Complex Systems Informatics and Modeling Quarterly (CSIMQ)
eISSN: 2255-9922
Published online by RTU Press, https://csimq-journals.rtu.lv
Article 213, Issue 39, June/July 2024, Pages 26–47
https://doi.org/10.7250/csimq.2024-39.02

Variability Modeling in Enterprise Architecture Management: Case Study and Survey on Existing Approaches

Ahmed Dehne[1*], Sahib Niyazov[1], and Kurt Sandkuhl[1,2]

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

ahmed.dehne@uni-rostock.de, sahib.niyazov@uni-rostock.de,
kurt.sandkuhl@uni-rostock.de

Abstract. Managing and dealing with variability in business processes and the IT landscape is a common challenge in the everyday practice of most enterprises and organizations. Recent studies have observed that digital transformation, Internet-of-Things solutions and the introduction of artificial intelligence cause changes and challenges in enterprises that simultaneously require variability on several levels, for instance, business processes, data architecture, and services. Enterprise architecture models are considered a suitable way to visualize and manage dependencies between different levels of an enterprise. However, the management of variability in enterprise architectures has not received much attention in scientific research. This article[†] aims to contribute to a better understanding of future investigation needs. Using a systematic literature analysis, the article structures the existing research work in the field and examines real-world challenges of variability based on a case study. We argue that there is a need for new constructs in enterprise architecture models that allow for expressing dependencies between variations on different enterprise architecture layers.

Keywords: Variability, Enterprise Architecture, Enterprise Architecture Management, Building Block.

1 Introduction

Managing and dealing with variability is a common challenge in the daily practice of many enterprises and organizations. Examples are business processes or workflows that exist in several variants, IT services delivered in different configurations depending on the target platform,

[*] Corresponding author

Reference: A. Dehne, S. Niyazov, and K. Sandkuhl, “Variability Modeling in Enterprise Architecture Management: Case Study and Survey on Existing Approaches,” Complex Systems Informatics and Modeling Quarterly, CSIMQ, no. 39, pp. 26–47, 2024. Available: https://doi.org/10.7250/csimq.2024-39.02

Additional information. Author’s ORCID iD: K. Sandkuhl – https://orcid.org/0000-0002-7431-8412. PII S225599222400213X. Article received: 22 December 2023. Accepted: 19 June 2024. Available online: 31 July 2024.

[†] This article is an extended version of work presented at the 13th Workshop on Business and IT Alignment (BITA): A. Dehne and K. Sandkuhl, “Variability modelling in enterprise architecture management – survey on existing approaches,” *Joint Proceedings of the BIR 2023 Workshops and Doctoral Consortium, co-located with 22nd International Conference on Perspectives in Business Informatics Research (BIR 2023)*, vol. 3514, pp. 94–107, 2023. Available: https://ceur-ws.org/Vol-3514/paper77.pdf

products with variations regarding their modules and configuration, business services varying in their customer frontend, or business offers depending on their delivery context. Different strategies can be observed to deal with variation with the extreme positions of allowing for no variability at all (rigid standardization) and full flexibility (no limitation for variability). The strategy of enterprises often aims for a combination of both, such as controlling the number of variants as a compromise between limiting complexity and allowing for flexibility.

Recent studies have observed that digital transformation, Internet-of-Things solutions and the introduction of artificial intelligence lead to changes in enterprises that require dealing with variability on several levels of an enterprise at the same time (see, e.g., [1], [2]), for instance, several variants of business processes that cause variations in the underlying data architecture, or variants of smart connected products that create the need for variations in IT services supporting them. Enterprise architecture (EA) models are considered a suitable way to visualize and manage dependencies between different levels of an enterprise, but there is not much work on managing variability in enterprise architectures across different architecture layers.

Data engineering is gaining importance in many organizations due to the increasing number of data-driven products and services and the use of AI solutions. From an EA perspective, data engineering should be based on and tightly coupled to an organization's data architecture to avoid incompatibilities or avoidable structural differences. However, business process management as part of the business architecture usually has a different focus on short execution times, efficient resource use or high process quality. This is not necessarily compatible with the aims of data engineering to provide data prepared for use in data-driven applications or AI. Our conjecture is that building blocks integrating business and data architecture or allowing for data-aware business process building blocks could help to ensure a high level of flexibility and, at the same time, control complexity.

In this context, we argue that digital transformation and the introduction of organizational AI solutions motivate new constructs in enterprise architecture models that allow for expressing dependencies between variations on different enterprise architecture layers. As a starting point for research in this field, the state of research regarding variability in enterprise architecture models and enterprise architecture management (EAM) has to be investigated. Furthermore, observations from real-world variability challenges in enterprises are expected to result in a better understanding of the problems. This article aims to structure the existing research work and contribute to a better understanding of future research needs. Furthermore, we present an initial approach for identifying cross-layer building blocks in EA models.

The article is structured as follows: Section 2 summarizes our research approach. Section 3 briefly defines the background for our work from variability management and enterprise architecture management. Section 4 contains details about the structured literature review. Section 5 presents the initial approach for developing cross-layer building blocks in EA models in an industrial case study. Section 6 gives an outlook on future work.

2 Methodology

The main objective of this research is to contribute to a better understanding of variability management in enterprise architectures. The long-term aim is to develop methodical and technological support for managing variability based on enterprise architecture building blocks. The project follows the paradigm of design science research (DSR). DSR is a research paradigm aiming at problem-solving in organizational settings, focusing on developing valid and reliable knowledge for designing the required solutions [3]. 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. This article concerns the first phases of a DSR project: problem investigation, requirements definition, and initial design of the envisioned artifact. Among the basic principles of DSR is the need to consider the relevance of the research that is expressed in business needs and rigorous evaluation of the research results against the existing scientific knowledge base. Thus, we conduct a literature analysis to structure the body of existing knowledge in the field. We also analyze an industrial case study to demonstrate business relevance and extract initial requirements for the envisioned method and tool support. Furthermore, we use the case study for a feasibility study on the possibility of designing variability handling suitable for enterprise architecture building blocks. This feasibility study is based on conceptual-deductive work.

Structured literature review
One part of the research work is to analyze the current state of research in the area of EAM and variability, focusing on approaches and constructs for variability management across different architecture layers. To gather the necessary information about the body of knowledge, we used the structured literature review (SLR) method according to Kitchenham [4]. Kitchenham's approach was developed to support the evaluation of what has been published on a specific research topic, to compare existing research, and to analyze potential research gaps. Based on [4], our research approach followed six steps. The results of these six steps are presented in Section 4 in detail.

First, we formulated the overall research question for the SLR (step 1): RQ-SLR: "What is the state of research on managing variability in enterprise architectures?". Step 2, the process of paper identification, starts with defining the overall search space, which basically consists of determining the literature sources to consider in light of the research question. Paper identification continues with the population phase (step 3). In this step, the search string is developed and applied by searching the literature sources. Afterwards, the step "paper selection" follows by defining inclusion and exclusion criteria and a manual selection of relevant papers found in the population phase (step 4). The data collection phase (step 5) focuses on extracting the information relevant to answering the research question from the set of identified relevant papers. The last step is the analysis of data and interpretation, i.e., to answer the research question defined in step 1 by using collected data from relevant papers. We finalize the article with a research summary and explain possible future work (cf. Section 6).

Case Study
Yin differentiates various kinds of case studies [5]: explanatory, exploratory, and descriptive. The case study presented in Section 4 has to be considered descriptive, as it describes the phenomenon of enterprise architecture variability and the real-life context in which it occurs. Based on the case study results, we conclude that there is a need to develop methodical and technological support for variability management that applies the concept of building blocks. The research question for the case study is RQ-CS: "How to capture EA variability in industrial practice?".

Conceptual-deductive Research
The basic principle of conceptual-deductive research is applying theoretical concepts to empirical data in order to gain new insights in the research field. In this article, we apply concepts from variability modeling to the structures of enterprise architecture models to identify reusable parts and their dependencies. The motivation is to investigate the feasibility of creating EA building blocks. The result can be considered as input to the first design and evaluation cycle for the envisioned DSR artifact, the methodical and technological support for managing variability.

3 Theoretical Background

This section describes the necessary theoretical information for this research approach, focusing on variability modeling and enterprise architecture management.

3.1 Enterprise Architecture Management

The focus of EAM is to holistically harmonize business and IT as well as the components and processes of enterprises. By reducing redundancies and complexity and using synergies, the company and its performance shall be improved in alignment with its goals, to name just a few examples of why companies utilize EAM. In general, EAM is a “management practice that creates, maintains and uses a coherent set of guidelines, architectural principles, and governance systems, which provides practical help and guidance for the design and development of an EA to achieve its visions and strategies” [6]. Therefore, EAM depicts a management philosophy, organizational function, and a systematic approach with tools and practices to develop the EA.

