case study

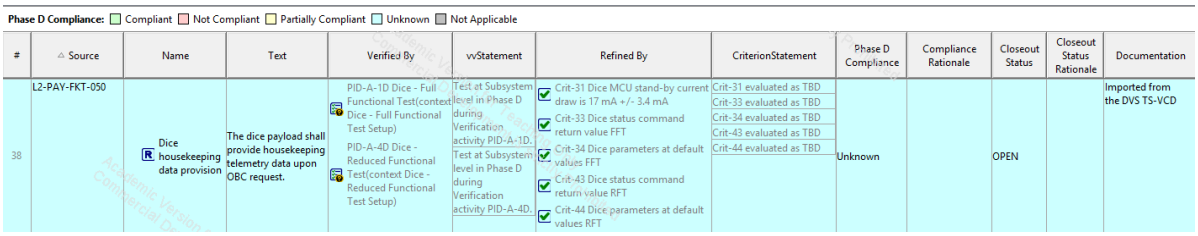
The initial requirements that were taken from the DVS project are shown in Figure 5.2.

Phase D Compliance: Compliant Not Compliant Partially Compliant Unknown Not Applicable

#	Source	Name	Text	Phase D Compliance	Closeout Status
29	CSF_1	Hardware Cleanliness	Before visiting the CSF, the hardware shall be visibly clean.	Unknown	OPEN
32	FDS-4.7.1.	Environment Cleanliness	Assembly, integration, testing and other ground processing activities shall be performed within a Class 100.000 clean environment (ISO	Unknown	OPEN
35	L2-PAY-FKT-010	Roll dice	The dice payload shall be able to roll the dice when commanded by the OBC.	Unknown	OPEN
36	L2-PAY-FKT-020	Capture dice	The dice payload shall be able to capture the dice following each roll.	Unknown	OPEN
37	L2-PAY-FKT-030	Take picture of dice	The dice payload shall be able to take an image of the rolled dice.	Unknown	OPEN
38	L2-PAY-FKT-050	Dice housekeeping data provision	The dice payload shall provide housekeeping telemetry data upon OBC request.	Unknown	OPEN
42	L2-PAY-FKT-210	Dice video quality and frame rate	The dice payload shall be able to take videos with the quality of 299k at a frame rate of 15 FPS.	Unknown	OPEN
43	L2-PAY-FKT-220	Dice picture format and quality	The dice payload shall take JPEG pictures with the quality of 640x480.	Unknown	OPEN
44	PL_TRSP_1	Survive Transport	The DVS Payload Module shall survive transport to and from test facilities.	Unknown	OPEN
45	PL_TRSP_1-1	Max allowed shock during transport	The DVS Payload Module shall experience no mechanical shock of higher then 25 g in magnitude.	Unknown	OPEN
46	PL_TRSP_1-2	Protect against moisture during transport	The DVS Payload Module shall not experience a relative humidity above 65% during transport.	Unknown	OPEN
47	PL_TRSP_1-3	Dice PL Transport Configuration	For transportation, the DVS Dice Payload shall be in launch configuration, with it's clamping motors in the fully clamped position.	Unknown	OPEN
48	PL_TRSP_1-4	Reduce likelihood of mechanical impact during transport	The Transportation setup of the DVS Payload Module shall protect the payload against mechanical impacts during transport.	Unknown	OPEN

Figure 5.2: Initial set of requirements from the project for the selected scope

The next step is to start populating the VCD view in the model with the relevant data. This is shown for one example requirement in Figure 5.3. As shown in the figure, the requirement is verified in two different verification activities, the full functional test and the reduced functional test. Additionally, there are 5 criteria related to showing compliance of the requirement in the refined by column. These will be described in detail in another process step.



Phase D Compliance: Compliant | Not Compliant | Partially Compliant | Unknown | Not Applicable

#	Source	Name	Text	Verified By	vvStatement	Refined By	CriterionStatement	Phase D Compliance	Compliance Rationale	Closeout Status	Closeout Status Rationale	Documentation
38	L2-PAY-FKT-050	Dice housekeeping data provision	The dice payload shall provide housekeeping telemetry data upon OBC request.	PID-A-1D Dice - Full Functional Test(context Dice - Full Functional Test Setup) PID-A-4D Dice - Reduced Functional Test(context Dice - Reduced Functional Test Setup)	Test at Subsystem level in Phase D during Verification activity PID-A-1D. Test at Subsystem level in Phase D during Verification activity PID-A-4D.	Crit-31 Dice MCU stand-by current draw is 17 mA +/- 3.4 mA Crit-33 Dice status command return value FFT Crit-34 Dice parameters at default values FFT Crit-43 Dice status command return value RFT Crit-44 Dice parameters at default values RFT	Crit-31 evaluated as TBD Crit-33 evaluated as TBD Crit-34 evaluated as TBD Crit-43 evaluated as TBD Crit-44 evaluated as TBD	Unknown		OPEN		Imported from the DVS TS-VCD

Figure 5.3: Example for one Requirement in the VCD

The last step in the rough planning process of the methodology is to define the AIV activities in the model. Two of them were already shown in the VCD extract. Figure 5.4 shows all seven AIV Activities to cover the transportation and functional testing procedures. The procedure to set up the payloads for the reduced functional test does not verify any requirements by itself. The description of how to set up was taken out from the reduced functional test procedure, while it remains part of the setup for the full functional test. This was done because the reduced functional test is also performed when the payloads are in a thermal vacuum chamber, which requires a different setup, while a full functional test always requires exactly the setup described in the procedure. This split in the reduced functional test promotes reuse.

The table indicates the requirements and the related criteria for each modeled AIV Activity. At the rough AIV planning step, it is not necessary to create all criteria yet for all the AIV activities. The criteria are already included here, because each verification activity was further specified in the detailed planning step.

#	AIV Act ID	Name	Level	Method	Stage	Vv Kind	Verifies	Related AIV Criteria
3	PID-34	PLM QM packing inside Cleanroom	Subsystem	Inspection	Phase D	Integration	162 Environment Cleanliness 125 Hardware Cleanliness 189 Protect against moisture during transport 190 Dice PL Transport Configuration	Crit-7 Inspect Payload Hardware Cleanliness before Transport Crit-2 Confirm dessicant bag placement inside 1st ESD wrapping Crit-1 Confirm Cleanroom is of class 100.000 / ISO 8 Crit-3 Confirm dessicant bag placement inside 2nd ESD wrapping Crit-8 Payload Module fully enclosed in first layer ESD wrapping Crit-6 Confirm that Dice Payload is in clamped position for transport Crit-4 Confirm humidity indicator placement inside 2nd ESD wrapping Crit-9 PLM fully enclosed in second layer ESD wrapping Crit-18 Confirm that dessicant bags are dry
4	PID-35	PLM QM packaging outside Cleanroom	Subsystem	Inspection	Phase D	Integration	191 Reduce likelihood of mechanical impact during transport 125 Hardware Cleanliness 188 Max allowed shock during transport	Crit-11 Shock indicators not triggered Crit-13 Stickers on outer ESD wrapping warning from opening in non-clean environment Crit-16 Fragile sticker placed Crit-17 Shock indicator placed on the outside of ESD wrapping Crit-15 Confirm Payload tight fit in transportation case Crit-5 Confirm shock indicators suitability Crit-10 Shock indicators installed in 3 axes Crit-14 Transported item identification sticker placed
8	PID-54	PLM QM UNpacking inside Cleanroom	Subsystem	Inspection	Phase D	Integration	189 Protect against moisture during transport 125 Hardware Cleanliness	Crit-28 Less then 50% relative humidity Crit-30 Payload Hardware Cleanliness after Transport
9	PID-55	PLM QM UNpacking outside Cleanroom	Subsystem	Inspection	Phase D	Integration	125 Hardware Cleanliness 188 Max allowed shock during transport 187 Survive Transport	Crit-24 Payload case undamaged after transport Crit-25 PLM external shock sensors not triggered Crit-26 PLM transport case internal shock sensor not triggered Crit-27 PLM second transport layer intact after transport
10	PID-56	Dice - Setup for reduced functional test	Subsystem	Inspection	Phase D	Integration		
12	PID-A-1D	Dice - Full Functional Test	Subsystem	Test	Phase D	Verification	152 Capture dice 154 Dice housekeeping data provision 159 Dice picture format and quality 158 Dice video quality and frame rate 151 Roll dice 153 Take picture of dice	Crit-31 Dice MCU stand-by current draw is 17 mA +/- 3.4 mA Crit-32 Dice Camera stand-by current draw is 200 mA +/- 20 mA Crit-33 Dice status command return value FFT Crit-34 Dice parameters at default values FFT Crit-35 Stop switches readout returns "12" when clamped FFT Crit-36 Stop switches readout returns "3" when unclamped FFT Crit-37 Stop switches readout returns "12" after clamping cycle Crit-38 LDR value higher after clamping cycle FFT Crit-39 Sweeper moves during unclamping Crit-40 Sweeper moves during clamping Crit-41 Check dice image quality Crit-42 Dice video quality
16	PID-A-4D	Dice - Reduced Functional Test	Subsystem	Test	Phase D	Verification	154 Dice housekeeping data provision 152 Capture dice	Crit-43 Dice status command return value RFT Crit-44 Dice parameters at default values RFT Crit-45 Stop switches readout returns "12" when clamped RFT Crit-46 Stop switches readout returns "3" when unclamped RFT Crit-47 LDR value higher after clamping cycle RFT

Figure 5.4: The seven AIV Activities defined to cover transportation and testing procedures.

2 Detailed AIV Activity Planning

The next step is to ensure that the modeled test is specified in terms of the criteria that need to be checked in order to verify the corresponding requirements. This serves to translate the general requirements into criteria that are specific to the planned test and are easily evaluated with the planned evidence that is generated. In this example, the full functional test will be detailed further. Figure 5.5 and Figure 5.6 show the model views that present the traceability between requirements, AIV activities, and criteria. These are the same view definitions as already presented in Figure 4.16c and Figure 4.16a, but now for the extended scope of the full functional test of a satellite payload.

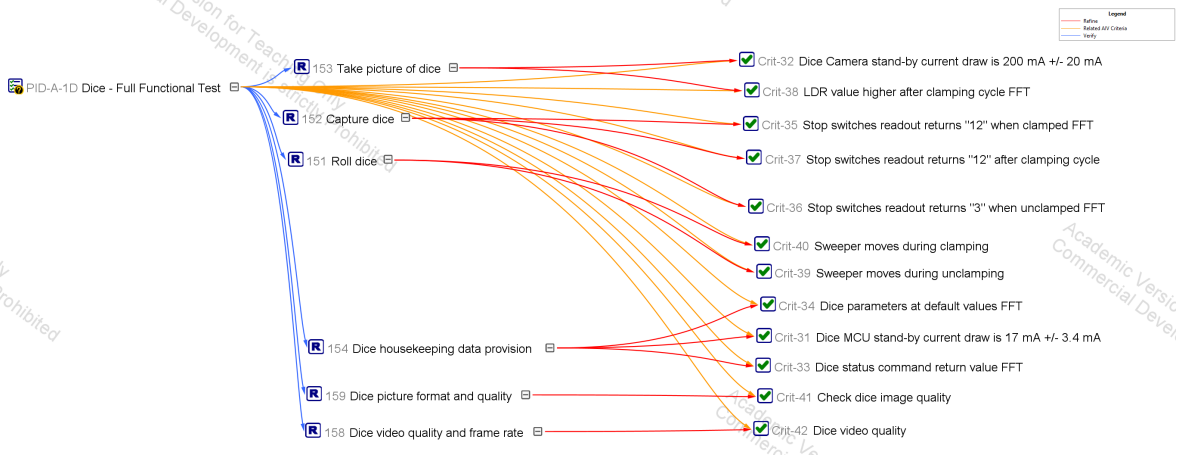


Figure 5.5: Relation map showing all requirements and criteria for the full functional test of the dice payload

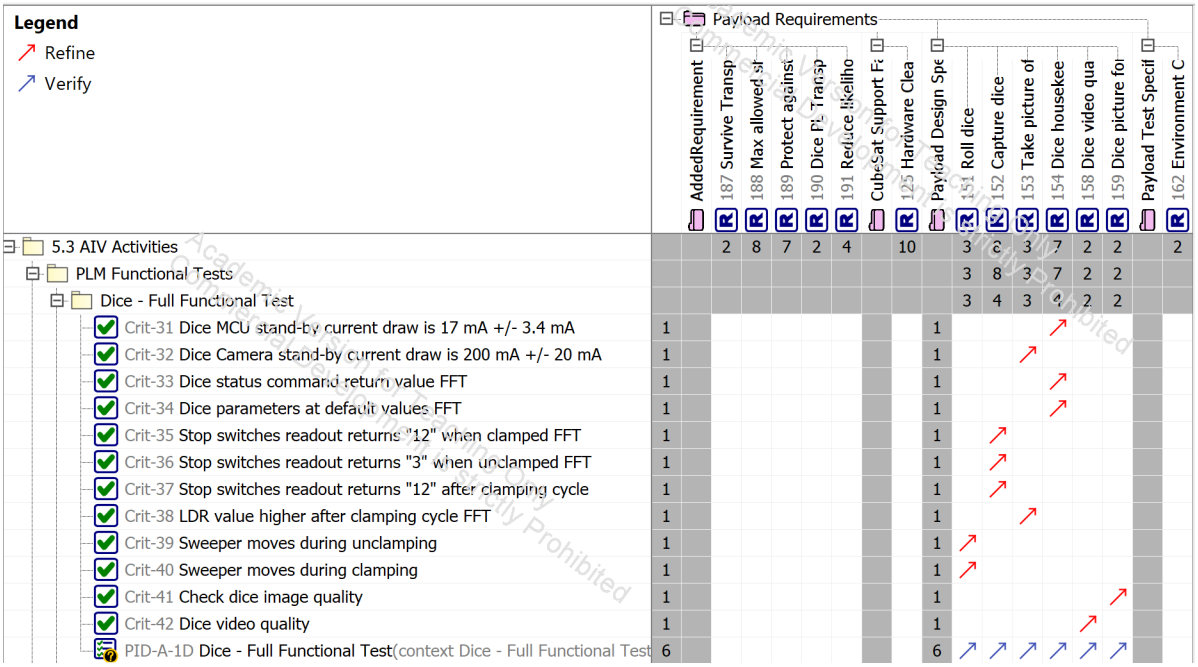


Figure 5.6: Traceability matrix from AIV Activities and Criteria to Requirements

Defining a criterion also includes the explicit modeling of the required evidence that contributes to evaluation of this criterion. The criteria defined to refine the requirement from Figure 5.3 about the Dice housekeeping data provision, are shown in Figure 5.7. The last column of the table indicates the required evidence that is expected to be generated during the procedure and to inform the criterion evaluation. It can be seen that the evidence that needs to be generated during the test activity has to include a picture of the current draw value measured by the power supply, the noted relevant values in the as-run procedure, and a full software log of the test. In addition, the table allows to specify the name, text, and additional notes for evaluation for each criterion. In total the full functional test activity involves 12 criteria derived from the 6 requirements that are verified during the test. Figure 5.8 shows the evidence related to the 4 criteria and the example requirement in another model view that is used to provide details about the evidence. In total there were 13 AIV evidence model elements modeled in the scope of the full functional test.

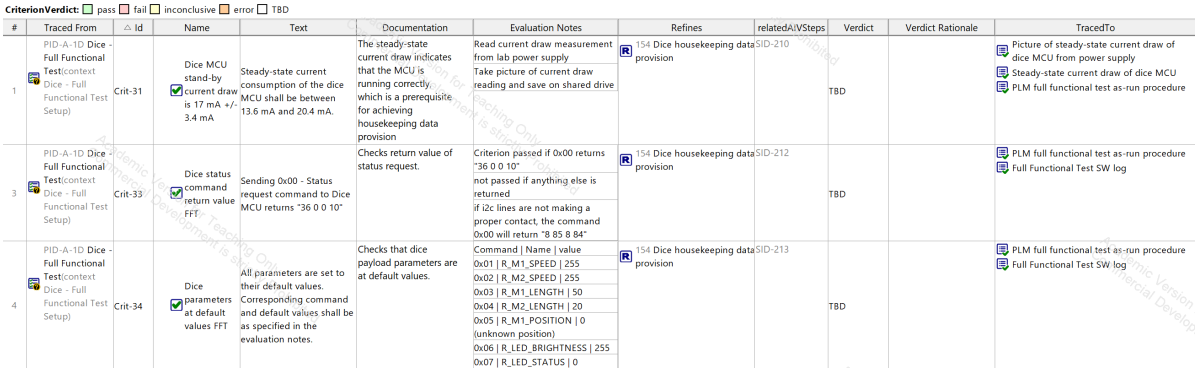


Figure 5.7: Detailed criteria for the example requirement

EvidenceStatus: planned collected outdated

#	Name	Status	Owned Properties	Traced From	RelatedRequirement
2	Picture of steady-state current draw of dice MCU from power supply	planned	status = planned	Crit-31 Dice MCU stand-by current draw is 17 mA +/- 3.4 mA	154 Dice housekeeping data provision
4	Steady-state current draw of dice MCU	planned	status = planned steady-state current of dice MCU : total current[ampere] = TBD A	Crit-31 Dice MCU stand-by current draw is 17 mA +/- 3.4 mA	154 Dice housekeeping data provision
12	Full Functional Test SW log	planned	status = planned	Crit-33 Dice status command return value FFT Crit-34 Dice parameters at default values FFT Crit-35 Stop switches readout returns "12" when clamped FFT Crit-36 Stop switches readout returns "3" when unclamped FFT Crit-37 Stop switches readout returns "12" after clamping cycle	154 Dice housekeeping data provision 152 Capture dice
13	PLM full functional test as-run procedure	planned	status = planned	Crit-33 Dice status command return value FFT Crit-34 Dice parameters at default values FFT Crit-35 Stop switches readout returns "12" when clamped FFT Crit-36 Stop switches readout returns "3" when unclamped FFT Crit-37 Stop switches readout returns "12" after clamping cycle Crit-39 Sweeper moves during unclamping Crit-40 Sweeper moves during clamping Crit-31 Dice MCU stand-by current draw is 17 mA +/- 3.4 mA Crit-32 Dice Camera stand-by current draw is 200 mA +/- 20 mA Crit-41 Check dice image quality Crit-42 Dice video quality	154 Dice housekeeping data provision 152 Capture dice 151 Roll dice 153 Take picture of dice 159 Dice picture format and quality 158 Dice video quality and frame rate

Figure 5.8: Table of required evidence to evaluate the example requirement.

The next steps are to model the test setup and the inputs and outputs of the activity, with their multiplicities and an indication who is responsible for providing an input or handling an output. The inputs and outputs are shown in Figure 5.9 and the setup is modeled in Figure 5.10.

#	△ Type	Direction	Multiplicity	ResponsibleStakeholder
3	2 channel lab power supply	in	1	ESA Cleanroom Operator
4	45 degree inclination mount	in	1	DVS
6	Camera	in	1	DVS
7	cleanroom gowns gloves hair and beard nets	in	(Unspecified)	ESA Cleanroom Operator
8	Dice Full Functional Test Python Script	in	1	DVS
9	ESD mat with bracelets	in	(Unspecified)	ESA Cleanroom Operator
10	Facom TX8 screwdriver	in	1	DVS
14	Operator Laptop	in	1	DVS
15	Payload Module QM	in	1	DVS
20	Red black banana connector wires	in	2	DVS
21	Screws 2.5x* ISO 14583 TX8 SS A4	in	8	DVS
24	TVAC Cable	in	1	DVS
25	TVAC_I2C_PCB with Arduino UNO	in	1	DVS
26	USB-B to USB-A cable 1.5 meters	in	1	DVS
28	Washer M2.5 ISO 7089 200HV SS A4	in	8	DVS

(a) Inputs

#	△ Type	Direction	Multiplicity	ResponsibleStakeholder
1	0x0D return value after clamping	out	1	DVS
2	0x0D return value before clamping	out	1	DVS
5	Bit rate value of dice camera video	out	1	DVS
11	Format of dice camera picture	out	1	DVS
12	FPS value of dice camera video	out	1	DVS
13	Full Functional Test SW log	out	1	
16	Picture of steady-state current draw of dice camera from power supply	out	1	DVS
17	Picture of steady-state current draw of dice MCU from power supply	out	1	DVS
18	Picture taken with dice camera	out	1	DVS
19	PLM full functional test as-run procedure	out	1	DVS
22	Steady-state current draw of dice camera	out	1	DVS
23	Steady-state current draw of dice MCU	out	1	DVS
27	Video taken with dice camera	out	1	DVS

(b) Outputs

Figure 5.9: Modeled inputs and outputs of full functional test activity

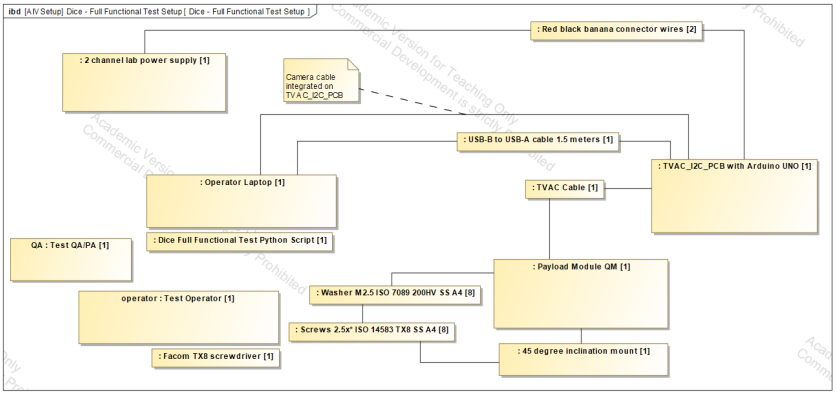


Figure 5.10: Modeled test setup for the Dice Payload Full Functional Test

In preparation for the test campaign, the setup was not included in the procedure to a deeper level, because it was already tested in previous tests that were outside the modeling scope. The setup model is included here for completeness, while the procedure describes how to correctly configure the setup. The more important aspect of the model here was the completeness of the setup, since every part of the setup needed to be brought by the student team. The model helped to ensure that the modeled setup and the input parameters of the activity are aligned. No parametric modeling was needed for this example. The full traceability diagram showing the digital thread is shown in Figure 5.11. This diagram shows all related requirements and the involved setup elements, as well as the criteria and the evidence to evaluate the criteria.

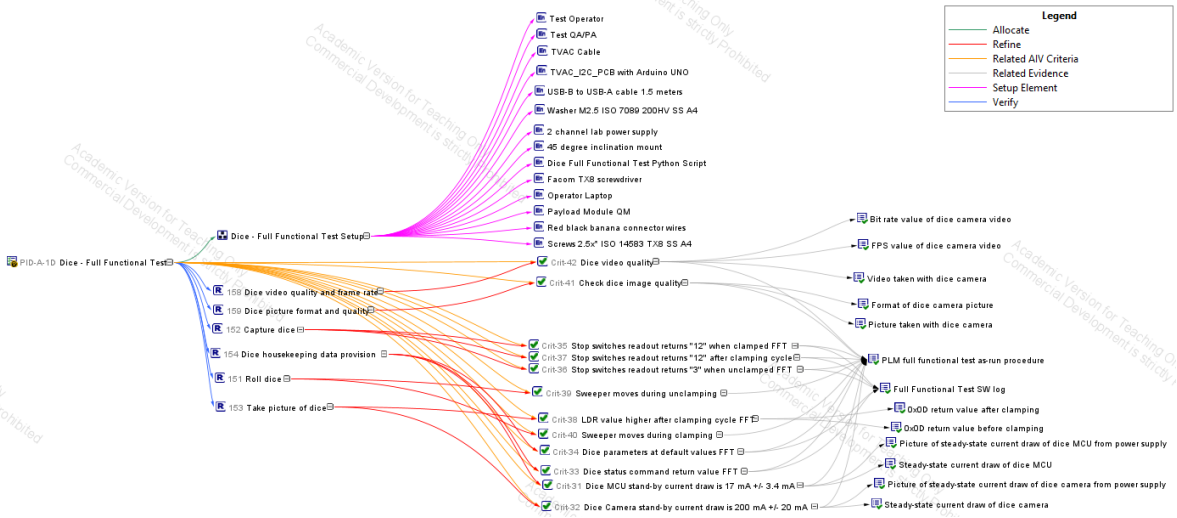


Figure 5.11: Full traceability view for the full functional test, showing digital thread

AIV Activity Execution and Postprocessing

The diagram that describes the step-by-step procedure as a SysML Activity Diagram is shown in Appendix G. It shows the complete sequence of steps involved in the diagram and how the inputs are transformed to outputs. The SysML activity description can be simulated in the tool to check for consistency and correct modeling. It is important to highlight that correct modeling does not mean that the procedure is correct, but it can be used as a sanity check to see if all required evidence is actually generated somewhere during the procedure. Creating the procedure was done in collaboration with the Da Vinci Satellite Payload team, which provided the technical expertise and reviewed the procedures for technical correctness. The actual full functional test was conducted at the ESA location ESEC Galaxia in Belgium by members of the Da Vinci Satellite payload team. The actual setup to conduct the full functional test in the ESA cleanroom is photographed and shown in Figure 5.12.

