

FEATURE MODEL CREATION THROUGH MISSION ANALYSIS IN THE CONTEXT OF VEHICLE FAMILIES

Angel M. Zarate Villazon[1], Dr. Michael Balchanos[1] & Prof. Dimitri N. Mavris[1]

[1]Aerospace Systems Design Laboratory, Georgia Institute of Technology, Atlanta, GA, USA

Abstract

Families of vehicles are widespread throughout various industries. Leveraging commonalities across vehicle designs is an attractive proposition as it has the potential to reduce development costs, streamline operations, and enable accelerated entries into new markets. While commercial aviation has successfully leveraged commonalities within well-defined market segments and missions, the process is challenging in emerging markets with high levels of technological and economic uncertainties, such as advanced air mobility or planetary exploration. In domains like these, the technical and economic unknowns complicate the architecture of families using traditional methods due to the lack of design heritage and the associated historical data. In particular, capturing the possible variability of the vehicle family consistently is a challenge that requires the consideration of multiple layers of system abstraction in a process called feature modeling. Typically, translating the different stakeholder needs into possible solutions within feature modeling requires lengthy sessions with subject matter experts. In an uncertain context where new needs can emerge, it can be difficult to evolve this feature model continuously. Generative Artificial Intelligence has emerged as a potential solution to respond to this challenge. This paper presents a pilot implementation of a process to use a Large Language Model (LLM) to elicit variability in a feature model with a use case related to a family of Advanced Air Mobility vehicles. The first step of this process consists of capturing the planned missions and associated stakeholder needs in SysML v2 through variable action definitions, illustrating the variability of the operational layer. Then, consistent with established modeling methodologies, the operational descriptions are used by the LLM to elicit the functional and physical layers of the feature model. An experiment was executed within this use case to demonstrate the importance of structure in the context and clarity in the prompt when using these LLMs. The results suggest that a structured representation of the operational variability in SysML v2, along with detailed instructions in the prompts can lead to better feature models. The quality of these models was measured through a qualitative evaluation in terms of correctness, consistency, and completeness of the resulting feature models. Overall, this exploratory study of the technical approach reveals the potential to leverage artificial intelligence to create comprehensive feature models that support the design of product lines in markets defined by uncertainty and emergent capabilities.

Keywords: product line engineering, product family, feature models, concept of operations

1. Introduction & Motivation

The competitive landscape in which most businesses have to survive creates schedule and budget pressures that make it imperative to reduce work redundancies. This has made the strategy of creating vehicle families that leverage commonalities in their members to support a variety of different missions a widespread practice across industries. Modern cars are developed with several shared parts, with a large number of models being derived from a common platform [1, 2]. In the commercial aviation industry, different aircraft variants compose families for each market segment among the larger OEMs [3]. With the rise of Advanced Air Mobility, an opportunity to approach the development

https://doi.org/10.71945/icas2026_0918

of a completely new vehicle family is emerging. The main argument behind this is that the development of Advanced Air Mobility vehicles for the various market needs can result in a family of vehicles with variants that satisfy each of the possible uses. However, Advanced Air Mobility presents specific complications such as uncertain market demands, challenging certification landscapes and evolving missions of interest as the market matures and operational issues emerge [4]. These complications present a significant systems engineering challenge when developing a family of vehicles.

At its core, the strategy to develop families of systems is motivated by the prospect of eliminating duplicated engineering efforts, which comes with its own benefits and drawbacks. Across industries, the development of vehicle families can lead to reduced development costs, streamlined operations, and an accelerated entry into new markets [5]. Particularly for commercial aircraft, airlines benefit from the family approach through reduced training requirements across the fleet thanks to the similarities in each variant enabled by the commonality approach [6, 7]. However, there are significant risks that need to be pondered when designing a family of vehicles where commonality can be leveraged. One of the biggest risks created by commonality is the resulting trade off of performance for the sake of commonality, which can jeopardize the mission effectiveness across the range of possible missions because of architectural decisions related to what to share across the family [8]. Another significant risk from commonality stems from single part failure across the family of vehicles, which can be particularly challenging in an immature certification landscape [9]. Consequently, to realize the benefits of approaching the design of these systems as a family while mitigating its inherent risks a careful methodological approach is necessary. One of the key challenges that emerges in the implementation of this approach is the need to manage the myriad of resulting alternatives, especially during early stages of the design process. As such, this paper focuses on a method to elicit and systematically capture the variability of a family of advanced air mobility vehicles in a traceable way. This allows to relate the stakeholders needs with design alternatives that can support their satisfaction in a structured way. The paper is structured as follows: first, a review of the approaches to design families of systems will be discussed. In particular, the review highlights the challenges in the process of systematically expressing the variability in a family of systems. Then, the approach to identify and propagate variability to support traceability and automation is presented, along with an evaluation of the framework to assess the resulting models. Finally, the results of different tests of this approach will be presented as applied in a case study of the variability of a family of advanced air mobility vehicles.

2. Background

Designing a family of complex vehicles requires a comprehensive approach that provides guidance through each stage of the process. There is a long heritage of research related to design frameworks to properly implement commonality in design. Earlier methods evolved with a marked reliance on mathematical modeling and optimization algorithms to make decisions related to commonality [10, 11]. Product Family Design has emerged as a discipline to design a group of related products derived from a product platform to satisfy a variety of market needs [10]. Its emphasis lies on an analytical approach that includes mathematical definitions to measure commonality [12] along with a reliance on modeling to leverage optimization algorithms.

These techniques have been applied to unmanned aerial systems, since the field has been identified as a potential area of opportunity for the development of vehicle families. For instance, Freeman et. al. present a methodology based on pattern recognition on a database of simulation outputs to guide component commonality decisions in families of unmanned aerial systems [13]. Similarly Chowdhury et. al. present a modification of the CP3 framework for product family design in the field of unmanned aerial systems [14]. In particular, the paper presents methods to measure commonality, along with the determination of system performance, to support decisions for reconfiguration of a vehicle. Overall, these methods are an example of the wealth of promising approaches for the design of product families. However, a problem that's persistent in the methodologies within this line of research, and manifested in the examples discussed, is the difficulty in scaling the system complexity. This is especially centered on enabling traceability of the assumptions and the decisions across the lifecycle due to the challenge of creating consistent engineering thread.

Handling the variability of a family of products throughout its lifecycle is key to design a family of

products. Product Line Engineering (PLE) has emerged as a method to support the management of variability in a family of related products [15]. PLE was first developed to aid in the configuration of software systems by automating the re-use of artifacts when composing a new system [16]. The processes in this field evolved and have been consolidated in a standard where features emerge as a fundamental construct to express the variability within a family of systems [17]. Furthermore, the emergence of Model-Based Systems Engineering (MBSE) was propitious for various efforts to combine Product Line Engineering and MBSE, as a way to better use data and knowledge to mitigate risks of design decisions by enhancing traceability. Different implementations of MBSE and Product Line Engineering started to emerge to solve challenges for the design of vehicle families, highlighting the potential of leveraging MBSE to solve the challenges related to the management and traceability of the variability associated with the family [18, 19]. Today, Model Based Product Line Engineering (MBPLE) is a framework with reported successful uses in various companies developing complex systems [16] to go from single system development to developing a family of systems.

2.1 Feature Modeling

One of the key challenges when designing a family of systems is expressing the various product characteristics of the members of the family. A basic concept to sort this challenge in the product line engineering process is the use of features, which are defined as a *"prominent or distinctive user-visible aspect, quality or characteristic of a software system or systems"* [20]. As such, features are considered *the lingua franca to express the product differences across the life cycle* [21]. The different features are aggregated in a feature model that contains the variability in the family of products. Each of the members of this family can be composed by selecting a specific set of features from a tree-like representation that portrays the various layers of variability. Figure 1 presents this process schematically.