TOGAF [7] is considered by many researchers to be the industry standard and defines three different architectural layers, which are visible in many other frameworks. Business architecture defines business strategy, governance, organization, and key business processes. Information Architecture often is divided into two sub-layers: Data Architecture and Application Architecture. The Data Architecture describes the structure of an organization’s logical and physical data assets and data management resources. The Application Architecture provides a blueprint for the individual application systems to be deployed, for their interactions and relationships to an organization’s core business processes. Technology Architecture describes the physical realization of an architectural solution. In addition to EAM frameworks, there are also different modeling languages to support different EAM activities. One such language is ArchiMate[‡], which is widely used for these purposes.

3.2 Variability Modeling

Variability modeling originated in software product line research and generative programming [8]. Complex software systems offer a rich set of functions and features to their users, but cause challenges to their developers: how to provide high flexibility with many possible variants for different application contexts and simultaneously restrict the systems’ complexity to achieve maintainability. Variability modeling offers a contribution to control the variety of the variants of systems by capturing and visualizing commonalities and dependencies between features and between the components providing feature implementations. For more than 20 years, variability modeling has been frequently used in the area of complex software systems and technical systems. Among the variability modeling approaches, feature models are considered particularly relevant for analyzing variability in EA models.

A feature is a “distinctive and user-visible aspect, quality, or characteristic of a software system or systems” [9]. The purpose of a feature model is to capture, structure, and visualize the commonality and variability of a domain or a set of products. Commonalities are the properties of products shared among all the products in a set, placing the products in the same category or family. Variability is the element of the products that differentiates and shows the configuration options, choices, and variation points that are possible between variants of the product aimed at satisfying customer needs and requirements. The variability and commonality are modeled as features and organized into a hierarchy of features and sub-features, sometimes called feature trees, in the feature model. The hierarchy and other properties of the feature model are visualized in a feature diagram. Feature diagrams express the relations between features with the relation types mandatory, optional, alternative, required, and mutually exclusive.

Different methodical approaches in the field and the exact syntax of feature diagrams were analyzed and compared in [10], and an overview of methods for feature model development is provided in [11]. Both papers focus on notations and approaches specialized for certain application fields, i.e., no generally accepted feature model development method exists. In Section 5, when applying feature modeling in the context of EAM, we use the metamodel of Kang et al. for FODA [9] but with the adaptation of the visual notation proposed in [12]. The visual notation was adopted because of the experience in previous projects where the practitioners clearly preferred the adapted notation as it is easier to remember the type of relationship (mandatory, optional, or alternative). Figure 1 compares the initial FODA notation with the adapted one.

[‡] https://www.opengroup.org/archimate-forum/archimate-overview

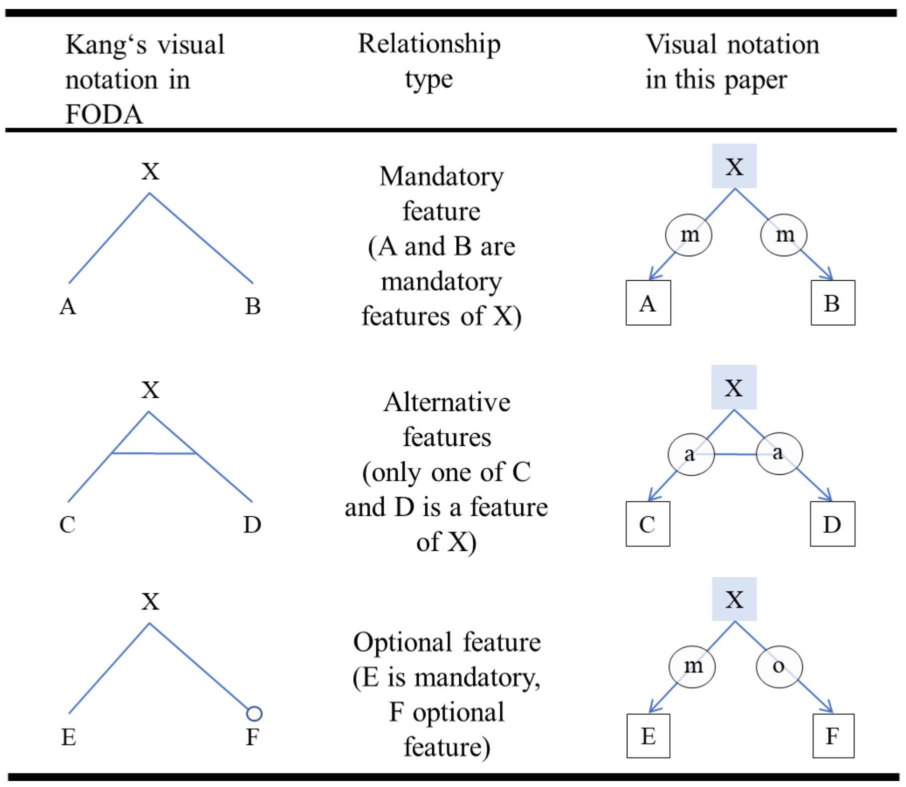

Kang's visual notation in FODA
Relationship type
Visual notation in this paper
X
A
B
Mandatory feature (A and B are mandatory features of X)
X
m
m
A
B
X
C
D
Alternative features (only one of C and D is a feature of X)
X
a
a
C
D
X
E
F
Optional feature (E is mandatory, F optional feature)
X
m
o
E
F

Figure 1. Feature model relationship types and notation

4 Structured Literature Review

Following the SLR approach by Kitchenham (see Section 2), we first developed several search strings (cf. Section 4.1) and selected the papers according to inclusion and exclusion criteria (cf. Section 4.2). Afterwards, we collected the data and summarized the results in Section 4.3. Section 4.4 adds the interpretation and discussion of the results.

4.1 Search String Development

After defining the research question presented in Section 2, we started the literature search with an initial population and identified "variability" and "enterprise architecture management" as the main keywords. We then gathered synonyms and associated terms for these two keywords, as shown in Table 1.

Table 1. Selected search terms for the SLR

Variability	Enterprise Architecture
Variability	Business Architecture
Variation	Application Architecture
	Software Architecture
	Information Architecture
	Technology Architecture
	Data Architecture

The synonyms for variability were chosen from previous experience in the field. The sub-categories of Enterprise Architecture were included as synonyms for EA to cover different categories. We also used the keywords to develop several search strings, beginning with the first two initial keywords and then adding synonyms to compare how the changes affect the results. We used the database 'Scopus' for the search string development and the actual search. Scopus was selected as it includes various data sources, such as most of the AISeL, Springer, IEEE, and Elsevier publications.

The first two search strings in Table 2 were the ones that identified the highest number of relevant papers at once. We had to modify the third search string by adding the keyword "enterprise architecture management" to it because, without this modification, we discovered many very general papers. After adding this keyword to the third search string, we were able to identify two relevant papers. Furthermore, we determined one relevant paper for each fourth, fifth, and sixth search string. We also tested different search strings but share only the most relevant ones in Table 2. For each string, we scanned only the titles first. We then checked the abstract, followed by an introduction and summary. Finally, we read all selected papers and applied the selection criteria explained in Table 3.

Table 2. Final search strings of the SLR and results

No.	Search String	Number of Results	Identified Papers	EA layer addressed in the papers
1	TITLE-ABS-KEY(variability) AND TITLE-ABS-KEY("enterprise architecture")	25	[13]–[16]	Business Architecture, Software Architecture, Technology Architecture, Application Architecture
2	TITLE-ABS-KEY(variability) AND TITLE-ABS-KEY("application architecture")	14	[17]–[19]	Technology Architecture, Application Architecture
3	TITLE-ABS-KEY(variability) AND TITLE-ABS-KEY("Software architecture") AND TITLE-ABS-KEY(enterprise AND architecture AND management)	6	[20], [21], [25]	Business Architecture, Software Architecture
4	TITLE-ABS-KEY(variability) AND TITLE-ABS-KEY("business architecture")	5	[22]	Business Architecture
5	TITLE-ABS-KEY(variability) AND TITLE-ABS-KEY("technology architecture")	8	[23]	Technology Architecture
6	TITLE-ABS-KEY(variability) AND TITLE-ABS-KEY("information architecture")	11	[24]	Information Architecture

The table also displays the outcomes of the utilized search phrases and the respective enterprise architecture layer to which each phrase pertains.