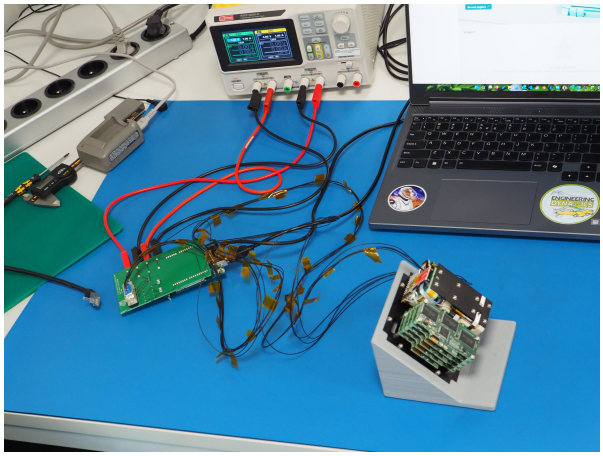


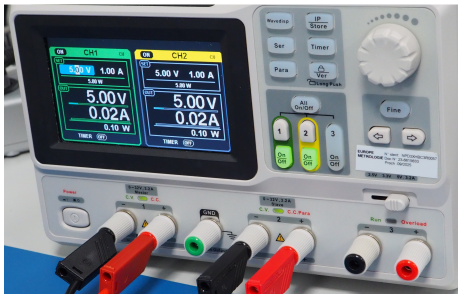
Figure 5.12: Actual setup for the full functional test in the cleanroom

For evaluating the criterion with ID Crit-31 (see Figure 5.7), certain evidence is required. This includes the noted down value of the power consumption in the procedure document, and a picture taken from the power supply showing the drawn current measurement value as additional proof. Both artifacts can be seen in Figure 5.13, where channel 1 of the power supply shows the current draw of the dice MCU.

StepID	Description	Input	Output	Remarks	Date/Time/ Signature
SID-210	Check current conspumption of the dice MCU	- 2 channel lab power supply - Camera	- Steady-state current draw of dice MCU - Picture of steady-state current draw of dice MCU from power supply		13/04/2026 15:38 ES

Crit ID	Name	Criterion Text	P/F	Evaluation Notes	Results
Crit-31	Dice MCU stand-by current draw is 17 mA +/- 3.4 mA	Steady-state current consumption of the dice MCU shall be between 13.6 mA and 20.4 mA.	P	- Read current draw measurement from lab power supply - Take picture of current draw reading and save on shared drive	0.02A

(a) Current measurement procedure extract with noted value



(b) Current measurement picture

Figure 5.13: Modeled inputs and outputs of full functional test activity

The collected evidence is postprocessed in the model using the same process as discussed in the toaster example. The evidence for the criterion with ID CRIT-31 is shown in Figure 5.14.

EvidenceStatus: planned collected outdated

#	Name	Status	Owned Properties	Traced From	RelatedRequirement
8	Picture of steady-state current draw of dice MCU from power supply	collected	power-supply-current-diceMCU.JPG status = collected	Crit-31 Dice MCU stand-by current draw is 17 mA +/- 3.4 mA	154 Dice housekeeping data provision
10	PLM full functional test as-run procedure	collected	test-procedure-as-run-Dice-FFT.pdf status = collected	Crit-33 Dice status command return value FFT Crit-34 Dice parameters at default values FFT Crit-35 Stop switches readout returns "12" when clamped FFT Crit-36 Stop switches readout returns "3" when unclamped FFT Crit-37 Stop switches readout returns "12" after clamping cycle Crit-39 Sweeper moves during unclamping Crit-40 Sweeper moves during clamping Crit-31 Dice MCU stand-by current draw is 17 mA +/- 3.4 mA Crit-32 Dice Camera stand-by current draw is 200 mA +/- 20 mA Crit-41 Check dice image quality Crit-42 Dice video quality	154 Dice housekeeping data provision 152 Capture dice 151 Roll dice 153 Take picture of dice 159 Dice picture format and quality 158 Dice video quality and frame rate
12	Steady-state current draw of dice MCU	collected	status = collected steady-state current of dice MCU : total current[ampere] = 0.02 A	Crit-31 Dice MCU stand-by current draw is 17 mA +/- 3.4 mA	154 Dice housekeeping data provision

Figure 5.14: Table with collected evidence during full functional test, postprocessed in the model

With the collected and updated evidence in the model, the criterion can be evaluated. The as-run procedure already requires the test operators to give a verdict during the procedure execution. The postprocessing task is to verify that the evaluation was done correctly and all evidence is collected to ensure the reasoning behind the evaluation is captured based on the generated raw evidence. The updated criterion is shown in Figure 5.15.

CriterionVerdict: pass fail inconclusive error TBD

#	Traced From	Id	Name	Text	Documentation	Evaluation Notes	Refines	relatedAIVSteps	Verdict	Verdict Rationale	TracedTo	RawEvidence
1	PID-A-1D Dice - Full Functional Test(context Dice - Full Functional Test Setup)	Crit-31	Dice MCU stand-by current draw is 17 mA +/- 3.4 mA	Steady-state current consumption of the dice MCU shall be between 13.6 mA and 20.4 mA.	The steady-state current draw indicates that the MCU is running correctly, which is a prerequisite for achieving housekeeping data provision	Read current draw measurement from lab power supply Take picture of current draw reading and save on shared drive	154 Dice housekeeping data provision	SID-210	pass	current consumption is measured as 20 mA.	Picture of steady-state current draw of dice MCU from power supply Steady-state current draw of dice MCU PLM full functional test as-run procedure	power-supply-current-dice MCU.JPG steady-state current of dice MCU : total current[ampere] = 0.02 A test-procedure-as-run-Dice-FFT.pdf

Figure 5.15: Updated Criteria based on collected evidence during full functional test.

The next steps in the MBSE AIV Methodology are to perform this for all collected evidence and all criteria, and then propagate the criteria evaluation up to the requirements in the VCD model view. Figure 5.16 shows the other two criteria related to the example requirement.

CriterionVerdict: pass fail inconclusive error TBD

#	Traced From	Id	Name	Text	Documentation	Evaluation Notes	Refines	relatedAIVSteps	Verdict	Verdict Rationale	TracedTo	RawEvidence
1	PID-A-1D Dice - Full Functional Test(context Dice - Full Functional Test Setup)	Crit-31	Dice MCU stand-by current draw is 17 mA +/- 3.4 mA	Steady-state current consumption of the dice MCU shall be between 13.6 mA and 20.4 mA.	The steady-state current draw indicates that the MCU is running correctly, which is a prerequisite for achieving housekeeping data provision	Read current draw measurement from lab power supply Take picture of current draw reading and save on shared drive	154 Dice housekeeping data provision	SID-210	pass	current consumption is measured as 20 mA.	Picture of steady-state current draw of dice MCU from power supply Steady-state current draw of dice MCU PLM full functional test as-run procedure	power-supply-current-dice MCU.JPG steady-state current of dice MCU : total current[ampere] = 0.02 A test-procedure-as-run-Dice-FFT.pdf
3	PID-A-1D Dice - Full Functional Test(context Dice - Full Functional Test Setup)	Crit-33	Dice status command return value FFT	Sending 0x00 - Status request command to Dice MCU returns "36 0 0 10"	Checks return value of status request.	Criterion passed if 0x00 returns "36 0 0 10" not passed if anything else is returned if i2c lines are not making a proper contact, the command 0x00 will return "8 85 8 84"	154 Dice housekeeping data provision	SID-212	pass	as-run procedure and log indicates correct return value	PLM full functional test as-run procedure Full Functional Test SW log	test-procedure-as-run-Dice-FFT.pdf DiceFFT_software_log.txt
4	PID-A-1D Dice - Full Functional Test(context Dice - Full Functional Test Setup)	Crit-34	Dice parameters at default values FFT	All parameters are set to their default values. Corresponding command and default values shall be as specified in the evaluation notes.	Checks that dice payload parameters are at default values.	Command | Name | value 0x01 | R_M1_SPEED | 255 0x02 | R_M2_SPEED | 255 0x03 | R_M1_LENGTH | 50 0x04 | R_M2_LENGTH | 20 0x05 | R_M1_POSITION | 0 (unknown position) 0x06 | R_LED_BRIGHTNESS | 255 0x07 | R_LED_STATUS | 0	154 Dice housekeeping data provision	SID-213	pass	as-run procedure and log indicates correct return value	PLM full functional test as-run procedure Full Functional Test SW log	test-procedure-as-run-Dice-FFT.pdf DiceFFT_software_log.txt

Figure 5.16: Updated criteria for the example requirement evaluated after full functional test execution

In the VCD, the example requirement about the dice housekeeping data provision can be updated now, since three of the five criteria are now evaluated as a pass. Another two criteria are related to other tests.

Phase D Compliance: Compliant Not Compliant Partially Compliant Unknown Not Applicable

#	Source	Name	Text	Verified By	vvStatement	Refined By	CriterionStatement	Phase D Compliance	Compliance Rationale	Closeout Status	Closeout Status Rationale
3	L2-PAY-FKT-050	Dice housekeeping data provision	The dice payload shall provide housekeeping telemetry data upon OBC request.	PID-A-1D Dice - Full Functional Test(context Dice - Full Functional Test Setup) PID-A-4D Dice - Reduced Functional Test(context Dice - Reduced Functional Test Setup)	Test at Subsystem level in Phase D during Verification activity PID-A-1D. Test at Subsystem level in Phase D during Verification activity	Crit-31 Dice MCU stand-by current draw is 17 mA +/- 3.4 mA Crit-33 Dice status command return value FFT Crit-34 Dice parameters at default values FFT Crit-43 Dice status command return value RFT Crit-44 Dice parameters at default values RFT	Crit-31 evaluated as pass Crit-33 evaluated as pass Crit-34 evaluated as pass Crit-43 evaluated as TBD Crit-44 evaluated as TBD	Partially Compliant	Partial compliance until reduced functional tests in TVAC chamber have been conducted.	OPEN	Open until Post-Test review with ESA.

Figure 5.17: Updated example requirement in VCD model view

This process was also performed for the packaging, transportation, and unpacking procedures of the payloads, in preparation of the transport from Delft to ESEC Galaxia.

Comparison of generated procedures

This section will compare the generated procedures against other procedures that were created within the overall project. To compare the MBSE-generated procedures with document-based procedures, a

structured checklist of questions was defined. The checklist was derived from the ECSS Test Procedure Document Requirements Definition (ECSS TPRO DRD) provided in ECSS-E-ST-10-03C [10] Annex C. The DRD defines the expected content of a test procedure, including traceability to the test specification, identification of the item under test, description of the test setup, required resources, step-by-step execution instructions, expected results, and provisions for recording the as-run procedure.

It should be noted that in the MBSE approach, the procedure is in the end "just another view of the model", that exposes certain information from the model in a way that is suitable for the specific stakeholder, in this case the AIV and PA engineers.

The purpose of the checklist is not to assess the technical correctness of the procedure itself, but to evaluate whether the procedure document provides the information needed to execute and review the activity in a structured, repeatable way. The questions therefore focus on document qualities that are relevant for test execution and conformance to the ECSS TPRO DRD. In particular, the questions assess whether information is explicit, complete, consistent, traceable, and usable by the operator or reviewer.

The following seven assessment questions were used:

1. **Traceability to requirements from VCD:** Does the procedure provide explicit traceability to the relevant test specification and requirements from a VCD or similar source?
2. **Item under test or assembly configuration:** Does the procedure clearly identify the item under test or item under assembly, including its configuration, relevant configuration list, deviations, and software version where applicable?
3. **Test or assembly setup:** Does the procedure describe the required test or assembly setup sufficiently to allow the activity to be prepared and repeated?
4. **Required resources, tools, and software:** Does the procedure identify all required GSE, tools, test scripts, test software, databases, and their versions where applicable?
5. **Step-by-step execution instructions:** Does the procedure provide detailed step-by-step instructions on how to perform the test?
6. **Expected results and evaluation criteria:** Does the procedure describe expected results, including tolerances and pass/fail criteria in relevant procedure steps?
7. **As-run recording:** Does the procedure support as-run recording, including structured placeholders for execution results, procedure variations with justification, and anomalies?

These questions cover the core information categories required by the ECSS TPRO DRD. The first question is derived from the DRD requirement that the procedure provides a mapping to the test specification and traceability towards the test requirements. The second question reflects the requirement to describe the item under test configuration, including references to configuration lists, deviations from the specified standard, and software version where applicable. The third and fourth questions address the DRD expectations that the procedure describes the test setup and identifies the required GSE, tools, test scripts, software, databases, instrumentation, and other resources needed for the activity. The fifth and sixth questions are derived from the requirement for detailed step-by-step instructions, including expected results, tolerances, and pass/fail criteria. Finally, the seventh question reflects the DRD statement that the procedure is filled in during execution and becomes the as-run procedure, as well as the requirement to provide placeholders for procedure variations and anomalies.

Each assessment question was evaluated using an integer rating scale from 0 to 3. A score of 0 indicates that the criterion is not addressed in the procedure. A score of 1 indicates that the criterion is only weakly addressed, for example through an implicit, incomplete, or unclear mention. A score of 2 indicates that the criterion is sufficiently addressed and usable for execution, but with remaining limitations or omissions. A score of 3 indicates that the criterion is addressed explicitly, completely, consistently, and in a way that directly supports execution, review, and repeatability of the activity. Each assigned value is explained with a short statement.

The checklist was intentionally limited to these seven questions to support a more focused comparison. Several detailed DRD requirements are consolidated into broader assessment questions. For example, the identification of GSE, tools, scripts, software, and databases was treated as one input-resources-related

question, while expected results, tolerances, and pass/fail criteria were assessed together because they determine whether the procedure supports objective evaluation of the activity outcome.

Each question can be answered through direct review of the procedure document. The evaluation therefore provides a basis for comparing document-based and MBSE-generated procedures. For the MBSE-generated procedures, the assessment assumes that the MBSE model is modeled following the previously presented modeling workflow.

The document-based procedures used for the comparison are existing AIV procedure documents from the student satellite project. For confidentiality and privacy reasons, the procedures are not reproduced in this thesis and are kept anonymized in the comparison as Procedure 1, Procedure 2, and Procedure 3.

Table 5.6: Evaluation of procedure documents against assessment questions

Assessment question	**Proc. 1**	**Proc. 2**	**Proc. 3**	**MBSE Proc.**
Traceability to requirements from VCD	**0** – Test requirements are not connected to, or present in, the project VCD.	**0** – No requirements cited.	**0** – No requirements cited.	**3** – Requirements are added automatically from the model; consistency is guaranteed if the model is correct.
Item under test or assembly configuration	**2** – HW and SW are clearly stated, but versions are not mentioned.	**3** – HW and SW are clearly stated, including versions.	**3** – HW and SW are clearly stated, including versions.	**3** – HW and SW are clearly stated, including versions.
Test or assembly setup	**2** – Setup is described in the procedure, including photos, but lacks an overview diagram or consolidated setup summary.	**2** – Setup is described in the procedure, including photos, but lacks an overview diagram or consolidated setup summary.	**0** – Not included.	**2** – Setup is described in BDD, IBD, and the procedure, but only SysML diagrams are used; no real pictures are included.
Required resources, tools, and software	**1** – HW setup is complete, but SW is missing.	**0** – Not included.	**0** – Not included.	**2** – Complete list of required inputs; completeness is guaranteed if the model is correct, but no pictures are included.
Step-by-step execution instructions	**3** – Detailed instructions are provided, supported by guiding pictures.	**3** – Detailed instructions are provided, supported by guiding pictures.	**3** – Detailed instructions are provided, supported by guiding pictures.	**2** – Detailed instructions are provided, but without guiding pictures.
Expected results and evaluation criteria	**1** – Expected results are stated for relevant procedure steps, but predefined pass/fail criteria are not directly linked to the steps.	**2** – Pass/fail criteria are defined for relevant procedure steps.	**2** – Pass/fail criteria are defined for relevant procedure steps.	**3** – Pass/fail criteria are defined for relevant procedure steps, including expected results and evaluation instructions. Traceability between requirements, criteria, and steps is guaranteed if the model is correct.
As-run recording	**3** – Supported.	**3** – Supported.	**3** – Supported.	**3** – Supported.

The comparison shows that the main advantage of the MBSE-generated procedure lies in its explicit traceability to requirements through the centrally maintained VCD-like model information. While the document-based procedures either do not cite requirements or do not connect them to the project VCD, the MBSE-generated procedure can automatically include the relevant requirements from the model. Further advantages are observed in the identification of required resources, tools, and software, where model-based generation supports completeness and consistency of the listed inputs. The MBSE-generated procedure also shows upsides with respect to expected results and evaluation criteria,

since pass/fail criteria, expected results, and evaluation instructions can be connected to the relevant procedure steps and traced back to requirements and verification criteria. A limitation is observed in the step-by-step execution instructions: while the generated procedure provides the required procedural structure, it lacks supporting pictures unless the document is manually post-processed after generation. Therefore, the model-based approach provides the strongest benefits for traceability, consistency, and structured verification logic, while manually prepared procedures can still offer advantages in operator guidance when they include contextual pictures or explanatory visual material that is not included automatically from the model side.

Evaluation of the MBSE AIV Methodology for the case study

Table 5.7 shows the different model complexities for the two cases and the number of AIV Activities and Requirements for a full environmental test campaign of a CubeSat, taken from the case study. The number for the full AIV campaign was derived from the full number of requirements that the project listed with regards to the payloads, and the modeled number of verification activities that resulted in modeling the flow of the whole environmental test campaign, without detailing any further the procedures, inputs and outputs, or criteria for the other activities not further modeled in the Da Vinci Satellite Payload Model for transport and functional testing.

Table 5.7: Overview of modeled AIV elements per project or case study

Project / Case Study	AIV Activities	Require-ments	AIV Criteria	Evidence	Procedure Steps
Toaster Example Model	1	4	4	9	12
Da Vinci Satellite Payload Model (Transport and Functional Tests)	7	13	40	38	175
Full CubeSat Environmental Test Campaign	24	66	–	–	–

The table indicates that the payload model is substantially larger and more complex than the initial toaster model. The application on this larger modeling scope showed that the approach is capable of handling an increased complexity, that goes significantly further than the first example model. Still, full validation of a real-life AIV campaign cannot be claimed since the scope of application was limited in complexity to transportation and functional testing of payloads and did not reach the number of requirements or AIV activities expected over the life cycle of a real project.

Observed Benefits

The main benefit observed when applying the MBSE AIV methodology in this case study is an increased level of rigor and consistency in AIV planning. Each verification activity is explicitly linked to requirements, while verification criteria and the corresponding evidence are systematically defined. This reduces ambiguity in what is being verified and how, and lowers the risk of missing verification aspects compared to a document-based approach. For example, the model ensured that all required evidence for each criterion was explicitly defined and generated during the procedure, allowing completeness of the verification activity to be checked prior to execution. This is ensured by the modeling process itself, and by the possibility to simulate the modeled AIV activity using the simulation capabilities of the SysML and the selected modeling tool.

A key contribution of the methodology is the explicit refinement of requirements into activity-specific, evaluable criteria. This introduces a structured layer between requirements and raw test results, where both the expected evidence and the evaluation rules are defined in advance. As a result, the verification process becomes more objective and reproducible. This is reflected in the procedure comparison, which shows that the included requirements are more consistently traced to procedure steps and pass/fail criteria. The members of the project team that were part of the case study reflected this as a major benefit of the generated procedures.

Furthermore, the methodology enables the creation of consistent traceability across requirements, activities, criteria, and verification evidence. This allows for verification results to be linked and assessed

directly within the model. In cases of non-compliance, the impact on requirements is immediately visible, supporting efficient impact assessment and change management.

Experienced limitations

The application of the MBSE AIV methodology in this case study introduced a higher initial modeling effort compared to a traditional document-based approach. Setting up the model structure, defining stereotypes, and tailoring the methodology required additional upfront investment, which may not always be feasible in time-constrained projects.

Furthermore, the effective use of the methodology is dependent on familiarity with the MBSE toolchain and its customization capabilities, such as report generation using VTL templates. This introduces a learning curve for both systems and domain engineers and may limit adoption in teams with limited MBSE experience.

It was found that the model-based approach does not incentivize the user to include pictures during procedure creation. This can be addressed either by post-processing the procedures after generation and including the pictures then, or be implemented as a minor addition to the procedure template to allow referencing pictures from the model directly.

The validation of the methodology in this case study is limited in scope, as it was applied only to selected AIV activities, specifically transportation and functional testing. Other important aspects of the AIV campaign, such as environmental testing (e.g., vibration and thermal vacuum), were not included in the modeling scope, limiting the generalizability of the results.

Finally, maintaining consistency between the model and generated evidence requires discipline and a change in mindset to put the model at the center of information collection and processing. Collected evidence and changes in procedures or requirements must be carefully propagated within the model to ensure that the digital thread remains accurate and up to date.

5.3.3. Validation against Use Cases

Use Case 1 - Define and Prepare Verification Activities

Defining and modeling verification activities is required to demonstrate compliance of system requirements. This includes mapping verification activities to the requirements under verification, defining verification criteria, specifying test setups, personnel, performance parameters, required inputs (e.g., tools, GSE, consumables), and expected outputs (e.g., AIV artefacts and verification evidence). Verification procedures are then constructed, exported, and finalized for approval and execution.

The toaster example showcased the possibility to follow the methodology to perform all the steps included in the description of the Use Case. This Use Case of the MBSE AIV Methodology is fulfilled.

Use Case 2 - Maintain Digital Thread from Requirements to Verification Evidence

The core objective of this use case is maintaining end-to-end traceability between requirements, verification activities, verification criteria, and verification evidence. For each requirement, verification activities and derived verification criteria are linked and their compliance status tracked. Verification criteria are associated with evidence produced during AIV activities and assigned verdicts. These verdicts are used to update requirement compliance and closeout status throughout the verification process and during review milestones.

The digital thread was showcased in the toaster example model, demonstrating end-to-end traceability from the first rough AIV planning activities to post processing of AIV Evidence generated during test activities. The example showed how documents and generated artifacts can be attached to model elements. This increases explainability of verdicts for AIV Criteria and allows reviewers to trace back the status of requirements to the actual evidence that was evaluated in the first place.

Use Case 3 - Track overall verification status

Provide an integrated overview of the verification status of system requirements and verification activities. This includes monitoring requirement close-out status, qualification status, and the progress of verification activities, enabling stakeholders to assess verification completeness and readiness for project reviews.

This Use Case is achieved mainly with the VCD-like model views showcased in the report. Each requirement has the needed parameters to track close-out status and compliance status. Additionally,

the AIV Criteria allow to translate the requirement statement into explicit conditions that need to be checked per AIV Activity. Over the course of a project life cycle, the amount of criteria evaluated to a "pass" verdict can be another indicator for the degree to which a system is known to be compliant to its requirements.

Use Case 4 - Define model philosophy consistent with AIV Plan
Keep an overview over the adopted model philosophy and differences in configuration between different models of the same hardware element. A model philosophy that is in line with the overall AIV Plan ensures that the required hardware models are available when needed to be integrated into higher level assemblies or test setups.

Both the model philosophy and the AIV Flow can be modeled using the methodology, and the hardware models can be set as inputs and outputs for events and activities within the AIV Flow. Modeling these relations can reveal dependencies between different activities, but the process of modeling an extensive AIV Flow is time consuming. The reason for this is mainly because the selected modeling tool is not optimized to capture project scheduling and event management.

Use Case 5 - Prepare for Qualification Review
Support the preparation of project review artefacts by generating documentation and evidence packages from the MBSE model. This includes compiling verification results, verification evidence, and traceability information required to demonstrate readiness for qualification reviews.

The modeling tool allows users to export any tables into Excel. These excel sheets can then be used as central deliverables for a classical, document-based review process. Another option is to transition to a model-based review process, where reviewers get access to certain parts of the model and leave review comments in the central model instead of in separate documents. Even though full export of a VCD was not shown in the report, and not performed in the frame of the thesis, there are lots of resources online that allow to extend the MBSE AIV Methodology with such features in the future.

Use Case 6 - Handle Anomalies and NCRs
Support the documentation and tracking of anomalies and non-conformance reports (NCRs) identified during AIV activities. While anomaly handling is not the primary focus of the methodology, the model can capture links between anomalies, affected system elements, and verification activities.

This Use Case is fulfilled, as demonstrated for the toaster example in the report. The ECSS mapping and the example show that NCRs can be captured in the model and connected to relevant model elements.

5.3.4. FEMMP Evaluation

The Framework for the Evaluation of MBSE Methodologies and Processes (FEMMP) [65], developed by Weilkiens et al., provides a framework for evaluating and comparing MBSE methodologies based on a structured set of criteria.

