# Toward a Unified Variability Language: Integrating Feature-Model into the ArchiMate Metamodel

Ahmed Dehne[1] and Kurt Sandkuhl[1,2]

[1] *Rostock University, Albert-Einstein-Str. 22, 18059 Rostock, Germany*
[2] *Jönköping University, Box 1026, 55111 Jönköping, Sweden*

**Abstract**

The rapid pace of digital transformation and emerging AI-driven solutions is driving variability across enterprise architectures (EA), spanning business processes, data structures, and supporting IT services. Effectively managing this variability requires a methodical approach that integrates cross-layer concerns. Motivated by this challenge, we propose an extension to the ArchiMate metamodel that embeds feature-modeling constructs directly into EA models. Our approach introduces three feature types—Mandatory, Optional, and Alternative—as first-class elements and defines "Building Blocks" to encapsulate modular combinations of business activities, application services, and data objects. We implement these extensions within the open-source Archi tool, leveraging its native meta-model customization capabilities. A detailed industrial case study demonstrates the practicality and expressiveness of our extended metamodel. This work advances method support for variability management in EA.

**Keywords**

Variability, Enterprise Architecture Management, ArchiMate Metamodel Extension, Feature Modeling, Enterprise Architecture building block

## 1. Introduction

The ongoing digital transformation, the emergence of innovative business models, and the proliferation of artificial intelligence (AI) solutions are driving a significant increase in variability within enterprises. This variability manifests across multiple layers of an organization simultaneously. For example, enterprises often face numerous variants of business processes, which in turn necessitate adaptations in the underlying data architectures. These observations are supported by various studies examining the impact of digital transformation on enterprise architecture( e.g., [1], [2], [3]) as well as the influence of AI on organizational structures and operations (e.g., [4], [5]). As a result, managing variability has become a routine but complex challenge in enterprise operations.

To address this, organizations adopt different strategies ranging from rigid standardization, where variability is minimized, to approaches that embrace full flexibility. Regardless of the chosen strategy, a deeper understanding of how changes affect different parts of the enterprise is essential. Enterprise Architecture (EA) models serve as a tool to visualize the interdependencies across various enterprise layers. However, current research and practice offer limited support for managing variability across these architecture layers in a cohesive manner.

At the same time, data engineering is gaining prominence in organizations due to the growing reliance on data-driven products and services, as well as AI-based solutions [6]. From an EA standpoint, effective data engineering should align closely with the organization's data architecture to prevent issues such as incompatibility and unnecessary structural divergence. We propose designing modular, data-aware building blocks integrated in business processes. Methodical support

1PoEM2025: Companion Proceedings of the 18th IFIP Working Conference on the Practice of Enterprise Modeling: PoEM Forum, Doctoral Consortium, Business Case and Tool Forum, Workshops, December 3-5, 2025, Geneva, Switzerland
ahmed.dehne@uni-rostock.de (A. 1); kurt.sandkuhl@uni-rostock.de (A. 2)
0000-0002-7431-8412 (A. 2)

for variability management can enable enterprises to respond more flexibly to change, improve consistency across architecture layers.

In previous work, variability challenges in EA have been investigated [7] and the prototype of a development method for identifying building blocks in EA models that integrate several architecture layers has been developed. Building on this work, this paper aims to improve this method by achieving a better integration of variability concepts into the modeling language ArchiMate, which is used to represent the building blocks. More specifically, we enhance ArchiMate by introducing key variability-modeling concepts, integrating these extensions into the Archi modeling environment, and demonstrating their application through a case study.

The remainder of the paper is organized as follows. Section 2 outlines our research methodology. Section 3 presents the necessary background in enterprise architecture management and reviews related work on EAM building blocks. In Section 4, we describe our extensions to ArchiMate—incorporating feature-model concepts—including a summary of our previous work (4.1), the tailored metamodel augmentations for seamless feature-model integration in Archi (4.2), and the core modeling elements underpinning the combined ArchiMate–feature model framework (4.3). Section 5 illustrates how the extended metamodel can be used. Section 6 discusses findings and outlines future work.

## 2. Research Approach

The main objective of this research is to contribute to a better understanding of variability management in enterprise architecture. The project follows the paradigm of design science research (DSR) [8]. DSR is a research paradigm aiming at problem-solving in organizational settings, focusing on developing valid and reliable knowledge for designing the required solutions. The envisioned solution, called "artefact" in DSR, in our research is methodical and technological support for managing variability based on enterprise architecture building blocks. DSR research projects typically consist of several phases and require the use of different research methods depending on the DSR phase and intended design solution. Based on a detailed problem investigation and requirements definition in previous work, this paper concerns the third phase of a DSR project: design and evaluation of the artefact.

The “design and evaluate” step in the DSR process typically includes several iterations in search of the best design of the artefact. Starting from the method prototype presented in previous work, this paper aims at a better integration of the representation of building blocks as models. In the first prototype, two models are used in combination – the EA model of the building block and a feature model to capture dependencies. In this iteration of the design, this integration of both models is in focus. More concretely, an extended ArchiMate meta-model is proposed and validated in a case study.