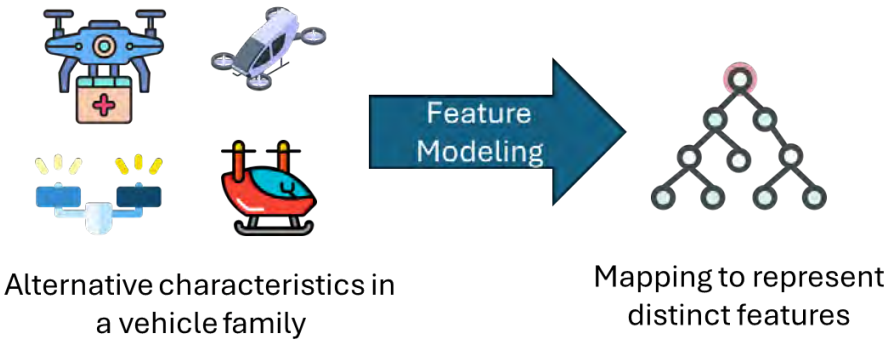


Figure 1 – The distinct characteristics that represent a vehicle family are captured in a feature model

Different methodologies exist to derive a comprehensive feature model. Two of the most important ones are Feature-Oriented Domain Analysis (FODA) and Orthogonal Variability Modeling (OVM). A key difference between these is the consideration of features that are common or not within the product line, with OVM treating variability as orthogonal and therefore not considering things in common [22]. On the other hand, FODA presents an approach where domain knowledge is captured in a set of modeling formalisms in a way that can comprehensively capture the variability and commonality of the family of systems as any distinctive quality is captured [20]. As the main benefit from the family approach lies on the reduction of duplicated efforts through commonality in the system there is a need to manage both, things in common and things that are variable and as a result FODA appears favorable for this approach.

2.2 Challenges Creating a Traceable Feature Model

The emergence of techniques like OVM and FODA solved the need to encode the different variability options. However, a key challenge remains to apply this process and construct relevant feature models. While domain requirement engineering is a core activity to define product lines, the feature modeling process lacks concrete guidelines to extract this variability through highly-specific domain

knowledge [23]. As a response, Martinson et.al. present a methodology to systematically construct feature models for software product lines. However, implementing a method like this directly in cyber-physical systems, like advanced air mobility vehicles, requires additional steps grounded in established design practices for complex systems.

A common approach in systems modeling is to evolve the systems through various layers of abstractions to obtain a model of the system. Different proposed methods have emerged to support this process like MOFLT [24, 25], ARCADIA [26], and RFLP [27]. A key trait in these frameworks is the process of encoding stakeholder expectations, and then leverage these through a logical set of processes to define the system of interest. This aims to establish traceability between the needs and the physical definition of the system [28]. Since these layers of abstraction are identified as one of the dimensions of variability [29], it is reasonable to use these processes to define the variability in the feature model.

Different papers have addressed the idea of leveraging the various layers of abstraction to define the variability in a family of related systems. Chiang et.al propose an approach that relates requirements to the family of products through features in a process called requirement encapsulation [30] Similarly, Lameh proposes a multi-level variability model where generic and variant missions are define to create a physical breakdown of components for automotive subsystems [31]. Other approaches include leveraging the MOFLT framework to propagate the variability at the various layers of abstraction of an airplane subsystem [24]. Finally, Hunter et. al. propose the creation of an automated configuration tool to define the flight software of a family of emergency response drones [32]. These examples illustrate the prevalence of the approach to design families of related products that focuses on layers of abstraction to define the variability, consistent with other modeling approaches, as shown in 2. However, advanced air mobility presents specific challenges to the established approach. The main challenge is related to the uncertain context in stakeholder needs that characterizes a nascent industry such as the advanced air mobility. This challenge brings up the main research question that guides this effort: How to create a comprehensive feature model that systematically captures the variety of stakeholder needs in an evolving market and enables their validation?

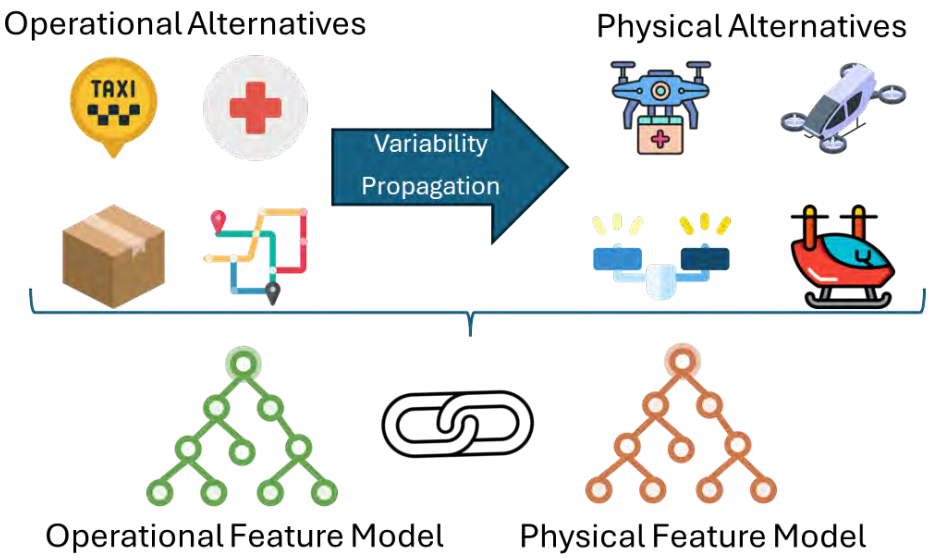


Figure 2 – The propagation of variability requires traceability in the feature models

3. Technical Approach

Defining the variability of a family of novel products heavily relies on a structured elicitation approach that can observe the various layers of abstraction for the family of systems. This technical approach to enable a systematic propagation in an evolving market requires two main aspects to consider: how to capture the variability that emerges and how to achieve the propagation of the variability when new market opportunities emerge. Through this section, the enablers for each of these aspects will be

defined, which allows to compose the technical approach demonstrated in this paper. The technical approach is presented generically so its transition to the design of other families of vehicles is not a limitation. Figure 3 presents a synthesized version of how the different enablers compose the approach.

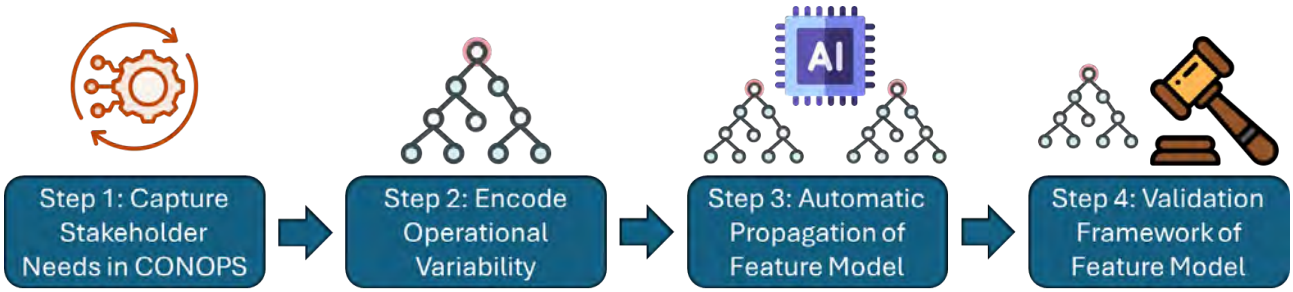


Figure 3 – Variability Modeling for a Novel Vehicle Family

3.1 Defining Alternative Missions

A crucial tool to define the stakeholder expectations in the systems engineering process is the Concept of Operations. Its description of the operation makes it a useful tool to communicate the specific stakeholder needs and possible requirements [33]. The CONOPs captures the needs of the stakeholders for the vehicle as well as relevant information about its operational environment, making it a reasonable starting point to capture the important features. In particular, the MOFLT and ARCADIA frameworks use descriptions of the operations as the starting point of the modeling process [24, 26]. Furthermore, a key part of these methods is to leverage the tools of Model-Based Systems Engineering to formalize the definition of the CONOPS [34, 24]. However, a characteristic of product lines is that there is a large number of possible missions. This means that there needs to be a process to model all these different missions and represent through them the variety of stakeholder needs.
Representing the different alternative missions of a family of vehicles requires a notation that supports variability. In this context, SysML v2 provides new capabilities to embed the variability in the different models built on this language [35, 36]. While other tools exist to create feature models [37, 38, 39], characteristics like the distinction of activities and features within SysML v2, along with its new variability capabilities make it a preferable approach to express the variability of the operational layer. With these tools, a variable CONOPS describing the range of operations expected for the family of systems can serve as an input that communicates the variety of stakeholder needs that motivate the variability of a family of products.