The usefulness of an MBSE methodology does not only depend on whether its concepts can be represented in a model. It also depends on whether the methodology can be applied coherently by practitioners, integrated into engineering processes, supported by suitable tool functionality, and used to produce useful engineering outputs. Rather than focusing only on the formal correctness of a modeling approach, FEMMP considers the broader applicability of a methodology in an engineering project and organizational context. This makes it suitable for assessing whether a methodology is understandable, usable, tailorable, sufficiently supported, and practically useful in realistic project settings.

The FEMMP framework evaluates MBSE methodologies across several categories and aspects. The categories are: Adoption, Basics, Efficiency, UX, Practicality, and Support. The aspects are: General, Information, Language, Model, Process, and Tool.

FEMMP also distinguishes between different types of evaluation responses. Some criteria are assessed using a qualitative scale, ranging from full compliance to not addressed, while other criteria are evaluated through yes/no answers or list-based responses, depending on the nature of the criterion.

In this thesis, FEMMP is used as a lightweight and practitioner-oriented evaluation framework. The intention is not to objectively rank the developed methodology against other MBSE methodologies, but to provide a transparent assessment of its strengths, limitations, and areas that remain outside

the current scope. This is particularly relevant because the methodology developed in this thesis is tailored specifically to post-CDR Assembly, Integration, and Verification activities in space projects. Applying FEMMP therefore helps to position the methodology relative to broader expectations for MBSE methods, while still acknowledging its focused scope.

The use of FEMMP is justified because the developed methodology includes several elements typically expected from an MBSE methodology: an ontology defining the relevant AIV concepts and relationships, a process description for applying these concepts, model views and viewpoints for structuring the information, and a tool-based implementation in SysML. FEMMP provides a suitable structure for evaluating these elements together, rather than assessing them in isolation. It also supports the identification of criteria that are well addressed, such as information capture, traceability, abstraction, tailoring, and procedure generation, as well as criteria that were not evaluated in detail, such as broader tool usability, training support, information security, and integration with specialty engineering models.

For this thesis, the FEMMP evaluation is therefore an assessment of the methodology's suitability for model-based AIV support. The evaluation focuses on the extent to which the methodology can support post-CDR AIV planning, execution, verification control, and reporting within a model-based environment. It does not constitute a complete assessment of the underlying modeling tool or a full industrial maturity assessment. Instead, it provides a structured overview of how well the developed methodology satisfies relevant expectations for an MBSE methodology and where further development or validation would be required.

The detailed results of the FEMMP evaluation can be found in Appendix F.

The FEMMP evaluation indicates that the developed methodology provides strong support for the model-based representation and management of AIV information. In particular, the methodology performs well in criteria related to information capture, abstraction, traceability, process consistency, tailoring, and reuse. The central model captures the information generated throughout the AIV process and allows requirements, verification activities, AIV criteria, evidence, setups, and procedures to be connected within a coherent digital thread. This supports one of the main objectives of the methodology: enabling structured and traceable AIV planning, execution, and verification control in post-CDR project phases.

A major strength identified in the evaluation is the explicit definition of the methodology through an ontology, process description, model views, and tool implementation. The presence of an ontology and its implementation supports a consistent interpretation of the modeled information. The methodology also supports different levels of abstraction, ranging from high-level AIV planning to detailed procedure definition and post-processing of collected evidence. This is particularly relevant for AIV, where high-level verification planning must remain connected to concrete test procedures and collected evidence.

The evaluation also shows that the methodology provides practical benefits in terms of the consistency of documentation. The implemented model views, such as the Verification Control Document view, support the structured presentation and review of model content. In addition, automated procedure generation reduces administrative effort once the relevant model information has been created. These capabilities demonstrate how the methodology can support document generation and reporting while maintaining the model as the authoritative source of information.

The methodology was also assessed positively with respect to tailoring and scalability. The dedicated process model and metamodel implementation make the methodology adaptable to specific project needs, while the case study application showed that it can be applied in a realistic space-system AIV context without major additional hurdles. Furthermore, the explicit modeling of AIV model elements, such as qualification models and flight models, supports the representation of product variants within the verification process.

However, the FEMMP evaluation also highlights several limitations. The methodology assumes prior knowledge of systems engineering, MBSE, and SysML, making it unsuitable for beginners without additional training. While experienced users can apply the methodology with limited additional effort, tailoring advanced aspects such as report templates or tool customizations requires more detailed knowledge of SysML, Velocity Template Language, and the modeling tool environment. The evaluation

also identifies limitations in tool connectivity: although an open API exists, using it is not straightforward and can create a barrier for integration with external tools. Some tools also have custom integrations, like MATLAB.

Several FEMMP criteria were not assessed in detail, including creativity support, information security, general tool intuitiveness, user interface readability, training support, and integration with specialty engineering models. These criteria were outside the main scope of the thesis or could not be evaluated sufficiently based on the performed case study. As a result, the FEMMP evaluation should be interpreted as an assessment of the methodology's suitability for model-based AIV support, rather than as a complete assessment of the underlying modeling tool or its broader organizational adoption.

Overall, the FEMMP evaluation supports the conclusion that the developed methodology is well suited for supporting post-CDR AIV activities in a model-based environment. Its main value lies in improving traceability, information consistency and control, and procedure generation. At the same time, the evaluation shows that successful application depends on the availability of MBSE expertise and that further work is required to assess broader usability, tool integration, training support, and the rollout and application of the methodology across a complete life cycle.

5.4. Chapter Summary

This chapter presents the verification and validation of the developed MBSE AIV methodology. The verification activities assess whether the methodology is built correctly, while the validation activities assess whether the methodology is suitable and useful for its intended purpose. Together, these activities evaluate the methodology from the perspectives of internal consistency, requirements compliance, standards alignment, practical applicability, and observed benefits and limitations.

The verification of the methodology is performed from four complementary perspectives. First, the internal consistency of the methodology is assessed by mapping the defined AIV ontology to its SysML v1 implementation. This shows that the ontology elements, attributes, and relations are mostly implemented consistently through stereotypes, properties, and SysML relationships. Partial deviations are highlighted because not all ontology relations can be mapped one-to-one to SysML constructs without additional textual explanation. Second, the methodology is evaluated against the previously defined methodology requirements. The majority of requirements are assessed as fulfilled, while partial deviations remain for interface consistency checking and the direct mapping to a complete ECSS test specification document. Third, the methodology is mapped to the ISO/IEC/IEEE 24641 process *Perform system verification and validation*. This shows that the methodology supports the preparation, execution, and result management of model-based verification and validation activities, although some advanced tool capabilities and configuration management integrations remain outside the scope. Fourth, the methodology is assessed against selected ECSS DRDs, including the Verification Plan, Verification Control Document, Test Specification, Test Procedure, and NCR Status List. This mapping shows that most required AIV information can be represented in the model and can therefore support the generation of deliverables aligned with ECSS.

The validation of the methodology is performed through application to complementary case studies. The toaster example demonstrates the complete methodology in a controlled and low-complexity setting and was already presented in a previous chapter. The Da Vinci Satellite CubeSat payload case study demonstrates the application of the methodology in a more realistic space engineering context. In this case, the methodology is applied to payload transportation procedures and functional testing of the Dice payload, including the modeling of AIV activities, requirements, criteria, evidence, setups, inputs, outputs, and generated procedures.

The case study application shows that the methodology scales beyond the simplified example and can support realistic subsystem-level AIV work. The Da Vinci Satellite payload model includes a substantially larger set of AIV activities, requirements, criteria, evidence elements, and procedure steps than the toaster example. The application demonstrates that the methodology improves rigor and consistency in AIV planning by explicitly connecting requirements, AIV activities, criteria, and evidence. It also supports the creation of a digital thread from requirements to verification evidence and provides structured model views for verification control and review preparation.

The validation also identifies limitations. Applying the methodology introduces additional upfront modeling effort compared to a traditional document-based approach. Effective use requires familiarity with MBSE, SysML, Cameo, and report generation mechanisms such as Velocity Template Language. The case study scope is also limited to selected AIV activities, mainly transportation and functional testing, while a complete environmental test campaign is not fully modeled in detail. In addition, maintaining the digital thread requires discipline, since collected evidence, procedure changes, and requirement updates must be consistently propagated in the model.

Finally, the methodology is evaluated using the FEMMP framework to assess its suitability as an MBSE methodology. The evaluation indicates strong support for information capture, abstraction, traceability, process consistency, tailoring, reuse, verification control, and procedure generation. At the same time, it highlights areas that require further work, including training support, broader usability assessment, tool integration, information security, and application across a full project life cycle. Overall, the chapter shows that the developed MBSE AIV methodology is internally consistent, largely requirements-compliant, aligned with relevant ISO and ECSS expectations, and practically applicable for selected post-CDR AIV activities in a realistic CubeSat payload context.

6

Discussion

6.1. Answering the Research Questions

Table 6.1 provides an overview of the research questions as stated at the beginning of the thesis report.

Table 6.1: Research Questions

ID	Research Topic / Research Question
RO	Develop an MBSE Methodology for Systems Engineering and Assembly, Integration and Verification in post-CDR project stages and implement it in part of a project.
RQ-1	How and to what extent do current MBSE approaches in literature address the post-CDR stage of projects?
HYP-1	Most MBSE approaches described in literature are focused on SE activities until CDR, while most of the SE concepts for post-CDR described in relevant SE standards and literature, such as AIV planning, verification control, and test specifications, are not covered in MBSE Methodologies found in literature.
RQ-2	How can an MBSE methodology be defined and implemented to effectively support Systems Engineering and AIV activities in post-CDR project stages?
HYP-2	It is possible to define and implement an MBSE methodology tailored to post-CDR project needs that supports Systems Engineering and AIV activities by focusing on requirements traceability, explicit modeling of verification activities, collected evidence, setups, and verification control.
RQ-3	What benefits and limitations can be observed when applying MBSE to the post-CDR phase of a project such as the Da Vinci Satellite?
HYP-3	Applying the MBSE approach results in improved consistency and traceability of requirements to other information, better planning and overview over the AIV process, and increased stakeholder confidence in the system definition and verification control, compared to a conventional document-centric approach.

6.1.1. RQ-1 evaluation

RQ-1 asked how and to what extent current MBSE approaches in the literature address the post-CDR stage of projects. Based on the literature review, it can be concluded that existing MBSE approaches address selected aspects of post-CDR engineering work, but do not yet provide comprehensive methodological support for the practical work of AIV planning, execution, verification control, and evidence management in later life cycle phases. In particular, full methodological support constitutes the crucial gap, since it requires complete documentation of the ontology/metamodel, a process definition, modeling guidelines, and implementation in a tool environment that aligns with space-domain terminology.

The reviewed approaches show that verification-related concepts can be represented in MBSE. OOSEM and the MagicGrid V&V extension provide concrete modeling patterns for test contexts, test cases, and traceability to requirements. The V&V strategy pattern contributes useful concepts for reasoning about verification activities, verification criteria, and evidence, but does not provide explicit modeling support for applying these concepts in an MBSE environment. The ESA MBSE Solution goes further

by introducing verification scenarios and verification activities as explicit model elements within a space systems MBSE context. The recent work by Mulholland et al. [49] provides one of the more complete examples, since it includes a metamodel, a SysML implementation, and process guidance for performing V&V in a model-based environment.

However, the reviewed approaches are limited with respect to the specific needs of post-CDR satellite projects. Most approaches focus on individual tests, verification scenarios, or early V&V reasoning, rather than supporting the full AIV information structure needed during implementation, integration, verification, and transition. In particular, aspects such as verification planning and control, explicit management of verification evidence, traceability to test setups and procedure steps, non-conformance handling, and alignment with space-domain documentation practices are only partially covered or are not addressed in sufficient detail. This confirms the observation made in subsection 2.4.4 that existing MBSE approaches do not yet cover several critical aspects of the AIV discipline in an integrated way. This assessment is also consistent with a parallel literature review by Buruso et al. [4], which similarly identifies limitations in the coverage of later life cycle verification and validation concerns.

6.1.2. RQ-2 evaluation

The presented methodology in section 4.2 covers the needs of SE and AIV for later project stages. For the AIV Planning, Execution, Postprocessing and Reporting, this was demonstrated on two case studies, a small toaster example model, and the Da Vinci Satellite Payloads. The methodology is demonstrated to enable end-to-end traceability and explicit modeling of AIV elements, like verification activities, setups, gathered evidence and the rules by which the evidence is used to evaluate test criteria.

The implementation on the DVS payloads functional tests and transportation procedures demonstrates feasibility in a real project context. The mapping to DRDs of the ECSS standards supports the claim that the model covers relevant information for the AIV process, while the mapping to the ISO 24641 standard shows the degree of conformance of the MBSE AIV Methodology to the expected methods and tool capabilities employed for verification and validation efforts conducted in a model-based environment. Furthermore, it is shown that the development of the methodology follows the formal structures for an MBSE approach as defined in the same ISO standard.

Overall, the results indicate that the proposed methodology provides a coherent and practically applicable framework for supporting Systems Engineering and AIV activities in post-CDR project stages. By combining a well-defined ontology, explicit modeling constructs, and alignment with established standards, the approach enables structured AIV planning and execution within a model-based environment. While the validation is limited to selected case studies, the demonstrated feasibility, traceability, and standards compliance support the conclusion that the methodology provides a suitable and effective response to RQ-2. Further applications in larger-scale projects would allow for a more comprehensive assessment of its scalability, robustness, and integration within industrial workflows.

6.1.3. RQ-3 evaluation

The application of the proposed MBSE AIV methodology to the Da Vinci Satellite (DVS) case study provides insight into the benefits and limitations of applying MBSE in post-CDR project stages. The evaluation focuses on selected AIV processes and compares the model-based approach to the previously used document-centric practices.

The results indicate clear improvements in terms of traceability, consistency, and transparency. The explicit modeling of AIV elements, such as activities, criteria, setups, and evidence, enables a structured linkage between requirements and verification artifacts. This improves the ability to track verification status, assess completeness, and ensure that all requirements are addressed through well-defined verification activities. In addition, the model provides a centralized representation of AIV-related information, reducing fragmentation across documents and supporting a more consistent interpretation of the system definition.

Furthermore, the methodology could enhance the planning and control of AIV processes. The structuring of activities into events and flows allows dependencies between verification activities to be made explicit, supporting a clearer overview of the AIV process. This could contribute to improved coordination and facilitates communication among stakeholders, as the model serves as a shared reference for both

technical and programmatic aspects of AIV. For a final verdict on this aspect, an evaluation on a larger scope is needed.

However, the evaluation also highlights several limitations. The demonstrated benefits are primarily qualitative and based on a limited application scope within the DVS project. A quantitative assessment of efficiency gains, error reduction, or time savings is not performed. In addition, the effort required to set up and maintain the model is non-negligible, especially in the absence of mature tool support and standardized modeling practices for post-CDR activities. This may pose a barrier to adoption in industrial contexts.

Looking forward, several areas for improvement can be identified. The integration of simulation and execution capabilities would strengthen the support for model-based verification and enable more direct use of the model during test execution. In addition, extending the methodology to better support validation activities and stakeholder-level evaluation would provide a more complete coverage of the V&V domain. Finally, transitioning the approach to SysML v2 and improving integration with existing toolchains could enhance usability, interoperability, and long-term applicability.

Overall, the results support HYP-3 to a large extent. The application demonstrates that the MBSE approach can improve traceability, consistency, and transparency of AIV processes compared to a document-centric approach. At the same time, the limitations identified indicate that further work is required to fully assess and realize the potential benefits of MBSE in post-CDR project stages.

6.2. Key Results

The developed MBSE AIV methodology demonstrates that model-based approaches can be effectively applied in post-CDR project phases, particularly for Assembly, Integration, and Verification activities. The methodology enables a structured representation of AIV-related information and integrates it within a consistent modeling framework. This extends the application of MBSE beyond its common emphasis on early life cycle architecture definition and shows how model-based methods can support later life cycle activities.

One of the key results is the establishment of end-to-end traceability between requirements, AIV activities, AIV criteria, and AIV evidence. This digital thread allows the verification logic to be followed from a requirement to the activity that verifies it, the criterion that defines how it is assessed, and the evidence that supports the final verdict. As a result, the methodology provides a structured basis for tracking verification status and understanding how compliance with requirements is established.

Another important result is the increased consistency of AIV-related information. By centralizing requirements, activities, criteria, evidence, setups, procedure information, and verification status in the model, inconsistencies between procedure documents, verification control information, and requirement databases can be reduced. Updates to model elements can be propagated more systematically across related artifacts, supporting a more coherent AIV information environment.

Furthermore, this thesis provides a structured AIV ontology and an example implementation of the concepts in SysML v1. The ontology introduces a conceptual distinction between requirements and AIV criteria. Requirements define what the system shall satisfy, while criteria translate these requirements into evaluable statements within the scope of specific AIV activities. This distinction supports a more precise representation of verification logic, because the model captures not only that a requirement is verified, but also how its satisfaction is assessed.

The methodology further supports AIV planning by explicitly modeling AIV activities, their dependencies, required inputs and outputs, required setups, and expected evidence. This improves transparency of the AIV process and supports better planning and coordination of verification activities. By representing dependencies and sequencing between activities, the model can help to identify prerequisite activities, required configurations, and potential bottlenecks within the AIV flow.

A modeling process and model-based workflow are proposed to support the use of MBSE in later life cycle phases. The workflow covers rough AIV planning, detailed planning, execution support, evidence collection, assessment, and reporting. This demonstrates how AIV information can be incrementally refined from high-level verification planning toward detailed procedure and evidence management.

Finally, the thesis demonstrates the alignment of the methodology with ECSS standards and terminology. In particular, the methodology shows how model content can support information typically captured in ECSS-related work products such as verification plans, verification control information, test specifications, test procedures, and non-conformance records. In addition, the methodology is evaluated with respect to ISO/IEC/IEEE 24641, showing its relevance as a structured model-based systems and software engineering approach.

6.3. Benefits

Compared to traditional document-based AIV approaches, the MBSE AIV methodology provides several advantages. The most notable benefit is improved traceability between requirements, AIV activities, criteria, evidence, and verification outcomes. This preserves the line of argumentation used to demonstrate compliance: requirements are refined into evaluable criteria, criteria are assessed using explicit evidence, and the resulting verdicts support the verification status of the requirements. This makes the verification rationale more explicit and easier to inspect.

A related benefit is improved information consistency. In a document-centric approach, requirements, procedures, verification control tables, evidence records, and non-conformance information are often maintained in separate documents. This increases the risk of duplicated, outdated, or inconsistent information. By representing AIV-related information in a shared model, the methodology provides a more consistent source of truth from which different views and documents can be generated.

This can be especially beneficial for student teams and other project environments with high personnel turnover. In such contexts, project knowledge is often distributed across documents, informal communication, and individual experience. A model-based representation can help preserve the reasoning behind verification decisions, making it easier for new team members to understand which requirements have been verified, which evidence supports the assessment, and which activities or issues remain open.

Furthermore, the methodology supports a more structured AIV planning process. Dependencies between activities are explicitly modeled, making it easier to identify prerequisites, bottlenecks, missing deliverables, and impacts of delays. The methodology also supports incremental refinement, allowing AIV information to evolve from rough planning in earlier phases toward detailed activity, setup, procedure, and evidence definitions in later phases.

The methodology can enable improved communication between systems engineering, AIV engineers, and other project stakeholders. The model provides a shared representation of requirements, activities, setups, evidence, and verification status. Provided that stakeholders are familiar with the modeling approach, this can improve review readiness and make verification information more accessible during technical discussions and formal reviews.

The methodology also enables reuse of AIV elements across projects. Activities, setups, procedure structures, criteria patterns, and reporting templates could be reused or adapted for similar verification campaigns. This is particularly relevant for recurring activities such as functional testing, environmental testing, or transport and handling verification, where similar verification logic and setups may appear across multiple projects.

A practical benefit is the ability to generate procedure documentation directly from the model. This reduces manual effort and helps ensure that documents remain consistent with the underlying AIV process definition.

Overall, the methodology supports a shift from a document-centric to a model-centric AIV approach, while maintaining compatibility with established ECSS standards and practices. It therefore provides a practical step toward digital continuity in later life cycle phases, where verification evidence, status, and decisions must remain traceable and auditable.

6.4. Limitations

Despite the presented benefits, several limitations need to be acknowledged. First, the development and application of the methodology are not considering features such as advanced simulation capabilities, automated data exchange with external tools, and integration with test execution environments. These

capabilities could likely be built on top of the existing conceptual foundation, but were outside the scope of this thesis.

Second, the presented validation of the methodology is limited. While the methodology was demonstrated using a simplified toaster example and the Da Vinci Satellite payload case study, it has not been applied across a complete large-scale industrial AIV campaign. Therefore, conclusions regarding scalability, robustness, long-term maintainability, and organizational adoption should be further investigated. Additional validation in larger and more complex project environments would be interesting.

A further limitation is the required initial modeling effort and MBSE expertise. The methodology relies on users being able to create, maintain, and interpret structured SysML models. This introduces a learning curve and may create adoption barriers in teams that are used to document-based AIV practices. The benefits of improved traceability and consistency therefore need to be considered against the effort required to establish and maintain the model as the central artifact in the workflow.

The application in the case study showed that the model-based working environment does not incentivize the modeler to add reference pictures or drawings into a test procedure. Such figures are considered very helpful for practical execution of the procedure. Even though adding pictures is possible in the model or as a post-processing step in the procedure document after generation, this is not ideal.

Another limitation is the reliance on a specific modeling language and tool environment, namely SysML v1 and Cameo Systems Modeler. While this environment provides useful capabilities for profile customization, traceability views, validation rules, tables, and document generation, it also constrains the immediate adoptability of the methodology in organizations that use different MBSE tools. Applying the methodology in other environments would require a mapping and tailoring of the ontology and workflow to the corresponding tool and language concepts.

In addition, the methodology has limited integration with external AIV or project management tools. This is particularly relevant for schedule management, resource planning, and automated test execution, where dedicated tools may be more suitable than SysML models. A promising approach is therefore not to model all project management and execution details directly in SysML, but to investigate tool integration strategies in which the model provides the AIV logic and traceability while specialized tools manage schedules, configurations, and test data.

A quantitative evaluation of benefits such as cost reduction, schedule improvement, defect reduction, or review efficiency could provide further insights into the benefits of a model-based approach compared to a document-based approach. The presented evaluation mainly demonstrates feasibility, consistency, traceability, and conceptual coverage. As a result, claims about practical efficiency gains should be treated as expected benefits rather than fully demonstrated outcomes.

Finally, the methodology focuses primarily on verification and AIV execution. System-level validation against stakeholder needs, operational validation, and later life cycle tasks are only partially addressed. The methodology therefore supports an important subset of the later life cycle digital thread, but does not constitute a complete end-to-end MBSE methodology for the full system life cycle after CDR.

6.5. Further Research Opportunities

A clear and, in the long term, necessary research opportunity is the translation of the methodology to SysML v2. SysML v2 provides improved language semantics, more formal model interaction capabilities, and stronger potential for integration with modern digital engineering environments. Translating the proposed AIV ontology to SysML v2 would allow the concepts developed in this thesis to be evaluated in a more semantically precise and tool-interoperable modeling environment.

Another important research direction is the connection of the post-CDR AIV methodology with early life cycle MBSE approaches. The methodology developed in this thesis focuses on later life cycle phases, but AIV information is not created only after CDR. Verification logic, test philosophy, model-based analyses, stakeholder needs, system requirements, and architectural decisions are established progressively throughout the project life cycle. A case study that applies the methodology from the beginning can demonstrate the gradual transition from rough AIV planning only to a full application of all process steps presented in the methodology. The methodology should also be applied and evaluated in

larger-scale industrial case studies. Such studies would make it possible to assess scalability, robustness, maintainability, and practical usefulness in realistic project conditions. They would also provide a basis for quantitative evaluation of benefits, such as reductions in documentation effort, improvements in review readiness, faster impact analysis, fewer inconsistencies, or improved verification close-out.

The methodology could also be extended with stronger support for system-level validation and stakeholder needs. While this thesis focuses primarily on verification and AIV execution, future work could investigate how validation scenarios, operational needs, mission objectives, and stakeholder expectations can be linked explicitly in the AIV ontology. This would support a more complete digital thread from stakeholder need to verification and validation evidence.

A further research opportunity is the integration of simulation and model-based verification capabilities. Simulation models could be linked to AIV criteria and evidence to support earlier verification, virtual testing, or comparison between predicted and measured behavior. This would connect the methodology to broader trends such as digital twins and fast prototyping. Future work could also investigate integration with test execution environments and real-time data acquisition systems. In a future implementation, evidence generated during test execution could be automatically imported, and used to update verification status of requirements. This could possibly work in a similar fashion to the existing synchronization of requirement management tools into the modeling environment. This would reduce manual evidence handling and strengthen the connection between real-world AIV execution and the system model. The fact that the approach is based on an ontology could provide interesting research into ontology-based AI systems that support engineers in various AIV-related tasks.