## 3. Background and Related Work

This section lays the theoretical foundation for this thesis. Section 3.1 introduces the fundamentals of Enterprise Architecture Management (EAM). Section 3.2 summarizes prior research on variability management in EAM, derived from a structured literature review. Section 3.3 presents the concept of Method Engineering. Finally, Section 3.4 reviews our earlier contributions, including the method requirements, the development of a prototype methodology, and insights gained from an industrial case study.

### 3.1. Enterprise Architecture Management

Modern enterprises operate through a complex network of stakeholders engaged in development, operation, and governance. These actors often require different views of organizational structures, processes, and components. To support alignment and effective communication across these domains,

architectural thinking has emerged as a strategic practice [9]. In this context, architecture refers to the core elements of an enterprise, their interconnections, and the guiding design principles.

Enterprise Architecture Management (EAM) offers a systematic approach to modeling and managing these elements. As a management discipline, EAM seeks to provide a cohesive representation of the enterprise, facilitating planning, transformation, and continuous improvement across various architectural layers [10]. The resulting enterprise architecture (EA) acts as a structured map—capturing the current state, interdependencies, and design logic of the system [11].

TOGAF, a well-established EAM framework, is widely recognized as an industry standard [12]. It defines three primary architectural domains: business architecture, information architecture (often divided into data and application architectures) and technology architecture, which encompasses infrastructure and physical systems.

To support these frameworks in practice, modeling languages such as ArchiMate are commonly used. ArchiMate provides a standardized notation that allows consistent representation across domains and supports communication between diverse stakeholders.

### 3.2. Variability in EAM

Our earlier study [7] explored how variability is addressed within the field of Enterprise Architecture Management through a structured literature review (SLR), based on the methodology proposed by Kitchenham [13]. The core research question guiding the review was: "What is the current state of research on managing variability in enterprise architecture?" The process involved defining selection criteria, systematically identifying relevant literature, extracting key insights, and synthesizing the results.

The review revealed a broad range of studies addressing variability at various architectural layers. A significant portion of the literature focuses on Business Architecture. For instance, works by Rurua et al. [14], Mani et al. [15], Asadi et al. [16], and Benavides et al [17] examine variability in enterprise domains, modeling approaches, and service-oriented systems. Technology Architecture also receives considerable attention, with Wille et al. and Wehling et al. [18] investigating methods for mining architectures and reducing complexity. In the realm of Application Architecture, research by Langermeier et al. [19] and Nerome & Numao [20] explores modular development and software product lines. Fewer studies address Software Architecture (e.g., Allian et al.) [21] or Information Architecture, with Adjoyan & Seriai [22]providing a rare example focused on dynamic, service-based modeling.

A key takeaway from this review is that most existing approaches concentrate on variability at a single architectural layer, often overlooking cross-layer effects. To address this limitation, our work advocates for an integrated, multi-layer approach to variability modeling, emphasizing reuse strategies at various levels. An updated scan of the literature for this paper found no new significant contributions beyond our own work [7].

## 4. ArchiMate Meta-Model Extension for Feature Modeling

Starting from a brief summary of our previous work, this section motivates and develops the extension of the ArchiMate meta-model and presents its concepts in detail.

### 4.1. Contributions from Previous Publications

In previous work [23], we applied Method Engineering principles to define method requirements and a prototype methodology for variability management in enterprise architecture (EA). The identified key requirements are: explicit modeling of cross-layer variability (business, application, and data layers); integration of disparate approaches into a cohesive variability framework; a unified modeling approach for all layers; use of standardized languages like ArchiMate for clarity and tool compatibility; seamless integration with commercial modeling tools; and visual representation of

architectural dependencies to aid configuration and reduce complexity. To meet these requirements, we developed a prototype method based on modular building blocks—configurable units derived from a reference architecture and adapted for specific business contexts, each comprising one or more business processes or functional activities, corresponding application and data components, and clearly defined interfaces for compatibility and reuse—modeled in ArchiMate and extended with feature models to describe configuration options, constraints, and interdependencies. The method follows a structured, sequential process of process modeling (analyzing enterprise processes to identify variability points and classify them into archetypes), variability analysis (capturing optional and alternative design elements with feature models), data architecture alignment (identifying data needs, sources, and integration logic), feature-model development (creating structured representations of the configuration space), and block definition (composing complete building blocks with traceable interfaces and reuse potential). Implemented in ArchiMate and validated in an industrial study, the approach proved effective at managing architectural complexity and enhancing adaptability in dynamic enterprise environments.