4.2 Paper Selection and Inclusion and Exclusion Criteria

This section explains the paper selection process as well as the inclusion and exclusion criteria used during and after the search term development. To structure the selection process and guarantee comparability between the papers, we declared inclusion criteria describing the information that a paper should contain to be relevant. Our criteria for a paper to be selected were as follows: The paper must cover at least the two subjects' variability and EAM; after that, we investigated whether the paper includes variability in all EAM layers or in specific EAM layers e.g., EA1 from EA1.1 to EA1.5. Following that, we investigated if a specific methodology (see criteria EA2) was used to measure the effect of the method when it was found. We grouped the criteria as shown in Table 3.

Throughout the search process, we determined much research about capturing the variability in different levels of systems and enterprises. However, the objective of this research is to discover papers that can deliver methods of managing variability in EAM. Thus, we excluded papers addressing variability outside the EAM context. For most of the variability-related papers, the target group was the tool users.

Table 3. Inclusion Criteria for the SLR

Variability and EAM	EAM Layer
Var. and EAM1: Does the paper address the topic of variability in the context of EAM?	EA1: Which layer of the organization deals with variability? – EA1.1. Business Architecture – EA1.2. Application Architecture – EA1.3. Software Architecture – EA1.4. Technology Architecture – EA1.5. Information Architecture
	EA2. Is a specific method used to measure the effects?

As explained in the previous section, the inclusion and exclusion criteria were applied to the full text of selected papers after reading their title, abstract, introduction, and summary. Table 4 shows the abstracted form of the Excel sheet used to record the results. It summarizes the 12 papers that fulfilled the criteria and requirements for the research question. The results and the content of the identified papers are described in the next section.

Table 4. Overview of inclusion criteria applied to the most relevant papers

PaperCriteria	**[13]**	**[14]**	**[15]**	**[16]**	**[17]**	**[19]**	**[20]**	**[21]**	**[22]**	**[23]**	**[24]**	**[25]**
Var. and EAM1	X	X	X	X	X	X	X	X	X	X	X	X
EA1.1				X			X	X	X			
EA1.2	X				X							
EA1.3			X									X
EA1.4		X				X				X		
EA1.5											X	
EA2												

4.3 Data Collection and Analysis – Presenting the Search Results

To structure the data collection process, we used another Excel sheet to extract essential information from each paper, as recommended by [25]. That information included the title, authors, year of publication, research question, used methods, sample description and size, results/insights, gaps/problems/critique, and a personal interpretation/ classification. We summarize the identified relevant papers in this section.

Table 5 summarizes the outcomes of the utilized search phrases and the respective layer architecture of each phrase. The table also shows the limitations of the used approaches regarding the architecture layers. All 11 selected papers are briefly described below.

N. *Rurua, R. Eshuis, M. Razavian – Representing variability in enterprise architecture: A case study [16]*

This paper discusses the variability in the designed business processes and their supporting application and infrastructure technology. The research used a design science approach to develop and capture the variability on the mentioned layer of architecture. Solution design is an extended enterprise architecture metamodel based on ArchiMate. For the extension, some new elements were added to cover concepts such as variation driver, constraint, dependency, and process variant. Variation points and their resolution options, as well as dependency options, are embedded. The solution analyzed the business processes of electronic invoicing for countries in Latin America as well as the existing enterprise architectures. Three experts were interviewed to evaluate the proposed solution's feasibility and applicability, as other experts were either heavily involved in the design process or unavailable. Overall, conducting a single case study from a specific business domain can make it difficult to validate the solution's generalizability. To mitigate this, the authors relied on established theories and techniques from the existing knowledge base on variability management in software engineering and studied multiple units of analysis across various Latin American countries. The case focused on financial processes where variability was caused by diverse fiscal regulations. The implications for similar European Union cases are uncertain, and nevertheless, the suggested approach did not expose whether it is possible to apply the same method on any other layer of architecture.

Table 5. Summary of the outcomes of the utilized search phrases and the respective architecture layer

Search String Nr.	**Paper Title**	**Architecture Layer**
1	N. Rurua, R. Eshuis, M. Razavian – Representing variability in enterprise architecture: A case study [16]	Business Architecture
1	A. P. Allian, B. Sena, E. Y. Nakagawa – Evaluating variability at the software architecture level: An overview [15]	Software Architecture
1	D. Wille, K. Wehling, C. Seidl, M. Pluchator, I. Schaefer – Variability mining of technical architectures [14]	Technology Architecture
1	M. Langermeier, P. Rosina, H. Oberkampf, T. Driessen, B. Bauer – Management of variability in modular ontology development [13]	Application Architecture
2	K. Wehling, D. Wille, C. Seidl, I. Schaefer – Decision support for reducing unnecessary IT complexity of application architectures [19]	Technology Architecture
2	T. Nerome, M. Numao – A product domain model based software product line engineering for web application [17]	Application Architecture
3	N. Mani, M. Helfert, C. Pahl – A domain-specific rule generation using model-driven architecture in controlled variability model [21]	Business Architecture
3	M. Asadi, B. Mohabbati, N. Kaviani, D. Gašević, M. Bošković, M. Hatala – Model-driven development of families of service-oriented architectures [20]	Business Architecture
4	D. Benavides, J. A. Galindo – Variability management in an unaware software product line company. An experience report [22]	Business Architecture
5	K. Wehling, D. Wille, C. Seidl, I. Schaefer – Automated recommendations for reducing unnecessary variability of technology architectures [23]	Technology Architecture
6	S. Adjoyan, A. Seriai – An architecture description language for dynamic service-oriented product lines [24]	Business Architecture
3	J.-M. Horcas, M. Pinto, L. Fuentes – Product Line line aArchitecture for Automatic automatic Evolution evolution of Multimulti-Tenant tenant Applications applications [25]	Application Architecture

A. P. Allian, B. Sena, E. Y. Nakagawa – Evaluating variability at the software architecture level: An overview [15]

The assessment of software architectures that incorporate variability data is an important undertaking. To date, a range of tools such as Product Line Architectures (PLAs), software architectures, reference architectures, and enterprise architectures have been deployed for assessment. Thus, this paper offers practitioners an overview and understanding of the most pertinent techniques and approaches developed for this type of evaluation. A Systematic Mapping Study (SMS) analyzed 1457 studies on evaluation techniques and methods. Following an initial exclusion process, 30 studies were identified to afford practitioners a comprehensive overview and insight into the most pertinent approaches. The SMS indicated the following findings:

- Maturity level is often overlooked in architectural assessment.
- Few studies focus on evaluating a maturity level in the context of software architectures.
- Only one architecture layer is typically evaluated for variability.

- Combining different evaluation techniques can result in more comprehensive results – software architecture is accepted based on subjective satisfaction with scenario and questionnaire evaluations.
- Constraints related to stakeholders must be prioritized during evaluation, especially when they conflict with evaluation criteria.
- Evaluation can hinder stakeholder learning and process improvements.

In this study, the research team employed various strategies to minimize potential threats to validity. These included defining the research question and search string through brainstorming sessions, reviewing protocols proposed by other researchers, and discussing extracted data among the team to avoid ambiguity. Although some studies lacked crucial details, the team sought out related studies with further information to mitigate the issue. Consequently, the study's validity was enhanced, and the findings were reliable. The SMS study discovered a lack of industrial participation and empirical evaluations in the field of software architecture variability.

While academic studies may provide examples to validate approaches, they do not offer concrete evidence of effectiveness. Furthermore, there is a shortage of fully supported tools to handle variability throughout the entire process. To address these concerns, the paper authors recommend increased empirical evaluations with industrial partners to emphasize the importance of handling variability in software architecture. This could lead to improved approaches that meet industrial requirements.

D. Wille, K. Wehling, C. Seidl, M. Pluchator, I. Schaefer – Variability mining of technical architectures [14]

The paper discusses the problem of the increasing number and variety of Technical Architectures (Tas) in companies that make it difficult to maintain and reuse solutions across systems due to missing information on the relations between existing variants of TAs, which results in technical debt and the risk of failure in productive systems. The authors propose an algorithm to analyze multiple TAs and identify common and varying parts based on the concept of variability mining. This information can help architects determine the potential for reuse and make maintenance decisions. The proposed method, using the algorithm (n-way), imports a set of TAs, analyzes them in parallel to identify common and varying parts, and clusters components based on structural relations to eliminate unrealistic variability. A rule-based system further improves results, and the identified variability is merged into a 150% model, providing architects with a unified view. The algorithm is efficient, automatic, and capable of analyzing an arbitrary number of TAs, with a customizable rule-based system improving accuracy. The resulting 150% model is useful for identifying commonalities and differences in TAs. Additionally, the case study was conducted with one industry partner and thus the results might not be generalizable to other contexts. Furthermore, the size of the sample used in the study is relatively small, and therefore, the results might not be representative of the entire population. Lastly, the study was conducted over a limited time span, and therefore, the findings might not be applicable in the long-term. In addition, the proposed method was only applied to the technology architecture and it was not discussed if the method could be applied to other enterprise architectures.