The methodology provides a structured approach to clearly separate requirements, criteria, and collected AIV evidence, but the current way to evaluate the criteria based on evidence and evaluate requirements based on criteria requires human input. Another way to approach this is to define rules for evaluation of criteria and requirements in the model in such a way that when the collected evidence changes, the criteria and requirements status is automatically updated as well. An implementation of this is not difficult to achieve in the tool, but it should be investigated if such an implementation should be allowed to automatically change the verification control baseline, or if these changes should be made by the responsible engineers, who need to defend the verification control baseline in front of a verification control board or customer.

Practical extensions of the presented methodology would be to increase the amount of automatic model checks, through SysML v1 validation rules. This could serve to increase the consistency of the model and prevent modeling mistakes, especially when this methodology is applied in a collaborative setting where multiple people work in the model.

Finally, further research can address organizational adoption. The successful use of a model-based AIV methodology depends not only on the ontology and tool implementation, but also on training, modeling guidelines, review practices, collaboration workflows, and stakeholder acceptance. Future work should therefore investigate how such a methodology can be introduced into engineering teams in a way that is useful, maintainable, and compatible with existing AIV practices.

Applying this methodology in more student satellite projects is of interest. Student teams in particular show a high student turnover rate (i.e. team members joining and leaving the team). This leads to a high risk of losing knowledge. This methodology and the model-based approach in general helps to capture knowledge in a traceable and formalized way and could potentially be a promising answer to keeping knowledge within the team. Introducing MBSE in a student team does come with its own risks and hurdles as well, especially since current modeling tools still have a demanding learning curve. Dedicated research could investigate how to minimize these hurdles while preserving the benefits of increased traceability and a formalized modeling process.

6.6. Conclusion

This chapter discussed the results of the thesis and evaluated the extent to which the developed MBSE AIV methodology answers the research questions. The findings show that MBSE can be meaningfully extended into post-CDR satellite project stages when the methodology is tailored to the specific information needs of AIV work. In particular, the methodology supports the explicit representation of

requirements, AIV activities, AIV criteria, required evidence, setups, procedure information, verification status, and non-conformances within a connected model-based environment.

The literature review showed that existing MBSE approaches provide useful concepts for verification and validation, but do not yet provide comprehensive methodological support for practical AIV planning, execution, evidence management, and verification control in later life cycle phases. In response to this gap, the thesis developed an MBSE AIV methodology consisting of an ontology, SysML v1 implementation, process definition, modeling methods, and model-based documentation support. The application to the Da Vinci Satellite payload case study indicates that such an approach can improve traceability, consistency, and transparency compared to a purely document-centric way of working.

A central result is that the methodology preserves the verification rationale more explicitly than traditional AIV documentation. Requirements are linked to AIV activities, refined into evaluable criteria, assessed using collected evidence, and reflected in verification status information. This supports a more inspectable digital thread from requirement definition to verification close-out. At the same time, the methodology does not remove the need for AIV documents, engineering judgement, or formal review processes. Rather, it provides a structured model-based foundation from which AIV information can be managed, reviewed, and potentially reported.

The evaluation also shows that the demonstrated benefits should be interpreted as a result of a feasibility study rather than complete validation. The methodology was applied to selected examples and to a limited scope within the DVS payload case study, but not to a full satellite AIV campaign or large-scale industrial project. For a full scale application of industrial size, the approach likely requires to build up modeling expertise in the project, tool support, and other organizational adoption effort. Quantitative benefits such as time savings, cost reduction, or defect reduction were not measured and remain expected benefits rather than proven outcomes.

Overall, the thesis supports the conclusion that a focused, AIV-oriented MBSE methodology can provide value in post-CDR satellite projects by improving traceability, information consistency, and verification control. The work contributes a concrete step toward digital continuity beyond early design phases and provides a foundation for future research on SysML v2 implementation, tool integration, automated evidence handling, model-based verification execution, and broader validation in industrial and student satellite projects.

References

[1] Aistė Aleksandravičienė and Aurelijus Morkevičius. "MagicGrid® BOOK OF KNOWLEDGE A Practical Guide to Systems Modeling using MagicGrid from Dassault Systèmes 2nd edition". In: (2021).

[2] Oliver Alt. "Modellbasierte Systementwicklung mit SysML". In: (2012).

[3] Loyd Baker et al. "Model Driven System Design Working Group: FOUNDATIONAL CONCEPTS FOR MODEL DRIVEN SYSTEM DESIGN". In: *INCOSE International Symposium* 6.1 (July 1996), pp. 1179–1185. ISSN: 2334-5837. DOI: 10.1002/j.2334-5837.1996.tb02139.x. URL: https://incose.onlinelibrary.wiley.com/doi/10.1002/j.2334-5837.1996.tb02139.x.

[4] Rehobot Bekele Buruso et al. "Model-Based Systems Engineering in Space Applications: A Comprehensive Literature Review". In: *Systems Engineering* (Mar. 2026). ISSN: 15206858. DOI: 10.1002/sys.70051.

[5] Da Vinci Satellite. *Da Vinci Satellite | Elevating Education*. 2025. URL: https://davincisatellite.nl/.

[6] David Long. *Schema and Metamodels and Ontologies, Oh My!* Dec. 2020. URL: https://www.incose.org/docs/default-source/enchantment/210113-long-metamodelsb4eafe8472db67488e78ff000036190a.pdf?sfvrsn=393199c6_4.

[7] Lenny Delligatti. *SysML Distilled*. Tech. rep. 2013.

[8] T. Doran. "IEEE 1220: For Practical Systems Engineering". In: *Computer* 39.5 (May 2006), pp. 92–94. ISSN: 0018-9162. DOI: 10.1109/MC.2006.164. URL: http://ieeexplore.ieee.org/document/1631953/.

[9] ECSS. *ECSS-E-ST-10-02C Rev.1 Space engineering: Verification*. Tech. rep. Feb. 2018.

[10] ECSS. "ECSS-E-ST-10-03-Rev.1 Space Engineering: Testing". In: (May 2022). URL: https://ecss.nl/standard/ecss-e-st-10-03c-rev-1-testing-31-may-2022/.

[11] ECSS. "ECSS-E-ST-10-24C-Rev.1: Interface management". In: (2024).

[12] ECSS. "ECSS-E-ST-10C Rev.1 Space Engineering: System Engineering General Requirements". In: (Feb. 2017). URL: https://ecss.nl/standard/ecss-e-st-10c-rev-1-system-engineering-general-requirements-15-february-2017/#:~:text=This%20standard%20specifies%20the%20system%20engineering%20implementation%20requirements,development%20of%20systems%20and%20products%20for%20space%20applications..

[13] ECSS. "ECSS-M-30-01A Space project management: Organization and conduct of reviews". In: (Sept. 1999).

[14] ECSS. "ECSS-M-ST-10C Rev. 1 Space project management: Project planning and implementation". In: (Mar. 2009).

[15] ECSS. *ECSS-Q-ST-10-09C Rev.1 Space product assurance Nonconformance control system*. Tech. rep. Mar. 2018.

[16] ECSS. "ESSB-ST-U-007: ESA Space Debris Mitigation Requirements". In: 1 (Oct. 2023).

[17] Avner Engel. *Verification, Validation, and Testing of Engineered Systems*. Wiley, May 2010. ISBN: 9780470527511. DOI: 10.1002/9780470618851.

[18] ESA. "ESA Agenda 2025". In: (Apr. 2021). URL: https://www.esa.int/About_Us/ESA_Publications/Agenda_2025.

[19] ESA. *ESA SysML Solution*. 2025. URL: https://essr.esa.int/project/esa-sysml-solution.

[20] ESA. "ESA'S TECHNOLOGY STRATEGY". In: 1.2 (Sept. 2022). URL: https://esamultimedia.esa.int/docs/technology/ESA-Technology-Strategy-2022.pdf.

[21] *ESA Agenda 2025 / ESA unclassified-for public release ESA AGENDA 2025*. Tech. rep. URL: www.morganstanley.com/ideas/investing-in-space..

[22] Jeff A Estefan. *Survey of Model-Based Systems Engineering (MBSE) Methodologies*. Tech. rep. 2008.

[23] Kevin Forsberg and Harold Mooz. "The Relationship of System Engineering to the Project Cycle". In: *INCOSE International Symposium* 1.1 (Oct. 1991), pp. 57–65. ISSN: 2334-5837. DOI: 10.1002/J.2334-5837.1991.TB01484.X. URL: /doi/pdf/10.1002/j.2334-5837.1991.tb01484.x%20https://onlinelibrary.wiley.com/doi/abs/10.1002/j.2334-5837.1991.tb01484.x%20https://incose.onlinelibrary.wiley.com/doi/10.1002/j.2334-5837.1991.tb01484.x.

[24] Sanford Fridenthal. *A Practical Guide to SysML: The Systems Modeling Language*. 2014. ISBN: 978-0128002025.

[25] Sanford. Friedenthal, Alan. Moore, and Rick. Steiner. *A practical guide to SysML : the systems modeling language*. 4th Edition. Elsevier/Morgan Kaufmann, 2015, p. 631. ISBN: 9780128002025.

[26] Alberto Gonzales Fernandez. "ESA UNCLASSIFIED-For ESA Official Use Only ESA MBSE Evolution: From ESA SysML Toolbox to ESA MBSE Solution". In: *MBSE2021 Conference*. 2021. URL: https://indico.esa.int/event/386/contributions/6221/attachments/4264/6373/0935%20-%20esa%20mbse%20evolution%20from%20esa%20sysml%20toolbox%20to%20esa%20mbse%20solution.pdf.

[27] Giancarlo Guizzardi. "On Ontology, ontologies, Conceptualizations, Modeling Languages, and (Meta)Models." In: Apr. 2006, pp. 18–39.

[28] Isha Amod Gujarathi, William R. Norris, and Albert E. Patterson. "Spiral Approach for Non-Software Product and Engineering System Development". In: *Systems Engineering* (2025). ISSN: 15206858. DOI: 10.1002/SYS.70003.

[29] High Tech Institute. *Professor Jon Holt about his MBSE training*. 2025. URL: https://www.hightechinstitute.nl/mbse-training-jon-holt/.

[30] Jon. Holt. *UML for systems engineering : watching the wheels*. Institution of Engineering and Technology, 2007, p. 357. ISBN: 0863413544.

[31] Jon. Holt and Simon. Perry. *SysML for systems engineering*. Institution of Engineering and Technology, 2008, p. 335. ISBN: 9780863418259.

[32] IEEE. *IEEE Std 1220-2005 (Revision of IEEE Std 1220-1998): IEEE Standard for Application and Management of the Systems Engineering Process*. IEEE, 2005. ISBN: 0738146919. URL: https://ieeexplore.ieee.org/document/1511885.

[33] IEEE. "IEEE Std 829™-2008 IEEE Standard for Software and System Test Documentation". In: (2008). DOI: 10.1109/IEEESTD.2008.4578383.

[34] INCOSE. *System and SE Definitions*. 2025. URL: https://www.incose.org/about-systems-engineering/system-and-se-definitions/.

[35] INCOSE. "SYSTEMS ENGINEERING VISION 2020". In: (Sept. 2007).

[36] ISO. "ISO 14300-1:2023 - Space systems-Programme management-Part 1: Structuring of a project". In: (2023). URL: www.nen.nl.

[37] ISO/IEC and IEEE. "ISO/IEC/IEEE 15288- Systems and software engineering - System life cycle processes". In: (2023).

[38] ISO/IEC and IEEE. "ISO/IEC/IEEE 24641 - Systems and Software engineering - Methods and tools for model-based systems and software engineering". In: (Mar. 2023). DOI: 10.1109/IEEESTD.2023.10123376. URL: https://ieeexplore.ieee.org/document/10123376/.

[39] ISO/IEC and IEEE. "ISO/IEC/IEEE 24748-1: Guidelines for life cycle management". In: (Feb. 2024). DOI: 10.1109/IEEESTD.2024.10476385. URL: https://ieeexplore.ieee.org/document/10476385/.

[40] ISO/IEC and IEEE. "ISO/IEC/IEEE 24748-2: Guidelines for the application of ISO/IEC/IEEE 15288 (System life cycle processes)". In: (Feb. 2018). DOI: 10.1109/IEEESTD.2024.10476373. URL: https://ieeexplore.ieee.org/document/10476373/.

[41] ISO/IEC and IEEE. "ISO/IEC/IEEE 24748-4: Systems Engineering Planning". In: (Mar. 2016). DOI: 10.1109/IEEESTD.2016.7470727. URL: https://ieeexplore.ieee.org/document/7470727/.

[42] Sukhwan Jung and Alejandro Salado. "Emergent knowledge patterns in verification artifacts". In: *Systems Engineering* (Nov. 2024). ISSN: 15206858. DOI: 10.1002/SYS.21771.

[43] Hanumanthrao Kannan and Alejandro Salado. "A Theory-driven Interpretation and Elaboration of Verification and Validation". In: (Apr. 2025). URL: https://arxiv.org/pdf/2506.10997.

[44] L Lindblad, M Witzmann, and S Vanden Bussche. "DATA-DRIVEN SYSTEMS ENGINEERING: TURNING MBSE INTO INDUSTRIAL REALITY". In: (2021).

[45] Paul W. Logan et al. "Model-Based Systems Engineering Metamodel: Roadmap for Effective Systems Engineering Process". In: *Systems Engineering/Test and Evaluation Conference [proceedings CD-ROM]*. Canberra, 2013.

[46] James N. Martin. *Systems Engineering Guidebook*. CRC Press, Apr. 2020. ISBN: 9780138737443. DOI: 10.1201/9780138737443.

[47] MBSE4U. *Author Tim Weilkiens - Model Based Systems Engineering 4 You*. 2025. URL: https://mbse4u.com/tim-weilkiens/.

[48] Aurelijus Morkevicius, Aiste Aleksandraviciene, and Zilvinas Strolia. "System Verification and Validation Approach Using the MagicGrid Framework". In: *INSIGHT* 26.1 (Mar. 2023), pp. 51–59. ISSN: 2156-485X. DOI: 10.1002/inst.12429.

[49] Rebecca Mulholland, Cameron Bentley, and Jeffrey Williams. "PERFORMING VERIFICATION AND VALIDATION ACTIVITIES IN A MODEL-BASED ENVIRONMENT". In: *INCOSE International Symposium* 35.1 (July 2025), pp. 1912–1939. ISSN: 2334-5837. DOI: 10.1002/IIS2.70098. URL: /doi/pdf/10.1002/iis2.70098%20https://onlinelibrary.wiley.com/doi/abs/10.1002/iis2.70098%20https://incose.onlinelibrary.wiley.com/doi/10.1002/iis2.70098%20https://www.incose.org/incose-content-library/content-details?item=10565.

[50] NASA. "NASA Systems Engineering Handbook". In: NASA SP-2016-6105 Rev2 (2016).

[51] NASA. "NPR 7123.1D NASA Procedural Requirements ". In: (2023). URL: https://nodis3.gsfc.nasa.gov..

[52] OMG. *Business Process Model & Notation™ (BPMN™) | Object Management Group*. 2025. URL: https://www.omg.org/bpmn/.

[53] OMG. *OMG SysML Home | OMG Systems Modeling Language*. 2025. URL: https://www.omgsysml.org/.

[54] OMG. *Welcome To UML Web Site!* 2025. URL: https://www.uml.org/.

[55] Mauricio Peña and Ricardo Valerdi. "Characterizing the impact of requirements volatility on systems engineering effort". In: *Systems Engineering* 18 (1 Jan. 2015), pp. 59–70. ISSN: 15206858. DOI: 10.1111/sys.21288.

[56] Alejandro Salado and Hanumanthrao Kannan. "Elemental patterns of verification strategies". In: *Systems Engineering* 22.5 (Sept. 2019), pp. 370–388. ISSN: 15206858. DOI: 10.1002/SYS.21481.

[57] Nieves Salor Moral et al. *HOW MBSE CAN HELP AIV DOMAIN. A PRACTICAL APPROACH*. Tech. rep. 2023, pp. 6–10. URL: https://www.researchgate.net/publication/369204044_HOW_MBSE_CAN_HELP_AIV_DOMAIN_A_PRACTICAL_APPROACH.

[58] Starion-Group. "CDP4-COMET™ User manual". In: (2024).

[59] SysML-Praxis. *SysML-Praxis.de*. 2025. URL: http://www.lacheweg21.de/sysml/.

[60] Terry Halpin. "Object-Role Modeling". In: (). URL: www.orm.net.

[61] Jean-Luc Voirin. "ARCADIA USER GUIDE Arcadia Principles and Contents Overview". In: (2023).

[62] W3C. *OWL 2 Web Ontology Language Document Overview (Second Edition)*. URL: https://www.w3.org/TR/2012/REC-owl2-overview-20121211/.

[63] David D. Walden et al. *INCOSE Systems Engineering Handbook*. Fourth Edition. 4. International Council on Systems Engineering (INCOSE), 2015.

[64] David D. Walden et al., eds. *INCOSE Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*. 5th ed. Prepared by the International Council on Systems Engineering (INCOSE). Hoboken, NJ: John Wiley & Sons, 2023. ISBN: 9781119814290.

[65] T Weilkiens et al. *DRAFT (20160527): Evaluating and Comparing MBSE Methodologies for Practitioners*. May 2016.

[66] Tim Weilkiens. *Definition MBSE Model - Model Based Systems Engineering 4 You*. 2022. URL: https://mbse4u.com/2022/01/18/definition-mbse-model/.

[67] Tim Weilkiens. *Systems Engineering with SysML/UML - Modelling, Analysis, Design*. 2006. ISBN: 978-3-89864-409-9.

[68] Tim. Weilkiens et al. *Model-based system architecture*. Wiley, 2016, p. 373. ISBN: 9781118893647.

[69] Ramymond Wolfgang, Tami Katz, and Lou Wheatcraft. *Guide to Verification and Validation*. International Council on Systems Engineering (INCOSE), May 2022. URL: http://www.incose.org.

A

Documentation Titles and Explanation

A.1. ECSS Documentation for QR

Table A.1: ECSS Documentation for Qualification Review

Document Title	ECSS Document	DRD Reference	Content
Verification Plan	ECSS-E-ST-10-02 [9]	Annex A	Describes how compliance with requirements will be demonstrated: verification approach, model philosophy, VCD/verification matrix, planning and required tools. May be combined with AIT plan.
AIT QM/FM Plan	ECSS-E-ST-10-03 [10]	Annex A	Complete testing process, sequencing, test matrix, GSE, facilities, responsibilities, documentation, schedule. Complements Verification Plan.
Space Debris Mitigation Report	ESSB-ST-U-007 [16]	Annex B	Covers debris mitigation implementation, disposal strategy, passivation/re-entry, hazardous materials, compliance matrix.
Other Related Plans	ECSS-E-ST-10C [12]	Annex D	System Engineering Management Plan. Includes verification, programmatic, engineering, and operations plans.
Coordinate System Document	ECSS-E-ST-10-09 [**ECSS-E-ST-10-09**]	Annex A	—
Technical Budget	ECSS-E-ST-10C [12]	Annex I	Parameter documentation and justification. Part of Design Definition File (DDF).
Design Definition File (next lower level)	ECSS-E-ST-10C [12]	Annex G	References requirements, functions, architectures, interfaces, budgets, constraints, and engineering data repository.
Interface Control Document	ECSS-E-ST-10-24 [11]	Annex B	Defines interface compliance and compatibility; includes definitions, organization by product tree, naming, and coordinate conventions.
Product/User Manual	ECSS-E-ST-10C [12]	Annex P	Functional and physical aspects of system; includes operations, configurations, interfaces, FDIR, storage, autonomy, data/signal specifications.
Verification Control Document	ECSS-E-ST-10-02 [9]	Annex B	Lists all requirements to be verified; includes method, level, stage, traceability matrix. Continuously updated.
Test Specification	ECSS-E-ST-10-03 [10]	Annex B	Defines test requirements: purpose, tools, sequence, pass/fail criteria, traceability, responsibilities, test schedule.
Mathematical Model Description	ECSS-E-ST-32 [**ECSS-E-ST-32**]	Annex I	—
Correlation Report	ECSS-E-ST-31 [**ECSS-E-ST-31**]	Annex C	—

Document Title	ECSS Document	DRD Reference	Content
Test Procedure	ECSS-E-ST-10-03 [10]	Annex C	Step-by-step test execution instructions; configuration, roles, results, pass/fail criteria. Forms "as-run" procedure.
Test Report	ECSS-E-ST-10-02 [9]	Annex C	References test spec/procedure, includes results, analysis, anomalies, verified requirements, open issues.
Verification Report	ECSS-E-ST-10-02 [9]	Annex F	Required if multiple verification methods are used. Includes results, deviations, references.
Design Justification File	ECSS-E-ST-10	Annex K	Integrates trade-offs, traceability, verification control, tests, inspections, analysis.
Review of Design Report	ECSS-E-ST-10-02 [9]	Annex D	Outcome of design review; documents considered, verified requirements, open issues.
Inspection Report	ECSS-E-ST-10-02 [9]	Annex E	Closes requirements via inspection: traceability, inspection dates, results, deviations.
GSE Specification	—	—	—
GSE Data Packages	—	—	—
Nonconformance Report	ECSS-Q-ST-10-09C-Rev1 [15]	Annex A	Describes test issues, time/place, spec/procedure references. Tracks issues until resolved.

A.2. IEEE 829-2008 Documentation

Table A.2: IEEE 829-2008 Test Documentation [32]

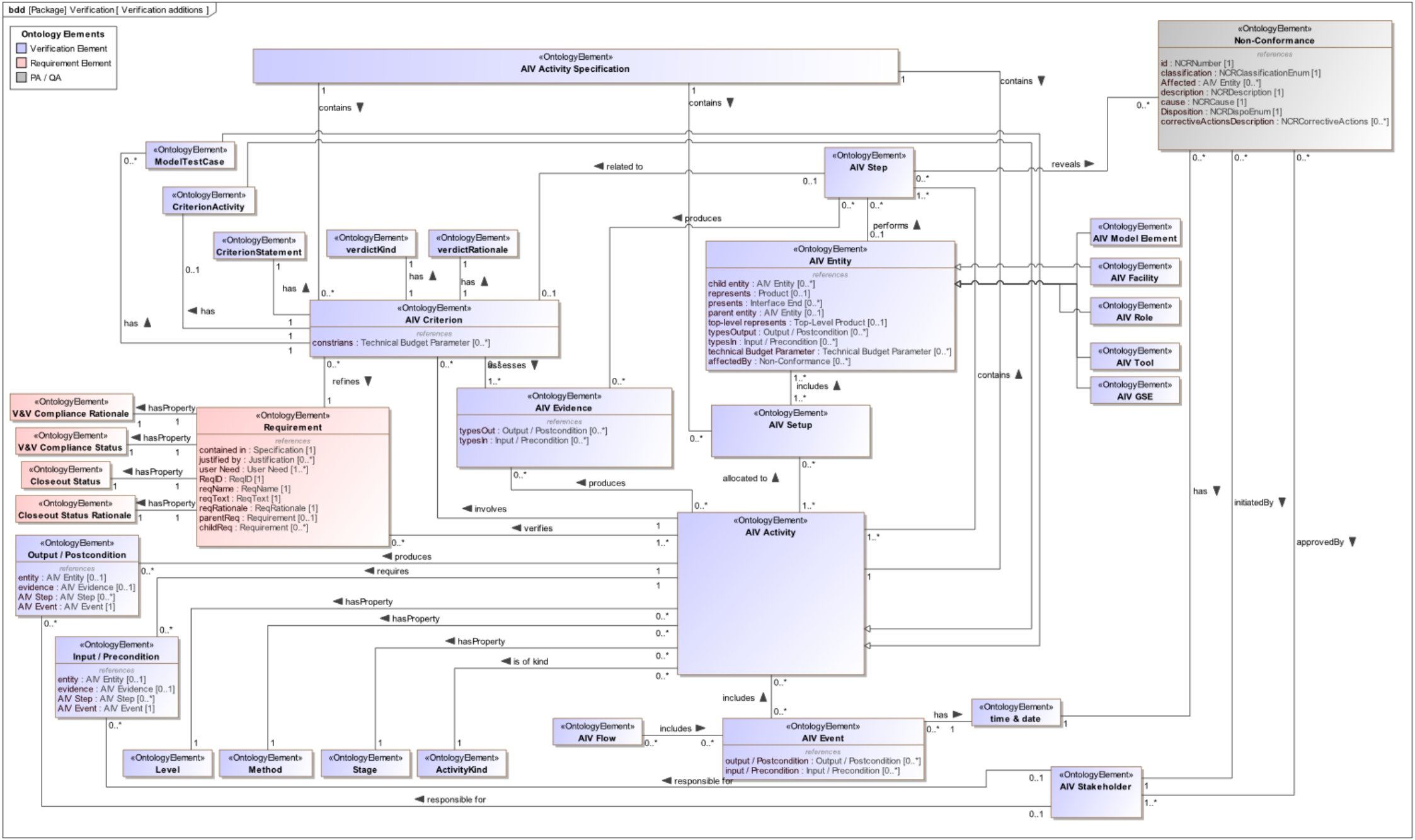
Document Title	Reference in IEEE 829-2008	Content
Master Test Plan	Clause 8	• Defines the overall strategy and coordination of all testing activities. • Describes system overview, key features, and testing flow. • Includes the master test schedule and prioritization schema (e.g. integrity levels). • Specifies test-related roles, responsibilities, tools, techniques, methods, and metrics. • Details requirements on documentation, reporting, and test administration.
Level Test Plan	Clause 9	• Created for a specific level of testing (e.g., Unit, Integration, System, Acceptance). • Defines scope, approach, resources, responsibilities, and risks. • Includes test items, traceability matrix, features to be tested, and pass/fail criteria. • Addresses test deliverables, infrastructure needs, schedules, and QA processes.
Test Design	Clause 10	• Refines the Level Test Plan by defining specific test conditions and sequences. • Organizes test cases into logical groups to achieve test coverage objectives. • Describes input data selection, expected results, and test design rationale.
Test Case	Clause 11	• Defines a single test objective and how it will be evaluated. • Includes test inputs and their dependencies (e.g., timing). • Lists expected outputs or behaviors with tolerances. • Specifies the test environment and required hardware/-software.
Test Procedure	Clause 12	• Provides step-by-step instructions to execute test cases. • Specifies inputs, outputs, required tools, and applicable test cases. • Covers logging, initialization, measurement, shutdown, restart, and contingency handling.