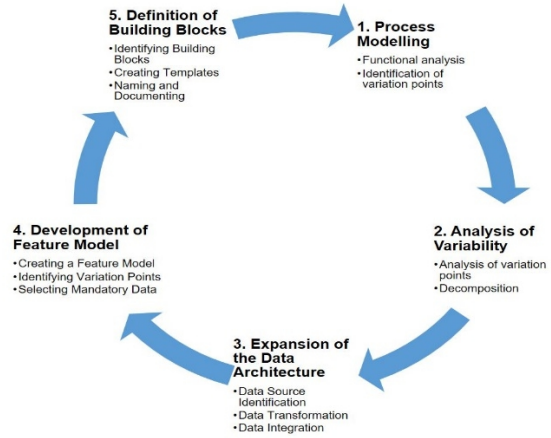



**Figure 1:** Methodical approach and method prototype

Figure 1 depicts the methodical approach and overall architecture of our prototype. To fulfill steps 1 (process modeling), 2 (variability analysis) and 3 (data-architecture expression), we leveraged ArchiMate within the Archi tool [24]. For step 4 (feature-model development), we employed Microsoft PowerPoint as a lightweight, flexible environment for designing feature models. The ArchiMate and PowerPoint artifacts defined our initial architectural building blocks, which we then manually re-modeled in ArchiMate using Archi. This dual-tool, dual-language workflow revealed a significant limitation: maintaining consistency across two distinct modeling languages and tools proved time-consuming and error-prone. Specifically:

- Synchronization Overhead: Whenever we updated the feature model, we had to manually propagate changes into the ArchiMate model—and vice versa—to keep both artifacts aligned, effectively doubling our effort and prolonging iteration cycles.
- Risk of Inconsistency: Manual translation between PowerPoint and ArchiMate increased the likelihood of discrepancies—missing attributes, mismatched relationships or outdated elements—that compromised the integrity of our documentation.
- Tooling Gaps: Neither Archi nor PowerPoint supports native interoperability, forcing ad-hoc export/import steps (e.g., screenshots, copy-and-paste, manual property mapping) that further amplified the potential for human error.
- Steep Learning Curve: Mastery of two distinct modeling paradigms and interfaces complicated onboarding and limited opportunities for cross-domain collaboration.

These challenges not only hindered productivity but also compromised the accuracy and maintainability of our enterprise architecture models. To overcome these limitations, we enhanced our approach by integrating ArchiMate and feature modeling into a single, unified language and tool. Building on the ArchiMate metamodel, we retained its core enterprise architecture concepts while embedding feature-model constructs directly within it.

## 4.2. Modeling Feature Modeling in Archi: Tailored Metamodel Extensions for Seamless Integration

For the extension of the ArchiMate metamodel to support feature-model notation. We evaluated three potential approaches—each designed to represent "Mandatory," "Optional," and "Alternative" feature values. Below, each approach is described in turn, with its respective advantages and disadvantages.

*Approach 1: Extend ArchiMate Concepts via Properties*

- Description: Augment existing ArchiMate elements (e.g., Business Process) by adding a "Feature Value" property (Mandatory, Optional, Alternative) directly to each relevant concept, leveraging the inherent property mechanism of the ArchiMate modeling language.
- Advantages:
  Rapid Implementation: Can be configured immediately via the built-in property editor.
  Low Overhead: No need to define new element types or write custom scripts.
- Disadvantages:
  Diagram Clutter: Every view must display the additional property, which may clutter the interface.
  Extraction Required: A separate script or function is needed to scan all elements and compile their feature-value assignments.
- Proposal: Creation of Building Blocks
- "Feature Value" Property: Add a single enumerated property (Mandatory, Optional, and Alternative) to relevant element types.
- Export Script: One simple script to scan elements and output their feature-value assignments.

*Approach 2: Model Feature Values as Relationship Properties*

- Description: Attach the "Feature Value" attribute to the relationships connecting ArchiMate elements, indicating whether each connection is Mandatory, Optional, or Alternative.
- Advantages:
  Quick Setup: Leverages the same property-based mechanism as Approach 1 for rapid deployment.
  Consistent Editing Workflow: Uses familiar property-editing features within Archi.
- Disadvantages:
  Similar Drawbacks to Approach 1: Relationship properties must be visible in every diagram, and additional scripting is required to aggregate and report feature assignments.

*Approach 3: Explicitly Define New ArchiMate Concepts in the Metamodel*

- Description: Introduce three new element types in the ArchiMate metamodel—Feature Mandatory, Feature Optional, and Feature Alternative—which can then be related back to any existing ArchiMate concept via standard relationships.
- Advantages:
  One-time Setup: New concept types need only be added once to the metamodel. Centralized Editing: All feature-value definitions live in a single place, making updates simple.
  Clean Diagrams: Feature values appear as distinct elements, styled and labeled consistently without cluttering the underlying elements.
- Disadvantages:
  Additional modeling step: Each Feature element requires explicit creation and linking to the relevant concepts, introducing one extra modeling activity per assignment.
- Proposal: Creation of Building Blocks
- New Element Types: In the metamodel, define three building-block element types—Feature Mandatory, Feature Optional, Feature Alternative. Standard Relationship Patterns: Create and document a "Feature Assignment" relationship pattern that links any ArchiMate element to one of these new Feature types. Reusable Viewpoint: Build a dedicated "Feature Model" viewpoint that automatically surfaces all Feature elements and their assignments in a single diagram.

