Review Article

MBPLE and modeling languages: From features to system models with SysML

Abstract

This chapter presents an industrial-strength Model-Based Product Line Engineering (MBPLE) approach and its integration into an end-to-end, model-based development process at a high-technology company. The approach is embedded in an End-to-End Model-Based Engineering (E2EMBE) framework that connects multiple domain-specific models via a digital thread and enforces a Single Source of Truth (SSOT)[1] for all engineering artifacts. Building on ISO/IEC 26580 feature-based product line engineering,[2] the method uses feature models to manage variability across SysML-based system models and other discipline-specific models. A hierarchical MBSE framework structures operational, functional and structural architectures and enables the definition of 150% models from which product-specific configurations are derived. Variability is captured through starting, atomic and propagated variation points that are systematically traced from high-level capabilities and use cases down to hardware and software components. The approach is realized with widely used tools such as IBM Rhapsody, Capella and PTC Pure::Variants,[3–5] complemented by custom integrations for automated feature propagation. The proposed method aims to reduce development time and non-recurring costs while improving reuse, consistency and quality for complex, often safety-critical systems in a highly regulated environment.

Keywords: feature catalogue, variation points, controlled reuse, configuration management, pure::variants

Volume 10 Issue 1 - 2026

Dieter Wagner
Systems Engineering Technical Lead, MBDA Deutschland GmbH, Germany

Correspondence: Dieter Wagner, Systems Engineering Technical Lead, MBDA Deutschland GmbH, Germany

Received: November 07, 2026 | **Published:** February 23, 2026

Abbreviations: MBPLE, model-based product line engineering; E2EMBE, end-to-end model-based engineering; SSOT, single source of truth; ECAD, electronic computer-aided design; MCAD, mechanic computer-aided design; OSLC, Open Services for Lifecycle Collaboration

Introduction

The complexity of modern engineering systems increasingly demands integrated approaches that span multiple modeling languages and abstraction levels. In high-technology industries, where products often serve safety-critical functions across diverse operational contexts, the challenge of managing variability while maintaining consistency and traceability becomes paramount. Traditional document-based development approaches struggle to cope with these challenges, particularly as development timelines lengthen while market demands accelerate.

This chapter addresses these challenges by presenting an industrial-strength Model-Based Product Line Engineering (MBPLE) approach embedded within an End-to-End Model-Based Engineering (E2EMBE) framework. The approach establishes digital continuity across multiple domain-specific models—from feature models capturing product line variability to SysML-based system architectures—through the concepts of digital thread and Single Source of Truth (SSOT). By connecting ISO/IEC 26580 feature-based product line engineering with hierarchical MBSE frameworks, the method enables systematic variability management across operational, functional, and structural architectures.

The primary contributions of this work are threefold: First, it demonstrates how feature models can be systematically mapped to SysML-based 150% models through starting, atomic, and propagated variation points. Second, it shows how this mapping enables controlled reuse and product-specific derivation while maintaining traceability from high-level capabilities to concrete hardware and software components. Third, it provides practical insights into implementing this approach using widely adopted industrial tools such as IBM Rhapsody, Capella, and PTC Pure::Variants, complemented by custom integrations.

The remainder of this chapter is structured as follows:

Section 1 establishes the industrial context and motivations driving the need for MBPLE in high-technology industries.

Section 2 provides theoretical foundations on feature modeling and SysML.

Section 3 presents the core methodology for mapping features to system models.

Section 4 discusses implementation aspects and tool support.

Section 5 presents related work, and

Section 6 concludes with a summary and outlook on future developments.

Model-Based Product Line Engineering; End-to-End Model-Based Engineering; SysML v1.x; SysML v2; feature model; variability management; digital thread; Single Source of Truth; Pure::Variants; IBM Rhapsody; Capella

High-technology industry context

The industrial context of this work is a company in the high-technology industry[6,2] that develops complex, often safety-critical systems and solutions for demanding customers in a highly regulated environment. Such companies typically operate internationally with several thousand employees and generate a multi-billion-euro annual turnover. Their products must satisfy diverse and partially conflicting stakeholder needs across different application domains, which differ significantly in terms of operational context and constraints.[7]

High-technology industry need for MBPLE

As the complexity of high-technology systems increases, development times have become longer and longer, and lead times of more than 10 years are not unusual. However, because the political, economic, and technological environment is changing ever faster, the need for new products is immediate and not in 10 years. The current document-based, waterfall-oriented development process has reached its limits and can no longer adequately cope with this growing complexity and time pressure.[6,7]

As illustrated in Figure 1, the traditional document-based approach shows clear signs of saturation. Around 2010, model-based systems engineering (MBSE) was introduced as a revolutionary improvement, but this measure alone soon reached its limits with respect to reducing development time. Consequently, a new improvement initiative was launched in 2022 with the goal of introducing a consistent, end-to-end model-based development approach. Unlike previous initiatives, the End-to-End Model-Based Engineering (E2EMBE) approach explicitly prepares the next revolution, End-to-End Product Line Engineering (E2EPLE) as a form of MBPLE, and even considers a subsequent evolution step that combines E2EPLE with artificial intelligence.

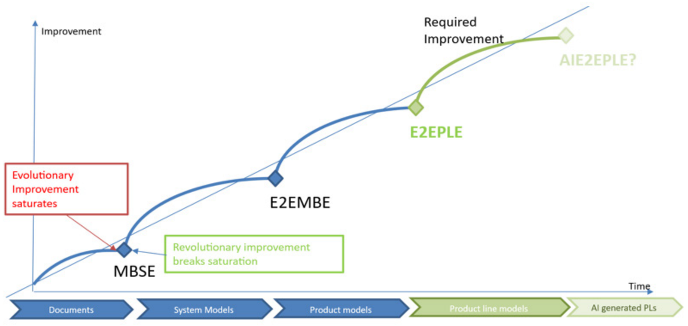


Figure 1 Evolution / revolution.

Enable digital continuity

The methodology E2EMBE, as implied by its name, is grounded on a model-based strategy. This approach encompasses the entirety of the product development process, emphasizing an end-to-end perspective.