Document Title	**Reference in IEEE 829-2008**	**Content**
Test Log	Clause 13	• Records chronological details of test execution sessions. • Documents date/time, involved personnel, test versions/-configurations. • Notes deviations, execution summaries, and encountered anomalies.
Anomaly Report	Clause 14	• Documents any observed deviation from expected behavior during testing. • Includes description, impact analysis, and proposed or implemented corrective actions.
Test Report	Clause 16	• Summarizes results and events of a specific test level. • Includes test coverage, open/closed anomalies, and rationale for test decisions. • Provides conclusions, pass/fail status, and residual risk estimates.
Master Test Report	Clause 17	• Consolidates all Level Test Reports into a project-wide summary. • Assesses compliance with overall testing objectives. • Supports final verification closure and stakeholder review.

B

Full Ontology and Implementation mapping of Ontology Elements

The following page shows the full ontology diagram.


bdd [Package] Verification [Verification additions]
Ontology Elements
Verification Element
Requirement Element
PA / QA
«OntologyElement» AIV Activity Specification
«OntologyElement» Non-Conformance
references
id : NCRNumber [1]
classification : NCRClassificationEnum [1]
Affected : AIV Entity [0..*]
description : NCRDescription [1]
cause : NCRCause [1]
Disposition : NCRDispoEnum [1]
correctiveActionsDescription : NCRCorrectiveActions [0..*]
«OntologyElement» ModelTestCase
«OntologyElement» CriterionActivity
«OntologyElement» CriterionStatement
«OntologyElement» verdictKind
«OntologyElement» verdictRationale
«OntologyElement» AIV Step
«OntologyElement» AIV Criterion
references
constrians : Technical Budget Parameter [0..*]
«OntologyElement» AIV Entity
references
child entity : AIV Entity [0..*]
represents : Product [0..1]
presents : Interface End [0..*]
parent entity : AIV Entity [0..1]
top-level represents : Top-Level Product [0..1]
typesOutput : Output / Postcondition [0..*]
typesIn : Input / Precondition [0..*]
technical Budget Parameter : Technical Budget Parameter [0..*]
affectedBy : Non-Conformance [0..*]
«OntologyElement» AIV Model Element
«OntologyElement» AIV Facility
«OntologyElement» AIV Role
«OntologyElement» AIV Tool
«OntologyElement» AIV GSE
«OntologyElement» V&V Compliance Rationale
«OntologyElement» V&V Compliance Status
«OntologyElement» Closeout Status
«OntologyElement» Closeout Status Rationale
«OntologyElement» Requirement
references
contained in : Specification [1]
justified by : Justification [0..*]
user Need : User Need [1..*]
ReqID : ReqID [1]
reqName : ReqName [1]
reqText : ReqText [1]
reqRationale : ReqRationale [1]
parentReq : Requirement [0..1]
childReq : Requirement [0..*]
«OntologyElement» AIV Evidence
references
typesOut : Output / Postcondition [0..*]
typesIn : Input / Precondition [0..*]
«OntologyElement» AIV Setup
«OntologyElement» AIV Activity
«OntologyElement» Output / Postcondition
references
entity : AIV Entity [0..1]
evidence : AIV Evidence [0..1]
AIV Step : AIV Step [0..*]
AIV Event : AIV Event [1]
«OntologyElement» Input / Precondition
references
entity : AIV Entity [0..1]
evidence : AIV Evidence [0..1]
AIV Step : AIV Step [0..*]
AIV Event : AIV Event [1]
«OntologyElement» Level
«OntologyElement» Method
«OntologyElement» Stage
«OntologyElement» ActivityKind
«OntologyElement» AIV Flow
«OntologyElement» AIV Event
references
output / Postcondition : Output / Postcondition [0..*]
input / Precondition : Input / Precondition [0..*]
«OntologyElement» time & date
«OntologyElement» AIV Stakeholder
contains
related to
reveals
produces
performs
has
refines
assesses
includes
hasProperty
involves
verifies
requires
allocated to
is of kind
responsible for
initiatedBy
approvedBy

The following pages show the complete implementation mapping from ontology elements to stereotypes. The first two pages show the mapping for ontology elements, the other pages show mapping for relations in the ontology.

#	Name	Implemented by	Documentation
1	Requirement	DVSRequirement [Class]	
2	Specification	Specification [Package]	
3	User Need	UserNeed [Class]	
4	Justification	+justification : String [0..*] Justification [Class]	
5	V&V Compliance Status	+Phase C Compliance : AIV-Profile::requirements::RequirementComplianceKind [1] +Phase D Compliance : AIV-Profile::requirements::RequirementComplianceKind [1] +Phase E Compliance : AIV-Profile::requirements::RequirementComplianceKind [1]	
6	V&V Compliance Rationale	+Compliance Rationale : String [0..*]	
7	Closeout Status	+CloseoutStatus : ESA Profile::Concept profile::Value Types::Close-out status type [1]	
8	Closeout Status Rationale	+CloseoutStatusRationale : String [1]	
9	ReqID	+Id : String [1] =	
10	ReqName		Mapped to Name field of Requirement
11	ReqText	+Text : String [1] =	
12	ReqRationale		Mapped to Documentation field of Requirement
13	Non-Conformance	Non-Conformance [Comment]	
14	NCRNumber	+NCR-ID : String [1]	
15	NCRClassificationEnum	+classification : AIV-Profile::verification::NCRClass [1]	
16	NCRDispoEnum	NCRDisposition	
17	NCRDescription		Mapped to body of the Non-Conformance model element
18	NCRCause	+NCRCause : String [1]	
19	NCRCorrectiveActions	+correctiveActionsDescription : String [0..*]	
20	AIV Flow	AIVFlow [Activity]	
21	AIV Evidence	AIVEvidence [Class, DataType]	
22	AIV Step	AIVStep [CallBehaviorAction]	
23	AIV Entity	AIVEntity [Class]	
24	AIV Activity	AIVActivity [Activity]	
25	AIV Setup	AIVSetup [Class]	
26	AIV Event	AIVEvent [Activity]	
27	Input / Precondition	InputPin	
28	Output / Postcondition	OutputPin	
29	Result Description	+actualResult : String [0..1] +expectedResult : String [0..1]	
30	verdictKind	+verdict : AIV-Profile::verification::CriterionVerdictKind [1] = TBD	
31	ActivityKind	+vvKind : AIV-Profile::verification::AIVVActivityKind [1]	
32	Level	+level : String [1]	
33	AIV Criterion	AIVCriterion [Class]	
34	Stage	+stage : String [1]	
35	Method	+method : SysML::Non-Normative	

12	ReqRationale		Mapped to Documentation field of Requirement
13	Non-Conformance	«» Non-Conformance [Comment]	
14	NCRNumber	+NCR-ID : String [1]	
15	NCRClassificationEnum	+classification : AIV-Profile::verification::NCRClass [1]	
16	NCRDispoEnum	NCRDisposition	
17	NCRDescription		Mapped to body of the Non-Conformance model element
18	NCRCause	+NCRCause : String [1]	
19	NCRCorrectiveActions	+correctiveActionsDescription : String [0..*]	
20	AIV Flow	AIVFlow [Activity]	
21	AIV Evidence	AIVEvidence [Class, DataType]	
22	AIV Step	AIVStep [CallBehaviorAction]	
23	AIV Entity	AIVEntity [Class]	
24	AIV Activity	AIVActivity [Activity]	
25	AIV Setup	AIVSetup [Class]	
26	AIV Event	AIVEvent [Activity]	
27	Input / Precondition	InputPin	
28	Output / Postcondition	OutputPin	
29	Result Description	+actualResult : String [0..1] +expectedResult : String [0..1]	
30	verdictKind	+verdict : AIV-Profile::verification::CriterionVerdictKind [1] = TBD	
31	ActivityKind	+vvKind : AIV-Profile::verification::AIVVActivityKind [1]	
32	Level	+level : String [1]	
33	AIV Criterion	AIVCriterion [Class]	
34	Stage	+stage : String [1]	
35	Method	+method : SysML::Non-Normative Extensions::Requirement::VerificationMethodKind [1]	
36	verdictRationale	+verdictRationale : String [1]	
37	CriterionStatement	AIVCriterion [Class] +Text : String [1] =	
38	ModelTestCase	«» ModelTestCase [Activity, Operation]	
39	CriterionActivity	Activity	
40	time & date	+plannedEndDate : date [1] +plannedStartDate : date [1]	
41	AIV Role	+type : Ontology Definition::Verification::AIVEntityKind [1]	
42	AIV Facility	+type : Ontology Definition::Verification::AIVEntityKind [1]	
43	AIV Tool	+type : Ontology Definition::Verification::AIVEntityKind [1]	
44	AIV GSE	+type : Ontology Definition::Verification::AIVEntityKind [1]	
45	AIV Model Element	+type : Ontology Definition::Verification::AIVEntityKind [1]	
46	AIV Stakeholder	«» AIVStakeholder [Class]	
47	AIV Activity Specification	Package	
48	Technical Budget Parameter	ValueProperty [Property]	

#	△ Type (Role B)	Name (Role B)	Multiplicity (Role B)	Name	Multiplicity (Role A)	Name (Role A)	Type (Role A)	ImplementedBy	Documentation
1	AIV Activity	supports	1..*	allocated to	0..*	allocatedTo	AIV Setup	Allocate [Abstraction]	
2	AIV Activity	organized in	1..*	contains	1..*	organizes	AIV Step		Implemented via SysML Containment Relation
3	AIV Activity	V&V Activity	0..*	hasProperty	1	method	Method	+method : SysML::Non-Normative Extensions::Requirement:: VerificationMethodKind [1]	
4	AIV Activity	V&V Activity	0..*	hasProperty	1	stage	Stage	+stage : String [1]	
5	AIV Activity	V&V Activity	0..*	hasProperty	1	level	Level	+level : String [1]	
6	AIV Activity	AIVActivity	1	involves	0..*	AIV Criterion	AIV Criterion	Trace [Abstraction]	
7	AIV Activity	Activity	0..*	is of kind	1	activityKind	ActivityKind	+vvKind : AIV-Profile::verification:: AIVVActivityKind [1]	
8	AIV Activity	verification Event	0..*	produces	0..*	artefact	AIV Evidence	Parameter	AIV Evidence types owned parameter of AIV ACtivity
9	AIV Activity	verification Activity	1	produces	0..*	outputs	Output / Postcondition		Implemented via SysML Containment Relation
10	AIV Activity	verification Activity	1	requires	0..*	inputs	Input / Precondition		Implemented via SysML Containment Relation
11	AIV Activity	verifiedBy	1..*	verifies	0..*	verifies	Requirement	Verify [Abstraction]	
12	AIV Activity Specification	AIV Activity Specification	1	contains	1	aIVActivity	AIV Activity		Implemented via SysML Containment Relation
13	AIV Activity Specification	AIV Activity Specification	1	contains	0..*	AIV Setup	AIV Setup		Implemented via SysML Containment Relation
14	AIV Activity Specification	AIV Activity Specification	1	contains	0..*	AIV Criterion	AIV Criterion		Implemented via SysML Containment Relation
15	AIV Criterion	isAssessedBy	0..*	assesses	1..*	assesses	AIV Evidence	Trace [Abstraction]	
16	AIV Criterion	AIV Criterion	0..*	constrains	0..*	technical Budget Parameter	Technical Budget Parameter		Implemented using parametric relation to Constraint Block that refines the Criterion.
17	AIV Criterion	verificationCriterion	1	has	1	verdict	verdictKind	+verdict : AIV-Profile::verification:: CriterionVerdictKind [1] = TBD	
18	AIV Criterion	V&V Criterion	1	has	1	verdictRationale	verdictRationale	+verdictRationale : String [1]	
19	AIV Criterion	V&V Criterion	1	has	1	criterionStatement	CriterionStatement	+Text : String [1] =	test
20	AIV Criterion	V&V Criterion	1	has	0..*	testCase	ModelTestCase		Implemented via SysML Containment Relation
									Implemented via SysML

20	AIV Criterion	V&V Criterion	1	has	0..*	testCase	ModelTestCase			Implemented via SysML Containment Relation
21	AIV Criterion	V&V Criterion	1	has	0..1	criterionProcedure	CriterionActivity			Implemented via SysML Containment Relation
22	AIV Criterion	refinedBy	0..*	refines	1	interprets	Requirement	«» Refine [Abstraction]		
23	AIV Entity	parent entity	0..1	composed of	0..*	child entity	AIV Entity	Association		
24	AIV Entity	AIV Entity	1	has	0..*	technical Budget Parameter	Technical Budget Parameter			Implemented via SysML Containment Relation
25	AIV Entity	performed by	0..1	performs	0..*	performs	AIV Step	«» AllocateActivityPartition [ActivityPartition] «» Allocate [Abstraction]		
26	AIV Entity	presented by	0..*	presents	0..*	presents	Interface End			Implemented using owned proxy port typed by ESA Physical Interface Stereotype or SysML Interface Block
27	AIV Entity	model	0..*	represents	0..1	represents	Product	Generalization		
28	AIV Entity	model	0..*	represents	0..1	top-level represents	Top-Level Product	Generalization		
29	AIV Entity	Affected	0..*	affects	0..*	affectedBy	Non-Conformance			Implemented using annotated item field of Non-Conformance (Comment)
30	AIV Event	AIV Event	0..*	has	1	time & date	time & date	+plannedStartDate : date [1] +plannedEndDate : date [1]		
31	AIV Event	AIVEvent	0..*	includes	0..*	AIV Actity	AIV Activity	CallBehaviorAction		AIV Activity types CallBehaviorAction owned by AIV Event
32	AIV Event	AIV Event	1	produces	0..*	output / Postcondition	Output / Postcondition			Implemented via SysML Containment Relation
33	AIV Event	AIV Event	1	requires	0..*	input / Precondition	Input / Precondition			Implemented via SysML Containment Relation
34	AIV Flow	AIV flow	0..*	includes	0..*	verification event	AIV Event	CallBehaviorAction		AIV Event types CallBehaviorAction owned by AIV Flow
35	AIV Setup	involved in	1..*	includes	1..*	verification element	AIV Entity	Association		
36	AIV Stakeholder	AIV Stakeholder	0..1	responsible for	0..*	inputs / Preconditions	Input / Precondition	«» Trace [Abstraction]		
37	AIV Stakeholder	AIV Stakeholder	0..1	responsible for	0..*	outputs / Postconditions	Output / Postcondition	«» Trace [Abstraction]		
38	AIV Step	origin	1	expected	0..*	expectedResult	Result Description	+expectedResult : String [0..1]		
39	AIV Step	V&V Step	0..*	produces	0..*	produces	AIV Evidence			AIV Evidence types owned output pin of AIV Activity

39	AIV Step	V&V Step	0..*	produces	0..*	produces	AIV Evidence		AIV Evidence types owned output pin of AIV Activity
40	AIV Step	AIV Step	0..*	produces	0..*	outputs / Postconditions	Output / Postcondition		Implemented via SysML Containment Relation
41	AIV Step	V&V Step	0..1	related to	0..1	V&V Criterion	AIV Criterion	CallBehaviorAction	AIV Step is CallBehaviorAction typed by Activity that is owned by AIV Criterion
42	AIV Step	AIV Step	0..*	requires	0..*	inputs / Preconditions	Input / Precondition		Implemented via SysML Containment Relation
43	AIV Step	relatedProcedureSteps	0..*	reveals	0..*	reveals	Non-Conformance		Implemented using annotated item property of non-conformance.
44	Input / Precondition	typesIn	0..*	typedBy	0..1	entity	AIV Entity		Using SysML Type field
45	Input / Precondition	typesIn	0..*	typedBy	0..1	evidence	AIV Evidence		Using SysML Type field
46	Justification	justified by	0..*	justifies	1..*	justifies	Requirement	«» Justify [Dependency]	
47	Non-Conformance	approvesFor	0..*	approvedBy	1..*	NCRApprover	AIV Stakeholder	+NCRApprover : String [1..*]	
48	Non-Conformance	non-Conformance	0..*	has	1	time & date	time & date	+dateOfDetection : date [1]	
49	Non-Conformance	initiatedNCRs	0..*	initiatedBy	1	initiator	AIV Stakeholder	+NCRInitiator : String [1..*]	
50	Output / Postcondition	typesOutput	0..*	typedBy	0..1	entity	AIV Entity		Using SysML Type field
51	Output / Postcondition	typesOut	0..*	typedBy	0..1	evidence	AIV Evidence		Using SysML Type field
52	Requirement	childReq	0..*	derivedFrom	0..1	parentReq	Requirement	DeriveReqt [Abstraction]	
53	Requirement	requirement	1	hasProperty	1	closeout Status	Closeout Status	+CloseoutStatus : ESA Profile::Concept profile::Value Types::Close-out status type [1]	
54	Requirement	requirement	1	hasProperty	1	closeout Status Rationale	Closeout Status Rationale	+CloseoutStatusRationale : String [1]	
55	Requirement	requirement	1	hasProperty	1	ReqID	ReqID	+Id : String [1] =	
56	Requirement	requirement	1	hasProperty	1	reqName	ReqName		Mapped to Name field of Requirement
57	Requirement	requirement	1	hasProperty	1	reqText	ReqText	+Text : String [1] =	
58	Requirement	rationaleFor	1	hasProperty	1	reqRationale	ReqRationale		Mapped to Documentation field of Requirement
59	Requirement	requirement	1..*	refines	1..*	user Need	User Need	«» Refine [Abstraction]	
60	Requirement	owned by	1	hasProperty	1	complianceRationale	V&V Compliance Rationale	+Compliance Rationale : String [0..*]	
								+Phase C Compliance : AIV-Profile::requirements::RequirementComplianceKind	

43	AIV Step	relatedProcedureSteps	0..*	reveals	0..*	reveals	Non-Conformance			Implemented using annotated item property of non-conformance.
44	Input / Precondition	typesIn	0..*	typedBy	0..1	entity	AIV Entity			Using SysML Type field
45	Input / Precondition	typesIn	0..*	typedBy	0..1	evidence	AIV Evidence			Using SysML Type field
46	Justification	justified by	0..*	justifies	1..*	justifies	Requirement	«» Justify [Dependency]		
47	Non-Conformance	approvesFor	0..*	approvedBy	1..*	NCRApprover	AIV Stakeholder	+NCRApprover : String [1..*]		
48	Non-Conformance	non-Conformance	0..*	has	1	time & date	time & date	+dateOfDetection : date [1]		
49	Non-Conformance	initiatedNCRs	0..*	initiatedBy	1	initiator	AIV Stakeholder	+NCRInitiator : String [1..*]		
50	Output / Postcondition	typesOutput	0..*	typedBy	0..1	entity	AIV Entity			Using SysML Type field
51	Output / Postcondition	typesOut	0..*	typedBy	0..1	evidence	AIV Evidence			Using SysML Type field
52	Requirement	childReq	0..*	derivedFrom	0..1	parentReq	Requirement	DeriveReqt [Abstraction]		
53	Requirement	requirement	1	hasProperty	1	closeout Status	Closeout Status	+CloseoutStatus : ESA Profile::Concept profile::Value Types::Close-out status type [1]		
54	Requirement	requirement	1	hasProperty	1	closeout Status Rationale	Closeout Status Rationale	+CloseoutStatusRationale : String [1]		
55	Requirement	requirement	1	hasProperty	1	ReqID	ReqID	+Id : String [1] =		
56	Requirement	requirement	1	hasProperty	1	reqName	ReqName			Mapped to Name field of Requirement
57	Requirement	requirement	1	hasProperty	1	reqText	ReqText	+Text : String [1] =		
58	Requirement	rationaleFor	1	hasProperty	1	reqRationale	ReqRationale			Mapped to Documentation field of Requirement
59	Requirement	requirement	1..*	refines	1..*	user Need	User Need	«» Refine [Abstraction]		
60	Requirement	owned by	1	hasProperty	1	complianceRationale	V&V Compliance Rationale	+Compliance Rationale : String [0..*]		
61	Requirement	owned by	1	hasProperty	1	complianceStatus	V&V Compliance Status	+Phase C Compliance : AIV-Profile::requirements::RequirementComplianceKind [1] +Phase D Compliance : AIV-Profile::requirements::RequirementComplianceKind [1] +Phase E Compliance : AIV-Profile::requirements::RequirementComplianceKind [1]		
62	Specification	contained in	1	contains	1..*	contains	Requirement			Implemented via SysML Containment Relation

C

Legend of ISO Life Cycle Processes Acronyms

This annex provides a legend for the acronyms used in the ISO life cycle process N2-chart shown in Figure C.1. The table below lists the process abbreviations and supporting categories used in the figure to make the chart easier to interpret. For a full explanation, the reader is referred to the source material from the INCOSE Systems Engineering Handbook, fifth edition [64].

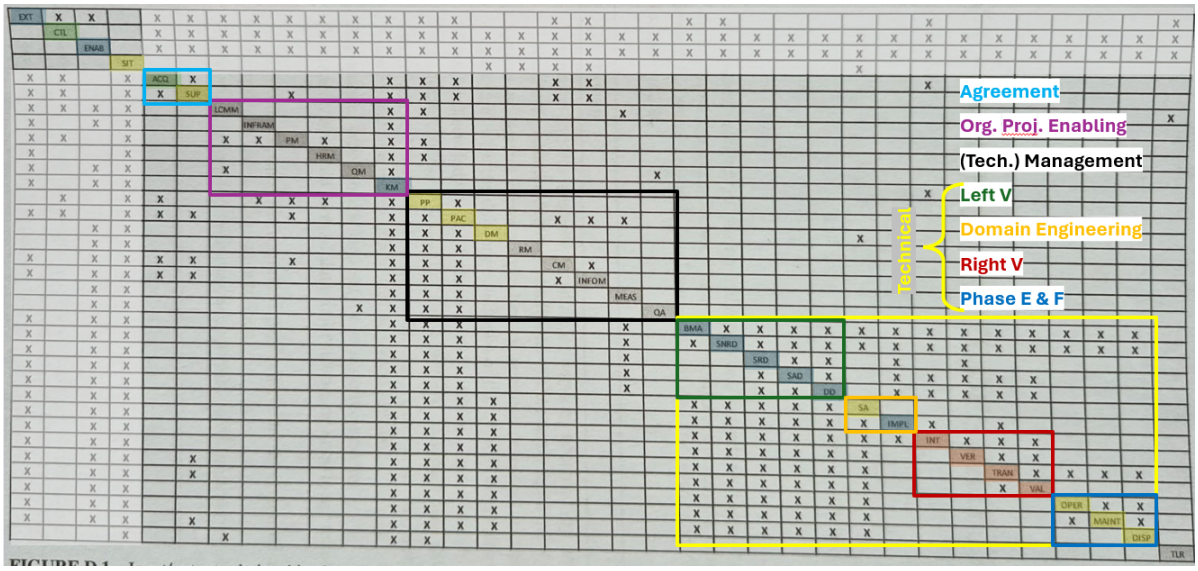


Figure C.1: ISO Life Cycle Processes N2-Chart by INCOSE [64]

Table C.1: Overview of INCOSE Life Cycle Process acronyms, from [64]

Acronym	**Meaning**
EXT	External
CTL	Controls
ENAB	Enablers
SIT	Situational
ACQ	Acquisition
SUP	Supply
LCMM	Life Cycle Model Management
INFRAM	Infrastructure Management
PM	Portfolio Management
HRM	Human Resource Management
QM	Quality Management
KM	Knowledge Management
PP	Project Planning
PAC	Project Assessment and Control
DM	Decision Management
RM	Risk Management
CM	Configuration Management
INFOM	Information Management
MEAS	Measurement
QA	Quality Assurance
BMA	Business or Mission Analysis
SNRD	Stakeholder Needs and Requirements Definition
SRD	System Requirements Definition
SAD	System Architecture Definition
DD	Design Definition
SA	System Analysis
IMPL	Implementation
INT	Integration
VER	Verification
TRAN	Transition
VAL	Validation
OPER	Operation
MAINT	Maintenance
DISP	Disposal
TLR	Tailoring

D

Generated Procedure from model

The following page shows the full generated procedure for the toaster operations example.