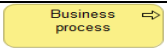
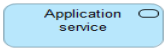
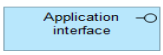
We chose Approach 3 because a single metamodel extension avoids repeated configuration, centralizes feature definitions for easier maintenance, and uses distinct element types to keep views clear and feature values visible.

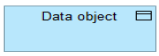
### 4.3. Core Modeling Elements for Seamless ArchiMate and Feature Model Integration

After deciding to extend the ArchiMate metamodel to support feature modeling, it was essential to define the elements incorporated into this extension to maintain conceptual consistency and ensure end-to-end traceability.

The extended metamodel includes a focused selection of ArchiMate constructs—Business Process, Business Activity, and Business Role from the Business Layer; Application Component, Application Service, and Application Interface from the Application Layer; and, where applicable, Technology Layer elements—augmented by Data Object to capture essential information for process execution and underpin core data-architecture aspects. At the same time, variability and configuration options are handled via three feature-modeling elements—Mandatory Feature, Optional Feature, and Alternative Feature—organized in a hierarchical structure to clearly distinguish each feature's status. Table 2 provides an overview of the key elements from ArchiMate and feature modeling.

**Table 1**
Overview of ArchiMate and Feature Modeling Elements Used in the Case Study

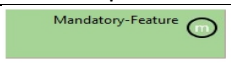
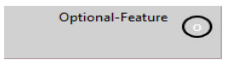
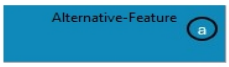
| Element name | Element picture | Modelling Language |
|---|---|---|
| Business process |   | Archimate |
| Application service |   | Archimate |
| Application interface |   | Archimate |
| Application Component |   | Archimate |

| | | |
|---|---|---|
| Data object |   | Archimate |
| Mandatory-Feature |   | Feature model |
| Optional-Feature |   | Feature model |
| Alternative-Feature |   | Feature model |

We used the specialization function in the Archi tool to adapt the metamodel, incorporating feature concepts via a specialized Plateau element—mirroring the idea of a feature as a stable architectural configuration—and a specialized Group element to represent building blocks as modular units of interrelated processes, data, and services. These specialized Plateau and Group concepts also offer the flexibility to establish diverse relationships with other ArchiMate elements such as Business Process, Application Component, and Application Service, ensuring seamless integration of feature models into the broader architecture. Table 2 presents the custom elements introduced in the extended ArchiMate metamodel, highlighting the additions made to support feature modeling and building block representation.

**Table 2**

Custom Elements Introduced in the Extended ArchiMate Metamodel

| Element name | Element picture |
|---|---|
| Mandatory-Feature |   |
| Optional-Feature |   |
| Alternative-Feature |   |
| Building-Block |   |

By adding feature-modeling elements to ArchiMate, we extended the metamodel to represent variability—configuration options, optional and mandatory components, and alternative structures—directly within enterprise architecture models. This bridges traditional ArchiMate modeling with feature-based variability management. Our customized metamodel now combines standard ArchiMate elements with new feature constructs, as shown in Figure 2

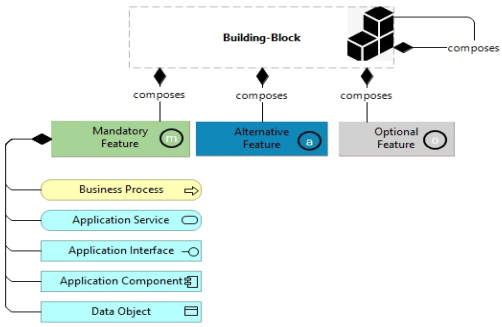



**Figure 2:** Metamodel Extension Combining ArchiMate and Feature Modeling

The extended metamodel shows how Building Blocks hold and express variability. Each block can stand alone or break down into sub-blocks for reuse, and it bundles features that drive configuration. Features come in three types—mandatory (always included), optional (chosen as needed), and alternative (pick one)—so you can model flexible architectures precisely. Every Business, Application, and Data layer element must link to at least one feature, grounding each component in its configuration. And because elements can join multiple features, shared services or data structures naturally span different setups or product variants.

## 5. Leveraging the Extended Metamodel to Modularize Existing Building Blocks

We applied the extended ArchiMate metamodel—enriched with feature-modeling elements and their interrelations—to the building blocks from our case study. In earlier work [23], we evaluated this on the "Geräteeinbau" (Device Installation) process, which details the steps for installing or removing electricity and water meters.