3.2 Automation of the approach to support mission evolution

Once the variability at the operational layer is defined, it needs to be propagated through the other layers of abstraction. This process however, typically relies on extensive domain knowledge about the problem. Furthermore, since this approach targets the definition of variability for families of novel vehicles, there is a need to support their evolution as new cases emerge. In this context, generative artificial intelligence (GenAI) tools can become a key enabler to propagate the variability across multiple layers of abstraction. These tools can leverage domain knowledge gained through its large training space and apply it automatically. Different attempts to test AI's potential for creating a feature model, particularly using Large Language Models (LLMs) have been reported in the literature [40, 16]. However they all make specific calls for establishing a formalized methodology, a process for which this paper aims to establish its first steps. Furthermore, the choice of the modeling environment is appropriate for this approach as the textual notation of SysML v2 make it naturally compatible with Large Language Models [41]. Based on these, leveraging AI to propagate the variability from the operational layer to the physical layer appears to be a promising approach that requires further exploration and evaluation.

3.3 Feature Model Validation

The automation of the process through AI requires the evaluation of the resulting output. Feature models describing a system's variability are typically evaluated based on its correctness, completeness and consistency [42]. Correctness refers to the ability of the feature model to reflect the right information. When working with feature models automatically created with a LLM, correctness refers to the lack of hallucinated physical alternatives, which are manifested as technologically infeasible alternatives. Completeness refers to the ability of the feature model to exhaustively describe the variability in the system. In this context, completeness refers to the quality of the automated feature model to elicit functions that would meet all the mission descriptions, and that each of the functions has physical alternatives to execute them. Finally, in terms of consistency, the validation refers to conforming to the application of the modeling standards. In the case of the automatically generated feature model, the emphasis is on its conformance with the rules of SysML v2. These traits provide a comprehensive evaluation of the tasks related to the creation of a feature model using generative artificial intelligence. However, other important aspects to consider such as efficiency and safety require the expansion of this proposed evaluation framework.

4. Use Case

To demonstrate the technical approach, a specific use case where the objective is the design of a family of Advanced Air Mobility (AAMs) vehicles is presented, as displayed in Figure 4. First, the various missions that are being proposed for AAMs are identified from the literature. These definitions are used to build a variability model of the operational layer of the vehicle family that encompasses this identified set. This provides key inputs to vary the instructions and the context that the LLM has to conduct the task. The resulting feature models are then qualitatively evaluated based on the completeness, correctness and consistency measures defined in the technical approach (see section 3.3). With this use case presentation, the technical approach can be defined concretely, supporting a discussion of the effectiveness of the proposed methodology.

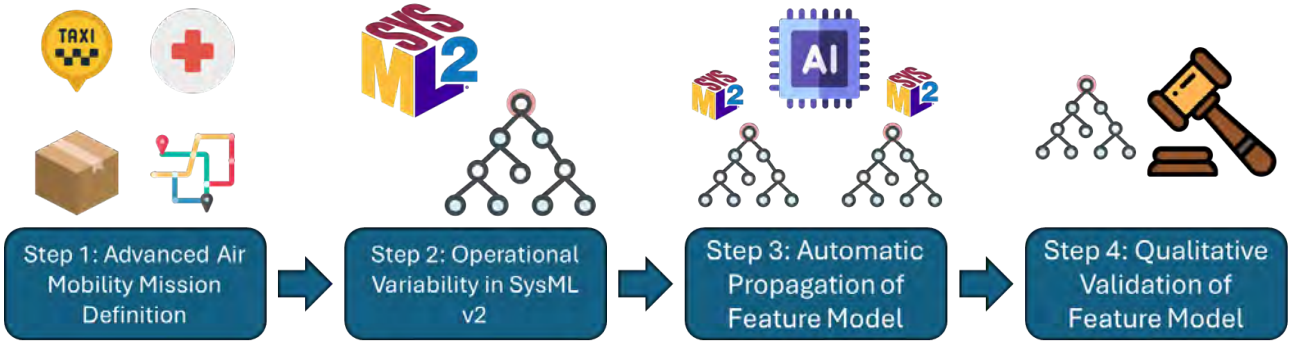


Figure 4 – Synthesis of the Proposed Use Case

4.1 Advanced Air Mobility Mission Definition

Currently, several different missions are being proposed for these systems. Raza et.al. present a thorough analysis of several potential use cases for advanced air mobility and developed their potential CONOPS [43]. Table 1 presents a summary of the various use cases presented. In particular, there are three main use cases analyzed: the air taxi focused on on-demand transportation of people, the air-metro focused on scheduled transportation of people and the last-mile delivery focusing on on-demand transportation of cargo. While there may certainly be other possible use cases, the focus of this paper is to describe the process to create a feature model for a vehicle family of advanced air mobility vehicles and not to provide an exhaustive exploration of the advanced air mobility landscape.

Table 1 – Comparison of Urban Air Mobility (UAM) Use Cases based on [43]

Feature	Air Taxi	Air Metro	Last-Mile Delivery (LMD)
Mission Description	Door-to-door ridesharing or ride-hailing service allowing customers to call VTOLs via apps. Emergency response falls under this category with appropriate vehicle modifications	Extension of public transportation with defined routes and timetables in high-traffic regions.	Transporting goods, food, or documents from distribution centers to customers.
Payload/ Passenger Capacity	1–6 passengers; $\approx$1000 lb payload.	3, 6, 15, or 16 passengers; $\approx$1000 lb payload.	Packages $<$ 5 lb (Walmart tests up to 10 lb).
Typical Range & Duration	10–70 miles per trip; 20–30 minutes.	10–70 miles per trip; vertiports $\approx$20 min apart.	Within 10 miles round trip; 15 mile radius prototypes.
Routing & Scheduling	On-demand; unscheduled and unplanned routes.	Scheduled service with predefined routes and regular timetables.	Unscheduled; flight schedules established as orders are submitted.
Infrastructure Requirements	Large number of vertiports (rooftops, parking structures) for door-to-door access.	100–300 vertiports/hubs, service stations, and charging stations.	Main distribution hubs, vertihubs, and ground stations.
Primary Competitors	Ground taxis, driverless/personal cars, commuter rail, rideshare apps.	Driverless cars, city buses, and subways.	Autonomous and human-driven ground delivery vehicles.
Profitability Timeline	Not viable by 2030 (est. \$100s to \$2000 per passenger).	Expected profitable by 2028–2030 (\$30–\$50 per trip).	Estimated profitable by 2030 (\$4.20 per delivery).

4.2 Operational Variability Model

With these missions identified, the next step is to create a variability model that reflects all this possible alternative use cases. A key enabler in this process is the concept of variation points and variability in SysML v2. In particular, the SysML v2 pilot implementation was deployed locally for the evaluation and renderings used in this paper [44]. The operational description is implemented in two packages, one containing the definitions of alternatives (as shown on Figure 5) and another focusing on the definition of the specific missions, leveraging the alternatives (as shown on Figure 6). The package contains *abstract use case definitions* to define to define the use case structure. These *abstract use case definitions* are then used to create *variation use case definitions* to specify specific alternatives for these definitions through the specification of *variant use cases*. The specific missions are created in a different package where an *abstract use case definition* is created to define the structure of the mission. Finally, the specific missions are created in a *variation use case definition* using the *abstract use case definition* for the missions as a base and leveraging *variant use cases* that redefine the variations according to each of the specified missions. This approach ensures the possibility of reusability and extensibility of the assets, even if it's not fully utilized at this stage.
The definition of the alternatives also relies on a textual notation. The first part considers the definition of alternatives via the *abstract use case*. The documentation added to each of these variants is omitted for brevity. These cases are the representative packages displayed in figures 5 and 6

```
package AAM_Family_OpsDef {
abstract use case def Routing {}
variation use case def Routing_Options :> Routing {
    variant use case Door_to_Door_On_Demand:Routing {}
    variant use case Scheduled_Corridors:Routing{}
    variant use case Hub_to_Door:Routing {}
}
```