Consequently, it is evident from the illustration presented in Figure 2 that E2EMBE does not focus on a single model exclusively; instead, it encompasses a wide array of models that are produced by employing domain-specific development tools. This highlights the versatility of E2EMBE in accommodating different types of models generated through the use of various tools tailored to specific domains. For instance, SysML tools are employed for systems engineering, UML models for software development, Mechanic Computer-Aided Design (MCAD) tools for mechanical designs, Electronic Computer-Aided Design (ECAD) tools for electrical and electronic circuit diagrams, and an assortment of other models created using specialized tools in their respective domains.

In order to ensure that these individual models do not exist in isolation, they are interconnected through interfaces, thereby establishing what is commonly referred to as a digital thread. A prominent example of a suitable interface for this purpose is the Open Services for Lifecycle Collaboration (OSLC).[8] Moreover, numerous tool providers within the industry offer solutions that facilitate the establishment and maintenance of a digital thread. This interconnectedness of all models through a digital thread enhances the overall coherence and integration of the product development process, fostering better communication and collaboration among the various domains involved in the project. The utilization of a digital thread effectively streamlines the exchange of information and data between the different models, promoting a more efficient and cohesive approach to product development. By establishing robust connections between the diverse models utilized throughout the product development lifecycle, the E2EMBE methodology contributes to enhancing the overall efficiency and effectiveness of the development process. The concept of a digital thread within the E2EMBE methodology underscores the significance of connectivity and integration in modern product development practices, emphasizing the importance of a holistic and interconnected approach to model-based engineering.

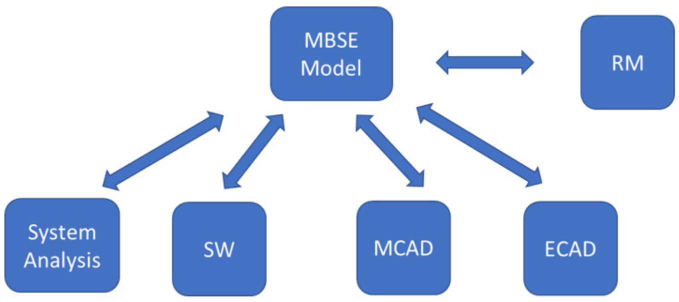


Figure 2 Distributed Model with digital thread.

Enable single source of truth

The fundamental principle of establishing a singular point of truth holds paramount significance within the E2EMBE methodology embracing digital continuity. This particular methodology hinges on the presence of a solitary developmental entity embodying a precise set of information, be it requirements or elements of the SysML model, residing within their singular model. The utilization of such entities in other models is exclusively permitted through referencing, thereby setting forth a singular point of reference, commonly referred to as the single source of truth (SSOT). This procedural norm safeguards the E2EPLE technique by ensuring uniformity through the consistent adoption of the most recent references throughout all models encapsulated within the E2EMBE structure. Through the enforcement of a sole point of reference, E2EPLE not only affirms the cohesion but also upholds the precision of all models interconnected within the framework. It is imperative to recognize that the adherence to a single source of truth is the cornerstone of maintaining integrity and accuracy within the E2EMBE paradigm, thereby fortifying the reliability and robustness of the overall system.

Enable controlled reuse

Enabling controlled reuse is an essential necessity within the HTC, as evidenced by the enduring implementation of the Design for Modularity and Reuse (DMR) strategy. These principal mandates that new programs should be constructed with modular and reusable frameworks, ensuring flexibility and efficiency in project development. In order to proficiently supervise the reuse process and monitor the utilization of particular modules across various projects and iterations, the adoption of a tool-supported methodology becomes imperative. The emergence of E2EPLE presents itself as a practical and effective solution for overseeing and regulating controlled reuse within the organizational framework. E2EPLE offers a comprehensive approach to managing and enforcing the reuse of modules, thereby enhancing the overall sustainability and scalability of projects within the HTC. By leveraging E2EPLE, the HTC can streamline its development processes, optimize resource utilization, and foster a culture of systematic and strategic reuse across all projects and initiatives.

Enable variability

When facing the task of reusing development artifacts, a significant hurdle arises in the form of having to cater to customer requirements that might not be fully satisfied by the existing artifacts. In order to tackle this particular challenge effectively, it becomes imperative to deploy a well-thought-out strategy that focuses on enabling variability within the development process. This strategy entails the creation of novel versions or variants of the existing development artifacts, specifically designed to align with and fulfill the unique needs of individual customers. The efficient management of these diverse variants is made possible through the utilization of the E2EPLE method, which plays a pivotal role in ensuring that the tailored solutions can seamlessly and effectively be integrated into the overall development workflow. By employing such methodologies, organizations can enhance their ability to adapt and respond to the ever-evolving demands and expectations of their clientele, thus fostering a more agile and customer-centric approach to software development.

Manage configuration

The intricate nature of E2EMBE, encompassing a wide array of models and their corresponding iterations, underscores the critical importance of efficient configuration management. E2EPLE emerges as a pivotal component in facilitating the smooth administration of these varied models and versions, guaranteeing a structured and effective configuration process. By leveraging E2EPLE, organizations can enhance their ability to manage the complexities inherent in E2EMBE, thus optimizing operational efficiency and ensuring seamless integration of diverse models and versions.

High technologie company implementation for MBPLE

In order to integrate all the aforementioned principles, an E2EPLE method was specifically developed to serve as the implementation of MBPLE within the HTC. This method exemplifies the application of HTC PLE through the E2EPLE approach within the realm of Model-Based Systems Engineering (MBSE). An illustrative example of this implementation is demonstrated through the light subset of Command-and-Control international Product Line (C2IntlPL).

System models with SysML

The MBSE model is the central element of the high-technology company E2EMBE method.[9,10,7] The goal is to create a 150% MBSE model for certain products and product families, from which product-specific MBSE model instances can then be generated. The following introduces the structure of the MBSE model, the hierarchy of MBSE models, the use of different variation points types, and the propagation of variation points within a Rhapsody model.