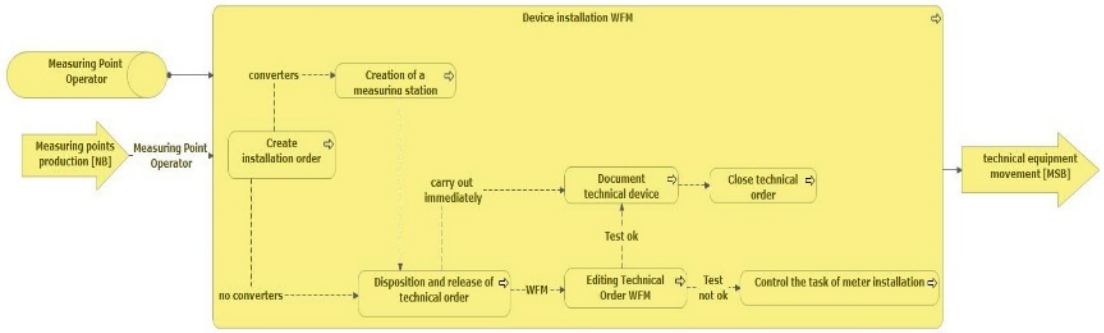



**Figure 3:** Process of Device installation WFM

The workflow has several variation points—for example, after creating an installation order, it checks for a converter and may branch into the "Editing Technical Order WFM [MSB/NB]" sub-process. We added ArchiMate Data Objects to fill the data-layer gap and grouped elements under Device Management (Device Type and Class). A PowerPoint feature model maps variability with six activity-set building blocks (mostly mandatory), marking "Document Technical Device Installation" and "Control Meter Installation" as alternatives, while shared data dependencies tie layers together.

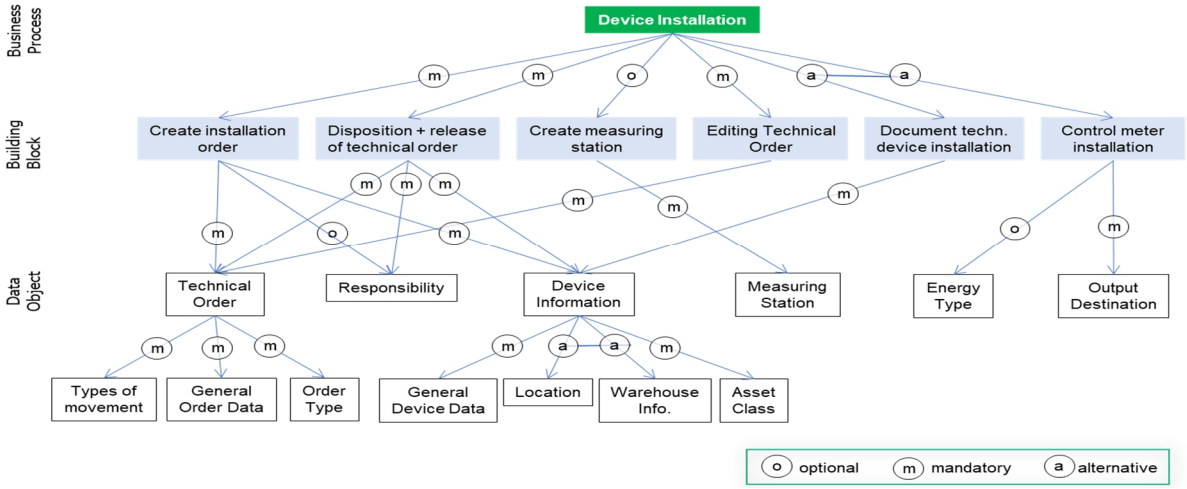



**Figure 4:** Feature Model showing building blocks identified for the device installation process

The feature model let us pinpoint process variants and candidate building blocks. We first built it in PowerPoint, defined the blocks, then re-modeled them in Archi with ArchiMate. For example, the Create Installation Order block links the Business Activity "Create Installation Order" to the Application Service "Installation Technical Order," provided by the Technical Order component and accessed via the TECHAUFT_AUFTRAG.FMX interface for editing its Data Objects.