Similarly, the definition of the various missions is defined in a package. Here is an abbreviated example of the textual describing how each of the specific alternative use cases is composed.

```
package AAM_Family_Operations {

private import AAM_Family_OpsDef::*;

variation use case def AAM_Operation_Variants :> AAM_Operation {
    variant use case AirTaxi : AAM_Operation {
        redefines routing   :>Routing_Options::Door_to_Door_On_Demand;
        redefines transport :> Payload::Under10Pax;
        redefines location  :>Location_Options::Many_Small_Vertiports;
        redefines control   :> Control_Options::Autonomous;
        redefines range     :> Range_Options::Short_Range;
    }
}}
```

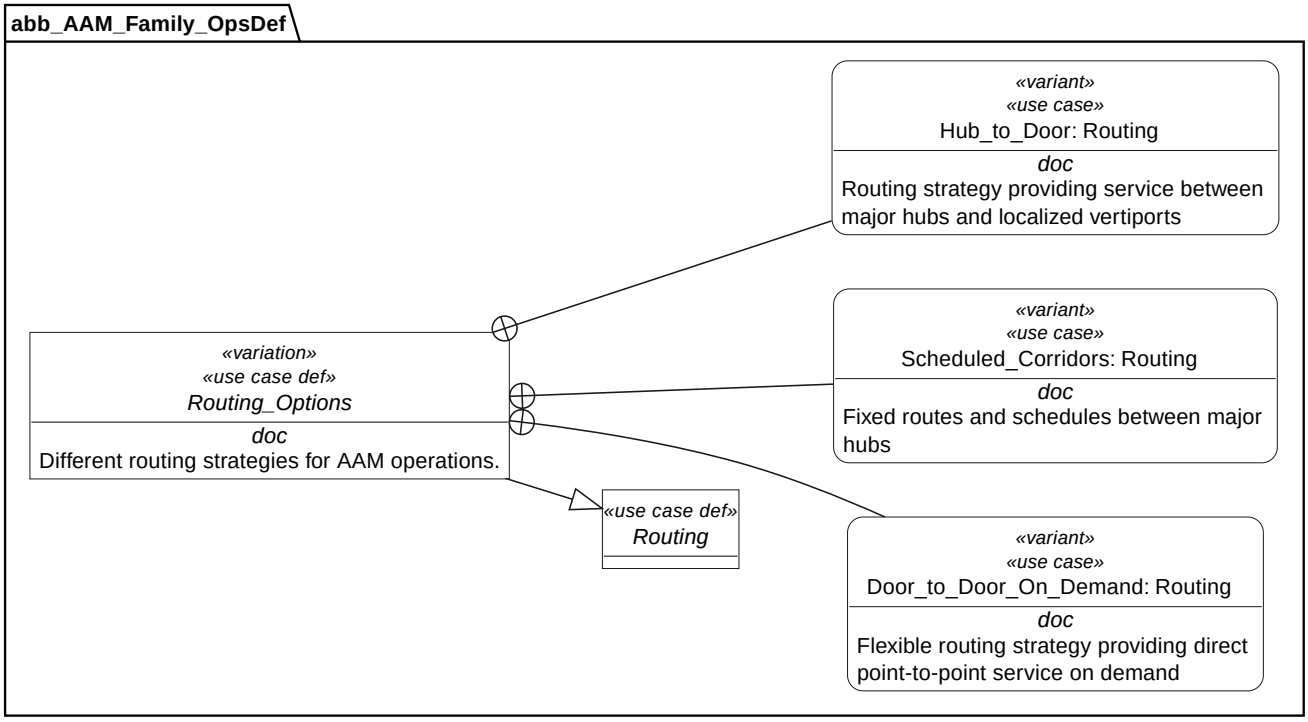


Figure 5 – Representative Definition of Alternative Actions in the Operational Variability Model

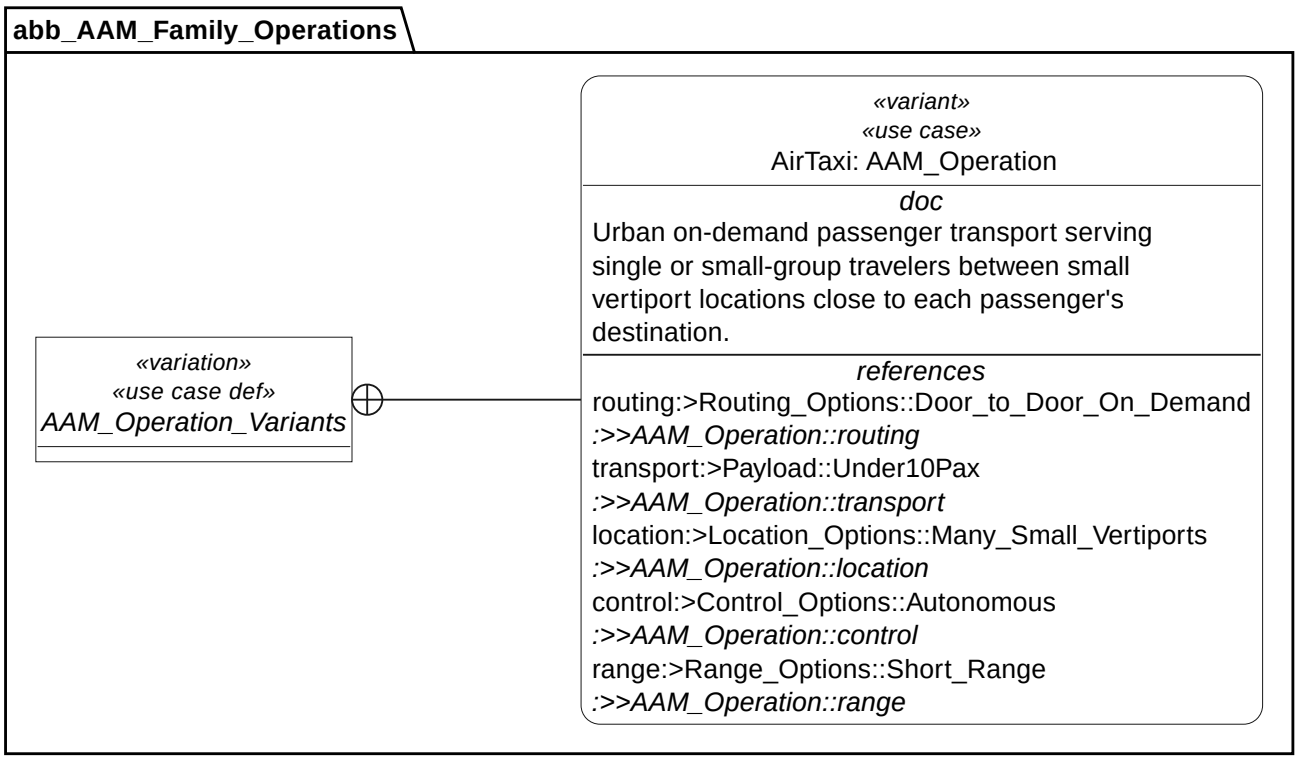


Figure 6 – Definition of Missions in the Operational Variability Model

4.3 Large Language Model experiments to complete a Feature Model

Large Language Models are a key enabler in the approach to automatically complete the various layers of abstraction in the Advanced Air Mobility Feature model . A key aspect to define in this process is the infrastructure to use them. There has been an emergence of *coding agents* in the recent months that allow for the deployment of Large Language Models directly in an Integrated Development Environment (IDE). This approach to execute the experiments was chosen since it allows for better control of the contexts and directly creates the resulting codes in the system, with no need to copy and paste the outputs into the SysML tool. In particular, the Google Antigravity [45] environment was selected to demonstrate this use case, which provides access to Large Language Models by Google and Anthropic directly from the tool. In this tool, a workspace dedicated to this experiments was defined that contained all the necessary information for different experiments that were executed. This was coupled with a version control tool to systematically record the different results of the experiments. However, custom agents were not created for this task and only the base infrastructure provided in Google Antigravity was leveraged.

A crucial part to ensure the effectiveness of tasks charged to LLMs as agents is the effective definition of context, as well as the prompts used to trigger the LLM actions [46]. As part of the context for this use case the workspace included the SysML v2 [44] specification to aid in the creation of consistent SysML v2 models, along with the definition of product line engineering package for SysML v2 proposed in [36]. These references were complemented by the work of Raza et. al. [43] to provide the agent with peer-reviewed context of problem and support the correctness of its responses.