MBSE model structure

The Figure 3 shows the structure of the high-technology company used MBSE frameworks based on the CESAMES framework.[11]

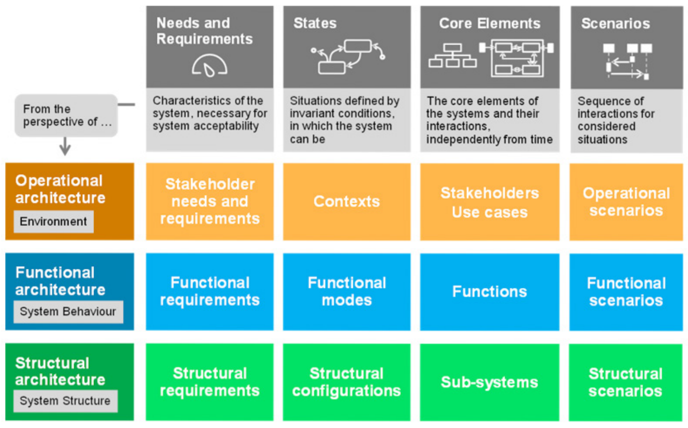

Needs and Requirements
States
Core Elements
Scenarios
From the perspective of ...
Characteristics of the system, necessary for system acceptability
Situations defined by invariant conditions, in which the system can be
The core elements of the systems and their interactions, independently from time
Sequence of interactions for considered situations
Operational architecture
Environment
Stakeholder needs and requirements
Contexts
Stakeholders Use cases
Operational scenarios
Functional architecture
System Behaviour
Functional requirements
Functional modes
Functions
Functional scenarios
Structural architecture
System Structure
Structural requirements
Structural configurations
Sub-systems
Structural scenarios

Figure 3 High-technology industry used MBSE framework.

In the following a breve description to the rows and columns of the framework is given:

The **Operational architecture**, within the realm of system design, elucidates the system from the perspective that operators have of it. Essentially, it delineates how operators engage with the system through the depiction of various use cases. This architectural framework portrays the System as a black box, with its external interfaces serving as points of interaction with the outside world. It provides a comprehensive understanding of the operational aspects of the system and how users navigate through its functionalities.

In a similar vein, the **Functional architecture** outlines the system's functions in a manner that is agnostic to any particular solution. This approach ensures that there are no constraints imposed by specific solutions or design parameters. This architectural blueprint also illustrates the interplay of functions through functional chains, depicting how they collaborate to achieve desired outcomes. The functions are intricately linked to the corresponding use cases, creating a cohesive structure that guides the system's behavior.

Finally, the **Structural architecture** delves into the realization of one or multiple possible configurations that meet the requirements set forth by the functional architecture. This layer presents a spectrum of physical architectures that can manifest the envisioned functionalities. The functions outlined in the functional architecture are assigned to specific elements within the physical architecture, which serve as the building blocks for implementing the desired functionalities. This allocation ensures a clear mapping between the abstract functions and their concrete realizations within the system.

The column dedicated to **Stakeholder needs and requirements** within the architectural framework contains the essential specifications and mandates for each architectural component, guiding the assignment of model elements to ensure alignment with stakeholder expectations and project objectives.

The column labeled "**Stats**" holds varying significance across the three different architectural structures due to the unique ways in which data is processed and interpreted within each system. Each architecture brings forth distinct methods and algorithms that influence the interpretation and utilization of the statistics presented in the column.

In the realm of **Operational architecture**, the delineation revolves around elucidating the intricate life cycle of system statistics encompassing aspects such as storage, transportation, operation, and beyond, which are paramount for the optimal functioning of the system. Conversely, within the domain of **Functional architecture**, the focus shifts towards defining the operational modes of the system, ranging from being off, on, functional, error, to training modes, thereby providing a comprehensive framework for understanding the various states in which the system can operate.

Lastly, the realm of **Structural architecture** introduces the concept of configurations and their definitions, which are strategically positioned within the overarching view of the system, thereby establishing a foundational framework for the system's structural organization and design.

The **Core Elements** column in all architectural layers encompasses a comprehensive list of essential components tailored to each specific type of architecture, serving as a key reference point for understanding the fundamental building blocks required for the development and implementation of the corresponding architectural structure.

The **Scenario** column within all architectural layers provides a detailed representation of the chronological sequence of activities and functions required to implement the various scenarios derived from use cases and functional chains, thus offering a comprehensive overview of the temporal progression within the system architecture.

MBSE model hierarchy

The above-described high-technology industry MBSE framework can be applied recursively. This means that the constructional layer decomposes the level of intestate (level n) always only one level down (level n+1). This means system level models decompose the system into subsystems. The subsystem level models decompose the subsystems into equipment.

This results than in one system model, n subsystem models and m equipment models. The diagram presented as Figure 4 vividly demonstrates the hierarchical structure of the product lines, showcasing a collection of distinct product lines that are interconnected and interdependent. As each product line offers a range of products, these individual products can be used as fundamental components or "building blocks" within the parent product line. For example, as part of the C2IntlPL, there is a equipment-level product line that provides products for data centers. In this example, equipment-level product line provides two product instances the "Tactical operation center" and the "Combat operation Center".

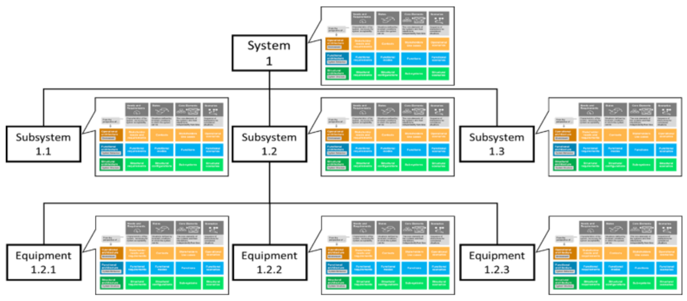


Figure 4 High-technology industry used MBSE framework hierarchy.

E2EPLE

To enable the feature based PLE for MBSE model approach it is necessary to have a correct scoping of the product line.[2,6] Since the MBSE models are hierarchical organized it is obvious to scope the product line approach in the same way. This means each model in each hierarchy is then its one product line.

Since each product line delivers a set of similar products, those products can be used as building blocks in the parent product line. In a system there is an equipment level PL for the computational HW the, the HW equipment can be used as a building block in the System.

The diagram presented as Figure 5 vividly demonstrates the hierarchical structure of the product lines, showcasing a collection of distinct product lines that are interconnected and interdependent. As each product line offers a range of products, these individual products can be used as fundamental components or "building blocks" within the parent product line. For example, as part of the C2IntlPL, there is a equipment-level product line that provides products for data centers. In this example, equipment-level product line provides two product instances the "Tactical operation center" and the "Combat operation Center".