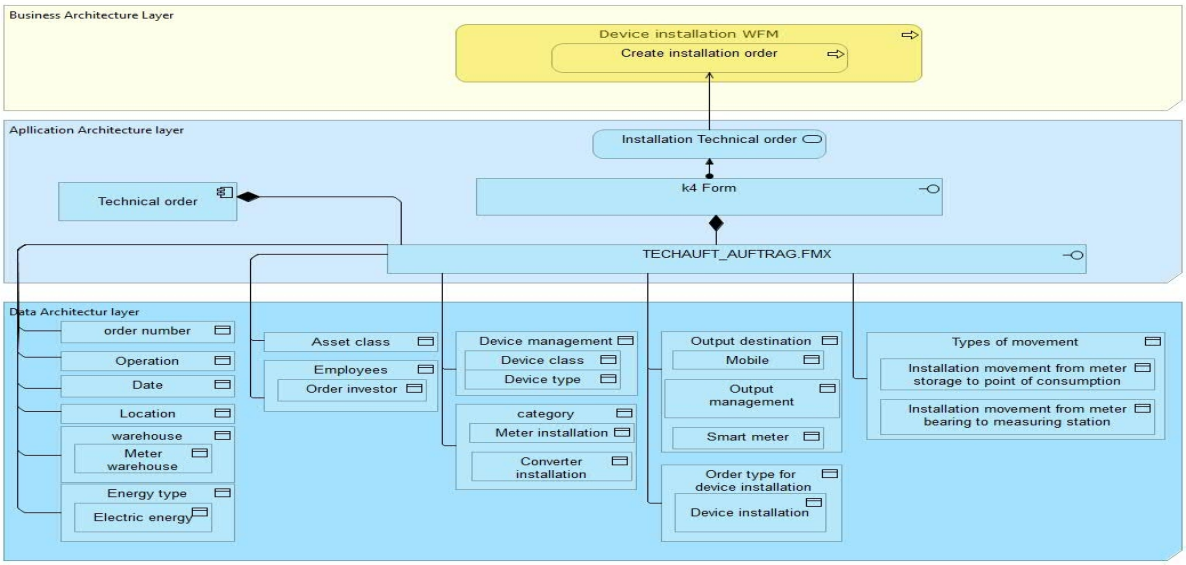



**Figure 5:** Building Block (Create installation order)

We demonstrate how the updated metamodel captures variability using the Create Installation Order block as an example. By mapping its features to the business activity, application service and interface, and data objects, we showcase the extended model's flexibility. The next section details this block's representation, highlighting its features and variability structures.

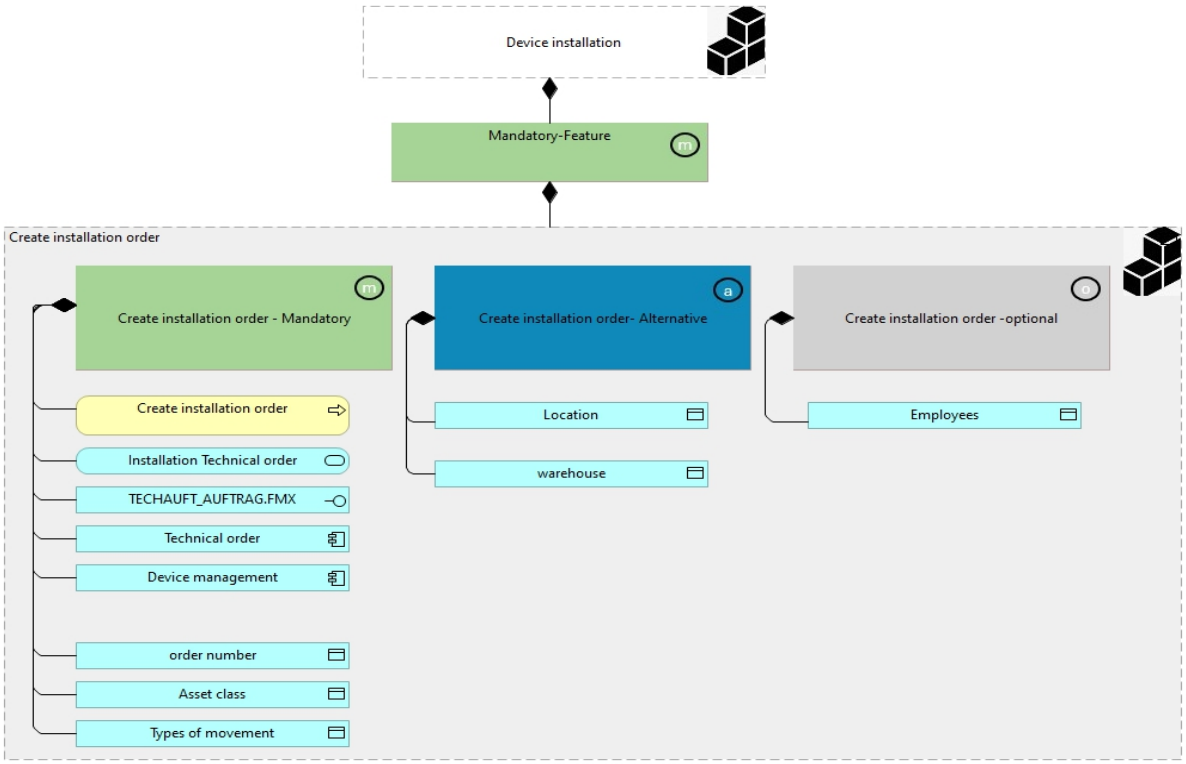



**Figure 6:** Integration of Feature Variability and ArchiMate Elements in the "Create Installation Order" Building Block

The Create Installation Order Building block spans three layers: at the Building Block Layer, it's defined as a mandatory component of the higher-level Device Installation block; at the Feature Layer, it captures three variants—Mandatory, Alternative, and Optional; and at the ArchiMate Layer, each variant maps to one or more ArchiMate elements—for example, the Mandatory variant includes the following elements:

- From the Business Layer: the business activity Create Installation Order,
- From the Application Layer: the application service Installation Technical Order, the application interface TECHAUFT_AUFTRAG.FMX, and the application component Device Management.
- From the Data Layer: the data objects Order Number, Asset Class, and Types of Movement.