Given the importance of contexts and prompts the main emphasis of these experiments centered around these parts of the task. A key step in this case was the definition of an experiment file, where the instructions for the agent were specified. This experiment file specified in tagged sections, the goal of the experiment, the location of references and additional context and finally the steps that the agent must follow. The steps for these experiments specifically tried to induce the large language models to engage in a chain-of-thought process. Chain-of-thought prompting has been recognized as an effective approach to induce reasoning in large language models [47]. In this specific case, the chain of thought prompting indicated that the model needs to reason through the functional decomposition, create an artifact for it in the form of a SysML v2 model and then use that information to generate the physical decomposition. Using these instructions, three different experiments were executed varying the specific commands in the prompt and context given to them. Each of the experiments was executed once in three different models: Gemini 3 Flash, Gemini 3.1 Pro (High) and Claude Sonnet 4.6 (Thinking). While this does not account for the inherent stochasticity associated with Large Language Models, this effort focuses on an exploratory qualitative analysis of a pilot implementation.

The first experiment focused on the unstructured definition of the missions. For this experiment, the use case models created in section 4.2 were omitted from the context and only the paper was used to reason. However, in this experiment the context included a description of the functional decomposition process, creating a more detailed description of the task and reducing ambiguity. The second experiment included the operational variability described in section 4.2 as part of the context. This provided a structured input that describes the various missions of the vehicle family. However, a key change in the prompt was introduced by dropping the definition of the functional decomposition from the experiment prompt. Finally, the third experiment leverages both important pieces, the definition of the use case using the SysML v2 structure as well as the description of the functional decomposition process. These experiments, with the main differences are displayed in Table 2.

Table 2 – Description of the Experiments

	Experiment 1	**Experiment 2**	**Experiment 3**
Structured Mission Definition	**NO**	**YES**	**YES**
Detailed Task Description	**YES**	**NO**	**YES**

4.4 Qualitative Assessment of the Feature Model

Each of the different experimental runs was analyzed qualitatively based on the framework proposed in section 3.3. In particular, the assessment centers on three dimensions: correctness, completeness and consistency. The validation is conducted qualitatively to the resulting functional and physical variability models respectively. For this use case, correctness indicates that the model did not hallucinate solution and provides sensible functions and physical solutions that encompass the variability. Similarly, completeness is evaluated subjectively and refers to the richness in variability and specifically for the case of the physical variability, it focuses on the coverage of the physical solutions to the functions identified functional decomposition. Finally, consistency is evaluated by ensuring the model can render a valid SysML v2 visualization. This visualization is tested using the Jupyter Notebook kernel provided with the SysML v2 pilot implementation in its April 2026 version.

4.4.1 Observations from Experiment 1

The first experiment focused on using the paper directly, in its PDF version, as the context of the mission, without adding any structure. This was only executed in the Gemini family of models since the Claude Sonnet model implementation available through the Antigravity environment could not read PDFs in the system. However, the model displayed resolve in working through these issues by attempting to use the web to find the paper before the experiment was aborted. The execution with the Gemini 3 Flash model resulted in a functional variability that is correct as there are no hallucinations or infeasible propositions, and while it is displayed consistently with the modeling standards, it lacks completeness as basic functions such as takeoff and landing were not identified. Similarly, the resulting physical feature model displayed a high degree of correctness and consistency as the resulting model displayed a valid SysML v2 representation with technologically feasible solutions. However, the model displays physical alternatives that adequately cover the proposed functional variability but, as the functional variability itself is incomplete, the feature model lacks basic parts where variability can be identified in the family of systems.

The same experiment run was executed with the Gemini 3.1 Pro (High) under the hypothesis that the additional reasoning capabilities of this model would provide further levels of detail. For the functional variability model, the model produced correct features, but it had a worse performance in terms of completeness and consistency. For the physical and the functional variability models, the resulting artifacts did not produce valid SysML v2 models as they had some missing keywords that violated the specification. Regarding the completeness, it reduced the amount of categories and alternatives for each category in the functional variability model compared to the output produced by Gemini 3 Flash, but it was able to reason through additional features such as a sensor suite in the physical variability model. These results slightly contradict the hypothesis that a more powerful model can reason through the problem better and provide better outputs. However, it is important to note though that the model gave an internal error at one point in the simulation and these results are reflective of it. Nonetheless, this result is reported and not rerun to underscore potential challenges related to unexpected outcomes when deploying fully automated approach leveraging this technology. One of the key takeaways relates to the definition of the missions of interest in a structured way that provides a meaningful view of the variability needs. As such the following experiments rely on the operational variability models presented in 4.2.

4.4.2 Observations from Experiment 2

The experiment 2 included the SysML model for the definition of the various use cases but did not define the task of functional decomposition in the prompt. This experiment was executed in Gemini 3 Flash, Gemini 3.1 Pro (High) and Claude Sonnet 4.6 within the Antigravity environment. All three models struggled with the creation of a correct and complete functional decomposition. In all of the generated functional variability models, the task limited itself to only repackage the operational variability model. This severely limits the assessment of correctness and completeness for the functional variability model. The models yielded mixed results in terms of consistency for the functional variability model since Gemini 3 Flash provided a nested variation that violates the SysML v2 specification.

The physical variability produced more promising results as these models created a correct representation of the variability of the system as they produced meaningful and technologically feasible parts.

However, the responses of each of the models were very different in terms of the features it included and the depth of the description. For example, the model yielded by Gemini 3 Flash displayed a bigger emphasis in powertrain and energy management features, while the model generated by Gemini 3.1 Pro (High) focused on cabin layouts and landing infrastructure. The model yielded by Claude Sonnet 4.6 displayed both emphases displayed by the Gemini models as well as additional physical features such as communications packages. Interestingly, the Claude output included a commented functional decomposition as part of the output that, while not in a SysML v2 compliant format, provided insight into the reasoning of the physical variability by the model. Nonetheless, the model did not elicit variability associated with some of the functions that were displayed in this functional decomposition such as those related to the human-machine interface. Finally, in terms of consistency for the physical layer with the PLE package for SysML v2, the Gemini 3 Flash model yielded an attempt to redefine this package which led to an inconsistent definition. However, the models yielded by Claude Sonnet 4.6 and Gemini 3.1 Pro (High) were compliant with the specification. These results indicated the importance of clarity in the prompting that is required to control the possible outputs. The functional variability that was produced did not follow the expected pattern as it indicated confusion between the use cases and the functions of the system. However, the additional considerations in the physical variability are suggestive of advantages introduced by the systematic description of the use cases introduced through the operational variability model.

4.4.3 Observations from Experiment 3

The third experiment re-introduced the explicit definition of the functional decomposition as part of the additional context section of the prompt. Just as in the previous experiments, this configuration was tested with the Gemini 3 Flash, Gemini 3.1 Pro (High) and Claude Sonnet 4.6 (thinking) models within the Antigravity environment. The functional model yielded by Gemini 3 Flash was not correct as it created generic functions that don't mean anything such as *Manage Cargo Hold*. These issues were expressed in the completeness of the model, exemplified by an omission in takeoff functional alternatives and only focusing on landing, even when takeoff and landing were defined together. The functional models yielded by Gemini 3.1 Pro (High) and Claude Sonnet 4.6 (thinking) were correct. Completeness is challenging to evaluate in this context, as it depends on the specific design cycle and its corresponding analyses that this approach is applied to, but the output in this model is appropriately complete for the architecture of an emerging family of advanced air mobility system. Finally, all three functional variability models were created consistently with the SysML v2 specification. These observations are aligned with the hypothesis that the more advanced models (Gemini 3.1 Pro and Claude Sonnet 4.6 (thinking)) would produce better results at a cost of latency over Gemini 3 Flash. The physical feature model displayed similar trends across the model. The physical feature model yielded by Gemini 3 Flash displayed challenges in defining correct alternatives. For example, it precluded the possibility of sensors in the crewed operations as it linked the sensors to the autonomous suite exclusively. The model completeness also has room for improvement as the amount of alternatives is limited to a handful of cases compared to other outputs from these models. The models that engage in more reasoning (Gemini 3.1 Pro (High) and Claude Sonnet 4.6 (thinking)) display a larger number of correct features with clear traces to the functional variability to support the completeness assessment. This detail even includes, in the model yielded by Claude, multi-depth variability definition in the definition of the airframe. However, neither reasoning model maintained full consistency with the modeling specification as the implementation of these yielded errors.