M. Langermeier, P. Rosina, H. Oberkampf, T. Driessen, B. Bauer – Management of variability in modular ontology development [13]

The paper discusses a method to organize a set of modular ontologies using the concepts of variability management (MOVO). In this method, a knowledge engineer (KE) selects modular ontologies to store in the ontology repository. Based on that an ontological variability model VM0 will be defined. The VM0 formalizes the dependencies between the annotated modular ontologies in the ontology. Variables are defined using features. Each feature can, but does not have to, be associated with one or more normative ontology. Based on VM0, KE creates a VM1 by selecting features, relationships, and extensions of stronger constraints depending on specific domain requirements. This form formalizes variables that have meaning and which are not associated with what is allowed and what is forbidden. After the creation of a consistent VM1 by the KE, the domain expert can easily create consistent configurations for his application ontology. The method has been used in the concept of variability management in software product line engineering and adapted to the domain of modular ontology management to be able to formalize possible combinations of the modules. The concept has been applied in connection with the variability in the application architecture level, but the paper did not discuss the ability to use the same method on the entire Enterprise architecture.

K. Wehling, D. Wille, C. Seidl, I. Schaefer – Decision support for reducing unnecessary IT complexity of application architectures [19]
The paper offers an approach to identify the variability in each application architecture by adapting a specific variability mining technique from the software product line (SPL). Driven from this an iterative decision process consisting of five phases can be offered to support experts in determining and reducing the unnecessary part of the application architecture's complexity by using the identified variability. The introduced approach was applied to a preliminary case study by considering data on application architectures obtained from one industry partner. An analysis of the usage of applications was performed in four different company sites with a focus on manufacturing customer-ordered products to see the impact of different variability levels on the applied approach. However, the applied method has been tested on the level of application architecture, it does not apply to other layers of architecture.

J.-M. Horcas, M. Pinto, L. Fuentes – Product line architecture for automatic evolution of multi-tenant applications [25]
The approach for multi-tenant applications is flexible, automated, and adaptable, treating tenants as cloneable features. Testing ensures correctness and efficiency. This paper proposes an approach using Product Line Architecture (PLA) to make changes and achieve valid software architecture for each tenant in cloud-based applications. The approach utilizes CVL (Common Variability Language) to model each tenant as a cloneable feature and employs three algorithms to automate the process of evolving the multi-tenant architecture. A running case study from the medical software solutions domain outlines the approach and demonstrates its effectiveness in dealing with a high number of tenants. The main goals of the study are to elicit changes in evolving cloud-based applications and obtain a valid software architecture.

The evolution algorithms can generate valid configuration models that satisfy the given variability model. The use of Choco allows the validation of the generated configuration models automatically and efficiently. Furthermore, the objective function helps to ensure that the generated configuration models are minimal, meaning that they contain the smallest number of resolved features necessary to satisfy the constraints. Overall, this approach provides a rigorous and reliable way to verify the correctness of the evolution algorithms. The authors of the paper recommend investigating the generalizability of the approach to other applications and cloud platforms. Additionally, it is important to evaluate the potential impact on the system's performance, security, and stability. The trade-offs between the cost and benefits, including non-financial costs, should be considered. To ensure the approach's use in real-world situations, the reliability and robustness must be assessed in various scenarios. Further research is needed to address these challenges and explore the approach's potential in practical settings. The applied approach was tested in the case study and proved to be successful in working in the defined environment. However, it did not apply to any other architecture layers.

T. Nerome, M. Numao – A product domain model based software product line engineering for web application [17]
The paper obtains an approach of using the software product line engineering (SPLE) for web application. While Software Product Line Engineering (SPLE) is widely used in industrial areas to develop embedded systems, it has not been widely utilized in the development of web applications involving business logic like banking and insurance applications. The issue of applying SPLE in financial areas has been identified, and SPLE activities can be broken down into domain engineering and application engineering. However, in the case of web application

development, only domain engineering exists to develop a range of products without application engineering. The proposed engineering approach applies software product line engineering (SPLE) for web applications. A domain engineering model called Product Domain Model is defined using UML-based metamodel with variability notation. An application architecture adopting dependency injection technology is defined for runtime, with the target being the logic of an instance product. A product generator is created to generate numerous resources based on the Product Domain Model.

The study showed that using web applications can streamline the development and maintenance of complex banking products. The modular and efficient development process helps to reduce costs while still delivering high-quality products that meet clients’ needs. Requirements analysis and traceability ensure product effectiveness and sustainability over time. The process will be improved to meet the banking industry's and clients' evolving needs. As mentioned, the study was applied in one industry field, which is banking, and covered the variation in the application layer. Nevertheless, the study did not deal with other architectural layers.

D. Benavides, J. A. Galindo – Variability management in an unaware software product line company. An experience report [22]
This paper presents an experience report on a software development company that had no knowledge of software product line approaches. The authors spent two months collecting information and observing variability management practices and identified areas for improvement. Despite being unfamiliar with product line engineering concepts, the company already had some variability management practices in place. The report briefly discusses how variability management is implemented in various parts of the company, such as business architecture and software assets management. The authors conclude that companies with potential for change could benefit from adopting software product line engineering practices. The case study company practices enterprise architecture management according to TOGAF standards. Variability in the business architecture is crucial due to changes in tax laws and procedures. Business rules are used to handle process variability, allowing changes without impacting the entire system. However, it is necessary to explicitly identify business rules, and business process families have not yet been identified. Variability management and product line engineering are vital in infrastructure architecture, as the case study company has approximately 100 applications that could be potentially considered for multiple product lines. TOGAF does not address the software product line engineering perspective at the product portfolio level. This case study focuses on the implementation of a configurable business process in an organization. A configurable business process is one that recognizes and explicitly represents variability, encompassing a range of similar processes within a specific domain. By configuring the process, the organization can define a concrete instance that meets its specific needs. The case study provides an example of two different tax refund processes to illustrate the concept of commonality and variability in processes. By incorporating a variation point, a configurable process is defined, enabling the organization to customize it according to its specific requirements.

This experience report details the implementation of variability management techniques in a company that was unaccustomed to software product line principles. The study discovered that variability management was prevalent across the organization and viewed as indispensable in many domains. The report proposes prospective areas for investigation and suggests conducting more structured experimental research in various enterprises to determine how to bolster their practices. Additionally, the report advocates improving the methodology description and leveraging lessons learned from previous research to enhance the process of adopting software product line principles. However, the experience report did not address how to implement the utilized variability approach in a different architectural layer of the company's software infrastructure.

N. Mani, M. Helfert, C. Pahl – A Domain-specific rule generation using model-driven architecture in controlled variability model [21]

The paper proposes an approach to the development of an automatic generation of the domain-specific rules by using variability feature model and ontology definition of domain model concepts coming from software product line engineering and Model Driven Architecture (MDA). Four-level MDA was used to prototype the application.

After the development of the MDA, the paper proposes using the Software product line engineering approach in the form of a feature model as standard variability model techniques to bridge between an assumed domain model and business process. Afterwards, a domain- specific language would be developed to describe the environment or configuration of the application in a specific domain.

Overall, the proposed approach can significantly reduce the time and effort required to generate domain-specific rules and adapt to changing business requirements. It enables non-technical domain experts to actively participate in the customization and configuration of their business processes. Additionally, it provides a systematic and structured approach to manage variability and generate customized domain-specific rules. The authors of the paper elieve that this approach has the potential to be applied in various domains and can be further extended to support more complex scenarios. However, the approach was applied in a specific domain and its main purpose was to be used in the relation with business processes. On the other side, the authors of the paper did not discuss if this approach can be applied in other architecture layers than the business architecture layer.

M. Asadi, B. Mohabbati, N. Kaviani, D. Gašević, M. Bošković, M. Hatala – Model-driven development of families of service-oriented architectures [20]

Developing Service Oriented Architecture (SOA) in a coherent process is important and SPLE can help in selecting business activities, unit services, service interfaces, and implementations. Combining Model-Driven Engineering (MDE) principles with SPLE can bridge the gap between business process management and software engineering. Domain engineering can be consistently conducted across all SOA layers to determine the families of SOA from business process specifications. Combining business requirements with software engineering requirements can provide implementation of service interfaces for business activities and unit services. The proposed method has two main life cycles: domain and application engineering.