This layered structure allows for a modular and configurable representation of architectural components, with each feature capturing a distinct configuration and linking to concrete business, application, and data elements.

Additionally, we identified the Creation of a Measuring Station Building block as an optional feature within the Device Installation building block. Its Feature Layer defines "Creation of a Measuring Station [Mandatory]," and in the ArchiMate Layer this feature links to a predefined set of elements according to the extended metamodel, structured as follows:

- Business Layer: the business activity Creation of a Measuring Station,
- Application Layer: the application service Creation of a Measuring Station and the application interface MP_EDIT.FMX,
- Data Layer: the data object Measuring Station.

This structured representation reflects the modular design of the architecture and demonstrates how variability is integrated and traced across all layers using the extended metamodel.

## 6. Summary and Future Work

In this work, we show that adding feature-modeling constructs—features, variation points, and binding mechanisms—to the ArchiMate metamodel creates a unified language for expressing software-product variability in enterprise-architecture artifacts. Embedding these notions into the Business, Application, and Technology layers yields a coherent framework that supports both high-level strategic planning and detailed product-line engineering, preserving ArchiMate's rigor while boosting its expressiveness for variant-rich designs.

Our industrial case studies demonstrate that the extended metamodel captures a wide range of product variants, letting architects define feature hierarchies alongside core services, application components, and infrastructure. Traceability links between features and architectural elements streamline impact analysis: when regulations change, customer needs shift, or platforms evolve, teams can quickly identify affected variants, gauge the scope of updates, and propagate changes consistently speeding decision cycles, reducing inconsistencies, and increasing confidence in large-scale deployments.

Beyond traceability, this richer modeling vocabulary fosters modular reuse. Architects can isolate variant-specific behaviors, design targeted extensions, and systematically assemble solutions from a common core. That modularity accelerates new offerings and clarifies governance by making variability constraints explicit and machine-readable. Early tool prototypes even show how automated validation catches configuration errors upstream, avoiding costly rework.

Despite these advances, broader adoption faces hurdles: many enterprise architects aren't yet familiar with feature-modeling concepts, so tailored training materials, examples, and workshops are needed. Tool support must be scaled to handle hundreds of variation points without performance hits, and research must integrate variability management into established governance lifecycles to keep variant definitions aligned with evolving business strategies.

Looking forward, we'll apply our method to additional industrial cases—spanning finance, manufacturing, and telecommunications—to refine metamodel extensions for optimal variability management. By iteratively modeling variation points in diverse contexts, we'll calibrate guidelines, tooling, and validation mechanisms, balancing expressiveness, performance, and usability. Ultimately, this continuous-improvement cycle will deliver a robust, widely applicable framework that empowers organizations to tackle complex product-line challenges with confidence.

## Declaration on Generative AI

During the preparation of this work, the author(s) used ChatGPT-5 in order to: Grammar and spelling check. After using these tool(s)/service(s), the author(s) reviewed and edited the content as needed and take(s) full responsibility for the publication's content.

## References


[1] Kaidalova, J., Sandkuhl, K., Seigerroth, U.: How digital transformation affects enterprise architecture management-a case study. International Journal of Information Systems and Project Management, vol. 6, 5–18 (2018)

[2] Zimmermann, A., Schmidt, R., Sandkuhl, K., Jugel, D., Bogner, J., Möhring, M.: Evolution of enterprise architecture for digital transformation. In: 2018 IEEE 22nd International Enterprise Distributed Object Computing Workshop (EDOCW), pp. 87–96 (2018)