4.4.4 Recap

The execution of these experiments proved the importance of the definition of the context and the prompt when defining tasks that leverage LLMs. It was observed that the definition of the various missions through a structured variability model that yields more correct and complete models. Furthermore, adding the definition of the functional decomposition into the prompt created even better model. Additionally, it was observed that the choice of model is important as the models with higher reasoning capabilities tended to produce more correct and complete models. However, consistency with the modeling standard was not guaranteed with any of them. In particular it was observed that

consistency errors emerged because of the resulting models' increased complexity. These results are synthesized in tables 3 and 4.

Table 3 – Model performance on functional variability across experiments

Experiment	**Model**	Correctness	Completeness	Consistency
Exp. 1 (PDF only)	Gemini 3 Flash	✓	✗	✓
	Gemini 3.1 Pro	✓	✗	✗
Exp. 2 (Ops. model, no decomp.)	Gemini 3 Flash	✗	✗	✗
	Gemini 3.1 Pro	✗	✗	∼
	Claude Sonnet 4.6	✗	✗	∼
Exp. 3 (Ops. model + decomp. task)	Gemini 3 Flash	✗	✗	✓
	Gemini 3.1 Pro	✓	✓	✓
	Claude Sonnet 4.6	✓	✓	✓

✓ = Good ∼ = Partial ✗ = Poor

Table 4 – Model performance on physical variability across experiments

Experiment	**Model**	Correctness	Completeness	Consistency
Exp. 1 (PDF only)	Gemini 3 Flash	✓	∼	✓
	Gemini 3.1 Pro	✓	∼	✗
Exp. 2 (Ops. model, no decomp.)	Gemini 3 Flash	✓	∼	✗
	Gemini 3.1 Pro	✓	∼	✓
	Claude Sonnet 4.6	✓	✓	✓
Exp. 3 (Ops. model + decomp. task)	Gemini 3 Flash	∼	✗	∼
	Gemini 3.1 Pro	✓	✓	∼
	Claude Sonnet 4.6	✓	✓	∼

✓ = Good ∼ = Partial ✗ = Poor

After modifying some of the advanced capabilities that were attempted by Claude Sonnet 4.6 to ensure a correct rendering, an extract of the models is presented in figures 7 and 8. These contain a subset of features in the functional and physical decompositions. They were created using the SysML v2 pilot implementation in Jupyter Notebook.

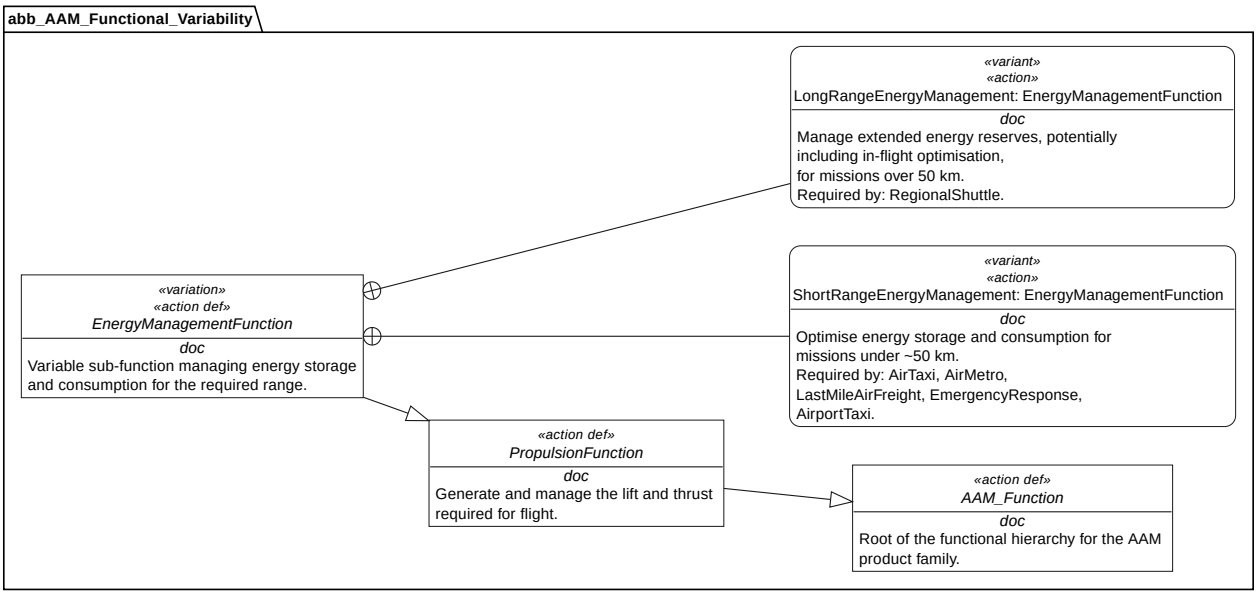


Figure 7 – Energy Management Functional Variability Model by Claude Sonnet 4.6 (thinking)

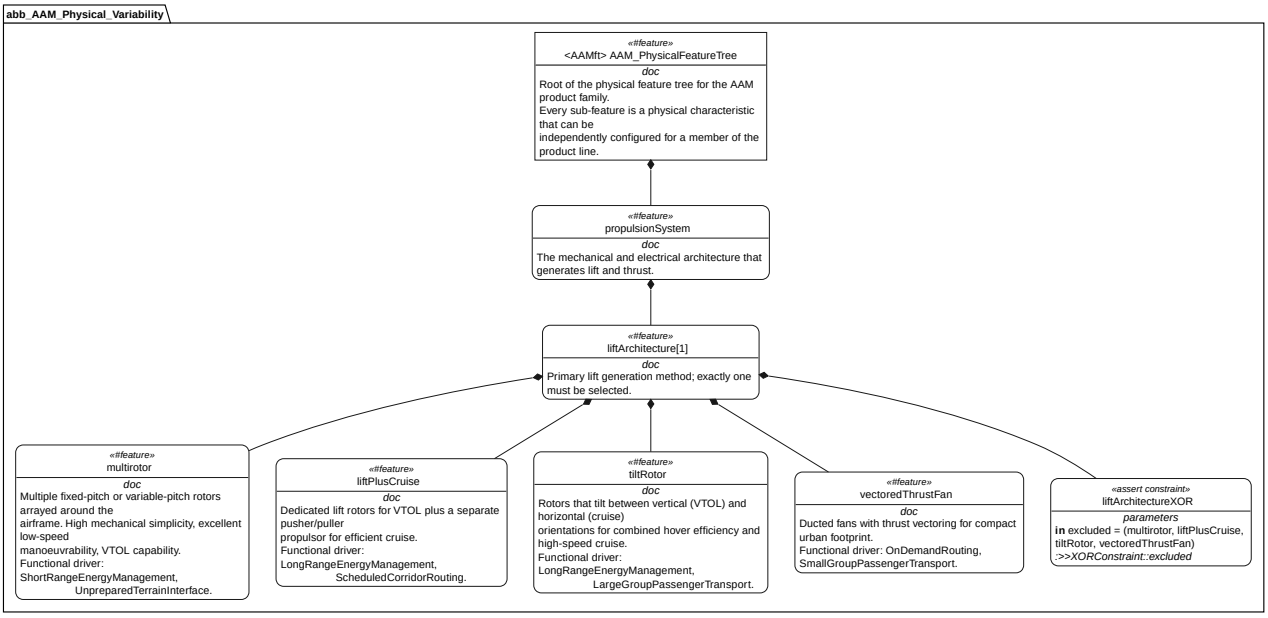


Figure 8 – Propulsion Physical Variability Model by Claude Sonnet 4.6 (thinking)

5. Technical Approach Discussion

The use case defined in this work allowed to provide an exploratory pilot implementation of the technical approach from which various lessons can be identified. One of the main lessons of the approach is the definition of variability at the use case level to support the definitions of the stakeholder expectations. Having this structure facilitated the derivation of functional and physical variability in a vehicle family as it defined specific possibilities that described the range of stakeholder expectations. As such, this Concept of Operations with explicit variability definitions is a key enabler of the approach. The difference in performance between having this structured representation and not having it illustrated in the automated experiments supports this view about the importance of this variability depiction. However, an identified limitation is that this view can create a reductionist view of the system, making the definition of the taxonomy to describe the mission itself a critical step for this technical approach. This taxonomy definition to express the various missions within a product line should be carefully tailored for each of the possible applications.