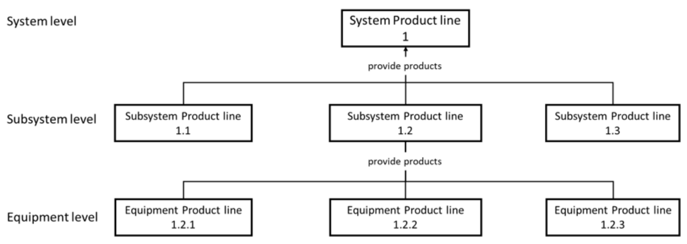

System level
System Product line 1
provide products
Subsystem level
Subsystem Product line 1.1
Subsystem Product line 1.2
Subsystem Product line 1.3
provide products
Equipment level
Equipment Product line 1.2.1
Equipment Product line 1.2.2
Equipment Product line 1.2.3

Figure 5 Product line hierarchy.

Structure of the product line model

The product line Model, shown in Figure 6 at any level is structured in folders for easy product line model navigation.

a) The references to the E2E Models stored in the "_Assets" folder. The references are point to the Shared Asset Superset and Un-shareable Asset [1]storage location in their right version.

b) The Feature Catalogue including all features as a building block, the constraints related to some of the features, the dependencies between features, and the provided products of the child product lines is stored in the "_Feature Catalogue" folder.

c) The provided products of the product line, represented as individual Bill of Features, are stored in the "_provided Products" folder.

d) An optional questionnaire to understand the customer needs is stored in the "_Questionnaire" folder. The answers of the questionnaire are stored into the corresponding Bill of Features.

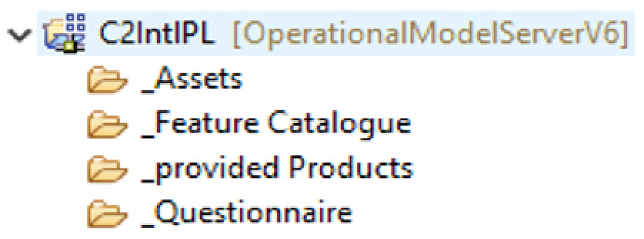

C2IntlPL [OperationalModelServerV6]
_Assets
_Feature Catalogue
_provided Products
_Questionnaire

Figure 6 Product line Model.

The provided products of the child product line are integrated as a read only Bill of Features into the Feature Catalogue. This allows

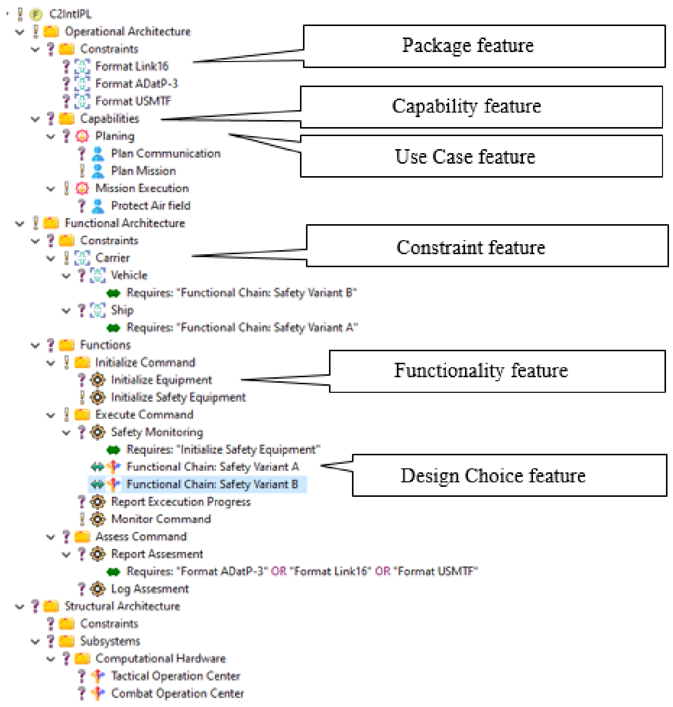
[1]Due to contractual reasons, not all assets are shareable

the product line engineer to see which features are included be the different products. Each Bill of Features also contains characteristic quantities of the concrete product and the answers of the optional questionnaire.

Feature catalogue

The high-technology company uses Feature Catalogue contains an additional semantic to the MBSE framework.

The additional semantic shown in Figure 7 is based on specialized features which are described in the following section:



```
C2IntIPL
  Operational Architecture
    Constraints
      Format Link16
      Format ADatP-3
      Format USMTF
    Capabilities
      Planing
        Plan Communication
        Plan Mission
      Mission Execution
        Protect Air field
  Functional Architecture
    Constraints
      Carrier
        Vehicle
          Requires: "Functional Chain: Safety Variant B"
        Ship
          Requires: "Functional Chain: Safety Variant A"
    Functions
      Initialize Command
        Initialize Equipment
        Initialize Safety Equipment
      Execute Command
        Safety Monitoring
          Requires: "Initialize Safety Equipment"
          Functional Chain: Safety Variant A
          Functional Chain: Safety Variant B
        Report Excecution Progress
        Monitor Command
      Assess Command
        Report Assesment
          Requires: "Format ADatP-3" OR "Format Link16" OR "Format USMTF"
        Log Assesment
  Structural Architecture
    Constraints
    Subsystems
      Computational Hardware
        Tactical Operation Center
        Combat Operation Center
```

Figure 7 High-technology industry used Feature Catalogue.

Package feature: To enable a better model organization a package feature is provided. The Package features are not assigned to any shared asset superset. It is a good practice to align the high-technology company MBSE framework model with the feature catalog structure as much as possible to enable a better orientation.

Capability feature: The Capability feature is used to capture the different capabilities of the system. Capabilities are the major building blocks of a system, therefore the concept of PLE features defined in **ISO 26580**,[2] is perfectly aligned to it. The main goal is to develop a high-technology industry owned set of capability which allows us the reuse for different customers. But we are also developed customer owned capabilities which are not sharable.

Use Case feature: In the MBSE framework the Use Case is the traditional way of describing the usage capabilities from the user's point of view. the high-technology company provides for each capability at least one Use Case to describe the high-technology company intended usage of the most of the capabilities in consequence each capability has at least one assigned Use Case. In reality each customer has a specific need for how to use those capabilities therefore each capability has multiple Use Cases. So, there is a collection of Use Cases, some of them owned by the high-technology company and others owned by the customer. Therefore, Use Cases have a high rate of variability.

Figure 8 shows the MBSE framework Model with capabilities (box with the gear symbol), use cases and their actors shown which corresponds to the feature model example.