[3] Sandkuhl, K., Seigerroth, U.: Digital Transformation of Enterprises: Case Studies and Transformation Paths. In: PACIS, p. 35 (2021)
[4] Rittelmeyer, J.D., Sandkuhl, K.: Effects of artificial intelligence on enterprise architectures-a structured literature review. In: 2021 IEEE 25th International Enterprise Distributed Object Computing Workshop (EDOCW), pp. 130–137 (2021)
[5] Park, S., yoon Lee, J., Lee, J.: AI system architecture design methodology based on IMO (Input-AI Model-Output) structure for successful AI adoption in organizations. Data & Knowledge Engineering, vol. 150, 102264 (2024)
[6] Fruhwirth, M., Ropposch, C., Pammer-Schindler, V.: Supporting Data-Driven Business Model Innovations: A structured literature review on tools and methods. Journal of Business Models, vol. 8, 7–25 (2020)
[7] Dehne, A., Sandkuhl, K.: Variability modelling in enterprise architecture management-survey on existing approaches. In: 22nd International Conference on Perspectives in Business Informatics Research Workshops and Doctoral Consortium, BIR-WS 2023 Ascoli Piceno 13 September 2023 through 15 September 2023, vol. 3514, pp. 94–107 (2023)
[8] Hevner, A., Chatterjee, S.: Design research in information systems: theory and practice. Springer Science & Business Media (2010)
[9] Winter, R.: Architectural thinking. Wirtschaftsinformatik, vol. 56, 395–398 (2014)
[10] Simon, D., Fischbach, K., Schoder, D.: Enterprise architecture management and its role in corporate strategic management. Information Systems and e-Business Management, vol. 12, 5–42 (2014)
[11] Ahlemann, F., Stettiner, E., Messerschmidt, M., Legner, C.: Strategic enterprise architecture management: challenges, best practices, and future developments. Springer Science & Business Media (2012)
[12] The Open Group: The TOGAF® Standard, 10th Edition. Architecture Development Method. Van Haren Publishing (2022)
[13] Kitchenham, B., Brereton, O.P., Budgen, D., Turner, M., Bailey, J., Linkman, S.: Systematic literature reviews in software engineering-a systematic literature review. Information and software technology, vol. 51, 7–15 (2009)
[14] Rurua, N., Eshuis, R., Razavian, M.: Representing Variability in Enterprise Architecture. Bus Inf Syst Eng, vol. 61, 215–227 (2019). doi: 10.1007/s12599-017-0511-3
[15] Mani, N., Helfert, M., Pahl, C.: A Domain-specific Rule Generation Using Model-Driven Architecture in Controlled Variability Model. Procedia Computer Science, vol. 112, 2354–2362 (2017). doi: 10.1016/j.procs.2017.08.206
[16] Asadi, M., Mohabbati, B., Kaviani, N., Gašević, D., Bošković, M., Hatala, M.: Model-driven development of families of Service-Oriented Architectures. In: Apel, S., Cook, W.R., Czarnecki, K., Kastner, C., Loughran, N., Nierstrasz, O. (eds.) Proceedings of the First International Workshop on Feature-Oriented Software Development, pp. 95–102. ACM, New York, NY, USA (2009). doi: 10.1145/1629716.1629735
[17] Benavides, D., Galindo, J.A.: Variability management in an unaware software product line company. In: Collet, P., Wąsowski, A., Weyer, T. (eds.) Proceedings of the Eighth International Workshop on Variability Modelling of Software-Intensive Systems, pp. 1–6. ACM, New York, NY, USA (2014). doi: 10.1145/2556624.2556633
[18] Wille, D., Wehling, K., Seidl, C., Pluchator, M., Schaefer, I.: Variability Mining of Technical Architectures. In: Cohen, M., Acher, M., Fuentes, L., Schall, D., Bosch, J., Capilla, R., Bagheri, E., Xiong, Y., Troya, J., Ruiz-Cortés, A. et al. (eds.) Proceedings of the 21st International Systems and Software Product Line Conference - Volume A, pp. 39–48. ACM, New York, NY, USA (2017). doi: 10.1145/3106195.3106202
[19] Langermeier, M., Rosina, P., Oberkampf, H., Driessen, T., Bauer, B.: Management of Variability in Modular Ontology Development. In: Hutchison, D., Kanade, T., Kittler, J.,

Kleinberg, J.M., Kobsa, A., Mattern, F., Mitchell, J.C., Naor, M., Nierstrasz, O., Pandu Rangan, C. et al. (eds.) Service-Oriented Computing – ICSOC 2013 Workshops. Lecture Notes in Computer Science, vol. 8377, pp. 225–239. Springer International Publishing, Cham (2014). doi: 10.1007/978-3-319-06859-6_20

[20] Nerome, T., Numao, M.: A Product Domain Model Based Software Product Line Engineering for Web Application. In: 2014 Second International Symposium on Computing and Networking, pp. 572–576. IEEE (2014). doi: 10.1109/CANDAR.2014.105

[21] Allian, A.P., Sena, B., Nakagawa, E.Y.: Evaluating variability at the software architecture level. In: Hung, C.-C., Papadopoulos, G.A. (eds.) Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing, pp. 2354–2361. ACM, New York, NY, USA (2019). doi: 10.1145/3297280.3297511

[22] Adjoyan, S., Seriai, A.: An Architecture Description Language for Dynamic Service-Oriented Product Lines. In: Proceedings of the 27th International Conference on Software Engineering and Knowledge Engineering. International Conferences on Software Engineering and Knowledge Engineering, pp. 231–236. KSI Research Inc. and Knowledge Systems Institute Graduate School (2015). doi: 10.18293/SEKE2015-217

[23] Dehne, A., Sandkuhl, K.: Towards Method Support for Variability Modelling in Enterprise Architecture Management. In: Řepa, V., Matulevičius, R., Laurenzi, E. (eds.) Perspectives in Business Informatics Research. Lecture Notes in Business Information Processing, vol. 529, pp. 119–134. Springer Nature Switzerland, Cham (2024). doi: 10.1007/978-3-031-71333-0_8

[24] Archi ArchiMate Modelling: Archi. Archimate modelling tool (2010)