DVS – TOASTER AIV SHOWCASE PROJECT

Exported AIV Procedure

Written by		**Reviewed by**	
Signature		**Signature**	

Operator 1		**Operator 2**	
Signature		**Signature**	

Estimated test duration		**Actual test duration**	

Document generated by: Nicolas Oidtmann

Issue: 0.1

Reference: XX00.YY-Z

Date of Issue: 23 March, 2026

1 TABLE OF CONTENTS

2 SCOPE OF THE ACTIVITY

Reasoning behind the test. Explain why it was needed, what type of test it is, whether it's functional, operational, integrated etc.

Requirements to verify are in the following page.

Title of the Activity

ToasterQualificationRun

Description

The purpose of this AIV activity is to verify that the toaster under qualification meets its performance requirements with respect to toasting duration and achieved toastiness level. The procedure defines a controlled and repeatable method to operate the toaster, measure the toasting time, and assess the resulting toast quality using objective evidence.

This activity contributes to the generation of verification evidence in the form of timing measurements, images, and evaluated toastiness levels, enabling an objective assessment of compliance with the defined AIV criteria. The structured execution of the procedure ensures consistency, traceability, and reproducibility of results within the verification campaign.

2.1 REQUIREMENTS TO VERIFY

The table below presents an overview of the requirements to be verified.

Req ID	Req Title	Req Text	Related Criteria
1	Toast Browning Level	The toaster shall produce a uniformly golden-brown toast surface within the levels 5 to 8 on the toastiness scale.	Crit-1 | Toastiness of Toast within 5 to 8
2	Toasting Duration	The toaster shall complete the toasting process within at most 180 seconds for a standard slice of bread.	Crit-2 | Toasting Duration

2.2 CRITERIA TO CHECK

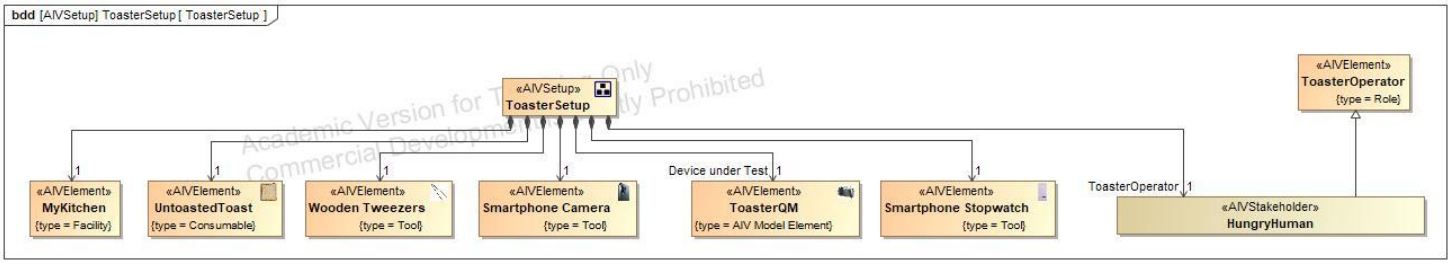
Crit ID	Crit Name	Crit Text	Evaluation Notes	Related Requirements	Related AIV Step
Crit-1	Toastiness of Toast within 5 to 8	The toasted toast has a toastiness level corresponding to at least a 5 and at most an 8 on the toastiness scale.	- The Toastiness Preference Scale should be attached - Indicate an integer number for front and back side of toast	1 | Toast Browning Level	SID-10 | Determine toastiness level of first side of toast on the TPS and take picture SID-12 | Determine toastiness level of second side of toast on the TPS and take picture
Crit-2	Toasting Duration	The total duration between starting the toasting process to ejection of the toasted toast takes at most up to 180 seconds.	- measure with stopwatch - Take picture of stopwatch time for later reference	2 | Toasting Duration	SID-8 | Take screenshot of stopwatch showing the measured toasting cycle duration.

3 SETUP & EQUIPMENT

3.1 SETUP

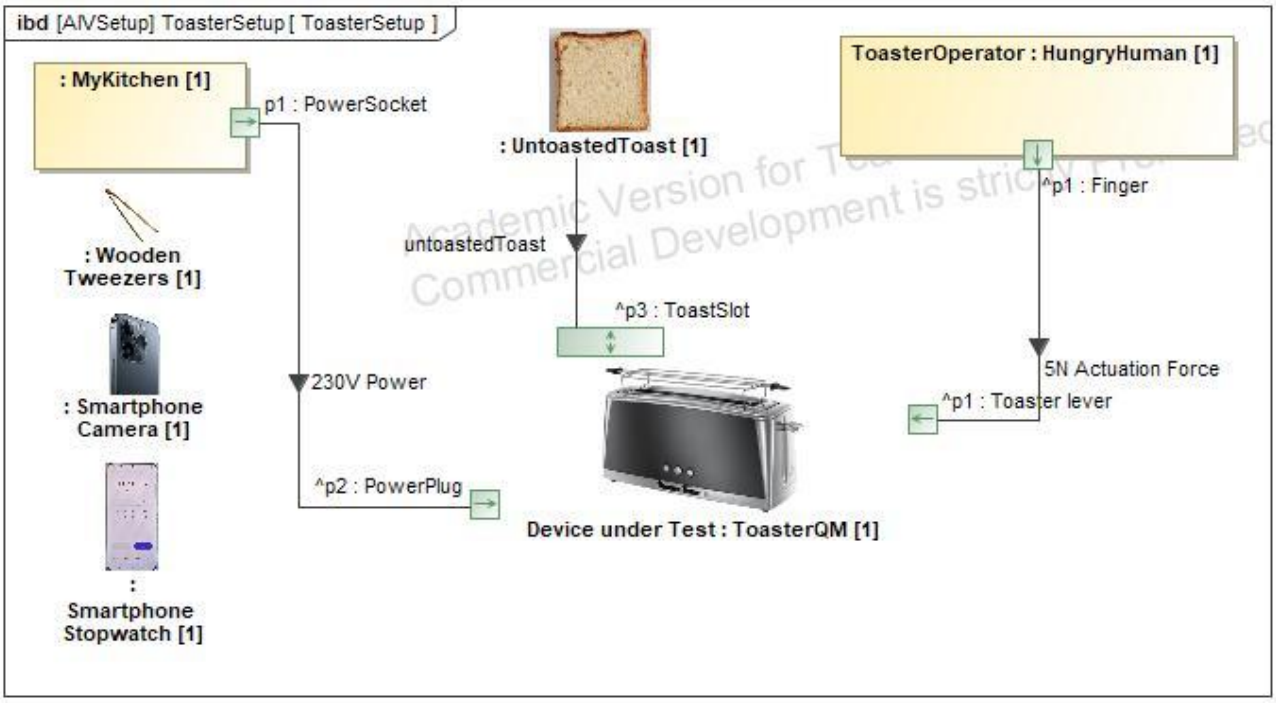
Describe how the test is set up, how the hardware is secured and how everything looks like.

SysML Block Definition Diagram: ToasterSetup



NA

SysML Internal Block Diagram: ToasterSetup



NA

Generic Table: ToasterSetup

#	Name	Type	Multiplicity	AIVEntityType
1		MyKitchen	1	Facility
2	Device under Test	ToasterQM	1	AIV Model Element
3		UntoastedToast	1	Consumable
4		Wooden Tweezers	1	Tool
5		Smartphone Camera	1	Tool
6		Smartphone Stopwatch	1	Tool
7	ToasterOperator	HungryHuman	1	Role

NA

Setup documentation

To create the setup to perform the procedure, you have to plug in the Toaster into a power plug. The operator will need to load the toaster with untoasted toast, and press down the toaster lever.

Next to that a Smartphone Camera and Stopwatch are required, as well as wooden tweezers to handle the toasted toast.

3.2 GSE & OTHER EQUIPMENT

The following Equipment shall be used for the test activity.

3.2.1 INPUTS

Number	Item	Quantity	relatedStep	Check	Comments
1	toaster : ToasterQM	1	SID-2 SID-3 SID-5		NA NA
Extra Remarks:					

Number	Item	Quantity	relatedStep	Check	Comments
2	kitchen : MyKitchen	1	SID-2		NA NA
Extra Remarks:					

Number	Item	Quantity	relatedStep	Check	Comments
3	toast : UntoastedToast	1	SID-3		NA NA

Extra Remarks:					
Number	**Item**	**Quantity**	**relatedStep**	**Check**	**Comments**
4	stopwatch : Smartphone Stopwatch	1	SID-4 SID-5 SID-7 SID-8		NA NA
Extra Remarks:					
Number	**Item**	**Quantity**	**relatedStep**	**Check**	**Comments**
5	wt : Wooden Tweezers	1	SID-9 SID-11		NA NA
Extra Remarks:					
Number	**Item**	**Quantity**	**relatedStep**	**Check**	**Comments**

6	cam : Smartphone Camera	1	SID-10 SID-12		NA NA
Extra Remarks:					

Number	Item	Quantity	relatedStep	Check	Comments
7	tps : TPS - ToastinessPreferenceScale	1	SID-10 SID-12		NA NA
Extra Remarks:					

Number	Item	Quantity	relatedStep	Check	Comments
8	setup : ToasterSetup	1	SID-1		NA To create the setup to perform the procedure, you have to plug in the Toaster into a power plug. The operator will need to load the toaster with untoasted toast, and press down the toaster lever.

					Next to that a Smartphone Camera and Stopwatch are required, as well as wooden tweezers to handle the toasted toast.
Extra Remarks:					

3.2.2 OUTPUTS

Number	Item	Quantity	relatedStep	Check	Comments
1	screenshot : Screenshot of stopped Stopwatch showing toasting duration	1	SID-8		NA NA
Extra Remarks:					

Number	Item	Quantity	relatedStep	Check	Comments
2	pic1 : Picture of toasted toast first side	1	SID-10		NA NA

Extra Remarks:

Number	Item	Quantity	relatedStep	Check	Comments
3	est1 : Estimated toastiness of toast first side	1	SID-10		NA NA

Extra Remarks:

Number	Item	Quantity	relatedStep	Check	Comments
4	pic2 : Picture of toasted toast second side	1	SID-12		NA NA

Extra Remarks:

Number	Item	Quantity	relatedStep	Check	Comments
5	est2 : Estimated toastiness of toast second side	1	SID-12		NA NA
Extra Remarks:					

Number	Item	Quantity	relatedStep	Check	Comments
6	toastedtoast : ToastedToast	1	SID-9 SID-10 SID-11 SID-12		NA NA
Extra Remarks:					

3.2.3 INTERMEDIATE ELEMENTS

Number	Item	Quantity	relatedStep	Check	Comments
1	5N Actuation Force	NA	SID-5	NA	NA

Extra Remarks:					

4 STEP-BY-STEP PROCEDURE

Note: fill in the columns related to “pass/fail criterion” only need to be considered if there is a criterion to be verified, otherwise to be ignored.

StepID	Description	Input	Output	Remarks	Date/Time/ Signature
SID-1	Check that all elements described in the setup are present in the correct quantities	- ToasterSetup			

Crit ID	Name	Criterion Text	P/F	Evaluation Notes	Results
NA	NA	NA			

StepID	Description	Input	Output	Remarks	Date/Time/ Signature
SID-2	Plug the toaster power plug into the Kitchen's power socket	- ToasterQM - MyKitchen			

Crit ID	Name	Criterion Text	P/F	Evaluation Notes	Results
NA	NA	NA			

StepID	Description	Input	Output	Remarks	Date/Time/ Signature
SID-3	Insert untoasted toast into the toaster.	- ToasterQM - UntoastedToast	- UntoastedToast		

Crit ID	Name	Criterion Text	P/F	Evaluation Notes	Results
NA	NA	NA			

StepID	Description	Input	Output	Remarks	Date/Time/ Signature
SID-4	Prepare stopwatch on smartphone to be able to actuate toaster lever and start stopwatch at the same time.	- Smartphone Stopwatch			

Crit ID	Name	Criterion Text	P/F	Evaluation Notes	Results
NA	NA	NA			

StepID	Description	Input	Output	Remarks	Date/Time/ Signature
SID-5	Actuate toaster lever by pressing it down until it locks into place and start stopwatch.	- ToasterQM - Smartphone Stopwatch	- 5N Actuation Force		

Crit ID	Name	Criterion Text	P/F	Evaluation Notes	Results
NA	NA	NA			

StepID	Description	Input	Output	Remarks	Date/Time/ Signature
SID-6	Observe toaster during toasting cycle and be prepare to stop the stopwatch.				

Crit ID	Name	Criterion Text	P/F	Evaluation Notes	Results
NA	NA	NA			

StepID	Description	Input	Output	Remarks	Date/Time/ Signature
SID-7	When toasting cycle is complete and the toast is ejected, stop the stopwatch.	- Smartphone Stopwatch			

Crit ID	Name	Criterion Text	P/F	Evaluation Notes	Results
NA	NA	NA			

StepID	Description	Input	Output	Remarks	Date/Time/ Signature
SID-8	NA	- Smartphone Stopwatch	- Screenshot of stopped Stopwatch showing toasting duration		

Crit ID	Name	Criterion Text	P/F	Evaluation Notes	Results
Crit-2	Toasting Duration	The total duration between starting the toasting process to ejection of the toasted toast takes at most up to 180 seconds.		- measure with stopwatch - Take picture of stopwatch time for later reference	

<table>
<tr><th>StepID</th><th>Description</th><th>Input</th><th>Output</th><th>Remarks</th><th>Date/Time/ Signature</th></tr>
<tr><td>SID-9</td><td>Use Wooden Tweezers to remove toast from toaster and place it on kitchen counter.</td><td>- ToastedToast
- Wooden Tweezers</td><td>- ToastedToast</td><td></td><td></td></tr>
<tr><td></td><td colspan="4">
<table>
<tr><th>Crit ID</th><th>Name</th><th>Criterion Text</th><th>P/F</th><th>Evaluation Notes</th><th>Results</th></tr>
<tr><td>NA</td><td>NA</td><td>NA</td><td></td><td></td><td></td></tr>
</table>
</td><td></td></tr>
</table>

<table>
<tr><th>StepID</th><th>Description</th><th>Input</th><th>Output</th><th>Remarks</th><th>Date/Time/ Signature</th></tr>
<tr><td>SID-10</td><td>NA</td><td>- ToastedToast
- Smartphone Camera
- TPS - ToastinessPreferenceScale</td><td>- Picture of toasted toast first side
- Estimated toastiness of toast first side
- ToastedToast</td><td></td><td></td></tr>
<tr><td></td><td colspan="4">
<table>
<tr><th>Crit ID</th><th>Name</th><th>Criterion Text</th><th>P/F</th><th>Evaluation Notes</th><th>Results</th></tr>
<tr><td>Crit-1</td><td>Toastiness of Toast within 5 to 8</td><td>The toasted toast has a toastiness level corresponding to at least a 5 and at most an 8 on the toastiness scale.</td><td></td><td>- The Toastiness Preference Scale should be attached
- Indicate an integer number for front and back side of toast</td><td></td></tr>
</table>
</td><td></td></tr>
</table>

StepID	Description	Input	Output	Remarks	Date/Time/ Signature
SID-11	Use Wooden Tweezers to flip toast and place it back on kitchen counter	- ToastedToast - Wooden Tweezers	- ToastedToast		

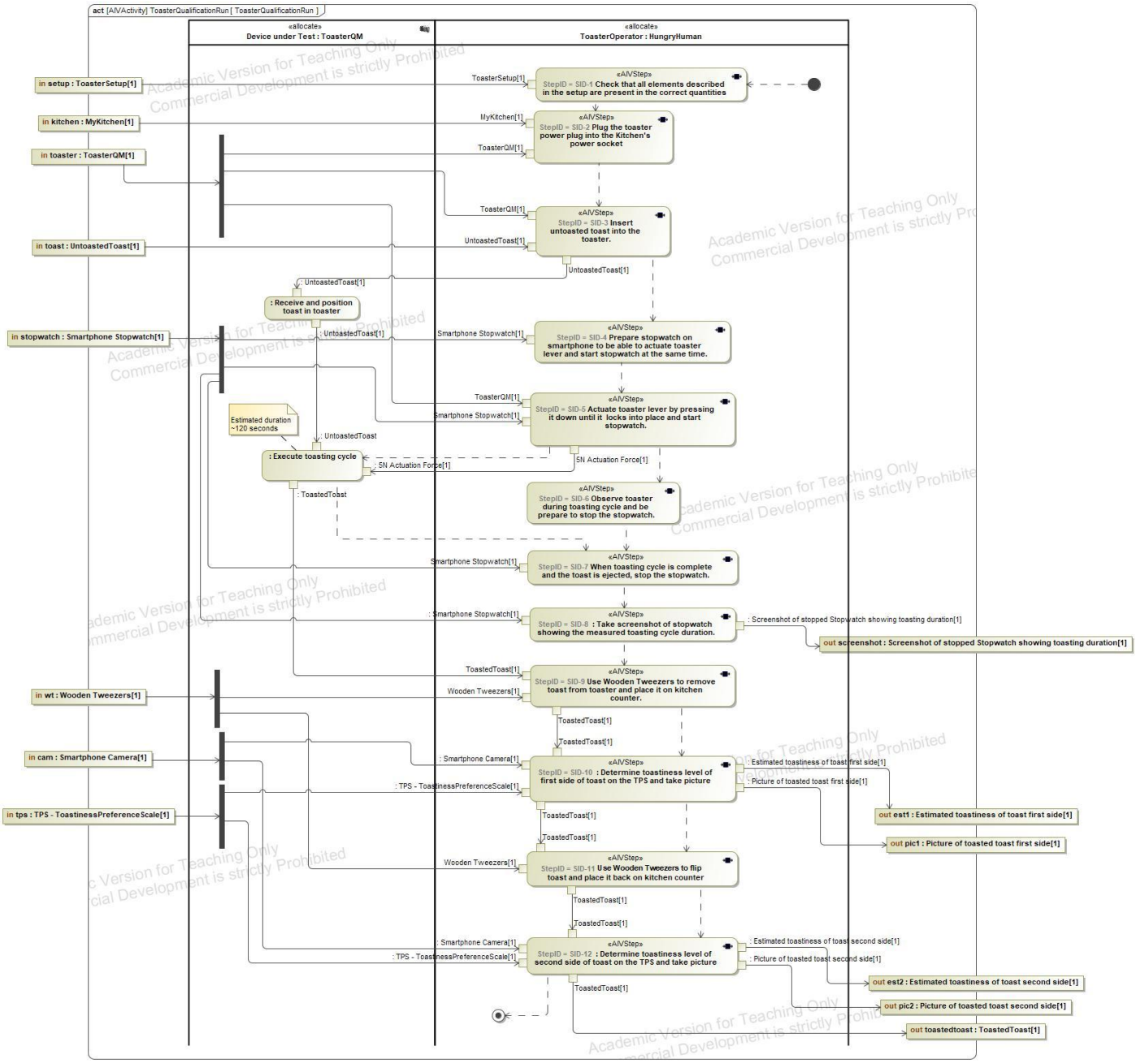
Crit ID	Name	Criterion Text	P/F	Evaluation Notes	Results
NA	NA	NA			

StepID	Description	Input	Output	Remarks	Date/Time/ Signature
SID-12	NA	- ToastedToast - Smartphone Camera - TPS - ToastinessPreferenceScale	- Picture of toasted toast second side - Estimated toastiness of toast second side - ToastedToast		

Crit ID	Name	Criterion Text	P/F	Evaluation Notes	Results
Crit-1	Toastiness of Toast within 5 to 8	The toasted toast has a toastiness level corresponding to at least a 5 and at most an 8 on the toastiness scale.		- The Toastiness Preference Scale should be attached - Indicate an integer number for front and back side of toast	

5 OTHER DIAGRAMS FROM MODEL

SysML Activity Diagram: ToasterQualificationRun



NA

Generic Table: RunningToasterProcedureSteps

#	Step ID	Name	Behavior	Argument	inMult	Result	outMult
1	SID-1	Check that all elements described in the setup are present in the correct quantities		03 ToasterAIV::AIVActivity - RunningToaster:: ToasterSetup[1]	1		
2	SID-2	Plug the toaster power plug into the Kitchen's power socket		03 ToasterAIV::AIVEntities:: ToasterQM[1] 03 ToasterAIV::AIVEntities:: MyKitchen[1]	1		
3	SID-3	Insert untoasted toast into the toaster.		03 ToasterAIV::AIVEntities:: ToasterQM[1] 03 ToasterAIV::AIVEntities:: UntoastedToast[1]	1	03 ToasterAIV::AIVEntities:: UntoastedToast[1]	1
4	SID-4	Prepare stopwatch on smartphone to be able to actuate toaster lever and start stopwatch at the same time.		03 ToasterAIV::AIVEntities:: Smartphone Stopwatch[1]	1		
5	SID-5	Actuate toaster lever by pressing it down until it locks into place and start stopwatch.		03 ToasterAIV::AIVEntities:: ToasterQM[1] 03 ToasterAIV::AIVEntities:: Smartphone Stopwatch[1]	1	02 ToasterDesign::ExchangeItems:: 5N Actuation Force[1]	1
6	SID-6	Observe toaster during toasting cycle and be prepare to stop the stopwatch.					
7	SID-7	When toasting cycle is complete and the toast is ejected, stop the stopwatch.		03 ToasterAIV::AIVEntities:: Smartphone Stopwatch[1]	1		
8	SID-8		Take screenshot of stopwatch showing the measured toasting cycle duration.	cam : 03 ToasterAIV::AIVEntities:: Smartphone Stopwatch[1]	1	screen : 03 ToasterAIV::AIV Evicence::Screenshot of stopped Stopwatch showing toasting duration[1]	1
9	SID-9	Use Wooden Tweezers to remove toast from toaster and place it on kitchen counter.		03 ToasterAIV::AIVEntities:: ToastedToast[1] 03 ToasterAIV::AIVEntities:: Wooden Tweezers[1]	1	03 ToasterAIV::AIVEntities:: ToastedToast[1]	1
10	SID-10		Determine toastiness level of first side of toast on the TPS and take picture(classifier behavior)	03 ToasterAIV::AIVEntities:: ToastedToast[1] cam : 03 ToasterAIV::AIVEntities:: Smartphone Camera[1] tps : 03 ToasterAIV::AIV Evicence::TPS - ToastinessPreferenceScale[1]	1	pic1 : 03 ToasterAIV::AIV Evicence:: Picture of toasted toast first side[1] est1 : 03 ToasterAIV::AIV Evicence:: Estimated toastiness of toast first side[1] 03 ToasterAIV::AIVEntities:: ToastedToast[1]	1
11	SID-11	Use Wooden Tweezers to flip toast and place it back on kitchen counter		03 ToasterAIV::AIVEntities:: ToastedToast[1] 03 ToasterAIV::AIVEntities:: Wooden Tweezers[1]	1	03 ToasterAIV::AIVEntities:: ToastedToast[1]	1
12	SID-12		Determine toastiness level of second side of toast on the TPS and take picture	03 ToasterAIV::AIVEntities:: ToastedToast[1] cam : 03 ToasterAIV::AIVEntities:: Smartphone Camera[1] tps : 03 ToasterAIV::AIV Evicence::TPS - ToastinessPreferenceScale[1]	1	pic2 : 03 ToasterAIV::AIV Evicence:: Picture of toasted toast second side[1] est2 : 03 ToasterAIV::AIV Evicence:: Estimated toastiness of toast second side[1] 03 ToasterAIV::AIVEntities:: ToastedToast[1]	1