1. Domain engineering: a software engineering process that involves investigating a family of business processes and creating reusable assets and reference architectures for software families. The process involves scoping the product line and creating variability models, followed by family requirement analysis, business process family design, and business process family annotation. The business process family realization involves implementing the business process model based on unit service models, while the service interface implementation deals with the development of service components. The process can be iterative and incremental, enabling the creation of business sub-processes and their relationships with corresponding requirements and features.
2. Application engineering: the paper describes the process of utilizing reused artifacts and adapting the reference architecture to create final products specific to the requirements of a particular business organization. The three main phases of application engineering include Application Requirement Analysis, where requirements are collected for the target business organization; Application Design, where a configuration is selected from the feature model and an initial business process model is created; and Application Implementation, where the business process is deployed and tested. These phases are iterative and incremental, allowing for changes to be made as necessary based on verification from stakeholders, business process participants, and software engineers. Examples of changes that may require going back to a previous phase include dissatisfaction with performance goals or deployment constraints.

Overall, the methodology has the potential to significantly improve the efficiency and effectiveness of developing families of software products for a wide range of industries and applications. By leveraging the power of SPLE and SOA, it is possible to reduce development costs, improve time-to-market, and increase product quality and customer satisfaction. Future work should focus on refining and evaluating the methodology and developing tools and frameworks to support its implementation in real-world settings.

Nevertheless, the paper did not expose a practical example of the proposed methodology and did not mention whether it can be applied to other layers of architecture than the business architecture layer.

K. Wehling, D. Wille, C. Seidl, I. Schaefer – Automated recommendations for reducing unnecessary variability of technology architectures [23]
The paper proposes an approach, which provides experts with recommendations for restructurings of related TAs to reduce unnecessary variability. The approach consists of three phases based on 150 % model which are includes the following:

- Rule-based analysis,
- Pattern analysis,
- Deduction of recommendations.

The case study analyzes four clusters of technical architectures related to different web servers and database systems. Each cluster was sorted by size and the top 20 Technology Architectures (TAs) were selected to form four sets. The sets were then analyzed using decision rules and pattern analysis. A 150% model was generated for each set and presented to the experts. The proposed algorithms were executed, and the resulting recommendations were evaluated by the experts. Comprehensive analysis including a comparison with manually deduced expert recommendations was not possible due to time constraints and is future work.

The evaluation of a domain-specific language (DSL) for TAs was based on the assessment of six industry experts who may not necessarily agree with the custom-tailored decision rules and criteria. Experts could use the DSL to create their own decision rules, but time constraints meant that not all unsuitable recommendations were identified. Additionally, some deduced recommendations may not be precise enough, but the experts found the provided recommendations useful for reducing unnecessary variability. The proposed method was applied in specific industry domains and on the layer of technology architecture.

S. Adjoyan, A. Seriai – An architecture description language for dynamic service-oriented product lines [24]
The paper offers an approach to describe the changing architecture of Dynamic Service-Oriented Product Lines (DSOPL). By introducing an Architecture Description Language (ADL) three types of information: architecture's structural elements, variability elements, and system's configuration are described. The suggested language is called DSOPL-ADL (Dynamic Service-Oriented Product Lines- Architecture Description Language) which allows the runtime variability of service-based product lines systems. The DSOPL-ADL is structured and composed of four sections: structural, variability, context, and configuration. Each of these sections has its own metamodel.

The suggested language, DSOPL-ADL, models runtime variability in service-based product lines. It comprises four sections, providing a comprehensive approach to managing variability. The paper focuses on spatial variability but suggests that temporal variability should be explored in future work. The lack of real-time configuration verification during late binding may be a limitation, but pre- and post-conditions compensate. Generating BPEL (Business Process Execution Language) processes from this architecture is also a promising future direction. Overall, the authors present a clear, well-structured approach, with potential use for architects and developers working on such systems. The authors used only an illustrative example for their approach. In addition, the proposed language did not deal with the variability in architecture at any layer.

4.4 Interpretation and Discussion

This section covers the last step of the SLR: interpreting its results. The identified papers will be examined in the context of the overall RQ to provide an overview of the current state of research regarding variability management in EAs.

The results show that not much research on variability management in EAs exists. Only a few papers were identified that intersect with the topic in different ways, depending on the chosen inclusion criteria. In total, we were able to identify 13 papers that overlap with the RQ in the following way: Rurua, Eshuis et al. [16] gave a solution design of an extended enterprise architecture metamodel based on ArchiMate to identify the variation on the business process layer, whereas the paper of Allian, Sena et al. [15] provides an overview of the most relevant techniques and methods used for evaluating variability in the software architecture layer. Wille, Wehling et al. [14] used the software product line and the automatic mining algorithm as an approach to variability mining at the technology architecture layer. The paper of Langemeier, Rosina et al. [13] uses the concept of variability management by having a knowledge engineer (KE) select modular ontologies to be stored in an ontology repository with the intent to add this to different variability models and identify each of those variables as a feature to make normalization between the variability models. Wehling, Wille et al. [19] offer an approach to identify the variability in each application architecture by adapting a specific variability mining technique from the software product line (SPL) to reduce the use of unnecessary IT complexity of application architectures. Similarly, using the software product line approach, complement Nerome, Numao [17] a web application architecture using the software product line (SPL) and UML-based metamodel with a notation of variability.

Horcas et al. paper [25] tries to solve the problem of variability in a multi-tenant application environment using a Product Line Architecture (PLA)-based approach consisting of cardinality-based variability models of the Common Variability Language (CVL) to model each tenant as a cloneable feature. Benavides and Galindo [22] obtain in their paper the usage of variation points to identify configuration processes out of variation of business processes in a specific domain. The papers of Asadi, Mohabbat, et al. [20] and Mani, Helfert et al. [21] show the usage of the software product line (SPL) technique and the Model Driven Architecture (MDA) in the application architecture. Wehling, Wille et al. [23] propose using the 150% model with its three phases to obtain a method for experts with recommendations for restructurings of related technology architecture to reduce unnecessary variability. Adjoyan, Seriai [24] introduced the DSOPL-ADL (Dynamic Service-Oriented Product Lines- Architecture Description Language). The DSOPL-ADL is structured and comprises four sections: structural, variability, context, and configuration.

In conclusion, the analysis of existing work in the field shows that the proposed approaches (1) primarily focus on variability in one architecture domain, (2) mostly propose ways to design enterprise architectures rather than to identify variability and improvement options in existing architectures, and (3) are evaluated in real-world case studies (5 out of 13 papers) which indicate the value of such case studies for evaluating approaches. Table 5 shows that all papers and approaches focus on one architecture layer; only two approaches [16, 22] also consider effects on one neighboring layer, and we did not find any paper that proposes a method to deal with variability in all EA layers. Furthermore, there is no approach focusing on data architecture variability. The missing coverage of all or at least three architecture layers and the lack of work on data architecture variability is a clear research gap.

5 Industrial Case Study on Developing Building Blocks

The industrial case study aims to explore the challenge of variability management in enterprise architectures in industrial practice to better understand the industrial challenges. Furthermore, we investigate the feasibility of identifying cross-layer building blocks from EA models. This section starts with a description of the case study and continues with presenting and applying an initial approach for creating EA building blocks.

5.1 Case Study Overview

A leading solution provider for the energy and water industry is developing software that enables the realization, optimization, and expansion of business processes in the energy and water sector. The solution covers the implementation of business processes, including the application, data, and technology layers in a digital environment. With the constant changes in business processes due to regulations and the need to adapt to new technologies, finding a method to capture adaptations in the architecture quickly is necessary. The architecture is documented using ArchiMate models. These models form the basis for both defining and implementing development roadmaps for the software products and defining instantiations for the company's clients. The instantiations usually are no mere configurations but only contain a selection of the functionality required for the client.

The company's aim to create customer solutions by "simply combining building blocks" was only part of their motivation to investigate decomposing the EA considering variability aspects. Another driver is the increasing use of artificial intelligence to discover anomalies in data, enable churn management or support prognosis functions. With such AI applications in mind, the "data footprint" of the software always has to meet certain integrity constraints that potentially could lead to a data architecture much more extensive than required for a certain combination of building blocks.

The envisioned method has to offer the ability to capture an enterprise architecture in a modular way. The modules have to represent independent elements that fulfil specific tasks and form the building blocks for the company's clients' operations. By identifying and structuring process modules, companies can systematically improve their processes and respond more agilely to new requirements.

The basic idea of building blocks is simple: building blocks consist of sequences of activities that are part of processes in combination with the data required for these activities and the services or functions required to create or process them. If possible, each building block should only include the minimum set of data required for the activities. On the other hand, the data architecture should be kept consistent and not be too fragmented. Dependencies between building blocks on activity, data, and service layer have to be visualized, for instance, by using feature models. The obvious conflict between only including the minimum set of data for activities and not fragmenting the data too much is the real challenge and must be defined more precisely.