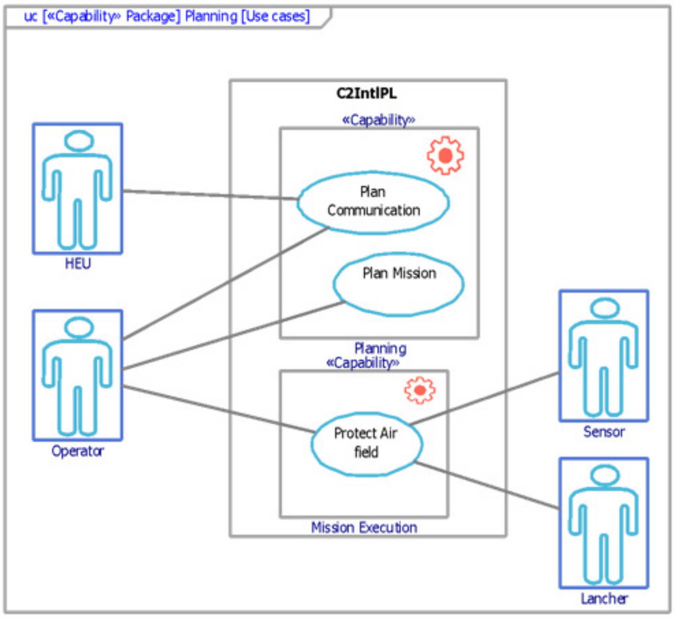


Figure 8 MBSE model with capabilities and use cases.

Constraint feature: The Constraint feature is used to capture constraints coming from customers, from national legislation, standard or high-technology industry product strategy. **Constraints reduce the solution space.** The Feature Catalogue example shows a high-technology industry product strategy constraint that only 3 supported communication formats[12–15] are available for any product.

Functionality feature: The functionality features are the atomic building blocks from which capabilities are assembled. To define the behavior of the capabilities the functions are connected together as a functional chain. These functional chains always start from an incoming event and end with an outgoing event.

Design choice feature: The functional chains may vary in functionality of fidelity level to satisfy the different customer needs. Therefore, the Design Choice feature is used to capture variability for functions and functional chains.

In the Feature Catalogue example, the function Safety Monitoring has two variants functional chains the Functional Chain: Safety Variant A and Functional Chain: Safety Variant B

Figure 9 shows the MBSE framework Model with a System_ Function Safety Monitoring and the 2 variations of functional chains to realize the System_Function. Additionally, the restriction to the specific Design choice feature SafetyVariantA and SafetyVariantB (see Feature Catalogue display name "Functional Chain: Safety Variant A" and "Functional Chain: Safety Variant b") is shown in the red boxes.

Usage of specialized features in MBSE:

The Table 1 indicated which specialized features type is used in the different architectures.

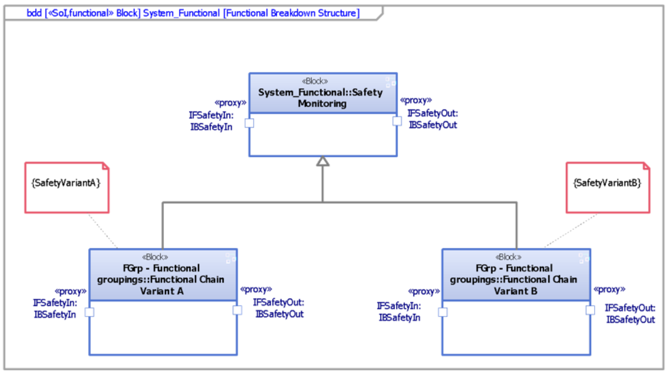

bdd [«SoI,functional» Block] System_Functional [Functional Breakdown Structure]
«Block»
System_Functional::Safety Monitoring
«proxy» IFSafetyIn: IBSafetyIn
«proxy» IFSafetyOut: IBSafetyOut
{SafetyVariantA}
{SafetyVariantB}
«Block»
FGrp - Functional groupings::Functional Chain Variant A
«Block»
FGrp - Functional groupings::Functional Chain Variant B
«proxy» IFSafetyIn: IBSafetyIn
«proxy» IFSafetyOut: IBSafetyOut

Figure 9 MBSE model with function and functional chains.

Table 1 Cross-reference feature type vs. architecture layer

Architecture feature type	Operational	Functional	Structural
Package feature	x	x	x
Capability feature	x		
Use Case feature	x		
Constraint Feature	x	x	x
Functionality feature		x	
Design Choice feature		x	x

The selection is based on best practices resulting from various pilots.

Variation points

To enable PLE for MBSE models the variability needs to be part of the model. There are several ways to add variability within the MBSE model. The location where the variability is located is called "Variation Point". Each variation point is exactly assigned to the corresponding feature of a feature model. It turns out that there are two different types of variation points relevant: a **"Atomic" (1)** a **"Starting" (2)** and a **"Propagated" (V) Variation Point**.

Figure 10 shows a simplified (collapsed hierarchy) view of the different Variation Points within the MBSE framework.

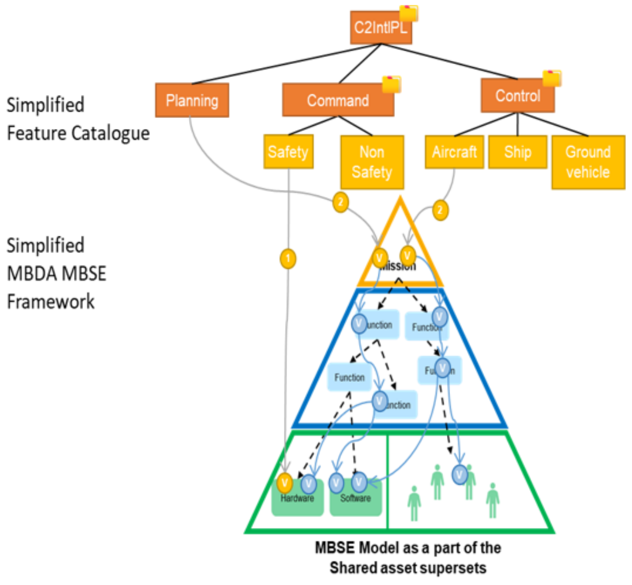

C2IntlPL
Simplified Feature Catalogue
Planning
Command
Control
Safety
Non Safety
Aircraft
Ship
Ground vehicle
Simplified MBDA MBSE Framework
Mission
Function
Hardware
Software
MBSE Model as a part of the Shared asset supersets

Figure 10 "Atomic", "Starting" and "propagated" variation points.