The automation of the approach is where the experimentation presented in the use case focused. Before dialing into the specific differences observed in the experiments some limitations need to be addressed. As evidenced by Figure 8, the variability that it can create is large and can become unmanageable. Consequently, the utility of automating this approach needs to be clearly established as it risks creating additional work to eliminate alternatives that are completely unnecessary. However, in cases where a product line might be feasible and there is a large degree of uncertainty about the stakeholder needs and regulatory environment such as Advance Air Mobility, the automated elicitation of a large number of alternatives can be beneficial and enrich the thoroughness of the architecting process. As demonstrated in the experiments and previously cited references, LLM's have the potential to do this task well. Nonetheless, the experiments demonstrated the need to carefully select the context and the prompt that the LLM can use as the success of the approach is closely related with this choice and further research can help identify strategies for this task. Furthermore, it could be observed that the different models can produce very different results, and it has been reported that even the same model can produce different outcomes due to inherent stochasticity of Large Language Models. As such, no strong conclusions about the preferability of a given model for this task can be drawn with these tests. The validation framework appeared to be appropriate for the qualitative assessment of the effectiveness for the task but further research to assess the efficiency and security is warranted based on the observed behaviors. Overall, this pilot exploratory demonstration in the Advanced Air Mobility use case highlights the potential of leveraging LLMs in the elicitation of variability to compose a feature model based on operational descriptions.

6. Concluding Remarks and Future Work

The design of families of products that can capture commonality to improve business efficiencies is crucial in the increasingly competitive business landscape. However, implementing this approach carries significant risks which must be implemented in a systematic approach to extract its benefits. Product Line Engineering emerges as a key approach to support the management of variability in complex systems. An important step of the Product Line Engineering process is the definition of variability through a feature model. However, there are challenges in the systematic propagation of variability through the various layers of abstraction, ranging from the operational variability and into the physical feature model. The technical approach presented is an attempt to create a more traceable process when defining this variability.

The technical approach consists of a four step process to define the physical feature model. It begins with the identification of the missions that could be supported by the product line and the various CONOPS for each. These are encoded into a model that describes the variability at the operational layer. Then, an automated propagation of this variability is proposed to create the feature model. This feature model is evaluated in terms of correctness, completeness and consistency to ensure its quality. The technical approach was demonstrated with the use case of a family of Advanced Air Mobility vehicles as they exhibit large degrees of business uncertainty and motivate the need for possible automated approaches.

The exploratory pilot implementation of the technical approach in this use case showed the potential

of automating this approach using consumer-grade frontier Large Language Models. In particular, the experiments revealed the importance of structured definitions of variability as part of the context as well as detailed instructions of the tasks in the prompts. However, these demonstrations revealed two main challenges to refine the approach. First, the structure of the definition of variability in the missions was clearly prevalent in the results of the model when defining the functional and physical variability. This can motivate future work in each field where this approach is applied to establish standards on how the missions are expressed in each of the fields and what characteristics in each of them are crucial to be identified. The second challenge relates to the implementation of the validation framework. While the qualitative evaluation implemented in this paper can provide a starting point for the evaluation of the resultant feature models, more thorough benchmarking processes could be devised for this task. These should focus not just on the effectiveness at generating the feature model but also on the efficiency, the robustness and the safety on the approach as the latest research suggests [48]. These evaluations would facilitate more meaningful comparisons between the models can be made. Several manual and automated approaches for the evaluation can be explored such as the LLM's as a judge approach, or including a human in the loop Additionally, more advanced workflows involving custom agents and skills can be investigated as these can be tailored to specific tasks within the described technical approach. Nonetheless, the focus on an operational variability model to describe the range of stakeholder expectations can prove useful regardless on the agent executing the variability propagation, whether it's humans or machines.

Acknowledgments

Schematics used images from Flaticon.com with permission.

Contact details

- Angel Mario Zarate Villazon: amzv3@gatech.edu
- Dr. Michael Balchanos: michael.balchanos@asdl.gatech.edu
- Prof. Dimitri Mavris: dimitri.mavris@aerospace.gatech.edu

Copyright Statement

The authors confirm that they, and/or their company or organization, hold copyright on all of the original material included in this paper. The authors also confirm that they have obtained permission, from the copyright holder of any third party material included in this paper, to publish it as part of their paper. The authors confirm that they give permission, or have obtained permission from the copyright holder of this paper, for the publication and distribution of this paper as part of the ICAS proceedings or as individual off-prints from the proceedings.

References

[1] Peter Hodges. Issues in Automotive Product Platform Strategies. SAE International, March 2004.

[2] Hugo G. Chalé Góngora, Alain Dauron, and Thierry Gaudré. A Commonsense-Driven Architecture Framework. Part 1: A Car Manufacturer's (naïve) Take on MBSE. *INCOSE International Symposium*, 22(1):1410–1424, 2012. _eprint: https://incose.onlinelibrary.wiley.com/doi/pdf/10.1002/j.2334-5837.2012.tb01412.x.

[3] Karen Willcox and Sean Wakayama. Simultaneous Optimization of a Multiple-Aircraft Family. *Journal of Aircraft*, 40(4):616–622, July 2003.

[4] Mohammad Sadrani, Filippos Adamidis, Laurie A. Garrow, and Constantinos Antoniou. Challenges in urban air mobility implementation: A comparative analysis of barriers in Germany and the United States. *Journal of Air Transport Management*, 126:102780, June 2025.

[5] Jianxin (Roger) Jiao, Timothy W. Simpson, and Zahed Siddique. Product family design and platform-based product development: a state-of-the-art review. *Journal of Intelligent Manufacturing*, 18(1):5–29, February 2007.

[6] J. E. Klansnic and R. J. Dittenberger. Boeing 757/767 Commonality Design Philosophy. page 810845, August 1981.

[7] Alexander Brüggen and Levin Klose. How fleet commonality influences low-cost airline operating performance: Empirical evidence. *Journal of Air Transport Management*, 16(6):299–303, November 2010.

[8] David Robertson and Karl Ulrich. Planning for Product Platforms. *MIT Sloan Management Review*, July 1998.

[9] Bruce G. Cameron and Edward F. Crawley. Crafting Platform Strategy Based on Anticipated Benefits and Cost. In Timothy W. Simpson, Jianxin Jiao, Zahed Siddique, and Katja Hölttä-Otto, editors, *Advances in Product Family and Product Platform Design: Methods & Applications*. Springer New York, New York, NY, 2014.

[10] Timothy W. Simpson, Tucker Marion, Olivier De Weck, Katja Ho¨ltta¨-Otto, Michael Kokkolaras, and Steve B. Shooter. Platform-Based Design and Development: Current Trends and Needs in Industry. In *Volume 1: 32nd Design Automation Conference, Parts A and B*, pages 801–810, Philadelphia, Pennsylvania, USA, January 2006. ASMEDC.

[11] Timothy W. Simpson, Jianxin Jiao, Zahed Siddique, and Katja Hölttä-Otto, editors. *Advances in Product Family and Product Platform Design: Methods & Applications*. Springer New York, New York, NY, 2014.

[12] Henri J. Thevenot and Timothy W. Simpson. Commonality indices for product family design: a detailed comparison. *Journal of Engineering Design*, 17(2):99–119, April 2006.

[13] Dane Freeman, Dongwook W Lim, Elena Garcia, and Dimitri N Mavris. METHODOLOGY FOR THE DESIGN OF UNMANNED AIRCRAFT PRODUCT FAMILIES. *Proceedings of the 28th Congress of the International Council of the Aeronautical Sciences*, 2012.

[14] Souma Chowdhury, Victor Maldonado, Weiyang Tong, and Achille Messac. New Modular Product-Platform-Planning Approach to Design Macroscale Reconfigurable Unmanned Aerial Vehicles. *Journal of Aircraft*, 53(2):309–322, March 2016.