5.2 Decomposing the Case Study EA into Building Blocks

By attempting to decompose a small part of an existing enterprise architecture model, we aim to investigate the feasibility of creating EA building blocks that include business, application, and data architecture. The demand for such building blocks originates from the business line managers and enterprise architects of the case study company, who observed substantial problems in meeting customer requirements regarding business process configuration and variations on the business architecture layer while at the same time keeping the application and data architecture implementing these requirements modular. Based on the case study, we expect to confirm the feasibility of forming building blocks and to collect ideas for the envisioned methodical approach.

The feasibility investigation happened more exploratively than following a defined procedure. We started from the idea that a building block for each sub-process of a business process would, if feasible, open the opportunity of high flexibility to compose new customer processes based on the building blocks of these sub-processes. For each sub-process we had the plan to form the corresponding building block by including the EA model elements of the application and data architecture layers into the building block that, in the model, is connected by relationships to the sub-process.

We selected one of these processes in the case study and applied our approach to it. The process we chose is known as “Geräteeinbau” in German, which translates to “Device installation” in English. This process outlines the necessary steps for installing or uninstalling a meter and other related devices. Primarily, the meter measures the amount of electricity and water consumed at a specific location. The model in Figure 2 shows the status of the “Device installation” process.

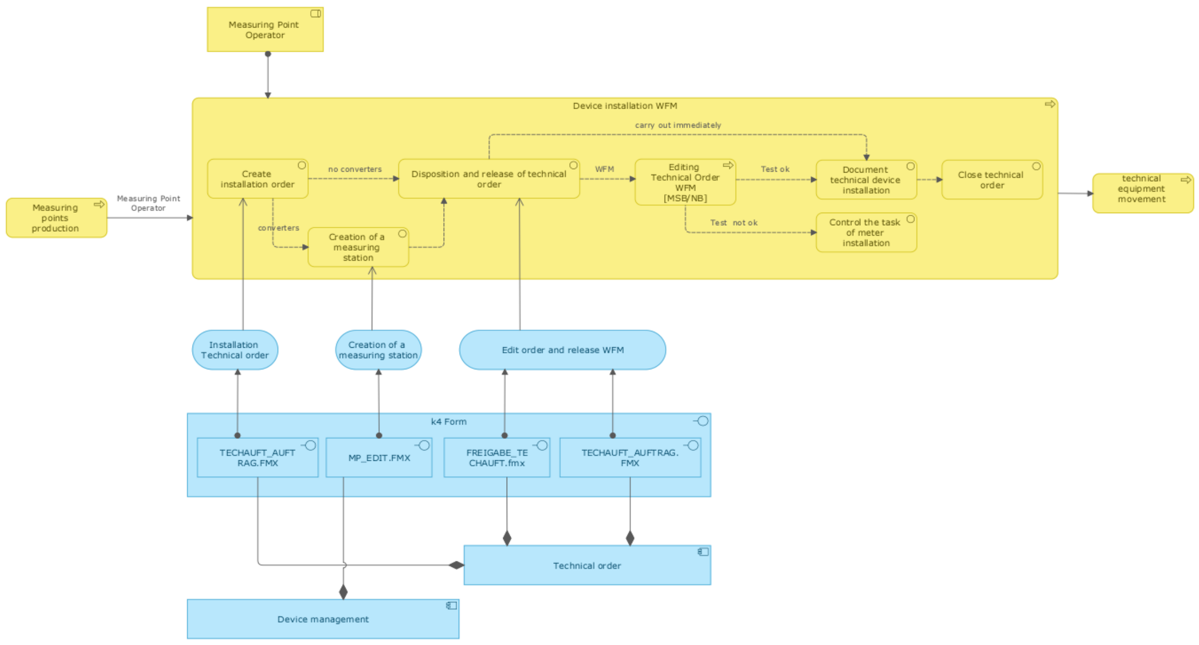
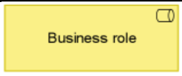
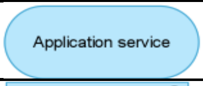
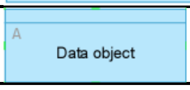

Measuring Point Operator
Device installation WFM
carry out immediately
Create installation order
no converters
Disposition and release of technical order
WFM
Editing Technical Order WFM [MSB/NB]
Test ok
Document technical device installation
Close technical order
technical equipment movement
Measuring points production
Measuring Point Operator
converters
Creation of a measuring station
Test not ok
Control the task of meter installation
Installation Technical order
Creation of a measuring station
Edit order and release WFM
k4 Form
TECHAUFT_AUFTRAG.FMX
MP_EDIT.FMX
FREIGABE_TECHAUFT.fmx
TECHAUFT_AUFTRAG.FMX
Technical order
Device management

Figure 2. Process of Device Installation WFM (Workforce Management)

We introduce its constituent elements to provide a clearer explanation of the model. The model was developed using the ArchiMate framework and incorporates the elements listed in Table 6.

Table 6. Overview of ArchiMate elements in the model

Element name	Element picture
Business role	
Business process	
Business activity	
Application service	
Application interface	
Application Component	
Data object	

The model illustrates that the Device installation WFM process is assigned to the Measuring Point Operator and can begin as a subsequent step to the Measuring Points production [NB] process. Within this process, there are several business activities that contain points of variation. For instance, after the business activity of creating an installation order, a variation point occurs to determine if a converter exists or not. Another variation was found after the sub-business process of editing technical order WFM [MSB/NB]. Additionally, the model displays how this process will be implemented in the application layer. To create an installation order, the application service Installation Technical Order will be called through its application interface TECHAUFT_AUFTRAG.FMX. This interface is also a component of the Device management application. Once the process is completed, another process of technical equipment movement [MSB] will follow.

During the analysis of the model, we discovered that the current architecture lacks the data architecture model. Therefore, we have made the necessary addition, as presented in the model in Figure 3.