"Atomic" variation point

The utilization of the "Atomic" Variation Point is specifically targeted towards the management of variability within the realm of Functional and Structural architecture exclusively. The term "Atomic" signifies the absence of any dependent child model elements, a characteristic that is uniquely viable within the Structural architecture domain. This concept holds significance due to the fact that the Structural architecture will be populated by the resulting products of the child product, thus making the "Atomic" Variation Point a pivotal design decision in constructing the building blocks of the system.

The building blocks themselves are structured in a hierarchical manner, starting from the System architecture encompassing Subsystems, followed by the Subsystem architecture incorporating Equipment, and finally, at the Equipment architecture comprising both hardware and software components. As illustrated in the abstract example depicted in Figure, the "Safety" feature is explicitly linked to a hardware element at the structural architecture, serving as a prime illustration of a design choice feature within the architecture. This association underlines the critical nature of design decisions in shaping the configuration and functionality of the system components. Such deliberate allocations of features to specific levels highlight the meticulous planning and strategic structuring involved in the architectural design process. The interplay between features, elements, and levels within the system architecture underscores the complexity and intricacy inherent in managing variability and design choices. The Design choice feature and Constraint feature are effectively implemented through the utilization of "atomic" variation points, highlighting a sophisticated approach to architectural design and decision-making. Overall, the "Atomic" Variation Point stands out as a key mechanism in orchestrating the variability management process within the Functional and Structural architecture paradigm.

"Starting" variation point

The variation point, known as "Starting", is model elements that are linked to other model elements and are either within the same architectural level or span the hierarchical structure of architectural levels. These connections of model elements or dependencies are crucial for the design of the 150% architecture and functionality of the system under consideration.

The "Starting" Variation Point is strategically positioned within the MBSE framework, typically associated with pivotal elements like capabilities, use cases, or functional chains. This strategic placement ensures that the foundational aspects of the variations are well-defined and aligned with the overarching objectives of the project.

In the illustrative scenario depicted in Figure 10, the presence of two distinct "starting" variation points is highlighted, each linked to specific features within the operational architecture. One use case is intricately tied to the "Planning" feature, while the other is associated with the child "Aircraft Variant" feature variate of the "Control" feature, showcasing the diverse applications of such variation points in system design.

The definition and implementation of the "starting" variation points are defined by MBSE framework model constraint elements, which are part of the SysML Meta model. This hierarchical relationship ensures that the variation points imposed by constraints these are in alignment with the broader system objectives and architectural constraints.

Both the capabilities and use cases linked to these variation points are intricately connected to their respective features within the feature catalogue, thereby establishing a clear and structured framework for product line-based system development. This integration of features and constraints not only enhances the clarity of the system design related to PLE, but also facilitates effective communication and collaboration among stakeholders.

Overall, the strategic placement and implementation of the “starting” variation points play a pivotal role in shaping the 150% system architecture, ensuring that key elements are appropriately linked and aligned with the product line requirements. By anchoring these points to high-level model elements with constraints, designers and engineers can effectively navigate the complexities of system design and development, ultimately leading to more robust and efficient systems.

“Propagated” variation points

Essential to the insertion of the “start” variation point is the absence of any break in the model dependency model, the meta model, which extends from the operational architecture to the structural architecture. This crucial aspect ensures that a capability or use case can be easily extracted from a 150% MBSE model. Upon conducting a model transformation (extraction), it is crucial that all model elements essential for the advancement of a capability or use case are retained. In order to facilitate the factory in identifying the appropriate elements, the concept of “propagated” variation points is assigned into all associated model elements pertaining to a capability or use case. This strategy ensures that the necessary components are effectively transmitted and preserved throughout the transformations process.

A specific propagation tool establishes a Pure::Variants constraint for “Functional Chain Variant A” as **(ProtectAirField) AND (SafetyVariantA)** and for “Functional Chain Variant B” as **(ProtectAirField) AND (SafetyVariantB)**. This logical framework empowers Pure::Variants to instantiate a Model-Based Systems Engineering (MBSE) framework model that exclusively incorporates one of the functional chains.

Figure 11 provides a visual depiction of a product preview wherein the design choice feature selected is SafetyVariantB, resulting in the greying out of SafetyVariantA and its corresponding anchored constraint.

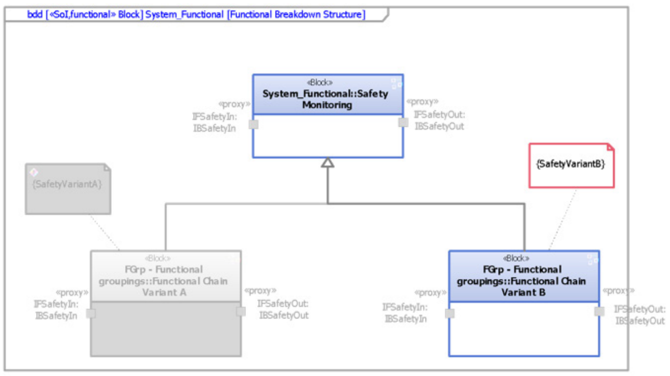


Figure 11 Preview of instantiated MBSE framework model.

The Propagation tool also creates a resulting Propagation Graph.

The depiction in Figure 12 showcases a “Propagation Graph” related to the use case titled “Protect Air Field”. This graph exemplifies how this particular use case is linked to three different Actors and two ports, all situated within the operational architecture. Furthermore, it is also connected to the “Safety Monitoring” functional chain.

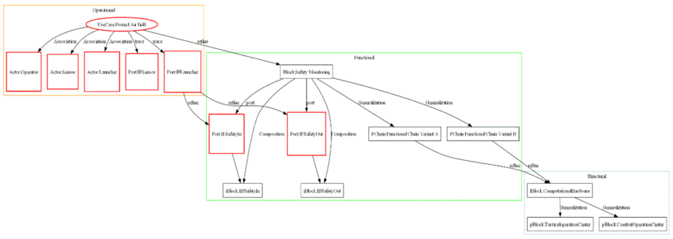


Figure 12 Propagation graph of the use case protect air field.