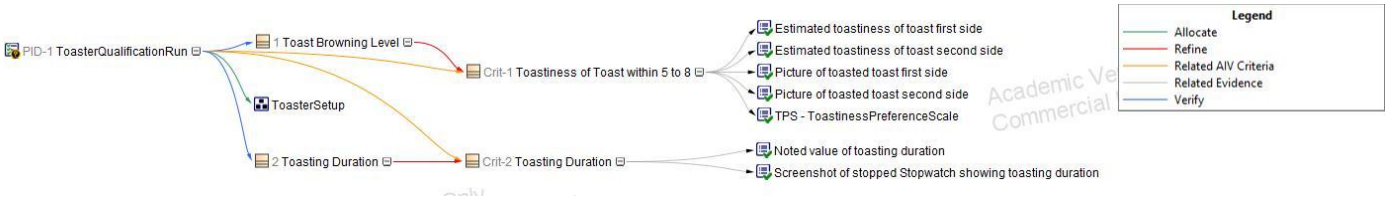
NA

Generic Table: InsOuts Table

#	Name	Type	Direct...	Multiplici...	ResponsibleStakeholder
1	cam	Smartphone Camera	in	1	HungryHuman
2	kitchen	MyKitchen	in	1	KitchenOwner
3	setup	ToasterSetup	in	1	HungryHuman
4	stopwatch	Smartphone Stopwatch	in	1	HungryHuman
5	toast	UntoastedToast	in	1	HungryHuman
6	toaster	ToasterQM	in	1	KitchenOwner
7	tps	TPS - ToastinessPreferenceScale	in	1	HungryHuman
8	wt	Wooden Tweezers	in	1	KitchenOwner
9	est1	Estimated toastiness of toast first side	out	1	HungryHuman
10	est2	Estimated toastiness of toast second side	out	1	HungryHuman
11	pic1	Picture of toasted toast first side	out	1	HungryHuman
12	pic2	Picture of toasted toast second side	out	1	HungryHuman
13	screenshot	Screenshot of stopped Stopwatch showing toasting duration	out	1	HungryHuman
14	toastedtoast	ToastedToast	out	1	HungryHuman

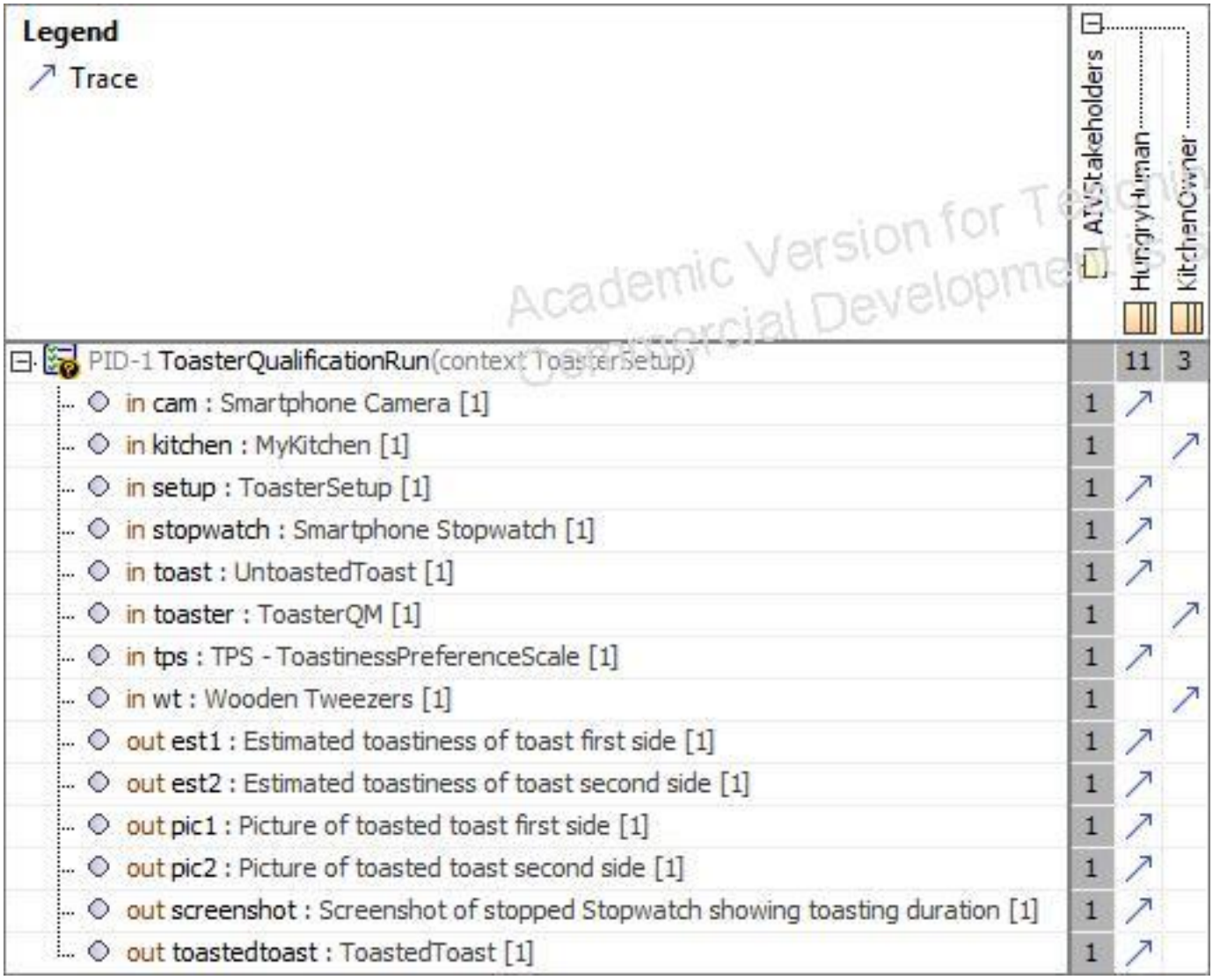
NA

Relation Map Diagram: ToasterQualificationRun



NA

Dependency Matrix: ToasterQualificationRun



NA

E

Procedure Generation Template

The following page shows the full template needed to generate the procedure for an AIV Activity in the MBSE AIV Methodology.

$COMPANYNAME – $PROJECT

$DocumentTitle

Written by		**Reviewed by**	
Signature		**Signature**	

Operator 1		**Operator 2**	
Signature		**Signature**	

Estimated test duration		**Actual test duration**	

Document generated by: $Author
Issue: $Revisions.lastChild.name
Reference: XX00.YY-Z
Date of Issue: $date.get('dd MMMMM, yyyy')

TUDelft

esa

1 TABLE OF CONTENTS

2 SCOPE OF THE ACTIVITY

Reasoning behind the test. Explain why it was needed, what type of test it is, whether it's functional, operational, integrated etc.

Requirements to verify are in the following page.

#import('helper', 'com.nomagic.requirements.reporthelper.SysMLReportHelper')

Getting all AIVActivities, AIVSteps, AIVProcedures from all elements in the scope

#set($AIVActList = $array.createArray())

#set($AIVStepList = $array.createArray())

#set($AIVCritList = $array.createArray())

#set($AIVSetupList = $array.createArray())

#set($debug = 0) ## set false to silence debug printing

#foreach($e in $elements)

#set($isAct = $report.containsStereotype($e, "AIVActivity"))

#set($isStep = $report.containsStereotype($e, "AIVStep"))

#set($isSetup = $report.containsStereotype($e, “AIVSetup”))

#set($isCrit = $report.containsStereotype($e, “AIVCriterion”))

#if($debug == 1)Name: $e.name | AIVActivity: $isAct | AIVStep: $isStep | AIVSetup: $isSetup

#end

#if($isAct)#set($ignore = $AIVActList.add($e))

#end

#if($isStep)#set($ignore = $AIVStepList.add($e))

#end

#if($isCrit)#set($ignore = $AIVCritList.add($e))

#end

#if($isSetup)#set($ignore = $AIVSetupList.add($e))

#end

#end

#set($AIVAct = $AIVActList.get(0))

Title of the Activity

$AIVAct.name

Description

$AIVAct.Documentation

2.1 REQUIREMENTS TO VERIFY

The table below presents an overview of the requirements to be verified.

#set($reqs = $array.createArray())

#set($crits = $array.createArray())

#foreach($verEl in $AIVAct.verifies)

#set($isReq=$report.containsStereotype($verEl, “DVSRequirement”))

#if($isReq)#set($void=$reqs.add($verEl))#end

#end

Req ID	Req Title	Req Text	Related Criteria
#forrow($req in $sorter.humanSort($reqs, “id”))$req.id	$req.name	$req.text	#foreach($crit in $req.RefinedBy) #if($AIVCritList.contains($crit)) $crit.id | $crit.name #end#end#endrow

2.2 CRITERIA TO CHECK

Crit ID	Crit Name	Crit Text	Evaluation Notes	Related Requirements	Related AIV Step
#forrow($crit in $sorter.humanSort($AIVCritList, "CritID"))$crit.id	$crit.name	$crit.text	#foreach($note in $crit.EvaluationNotes) - $note #end	#foreach($req in $crit.Refines) #if($reqs.contains($req)) $req.id | $req.name #end#end	#set($shared=$array.createArray()) #foreach($ob in $crit.OwnedBehavior)#set($void = $shared.addAll($ob.get_callBehaviorActionOfBehavior()))#end #set($void=$shared.retainAll($AIVStepList)) #foreach($crit_step in $shared) $crit_step.StepID | $crit_step.Behavior.name #end#endrow

3 SETUP & EQUIPMENT

3.1 SETUP

Describe how the test is set up, how the hardware is secured and how everything looks like.

Get diagrams owned by related AIVSetup Model Element

#foreach($AIVSetup in $AIVSetupList)

#foreach($dia in $AIVSetup.getOwnedDiagram())

$dia.diagramType: $dia.name

$dia.image

$dia.Documentation

#end

Print Setup documentation

#if($AIVSetup.Documentation && $AIVSetup.Documentation != “”)

Setup documentation

$AIVSetup.Documentation

#end

#end

3.2 GSE & OTHER EQUIPMENT

The following Equipment shall be used for the test activity.

#set($pars = $AIVActList.get(0).getOwnedParameter())

```
#set($in_pars = $array.createArray())

#set($out_pars = $array.createArray())

#set($int_products = $array.createHashSet())

#set($pinTypes = $array.createArray())

#foreach($vvstep in $AIVStepList)#foreach($pin in $vvstep.getInput())#if($pin.getType())#set($void =
$pinTypes.add($pin.getType()))#end#end#foreach($pin in $vvstep.getOutput())#if($pin.getType())#set($void =
$pinTypes.add($pin.getType()))#end#end#end

#foreach($par in
$pars)#if($par.getDirection().toString()==”in”)#set($void=$in_pars.add($par))#elseif($par.getDirection().toString()==”out”)#set($void=$out_pars.ad
d($par))#end#end

#foreach($ptype in $pinTypes)#set($found=false)#foreach($par in
$pars)#if($par.getType()==$ptype)#set($found=true)#break#end#end#if(!$found)#set($void=$int_products.add($ptype))#end#end
```

3.2.1 INPUTS

```
#set($counter = 1)

#foreach($par in $in_pars)
```

Number	Item	Quantity	relatedStep	Check	Comments
$counter #set($counter =	$par.name : $par.Type.name	$par.Multiplicity	#set($stepIdSet = $array.createHashSet())		$par.Documentation $par.Type.Documentation

$counter + 1)			#foreach($pin in $par.Type.get_typedElementOfType())#set($stepId = $report.getStereotypePropertyString($pin.Owner, "AIVStep", "StepID"))#if($stepId && $stepId != "")#set($void=$stepIdSet.add($stepId))#end#end#foreach($id in $sorter.humanSort($stepIdSet))$id #end		
Extra Remarks:					

116 #end

3.2.2 OUTPUTS

#set($counter = 1)

#foreach($par in $out_pars)

Number	Item	Quantity	relatedStep	Check	Comments
$counter #set($counter = $counter + 1)	$par.name : $par.Type.name	$par.Multiplicity	#set($stepIdSet = $array.createHashSet()) #foreach($pin in $par.Type.get_typedElementOfType())#set($stepId = $report.getStereotypePropertyString($pin.Owner, "AIVStep", "StepID"))#if($stepId && $stepId != "")#set($void=$stepIdSet.add($stepId))#end#end#foreach($id in $sorter.hum		$par.Documentation $par.Type.Documentation

			anSort($stepIdSet))$id #end		
Extra Remarks:					

#end

3.2.3 INTERMEDIATE ELEMENTS

#set($counter = 1)

#foreach($e in $int_products)

Number	Item	Quantity	relatedStep	Check	Comments
$counter #set($counter = $counter + 1)	$e.name	NA	#set($s=$array.createHashSet())#foreach($pin in $e.get_typedElementOfType())#if($report.containsStereotype($pin.Owner,"AIVStep"))#set($sid=$report.getStereotypePropertyString($pin.Owner,"AI	NA	$e.Documentation

			VStep","Step ID"))#if($sid && $sid!="")#set($void=$s.add($sid))#end#end#end #foreach($id in $sorter.humanSort($s))$id #if($foreach.hasNext)#end#end		
Extra Remarks:					

#end

4 STEP-BY-STEP PROCEDURE

Note: fill in the columns related to “pass/fail criterion” only need to be considered if there is a criterion to be verified, otherwise to be ignored.

#foreach($vvstep in $sorter.humanSort($AIVStepList, “StepID”))

<table>
<tr><th>StepID</th><th>Description</th><th>Input</th><th>Output</th><th>Remarks</th><th>Date/Time/ Signature</th></tr>
<tr><td rowspan="3">$vvstep.tags.get("AIV Step").get("StepID")</td><td>#set($behav = $vvstep.getBehavior())#set($hasProc = $report.containsStereotype($behav, "AIVProcedure"))
#if($hasProc)$behav.name
#{else}$vvstep.name#end</td><td>#foreach($in_pin in $vvstep.getInput())
- $in_pin.Type.name
#end</td><td>#foreach($out_pin in $vvstep.getOutput())
- $out_pin.Type.name
#end</td><td></td><td rowspan="3"></td></tr>
<tr><td colspan="4"><table>
<tr><th>Crit ID</th><th>Name</th><th>Criterion Text</th><th>P/F</th><th>Evaluation Notes</th><th>Results</th></tr>
<tr><td>$behav.owner.id</td><td>$behav.owner.name</td><td>$behav.owner.Text</td><td></td><td>#foreach($note in $behav.owner.EvaluationNotes)
- $note
#end</td><td></td></tr>
</table></td></tr>
</table>

#end

5 OTHER DIAGRAMS FROM MODEL

#foreach($dia in $AIVAct.getOwnedDiagram())

$dia.diagramType: $dia.name

$dia.image

$dia.Documentation

```
138
139     #end
```

F

FEMMP Evaluation

The FEMMP Evaluation follows the method described in [65]. Table F.1 shows the types of evaluation metrics used in the FEMMP framework. Table F.2 shows an explanation for the qualitative assessment scale used.

Table F.1: Evaluation metric types used in the framework [65]

Metric Type	**Explanation**
Yes/No Question	A binary assessment indicating whether a criterion is fulfilled or not. The answer includes a free-text justification.
Statement	A free-text statement that answers the question.
Selection/List	A list-based response in which the relevant items are named and explained.
Qualitative Assessment Scale	A graded assessment scale used to evaluate how well a criterion is addressed. The scale distinguishes between full compliance, acceptable performance, limited applicability, generalisation, and not addressed.

Table F.2: Qualitative assessment scale [65]

Grade	**Name**	**Explanation**
A	Fully Compliant	The methodology covers the item exhaustively and addresses it well.
B	Acceptable Performance	Minor constraints or limitations apply, but they are documented well.
C	Limited Applicability	Major constraints or limitations apply that require considerable extra effort, cumbersome workarounds, or extensive customisation.
G	Generalisation	Compliance is claimed, but no conclusive information on the practical application is provided.
X	Not Addressed	The criterion is not addressed, not implicit, and no reasons for its omission are provided.

Table F.3: Evaluation criteria for MBSE methodology assessment [65]

Title	Description	ID	Wgt	Aspect	Category	Response Type
Learning Curve	How steep does the learning curve feel?	G-A-01	3	General	Adopting	Scale
Suitability for Beginners	How easy to understand is the methodology for (MB)SE novices?	G-A-02	3	General	Adopting	Statement
Training Effort for SEs	How much training does it take for experienced SEs to implement the methodology correctly?	G-A-03	2	General	Adopting	Statement
Industry Domains	Which industry domains does the methodology support particularly well/ has it been developed for?	G-B-01	1	General	Basics	
Creativity Support	How does the methodology foster a creative environment for the systems engineering team?	G-B-02	2	General	Basics	Statement
Support of Other Standards	Does the methodology also support other standards or norms? If so, how well?	G-B-03	1	General	Basics	Statement
Process Info Capture	How well does the tool capture the information generated throughout the process?	I-B-01	3	Information	Basics	Scale
MBSE-SE Info Exchange	How well does the tool facilitate the exchange of the information between the MBSE domain and the non-MBSE tools	I-B-02	3	Information	Basics	Scale
Philosophy	Are Model Elements clearly distinguished from Diagram Elements (separation of content from representation)?	L-B-01	3	Language	Basics	Yes / No
Modelling Features	What modelling features and approaches are included from the reference framework ISO 42010 (Architecture)	M-B-01	2	Model	Basics	List
Modelling ISO 15288	How well does the MBSE methodology support modelling of the ISO 15288 processes?	M-B-02	3	Model	Basics	Scale
Abstraction	How well does the model support keeping it abstract through multiple levels?	M-B-03	2	Model	Basics	Scale
Meta-Model	Does the methodology provide a (semantic) meta-model, ontology or similar?	M-B-04	3	Model	Basics	Yes / No
ISO Standard	What process steps of ISO 15288 are covered?	P-B-01	3	Process	Basics	List
Framework	What views from the reference framework are used (MODAF, DODAF...)?	P-B-02	3	Process	Basics	List
Precision	How precisly is the process implemented in terms of semantics and sequence, i.e. is it strongly enforced, or can 'wildcard' be used as 'work arounds' that would reduce the model quality?	T-B-01	3	Tool	Basics	Scale
Information Security	How secure is the data/information exchange between parties	T-B-02	2	Tool	Basics	Scale
Perspectives	To what level is the creation of experts' perspectives automated	T-E-01	1	Tool	Efficiency	Scale
Reporting	How quickly are standard/custom reports, is design documentation created	T-E-02	1	Tool	Efficiency	Scale
Admin	How well does the tool help to minimise work that isn't creating any value	T-E-03	1	Tool	Efficiency	Scale
Checking	Does the tool support consistency checking of the model? (Automated detection of wrong content and/or formats, flagging of , 'loose ends' etc.)?	T-E-04	2	Tool	Efficiency	Yes / No
Navigation	How easy is it to find the correct model element (are elements links, users guided in the process, information well aggregated, need to 'jump' screens)?	T-E-05	2	Tool	UX	Scale
Intuitition	How intuitive is the tool to work with (compliance with UX conventions, standard tool reactions e.g. tool tips, double/right click, drag&drop, delete, Keyboard shortcuts, spell check, familiar operations e.g. as MS-Office)?	T-U-01	2	Tool	UX	Scale
View	How easy is it to configure the UX dynamically (define a matrix with sorting & filtering of columns and rows, store customised view, annotation, comment)?	T-U-02	1	Tool	UX	Scale
UI	How readable is the UI (good use of screen estate and colour, zoom, can fonts and sizes be changed, is information well presented)?	T-U-03	1	Tool	UX	Scale
Multi-User Environment	How well does the tool support the distribution of the work among different parties.	T-U-04	2	Tool	UX	Scale
Scope	For what engineering purpose is the methodology suited(innovation, improved products, refactoring, reverse engineering, etc.)?	G-P-01	2	General	Practicality	List

Continued on next page

Title	Description	ID	Wgt	Aspect	Category	Response Type
Tailoring	How easy is it to tailor the methodology (add, delete or change processes or process steps, object definitions or toggle tool features on and off)?	G-P-02	3	General	Practicality	Scale
Complexity	How often is the methodology 'interrupted' by relying on external processes and/or non integrated tools, e.g. by a 'paper-review'?	G-P-03	2	General	Practicality	Scale
Simulation	How well does the methodology provide for an integrated simulation of the various model abstractions?	G-P-04	2	General	Practicality	Statement
Language	What Modelling Language is used (If NOT SysML: How well does it define the real-world semantics of the engineering, are elements strictly typed, is their meaning unambiguous, do they have a defined purpose etc.)?	L-P-01	3	Language	Practicality	List
Integration	How well can the model be integrated with specialty engineering models	L-P-02	1	Language	Practicality	Scale
Scalability	How well does the model scale (suitable for large projects, 'grows' with time without becoming cumbersome, does it require partitionaing e.g. in a tree)?	M-P-01	3	Model	Practicality	Scale
Variants	How well does the methodology support the variant management?	M-P-02	2	Model	Practicality	Scale
Independent View Generation	How well does the methodology support the generation of views that can be read in another MBSE language?	M-P-03	2	Model	Practicality	Scale
Redundancy	How well does the methodology prevent duplication (of work, model elements, artefacts, communications and reports)?	M-P-04	2	Model	Practicality	Statement
Consistency	Is the process self-contained (are in-/outputs to all steps connected)?	P-P-01	3	Process	Practicality	Yes / No
Connectivity	How easily can the information be exchanged with other tools (what standard API are provided by the tool, what API can be added, Is im-/export open protocols & guided by a wizard, can it be rolled–back, what the quality control mechanism etc.)?	T-P-01	3	Tool	Practicality	Scale
Reuseablity	Does the tool allow to reuse any type of Modelling Element across projects	T-P-02	2	Tool	Practicality	Yes / No
Documentation	How well is the methodology supported (books, manuals, case studies, on-line help, community, websites, interactive support, user feedback etc.)?	G-S-01	3	General	Support	Scale
Training	How well is training supported (availability, consultants, coaches, e-training, background knowledge required)?	G-S-02	1	General	Support	Scale
Support	How well is the tool supported (vendor response times, 24/7 helpline etc.)?	T-S-01	1	Tool	Support	Scale

Table F.4: FEMMP evaluation table

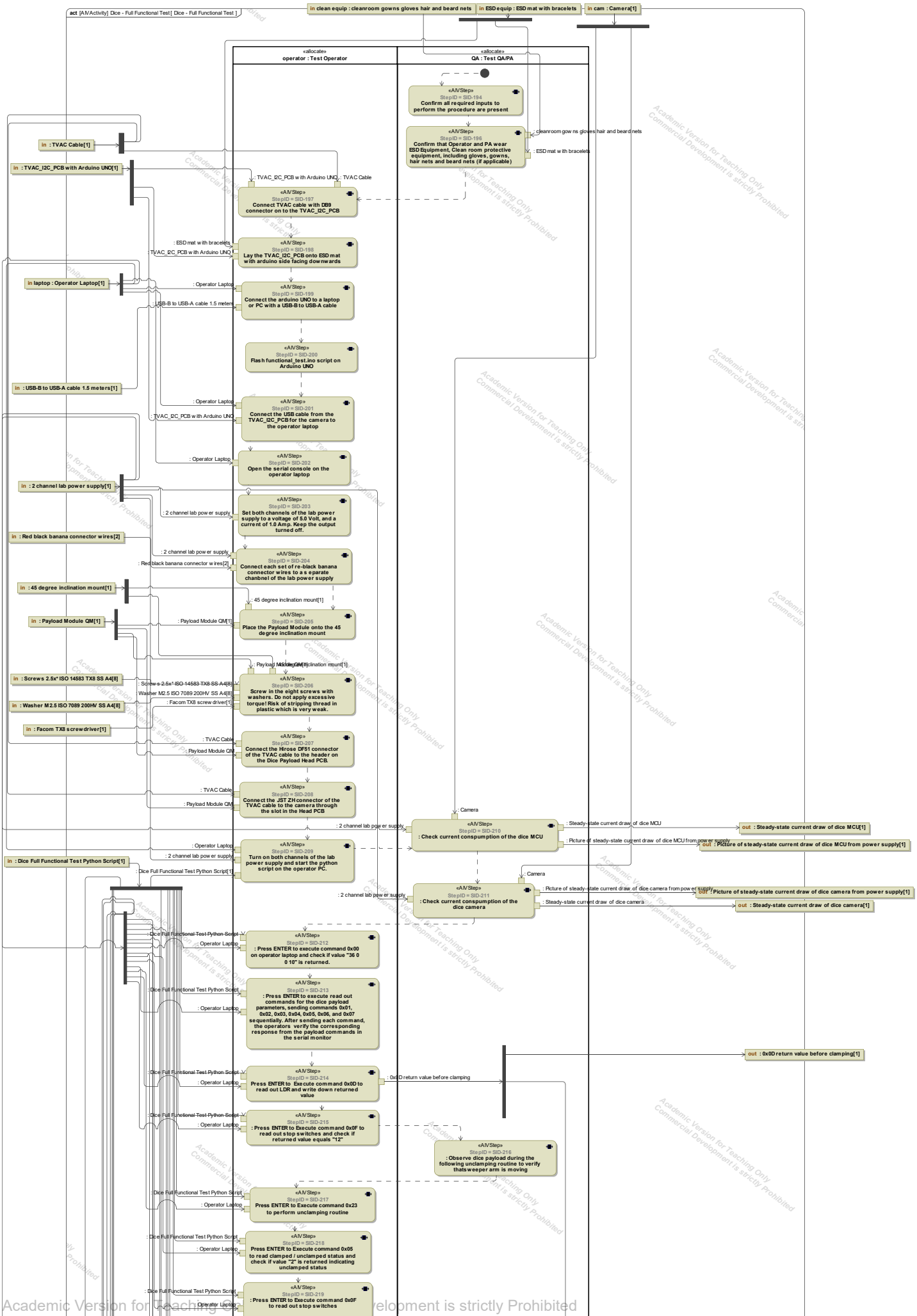
ID	Criterion Title	Evaluation	Rationale
G-A-01	Learning Curve	G	Application of this methodology requires knowledge of MBSE and SysML, but easy to learn for experienced users due to the metamodel and process description and the model view templates, adapting templates for document generation or tailoring of the methodology requires in depth knowledge of VTL and SysML.
G-A-02	Suitability for Beginners	G	Approach is unsuitable for beginners without SE or MBSE experience.
G-A-03	Training Effort for SEs	-	With a solid foundation of MBSE and SysML, additional training effort for this methodology is minimal
G-B-01	Industry Domains	-	Tailored for space industry common terminology
G-B-02	Creativity Support	-	Not evaluated.
G-B-03	Support of Other Standards	-	The methodology is aligned with relevant ECSS and ISO Standards
I-B-01	Process Info Capture	A	All information that is generated is captured in the central model.
I-B-02	MBSE-SE Info Exchange	A	Tool allows to export all information in accessible format, e.g. CSV, and custom tool integrations exist, e.g. MATLAB
L-B-01	Philosophy	Yes	The selected modeling language (SysML) provides this separation.
M-B-01	Modelling Features	-	Not evaluated
M-B-02	Modelling ISO 15288	A	Supports the SE processes Integration, Verification, and Validation in ISO 15288
M-B-03	Abstraction	A	The methodology supports activities on different abstraction levels, from high level AIV process planning to detailed definition of procedures, and post-processing of collected evidence, including propagating the evidence evaluation back up the abstraction layers, through criteria and requirements closeout.
M-B-04	Meta-Model	Yes	A full ontology and its implementation is part of the methodology.
P-B-01	ISO Standard	-	Integration, Verification, Validation
P-B-02	Framework	-	model views correspond to ECSS DRD framework (VCD and procedure documents).
T-B-01	Precision	B	Correct modeling practice is encouraged by tool customization, e.g. suggesting correct model elements on creation, not allowing wrong elements to be created, standardized model views like VCD. Users can circumvent this and it doesn't guarantee fault-free modeling.
T-B-02	Information Security	-	not assessed.
T-E-01	Perspectives	B	Implemented through the VCD model view and automatic procedure generation.
T-E-02	Reporting	B	Only for procedures, the document generation capabilities were implemented, tool allows to create more document templates and extend following the same process.
T-E-03	Admin	A	The tool helps significantly in generating a full test procedure once modeling is done, which would otherwise be additional work. The methodology helps to ensure traceability throughout the life cycle.
T-E-04	Checking	B	Some automated consistency checks are implemented, others rely on model views that present model content in an easily reviewable manner, e.g. overview tables, traceability views, SysML dependency matrices.
T-E-05	Navigation	A	Model organization follows a clear pattern. Model elements traceability is defined in the metamodel and can be fully queried in the model.
T-U-01	Intuitition	-	not assessed
T-U-02	View	A	Easy to do for experienced and novel users of the tool.
T-U-03	UI	-	not assessed, but the tool is used by many organizations in the industry.
T-U-04	Multi-User Environment	A	Collaborative server setup and user management provided by the tool.
G-P-01	Scope	-	Scope of the methodology is to support Integration, Verification, and Validation in a model-based environment.
G-P-02	Tailoring	A	Dedicated process model that shows process step dependencies and informs tailoring. Metamodel implementations and tool customizations can be tailored.