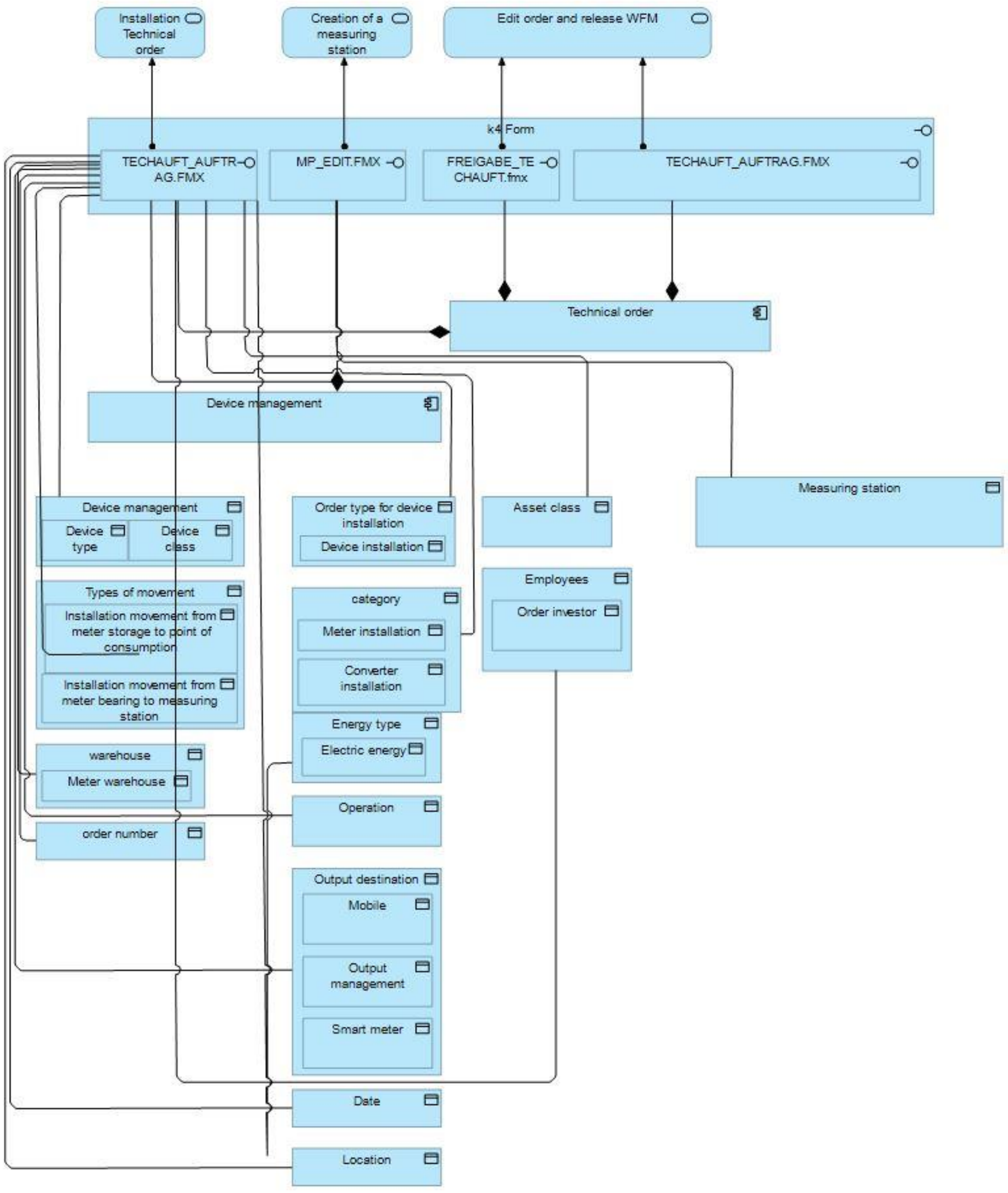


Figure 3. Data architecture of Device installation WFM process

We utilized the element (Data object) from ArchiMate to depict the data architecture layer. In certain scenarios, we found it necessary to include a higher-level category for the data objects. For instance, Device management serves as a top category (data object) encompassing both device type and device class as subordinate data objects.

When analyzing the data architecture, we observed difficulties in extracting and visualizing the relationships between business and data architecture from the model. However, these relationships are essential as they express important dependencies. A feature model (see Section 3.2) would be a suitable tool for more explicit visualization of the dependencies. Thus, we developed a feature model that showed the different sub-processes of the Process of Device Installation that would qualify as building blocks and what parts of the data architecture they require. As shown in Figure 4, this is a possibility to visualize dependencies between activities and also if they can or must be combined.

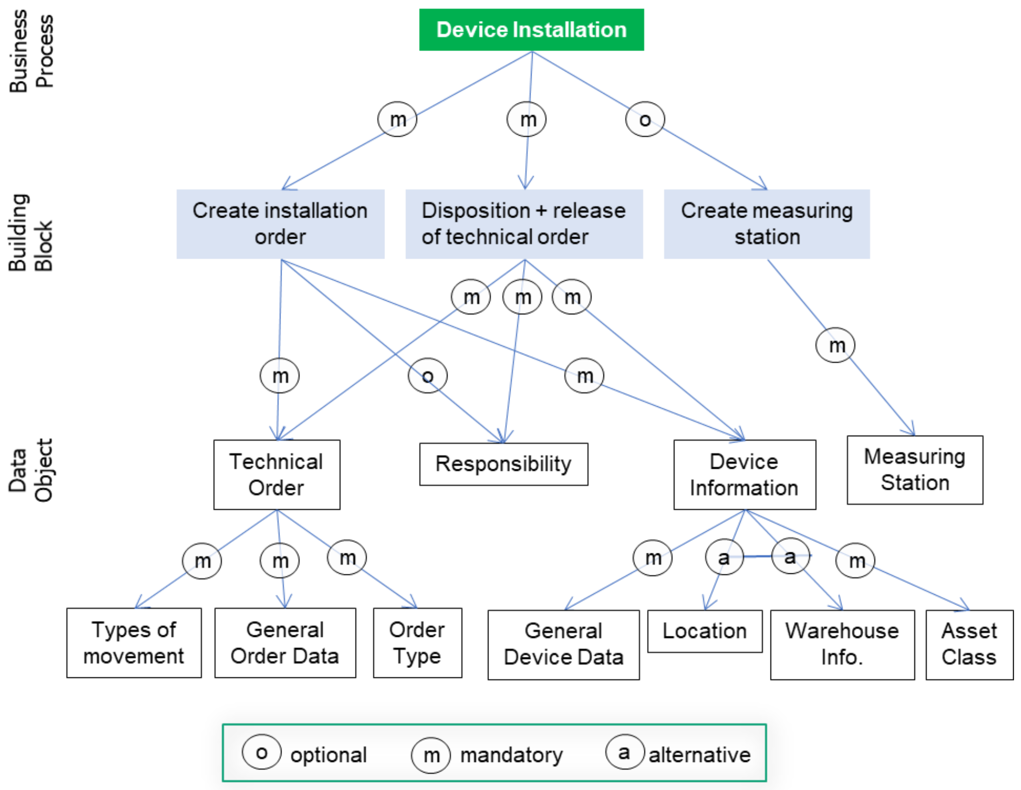


Figure 4. Feature Model to support Building Block Configuration

For the Business Process of "Device Installation" (the root of the example feature model), three activity sets are candidates for building blocks and are shown on the second level. The relation type between the business process and the building block indicates if the building block is mandatory for the process or optional. The required (mandatory or optional) data is shown on the third level for each building block. In our example, it becomes clear that the required data for the building blocks does not overlap, i.e., the building blocks are independent of each other regarding data use. However, the data architecture is quite "entangled" as transaction data (e.g., order information) and basic data (e.g., device information) are stored together. A revision of the data architecture will improve the possibility of using the data more flexibly for other process-oriented building blocks.

Utilizing the feature model, we successfully established four distinct variants of the process and identified several potential building blocks. Consequently, we present a few of these building blocks in the subsequent sections.

First, we have the building block represented by the business activity "Create installation order" (Figure 5) and the application service "Installation Technical order". This service is part of the application component "Technical order" and can be accessed through the application interface "TECHAUFT_AUFTRAG.FMX". The application interface "TECHAUFT_AUFTRAG.FMX" is a software page used to edit data objects.

Another building block is the "Disposition and release of technical orders" (Figure 6), which also includes the business activity of disposing and releasing technical orders. It encompasses the application service of editing orders and releasing WFM (Workforce Management). Additionally, two application interfaces, FREIGABE_TECHAUFT.fmx and TECHAUFT_AUFTRAG.FMX, are utilized, along with one application component, the technical order. It should be noted that no data objects are required for this building block.

One additional building block is the “Creation of a measuring station” (Figure 7). It involves a specific business activity, also referred to as the “Creation of a measuring station”. This activity is supported by the application services that are responsible for facilitating the “Creation of a measuring station”. These application services are linked to the application interface (MP_EDIT.FMX), which is a part of the wider application component known as “Device management”. Moreover, these services utilize the data object called “Measuring station”.

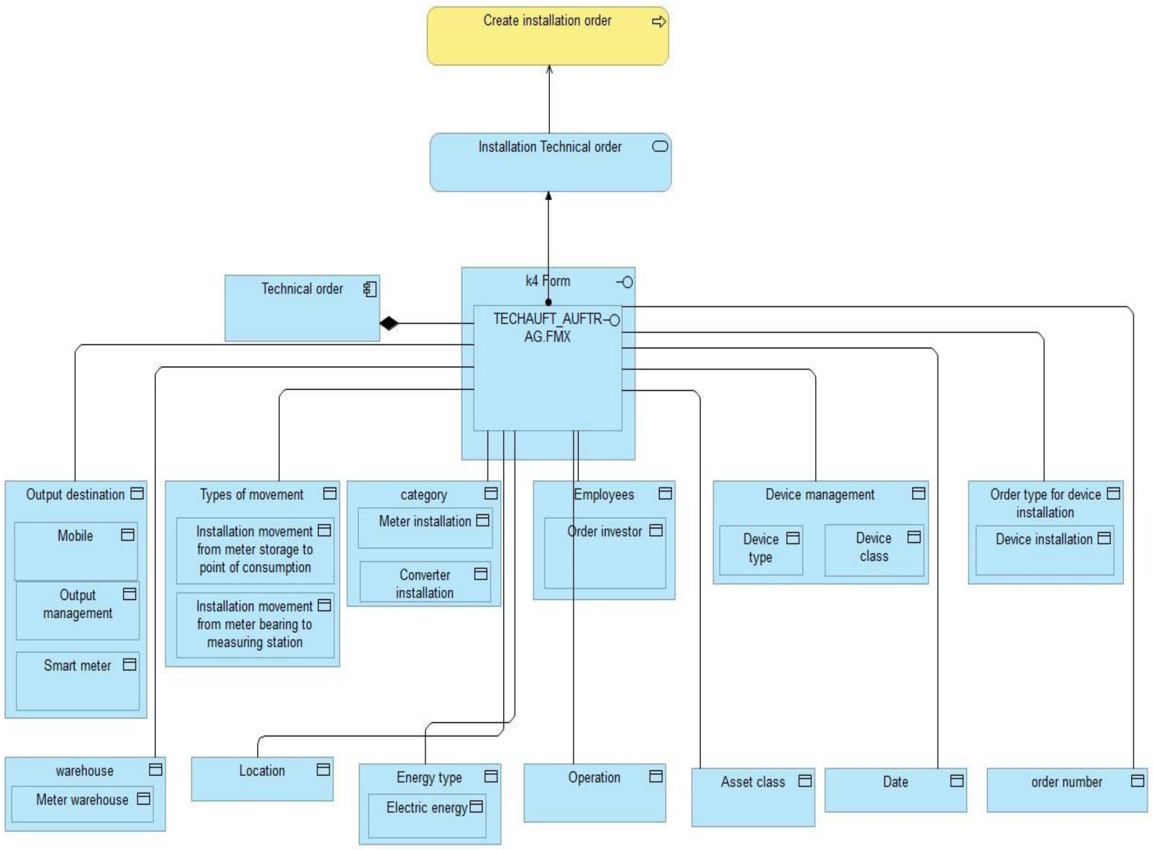
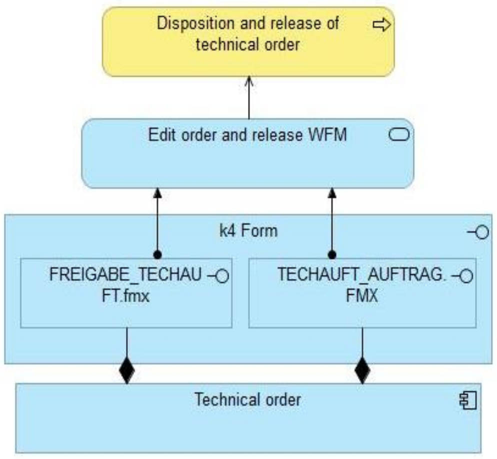