Within the functional architecture, the “Safety Monitoring” functional chain serves as a fundamental component, showcasing two distinct specializations: “Functional Chain Variant A” and “Functional Chain Variant B”. These variants represent the concept of an “atomic” variation point, allowing flexibility in choosing the appropriate variant based on specific customer requirements or feature catalog dependencies.

The “atomic” variation point in conjunction two or more specialized variants ensures that based on the system’s configuration needs only one of them will be ultimately incorporated into the final product. This strategic selection process is crucial for tailoring the system according to its intended purpose and functionality.

Moreover, the “Safety Monitoring” component features two ports “IFSafetyIn” and “IFSafetyOut,” each defined by interface blocks “IBSafetyIn” and “IBSafetyOut”. These ports and interface blocks play a vital role within the functional architecture, enabling seamless communication and data exchange between different system elements.

Additionally, the operational architecture includes a port labeled “IFLaucher,” which exhibits two optional trace dependencies linked to ports “IFSafetyIn” and “IFSafetyOut”. These dependencies are illustrated by a trace relationship between ports from different architectural levels and illustrate the hierarchical structure of the interface.

The two specialized functional chains are further connected to a logical subsystem named “Computational Hardware” within the structural architecture. This logical subsystem branches out into two specialized physical subsystems: the “Tactical Operation Center” and the “Combat Operation Center,” each serving distinct purposes within the overall system framework. To distinguish between these two hardware solutions, an “atomic” variation point is mandatory.

It is important to note that the presented propagation graph provides a simplified excerpt of the real C2IntlPL MBSE framework model. The actual model entails a significantly more intricate and elaborate propagation graph, reflecting the comprehensive nature of the system architecture and its interconnected components.

MBSE framework model Meta model

The MBSE metamodel can be succinctly described as the fundamental model that defines the dependencies of the various model elements within the MBSE framework. It serves as the underlying structure that guides the development and implementation of the MBSE framework. This special representation provides a detailed insight into the relationship model, the MBSE metamodel, in the context of the HTC MBSE framework. By using the Rhapsody add-in tool, it is possible to generate an individual MBSE metamodel that shows exactly which elements of the HTC MBSE framework were actually used.

The one in Figure 13 Shows the Meta Model of light version of the C2IntlPL. The HTC MBSE model elements that are shown are partly elements of the SysML meta model and partly HTC specialized SysML elements. Use case, Block, Port, actor and interface Block are SysML metamodel elements and Functional chain, logicalBlock and physicalBlock are HTC specialized SysML blocks.

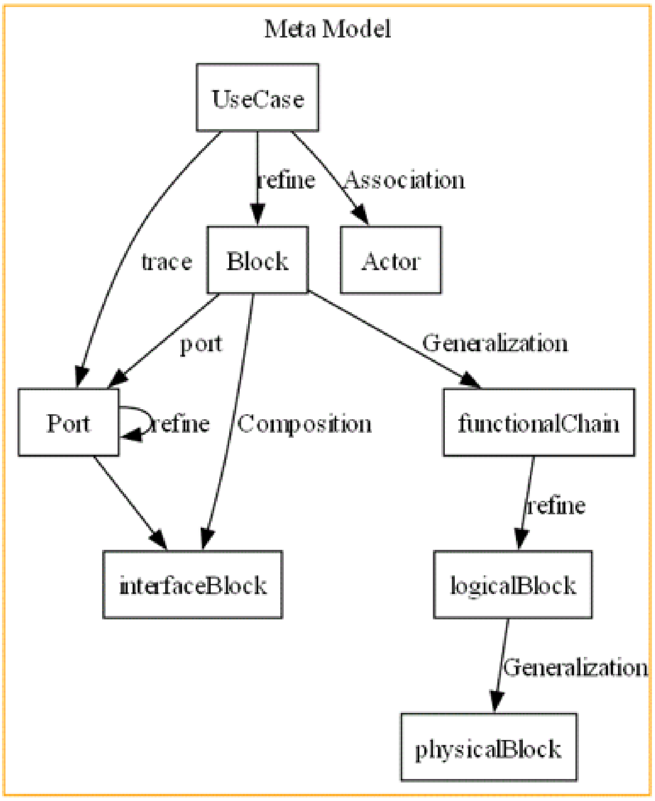


Figure 13 Meta Model of light version of the C2IntlPL.

Tool

In the high-technology company two standard MBSE tools are used:

A. IBM Rhapsody[4]

B. Open-Source Capella[5]

The high-technology company MBSE framework is applied to both of them.

The high-technology company PLE tool is used:

I. PTC Pure::Variants[3]

Pure::Variants[3] support both of the MBSE tools so we can apply the same PLE method.

But with a tool like Capella,[5] feature propagation is automatic based on the dependency tree from high-level model elements to lower-level elements. The high-technology company also uses IBM Rhapsody,[4] which does not support automatic feature propagation. To enable this feature propagation, the high-technology company has developed a Rhapsody[4] add-in. Once a high-level MBSE framework model element is anchored to a feature, the add-in performs the propagation and adds the same feature as "Propagated" Variation Points.

Conclusion

This chapter presented a comprehensive Model-Based Product Line Engineering (MBPLE) approach for the high-technology sector. The proposed End-to-End Product Line Engineering (E2EPLE) methodology integrates variability management systematically into an End-to-End Model-Based Engineering (E2EMBE) framework, addressing the challenges of managing complexity, reducing development time, and ensuring consistency across multiple engineering disciplines in safety-critical systems.

The approach leverages a hierarchical MBSE framework with feature-based variability management through digital thread and Single Source of Truth (SSOT)[1] principles. By using industry-standard tools (IBM Rhapsody, Capella, PTC Pure::Variants)[3–5] enhanced with custom add-ins for automated feature propagation, the methodology enables effective management of 150% models and systematic derivation of product-specific configurations while maintaining traceability across all engineering artifacts.

The systematic application of variation points (starting, atomic, propagated) combined with the hierarchical MBSE framework enables controlled reuse and configuration management at scale. This approach is expected to significantly reduce non-recurring engineering costs, accelerate time-to-market, and improve product quality through enhanced consistency and traceability—critical factors for safety-critical systems in highly regulated domains.

Future work

This chapter has presented an industrial-strength MBPLE approach that integrates feature modeling with SysML-based system models within an E2EMBE framework. While the described methodology has been successfully deployed in a high-technology industrial context, several directions for future work and recommendations emerge from this experience.