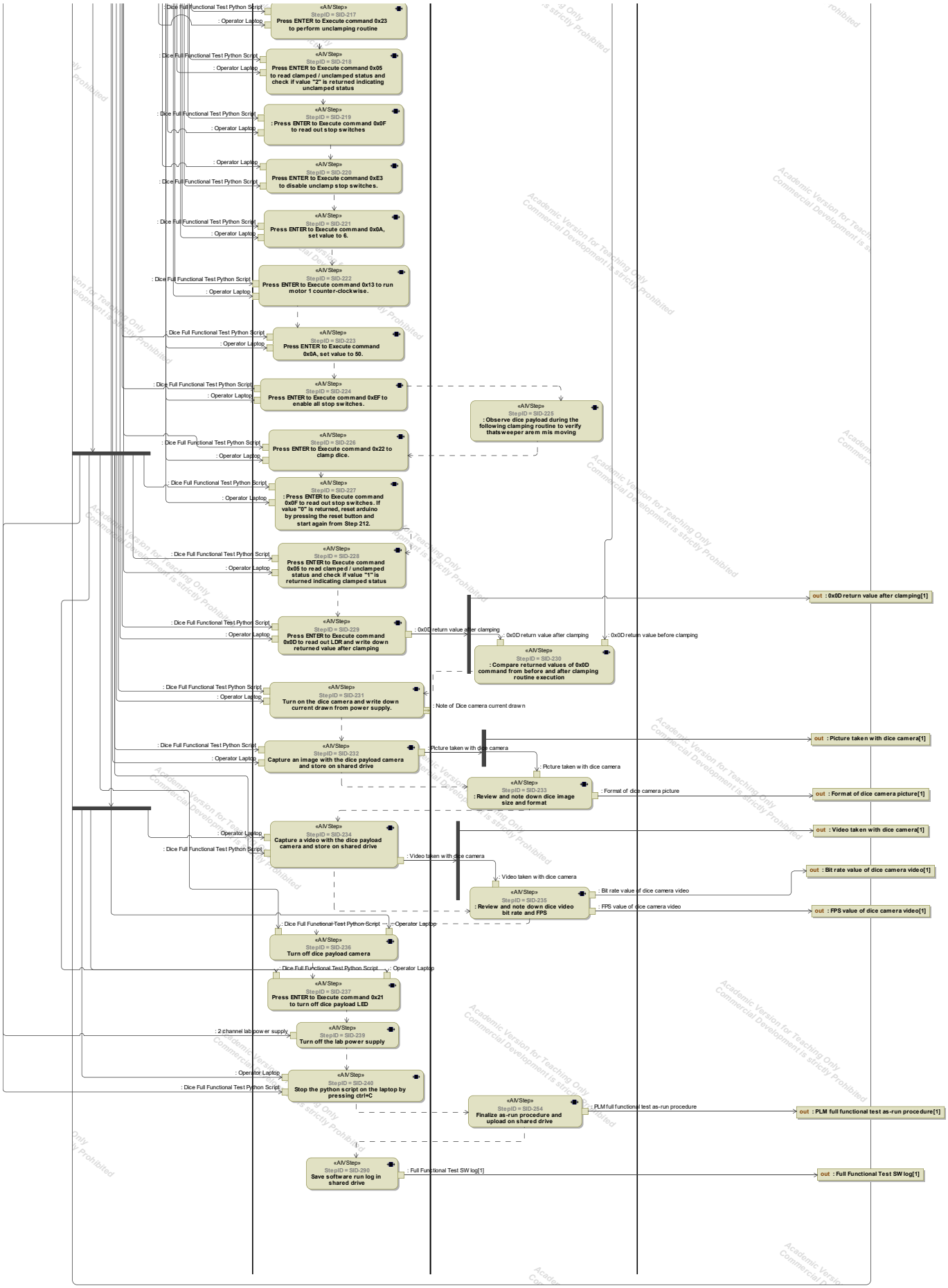
ID	Criterion Title	Evaluation	Rationale
G-P-03	Complexity	B	Interruptions in modeling can occur at review activities or when performing AIV activities. Planning and preparing for these events is the main purpose of the methodology.
G-P-04	Simulation	A	Tool supports simulation of full flow of AIV activities or a single procedure.
L-P-01	Language	-	SysML v1 is used for modeling.
L-P-02	Integration	-	not assessed.
M-P-01	Scalability	A	The methodology was applied to a realistic case study, without showing major additional hurdles.
M-P-02	Variants	A	Variants of the product are explicitly supported through the use of AIV Model Elements (Qualification Model, Flight Model, etc.)
M-P-03	Independent View Generation	-	not evaluated
M-P-04	Redundancy	A	The model-based approach allows to reference model elements across views. Duplication should not happen if proper modeling process is followed. Consistency checks help to highlight redundancy related issues.
P-P-01	Consistency	Yes	For all modeling elements and actual verification evidence, it is explained how to use them in subsequent steps.
T-P-01	Connectivity	C	Tool has an open API, but using it is not easy.
T-P-02	Reuseablity	A	The methodology allows to reuse various model elements across verification scopes.
G-S-01	Documentation	B	Detailed description of how to apply the methodology, and all related work packages are available in the frame of this thesis.
G-S-02	Training	-	not assessed
T-S-01	Support	X	

G

Modeled Functional Test Procedure

The following pages show the modeled test procedure as an activity diagram. This forms the basis for generating the step-by-step procedure document and can be simulated.





H

Diagrams from the model

This Annex contains several model diagrams referenced in the main text.

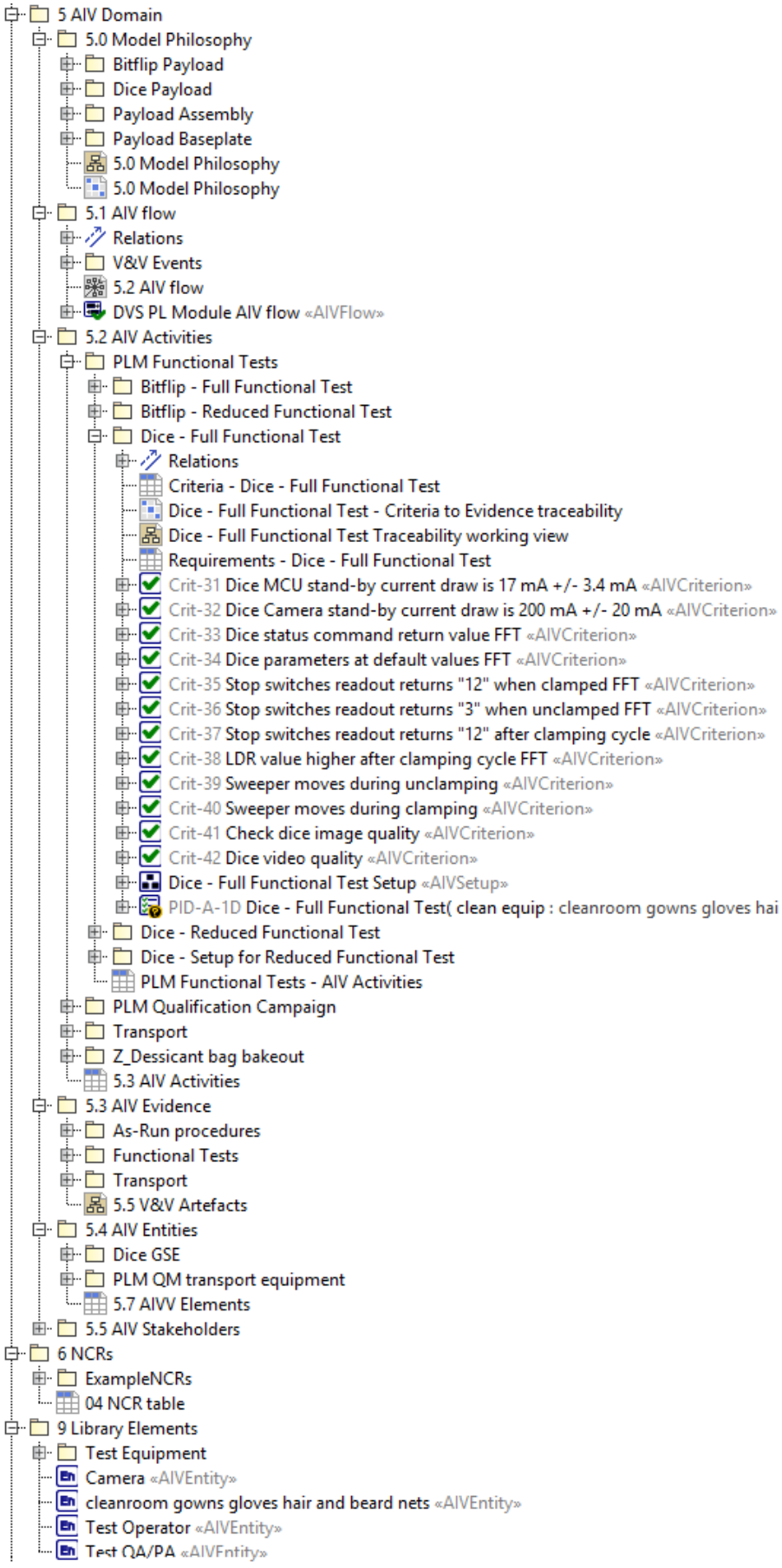


Figure H.1: Detailed Containment Tree Structure for AIV Modeling

#	Name	Documentation	relatedModelElement
1	<1> Introduction	The VP shall contain a description of the purpose, objective, content and the reason prompting its preparation. Open issues, assumptions and constraints relevant to this document shall be stated and described.	
2	<2> Applicable and Reference Documents	The VP shall list the applicable and reference documents in support to the generation of the document.	
3	<3> Definitions and abbreviations	The VP shall list the applicable dictionary or glossary and the meaning of specific terms or abbreviations utilized in the document.	
4	<4> Verification subject	The VP shall briefly describe the subject of the verification process.	AIVEntity [Class]
5	<5> Verification approach	The VP shall describe the basic verification concepts and definitions (methods, levels and stages).	+method : SysML::Non-Normative Extensions::Requirement::VerificationMethodKind [1] +stage : String [1] +level : String [1]
6	<6> Model philosophy	The VP shall describe the selected models and the associated model philosophy, product matrix.	AIVEntity [Class]
7	<7> Verification Strategy	a. The VP shall describe the selected combination of the different verification methods at the applicable verification levels and stages, in general and for each requirement type/group (including software). b. The allocation of the requirements to the specific verification tasks shall be given.	AIVActivity [Activity] +level : String [1] +method : SysML::Non-Normative Extensions::Requirement::VerificationMethodKind [1] +stage : String [1] DVSRequirement [Class]
8	<8> Verification programme	a. The VP shall document the verification activities and associated planning in the applicable verification stages. b. Analysis, review-of-design, inspection and test programmes should be detailed through dedicated activity sheets, or through reference to the AIT Plan.	AIVActivity [Activity] +level : String [1] +stage : String [1] +method : SysML::Non-Normative Extensions::Requirement::VerificationMethodKind [1] AIVFlow [Activity] AIVEvent [Activity]
9	<9> Verification tools	The VP shall describe high level definitions of the verification tools to be used, such as S/W facilities, special tools, simulators, analytical tools.	AIVEntity [Class]
10	<10> Verification control methodology	The VP shall describe the proposed methodology to be utilized for verification monitoring and control including the use of a verification data base.	
11	<11> Documentation	The VP shall list the involved verification documents and describe their content.	
12	<12> Organization and management	a. The VP shall describe the responsibility and management tools applicable to the described verification process. b. It shall describe the responsibilities within the project team, the relation to product assurance, quality control and configuration control (including anomaly handling and change control) as well as responsibility sharing with external partners. c. The relevant reviews shall be planned and responsibilities described.	

Figure H.2: ECSS Verification Plan (VP) DRD structure (ECSS-E-ST-10-02C Rev.1 Annex A)

#	Name	Documentation	relatedModelElement
1	<1> Introduction	a. The VCD shall contain a description of the purpose, objective, content and the reason prompting its preparation. b. Open issues, assumptions and constraints relevant to this document shall be stated and described. c. The VCD content shall be phased with the product life-cycle such that the initial issue contains the verification matrix, intermediate issues cover the planned on-ground verifications and their executions evidence (in particular for qualification and acceptance completion), the in-orbit and post landing activities; final issue provides evidence of the close-out of the overall verification process.	
2	<2> Applicable and reference documents	The VCD shall list the applicable and reference documents in support to the generation of the document.	
3	<3> Definitions and abbreviations	The VCD shall list the applicable dictionary or glossary and the meaning of specific terms or abbreviations utilized in the document.	
4	<4> Verification subject	a. The VCD shall describe the verification control approach applied to the product, the involved documentation and the computerized tool used to support the process. b. The VCD shall include the requirements to be verified (with reference to the specifications involved), call up the verification methods, levels and stages definitions and explain the verification close-out criteria.	AIVEntity [Class] DVSRequirement [Class] AIVActivity [Activity] +level : String [1] +method : SysML::Non-Normative Extensions::Requirement::VerificationMethodKind [1] +stage : String [1] +CloseoutStatus : ESA Profile::Concept profile::Value Types::Close-out status type [1] +CloseoutStatusRationale : String [1] +Compliance Rationale : String [0..*] +Phase C Compliance : AIV-Profile::requirements::RequirementComplianceKind [1] +Phase D Compliance : AIV-Profile::requirements::RequirementComplianceKind [1] +Phase E Compliance : AIV-Profile::requirements::RequirementComplianceKind [1]
5	<5> Verification summary status	Each issue of the VCD shall summarize the current Verification Close-out status.	
6	<6> Verification control data	The VCD shall collect in the form of a matrix, for each requirement, the following verification information: 1. Requirement identifier, 2. Requirement text 3. traceability between requirement, 4. Levels and stages of verification, 5. methods, 6. link to the relevant section of the verification plan and any planning document, 7. References to any documentation that demonstrates compliance to the requirements, 8. Status of Compliance (yes, no, partial), 9. Close-out status (open/closed), 10. Reasons of the close-out status,	+Id : String [1] = +Text : String [1] = +/Derived : SysML::Requirements::AbstractRequirement [*] +/DerivedFrom : SysML::Requirements::AbstractRequirement [*] +level : String [1] +method : SysML::Non-Normative Extensions::Requirement::VerificationMethodKind [1] +stage : String [1] +CloseoutStatus : ESA Profile::Concept profile::Value Types::Close-out status type [1] +CloseoutStatusRationale : String [1] +Compliance Rationale : String [0..*] +Phase C Compliance : AIV-Profile::requirements::RequirementComplianceKind [1] +Phase D Compliance : AIV-Profile::requirements::RequirementComplianceKind [1] +Phase E Compliance : AIV-Profile::requirements::RequirementComplianceKind [1] +source : String [0..*] +/Master : SysML::Requirements::AbstractRequirement +/VerifiedBy : AIV-Profile::verification::AIVActivity [*]

Figure H.3: ECSS Verification Control Document (VCD) DRD structure (ECSS-E-ST-10-02C Rev.1 Annex B)

#	Name	Documentation	relatedModelElement
1	<1> Introduction	a. The TSPE shall contain a description of the purpose, objective, content and the reason prompting its preparation. b. Any open issue, assumption and constraint relevant to this document shall be stated and described.	
2	<2> Applicable and reference documents	The TSPE shall list the applicable and reference documents in support to the generation of the document.	
3	<3> Definitions and abbreviations	The TSPE shall list the applicable dictionary or glossary and the meaning of specific terms or abbreviations utilized in the document.	
4	<4> Requirements to be verified	The TSPE shall list the requirements to be verified (extracted from the VCD) in the specific test and provides traceability where in the test the requirement is covered.	DVSRequirement [Class]
5	<5> Test approach and test requirements	The TSPE shall summarize the approach to the test activity and the associated requirements as well as the prerequisites to start the test.	DVSRequirement [Class]
6	<6> Test description	The TSPE shall summarize the configuration of the item under test, the test set-up, the necessary GSE, the test tools, the test conditions and the applicable constraints.	AIVEntity [Class] AIVSetup [Class] AIVActivity [Activity] +vvKind : AIV-Profile::verification::AIVVActivityKind [1]
7	<7>Test facility	The TSPE shall describe the applicable test facility requirements together with the instrumentation and measurement uncertainties, data acquisition and test space segment equipment to be used	AIVEntity [Class]
8	<8> Test sequence	a. The TSPE shall describe the test activity flow and the associated requirements. b. When constraints are identified on activities sequence, the TSPE shall specify them including necessary timely information between test steps.	AIVActivity [Activity]
9	<9> Pass/fail criteria	The TSPE shall list the test pass/fail criteria in relation to the inputs and output.	AIVStep [CallBehaviorAction] AIVCriterion [Class]
10	<10> Test documentation	The TSPE shall list the requirements for the involved documentation, including test procedure, test report and PA and QA records.	AIVEvidence [Class, DataType]
11	<11> Test organization	The TSPE shall describe the overall test responsibilities, participants to be involved and the schedule outline.	AIVStakeholder [Class]

Figure H.4: ECSS Test Specification (TSPE) DRD structure (ECSS-E-ST-10-03C Rev.1 Annex B)

#	Name	Documentation	relatedModelElement
1	<1> Introduction	a. The TPRO shall contain a description of the purpose, objective, content and the reason prompting its preparation. b. Any open issue, assumption and constraint relevant to this document shall be stated and described.	
2	<2> Applicable and reference documents	The TPRO shall list the applicable and reference documents in support to the generation of the document.	
3	<3> Definitions and abbreviations	The TPRO shall list the applicable dictionary or glossary and the meaning of specific terms or abbreviations utilized in the document.	
4	<4> Requirements mapping w.r.t. the TSPE	The TPRO shall provide a mapping matrix to the TSPE giving traceability towards the test requirement.	DVSRequirement [Class]
5	<5> Item under test	a. The TPRO shall describe the item under test configuration, including any reference to the relevant test configuration list, and any deviation from the specified standard. b. The software version of the item under test shall be identified.	AIVEntity [Class]
6	<6> Test set-up	The TPRO shall describe the test set-up to be used.	AIVSetup [Class]
7	<7>GSE and test tools required	The TPRO shall identify the GSE and test tools to be used in the test activity including test script(s), test software and database(s) versioning number.	AIVEntity [Class]
8	<8> Test instrumentation	The TPRO shall identify the test instrumentation, with measurement uncertainties, to be used, including fixtures.	AIVEntity [Class]
9	<9> Test facility	The TPRO shall identify the applicable test facility and any data handling system.	AIVEntity [Class]
10	<10> Test conditions	The TPRO shall list the applicable standards, the applicable test conditions, in terms of levels, duration and tolerances, and the test data acquisition and reduction.	AIVActivity [Activity] AIVEvidence [Class, DataType] AIVCriterion [Class]
11	<11> Documentation	The TPRO shall describe how the applicable documentation is used to support the test activity.	
12	<12> Participants	The TPRO shall list the allocation of responsibilities and resources.	AIVSetup [Class] AIVStakeholder [Class]
13	<13> Test constraints and operations	a.The TPRO shall identify special, safety and hazard conditions, operational constraints, rules for test management relating to changes in procedure, failures, reporting and signing off procedure. b. The TPRO shall describe QA and PA aspects applicable to the test. c. The TPRO shall contain a placeholder for identifying: 1. procedure variations, together with justification, and 2. anomalies.	AIVActivity [Activity] AIVStakeholder [Class] AIVSetup [Class] +verdict : AIV-Profile::verification::CriterionVerdictKind [1] = TBD AIVCriterion [Class]
14	<14> Step-by-step procedure	a. The TPRO shall provide detailed instructions, including expected results, with tolerances, pass/fail criteria, and identification of specific steps to be witnessed by QA personnel. b. The step-by-step instructions may be organized in specific tables. c. When the procedure is automated, the listing of the automated procedure shall be documented to a level allowing consistency check with the TPRO and the TPSE.	AIVStep [CallBehaviorAction] +verdict : AIV-Profile::verification::CriterionVerdictKind [1] = TBD AIVCriterion [Class]

Figure H.5: ECSS Test Procedure (TPRO) DRD structure (ECSS-E-ST-10-03C Rev.1 Annex C)

#	Name	Documentation	relatedModelElement
1	<1> Company	The NCR status shall identify the supplier of the nonconforming item	«» Non-Conformance [Comment] +NCRInitiator : String [1..*]
2	<2> NCR identifier	The NCR status shall uniquely identify the NCR.	«» Non-Conformance [Comment] +NCR-ID : String [1]
3	<3> Item identifier	The NCR status shall identify the nonconforming item.	«» Non-Conformance [Comment]
4	<4> Description	The NCR status shall provide short description of the nonconformance.	«» Non-Conformance [Comment] +NCRTitle : String [1]
5	<5> Date	The NCR status shall contain the date of last NRB meeting.	«» Non-Conformance [Comment] +dateOfDetection : date [1]
6	<6> Disposition	The NCR status shall describe the disposition.	«» Non-Conformance [Comment] +disposition : AIV-Profile::verification::NCRDisposition [1]
7	<7> Implementation status	The NCR status shall describe the implementation status of the disposition.	«» Non-Conformance [Comment] +correctiveActionsDescription : String [0..*]
8	<8> RFW	The NCR status shall refer to a RFW, if a related RFW exists.	«» Non-Conformance [Comment] +correctiveActionsDescription : String [0..*]
9	<9> Status	The NCR status shall provide the status as “open” or “closed”.	«» Non-Conformance [Comment] +NCRStatus : AIV-Profile::verification::NCRStatus [1]

Figure H.6: ECSS NCR Status List DRD structure (ECSS-Q-ST-10-09C Rev.1 Annex B)