[15] INCOSE. Feature-based Systems and Software Product Line Engineering: A Primer, 2023.

[16] Marco Forlingieri, Tim Weilkiens, and Hugo Guillermo Chalé-Gongora. *Model-based product line engineering (MBPLE): the feature-based path to product lines success*. John Wiley & Sons, Incorporated, Newark, 2025.

[17] ISO. ISO/IEC 26580:2021: Software and Systems Engineering - Methods and Tools for the Feature-Based Approach to Software and Systems Product Line Engineering.

[18] Hugo G. Chalé Góngora, Marco Ferrogalini, and Christophe Moreau. How to Boost Product Line Engineering with MBSE - A Case Study of a Rolling Stock Product Line. In Frédéric Boulanger, Daniel Krob, Gérard Morel, and Jean-Claude Roussel, editors, *Complex Systems Design & Management*, pages 239–256, Cham, 2015. Springer International Publishing.

[19] Fatma Karagoz, Burak Bagdatli, Yu Cai, and Dimitri N. Mavris. Interactive Model-based Environment for Aircraft Family Conceptual Design Decisions. In *AIAA Scitech 2021 Forum*, AIAA SciTech Forum. American Institute of Aeronautics and Astronautics, January 2021.

[20] Kyo C. Kang, Sholom G. Cohen, James A. Hess, William E. Novak, and A. S. Peterson. Feature-Oriented Domain Analysis (FODA) Feasibility Study:. Technical report, Defense Technical Information Center, Fort Belvoir, VA, November 1990.

[21] Paul C. Clements. Product Line Engineering Comes to the Industrial Mainstream. *INSIGHT*, 22(2):7–14, 2019. _eprint: https://incose.onlinelibrary.wiley.com/doi/pdf/10.1002/inst.12241.

[22] Klaus Pohl, Günter Böckle, and Frank Van Der Linden. *Software Product Line Engineering*. Springer, Berlin, Heidelberg, 2005.

[23] Johan Martinson, Wardah Mahmood, Jude Gyimah, and Thorsten Berger. FM-PRO: A Feature Modeling Process. *IEEE Transactions on Software Engineering*, 51(1):262–282, January 2025.

[24] Raphael Henrique Madeira, Davi Henrique De Sousa Pinto, and Marco Forlingieri. Variability on System Architecture using Airbus MBPLE for MOFLT Framework. In *INCOSE International Symposium*, volume 33, pages 601–615, July 2023.

[25] Surya Prakash Baskaran, Jorge Camacho Casero, Burak Bagdatli, and Dimitri Mavris. MBSE-Driven MDAO in Co-Architecture and Concurrent Engineering Workflows for Aircraft Development. In *AIAA SCITECH 2026 Forum*. American Institute of Aeronautics and Astronautics, January 2026. _eprint: https://arc.aiaa.org/doi/pdf/10.2514/6.2026-0193.

[26] Pascal Roques. MBSE with the ARCADIA Method and the Capella Tool. In *8th European Congress on Embedded Real Time Software and Systems (ERTS 2016),*, January 2016.

[27] Kevin Baughey. Functional and Logical Structures: A Systems Engineering Approach. pages 2011–01–0517, Detroit, Michigan, United States, April 2011.

[28] E. N. Omoarebun, S. Cimtalay, and D. N. Mavris. Formalizing the decomposition process between elements in the RFLP framework using axiomatic design theory. AIAA AVIATION Forum, 2023.

[29] Marco Forlingieri. The four dimensions of Variability and their impact on MBPLE: How to approach variability in the development of aircraft product lines at Airbus. In *Proceedings of the 16th International Working Conference on Variability Modelling of Software-Intensive Systems*, VaMoS '22, pages 1–4, New York, NY, USA, February 2022. Association for Computing Machinery.

[30] Thomas Chiang, Richard Paige, Alan Wassyng, and Sebastien Mosser. Mapping Requirements to Features to Create Traceability in Product Line Models. In *Proceedings of the ACM/IEEE 27th International Conference on Model Driven Engineering Languages and Systems*, MODELS Companion '24, pages 1180–1187, New York, NY, USA, October 2024. Association for Computing Machinery.

[31] José Lameh. *Towards a Systemic Product Line Engineering Approach for Systems Engineering Projects : from Models to Configurations*. PhD Thesis, Université Paris-Saclay, November 2025.

[32] Michael Hunter, Usman Gohar, Myra Cohen, Robyn Lutz, and Jane Cleland-Huang. A Family-Based Approach to Safety Cases for Controlled Airspaces in Small Uncrewed Aerial Systems. In *AIAA AVIATION FORUM AND ASCEND 2024*, AIAA Aviation Forum and ASCEND co-located Conference Proceedings. American Institute of Aeronautics and Astronautics, July 2024.

[33] Hirshorn, Steven R., Linda D. Voss, and Linda K. Bromley. NASA Systems Engineering Handbook. Technical Report HQ-E-DAA-TN38707, 2017.

[34] Howard D. Gans. Development of a Space Vehicle CONOPS Using SysML and the Unified Profile for DoDAF and MoDAF (UPDM). In *AIAA SPACE and Astronautics Forum and Exposition*, Orlando, FL, September 2017. American Institute of Aeronautics and Astronautics.

[35] Jordan Epp, Thomas Robert, Olivier Ruch, and Alison Olechowski. Towards SysML v2 as a Variability Modeling Language. In *2023 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*, pages 251–256, October 2023.

[36] Tim Weilkiens, Marco Forlingieri, and Vince Molnár. Next Generation MBPLE with SysML v2: Feature Modeling, Variability Modeling, and API Potentials. *INCOSE International Symposium*, 35(1):1589–1602, 2025. _eprint: https://incose.onlinelibrary.wiley.com/doi/pdf/10.1002/iis2.70058.

[37] Danilo Beuche, Holger Papajewski, and Wolfgang Schröder-Preikschat. Variability management with feature models. *Science of Computer Programming*, 53(3):333–352, December 2004.

[38] Charles Krueger and Paul Clements. Systems and Software Product Line Engineering. *Encyclopedia of Software Engineering*, 2013.

[39] Thomas Thüm, Christian Kästner, Fabian Benduhn, Jens Meinicke, Gunter Saake, and Thomas Leich. FeatureIDE: An extensible framework for feature-oriented software development. *Science of Computer Programming*, 79:70–85, January 2014.

[40] José Lameh, Alexandra Dubray, and Marija Jankovic. Automating Feature Modeling in Product Line Engineering for Systems Engineering: The Application of Natural Language Processing:. In *Proceedings of the 13th International Conference on Model-Based Software and Systems Engineering*, pages 450–457, Porto, Portugal, 2025. SCITEPRESS - Science and Technology Publications.

[41] John K. DeHart. Leveraging Large Language Models for Direct Interaction with SysML v2. *INCOSE International Symposium*, 34(1):2168–2185, 2024. _eprint: https://incose.onlinelibrary.wiley.com/doi/pdf/10.1002/iis2.13262.

[42] Anjali Sree-Kumar, Elena Planas, and Robert Clarisó. Extracting software product line feature models from natural language specifications. In *Proceedings of the 22nd International Systems and Software Product Line Conference - Volume 1*, SPLC '18, pages 43–53, New York, NY, USA, September 2018. Association for Computing Machinery.

[43] Waleed Raza, Justus Renkhoff, Olayemi Ogirimah, Gurvir Kaur Bawa, and Richard S. Stansbury. Advanced Air Mobility: Innovations, Applications, Challenges, and Future Potential. *Journal of Air Transportation*, 33(2):169–187, 2025. _eprint: https://doi.org/10.2514/1.D0440.

[44] Systems-Modeling/SysML-v2-Pilot-Implementation, May 2026.

[45] Google. Google Antigravity Documentation.

[46] Prithvi Rajasekaran, Ethan Dixon, Carly Ryan, and Jeremy Hadfield. Effective context engineering for AI agents, September 2025.

[47] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc V. Le, and Denny Zhou. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. *Advances in Neural Information Processing Systems*, 35:24824–24837, December 2022.

[48] Meltem Subasioglu, Turan Bulmus, Wafae Bakkali, and Anant Nawalgaria. Agent Quality. May 2026.