Enhanced tool integration and automation

The current implementation relies on a combination of widely adopted tools (IBM Rhapsody, Capella, PTC Pure::Variants) complemented by custom integrations. Future work should focus on:

a) Developing more sophisticated bidirectional synchronization mechanisms between feature models and system models to reduce manual effort in maintaining consistency.

b) Enhancing automated traceability support that can dynamically update trace links as models evolve.

c) Creating intelligent validation mechanisms that can detect inconsistencies between feature selections and architectural constraints earlier in the development process.

d) Implementing model transformation engines that can automatically generate product-specific configurations from feature selections with minimal manual intervention.

These enhancements would significantly reduce the overhead of maintaining the MBPLE infrastructure and make the approach more accessible to organizations with limited resources for custom tool development.

Extended evaluation and industrial case studies

While the approach has been validated in a high-technology industrial environment, broader empirical evaluation is needed:

1) Conduct longitudinal studies tracking the quantitative benefits of MBPLE adoption across multiple product generations, measuring metrics such as development time reduction, reuse rates, defect density, and time-to-market improvements
2) Perform comparative case studies across different industrial domains (aerospace, automotive, medical devices) to identify domain-specific adaptation patterns and generalization opportunities
3) Investigate the scalability limits of the approach when applied to extremely large product lines with thousands of features and complex constraint networks
4) Analyze the organizational and process implications of MBPLE adoption, including required changes to team structures, skill sets, and development workflows

Integration with emerging technologies

Several emerging technologies offer potential synergies with the presented MBPLE approach:

A. Artificial Intelligence and Machine Learning: Explore AI-assisted feature modeling, automated variation point identification, and intelligent product configuration recommendations based on historical data and customer requirements

B. SysML v2 and Next-Generation Modeling Languages: Investigate how the transition to SysML v2 and other emerging modeling standards (such as UAF, ARCADIA) affects the feature-to-model mapping approach and identify opportunities for improved integration

C. Digital Twin Technology: Extend the MBPLE framework to support digital twin creation and management, enabling runtime variability management and product-specific simulation

D. Cloud-Based Collaborative Modeling: Develop distributed MBPLE environments that support geographically dispersed teams working on shared product line artifacts with real-time collaboration capabilities

Methodological refinements

Further refinement of the core methodology would benefit from:

a) Developing formal semantics for the feature-to-SysML mapping to enable rigorous verification and validation of product line models.

b) Establishing systematic patterns and best practices for different types of variation points (starting, atomic, and propagated) based on architectural contexts.

c) Creating comprehensive guidelines for 150% model structuring that balance completeness with maintainability.

d) Defining metrics for assessing the quality and effectiveness of MBPLE implementations.

Standardization and community building

To facilitate broader adoption and maturation of MBPLE approaches:

i. Work toward standardization of feature modeling notations and their integration with established modeling standards (ISO/IEC 26580, OMG SysML).

ii. Contribute to industry consortia (INCOSE, OMG) to promote best practices and interoperability.

iii. Develop open-source reference implementations and educational materials to lower adoption barriers.

iv. Establish communities of practice for knowledge sharing among MBPLE practitioners.

The successful implementation of these future directions will further strengthen the E2EPLE approach and solidify its position as a mature, industry-proven methodology for managing complexity in high-technology product development. As organizations continue to face increasing system complexity and accelerating market demands, the benefits of systematic variability management through MBPLE will become even more critical for competitive success.[16–20]

Acknowledgement

None.

Conflict of interest

Author has no conflicts of interest.

Funding

None.

References

1. Brackenbury W, Liu R, Mondal M, et al. Draining the data swamp: a similarity-based approach for multi-source data. In: *Proceedings of the Workshop on Human-In-the-Loop Data Analytics (HILDA '18).* ACM; 2018.
2. International Organization for Standardization. ISO/IEC 26580:2021. *Software and systems engineering—Methods and tools for feature-based product line engineering.* Geneva, Switzerland; 2021.
3. Pure-systems GmbH. *pure:variants—variant management tool for product line engineering.* Version 6.0. Magdeburg, Gkermany.
4. IBM Corporation. *IBM Engineering Systems Design Rhapsody—Model--based systems engineering tool*. Version 10.0. IBM.
5. Eclipse Foundation. *Capella—open source solution for Model-Based Systems Engineering (MBSE).* Version 7.0.
6. International Organization for Standardization. ISO/IEC 26550:2015. *Software and systems engineering—Reference model for product line engineering and management.* Geneva, Switzerland; 2015.
7. INCOSE. Walden DD. *INCOSE systems engineering handbook: a guide for system life cycle processes and activities.* 5th ed. John Wiley & Sons; 2023.
8. OASIS. *Open Services for Lifecycle Collaboration (OSLC) Core Specification Version 3.0.* OASIS Standard; 2020.
9. Object Management Group. *OMG Systems Modeling Language (OMG SysML™), Version 1.6.* OMG Document No. formal/2019-11-01. 2019.
10. Object Management Group. *Systems Modeling Language (SysML®) v2, Version 2024-05.* OMG Document No. formal/2024-05-01. 2024.
11. Roques P. *Systems architecture modeling with the arcadia method: a practical guide to capella.* Elsevier; 2016.
12. Object Management Group. *Unified Modeling Language®, Version 2.5.1.* OMG Document No. formal/2017-12-05. 2017.
13. United States Department of Defense. MIL-STD-6040. *Link 16 Standard.* Washington, DC; 2022.

14. NATO Standardization Office. STANAG 5516, Allied Data Publication-3 (ADatP-3): *NATO Message Text Formatting System (FORMETS).* 2021.
15. United States Department of Defense. MIL-STD-6017. *United States Message Text Format (USMTF).* Washington, DC; 2020.
16. Clements P, Krueger C. Product line engineering in context. *INCOSE Insight.* 2002;5(4):18–21.
17. Pohl K, Böckle G, van der Linden FJ. *Software product line engineering: foundations, principles, and techniques.* Springer-Verlag; 2005.
18. Clements P, Northrop L. *Software product lines: practices and patterns.* Addison-Wesley Professional; 2001.
19. International Organization for Standardization; IEEE. ISO/IEC/IEEE 15288:2023. *Systems and software engineering—System life cycle processes.* Geneva, Switzerland; 2023.
20. International Organization for Standardization; IEEE. ISO/IEC/IEEE 42010:2022. *Systems and software engineering—Architecture description.* Geneva, Switzerland; 2022.