Figure 5. Building Block **“**Create installation order**”**



Disposition and release of technical order
Edit order and release WFM
k4 Form
FREIGABE_TECHAU FT.fmx
TECHAUFT_AUFTRAG. FMX
Technical order

Figure 6. Building Block “Disposition and release of technical order”

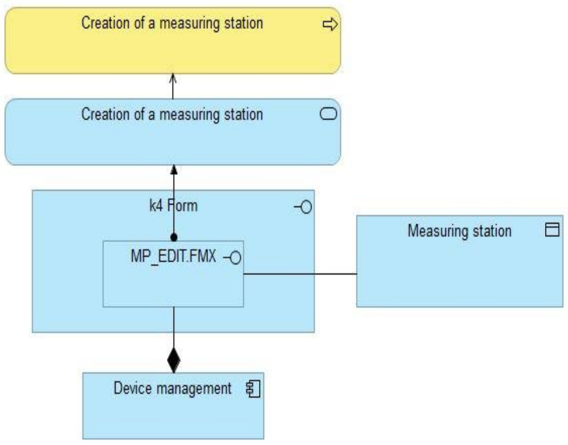

Creation of a measuring station
Creation of a measuring station
k4 Form
MP_EDIT.FMX
Measuring station
Device management

Figure 7. Building Block **“**Creation of a measuring station**”**

There are several observations and lessons to be learned from the case study:

- It was feasible to decompose the selected part of the enterprise architecture into building blocks covering the EA layers of business, application and data architecture.
- All three EA layers are required in a building block to fully express the dependencies of the different layers and effects of using a building block.
- Feature models can be applied to visualize dependencies between business and data architecture, which were difficult to extract from the EA model. Thus, a feature model could be a valuable contribution to the support of the envisioned method.
- Starting from single sub-processes to develop building blocks only makes sense if the sub-processes represent parts of business services that potentially are required in several business services. Otherwise, extracting building blocks seems unnecessary.
- Revision of the data architecture in more independent units could help to reduce the tangledness of business and data architecture. This should be part of the method and requires further investigation.

The biggest limitation of the work so far is that the above observations are based on only a single case study. More empirical work and a full decomposition of an EA are needed in future work.

6 Summary, Future Work and Limitations

The work in this article focused on two research questions. For RQ-SLR “What is the state of research on managing variability in enterprise architectures?”, Section 4 shows that some research exists, but this existing work focuses on treating variability on one architecture layer (e.g., variability in the business architecture) with a few approaches also tackling the effects of this variability on one of the neighboring layers. However, how to control variability in business architecture and simultaneously capture the induced variability of applications and data architecture has not been the subject of research. We argue that such an architecture-spanning approach is required to ease the implementation of complex changes in enterprises, such as digital transformation or the introduction of artificial intelligence, which motivates further research in this field.

For RQ-CS “How to capture EA variability in industrial practice?”, Section 5 shows that a combination of the feature model and enterprise architecture model consisting of a set of activities, the relevant data, and services could be a suitable approach. However, only one case study is not sufficient to confirm this result. More work is required to define the detailed process, conceptualize a formal representation (possibly as an extension of ArchiMate), and apply it in more use cases.

In this context, it would be worth investigating the integration of reuse approaches “in the large”, such as reference models, and reuse in “in the small”, such as what is enabled by the approaches tackling variability management. We believe feature modeling would be a promising way to integrate these different streams of research.

For the envisioned methodical approach for identifying EA building block we identified several activities and steps that could form the skeleton for a future systematic process: define application area and scope of the study; acquire EA models from enterprise under consideration covering the scope; decompose EA model starting from the business architecture; in the business architecture use sub-processes of business processes as starting point and "candidates" for building blocks; derive building block encompassing all EA model elements on application and data architecture layer that have relationships to each sub-process selected as starting point; visualize the dependencies between the architecture layers of building blocks by using feature models; analyze the resulting building blocks for improvement potential by revising the data architecture; develop and implement roadmap for implementing the required revision on data architecture layer; plan revision of the building blocks when data architecture revision is finished. These steps require refinement, confirmation, and evaluation in future work.

References

[1] J. D. Rittelmeyer and K. Sandkuhl, "Effects of artificial intelligence on enterprise architectures-a structured literature review," in *2021 IEEE 25th International Enterprise Distributed Object Computing Workshop (EDOCW)*, 2021, pp. 130–137. Available: https://doi.org/10.1109/EDOCW52865.2021.00042

[2] J. Kaidalova, K. Sandkuhl, and U. Seigerroth, "How digital transformation affects enterprise architecture management – a case study," *International Journal of Information Systems and Project Management*, vol. 6, no. 3, pp. 5–18, 2018. Available: https://doi.org/10.12821/ijispm060301

[3] A. Hevner and S. Chatterjee, *Design research in information systems: theory and practice*. Springer Science & Business Media, 2010. Available: https://doi.org/10.1007/978-1-4419-5653-8

[4] B. Kitchenham, O. P. Brereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman, "Systematic literature reviews in software engineering-a systematic literature review," *Information and Software Technology*, vol. 51, no. 1, pp. 7–15, 2009. Available: https://doi.org/10.1016/j.infsof.2008.09.009

[5] R. K. Yin, *Case study research: Design and methods*. Sage, 2009.

[6] F. Ahlemann, E. Stettiner, M. Messerschmidt, and C. Legner, *Strategic Enterprise Architecture Management: challenges, best practices, and future developments*. Springer Science & Business Media, 2012. Available: https://doi.org/10.1007/978-3-642-24223-6

[7] A. Josey, *TOGAF$^{®}$ version 9.1 – A Pocket Guide*. Van Haren, 2016.

[8] B. Barth, G. Butler, K. Czarnecki, and U. Eisenecker, "Generative programming," *Object-Oriented Technology, ECOOP 2001. Lecture Notes in Computer Science*, vol. 2323, pp. 135–149. Available: https://doi.org/10.1007/3-540-47853-1_11

[9] K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak, and A. S. Peterson, "Feature-oriented domain analysis (FODA) feasibility study," *Technical Report CMU/SEI-90-TR-021*, Software Engineering Institute, 1990.

[10] C. Thörn and K. Sandkuhl, "Feature modeling: managing variability in complex systems," *Complex Systems in Knowledge-based Environments: Theory, Models and Applications*, pp. 129–162, 2009. Available: https://doi.org/10.1007/978-3-540-88075-2_6

[11] L. Li *et al.*, "A survey of feature modeling methods: Historical evolution and new development," *Robotics and Computer-Integrated Manufacturing*, vol. 61, p. 101851, 2020. Available: https://doi.org/10.1016/j.rcim.2019.101851

[12] K. Sandkuhl, "Quantified Products: Case Studies, Features and their Design Implications," *Informatica*, vol. 34, no. 4, pp. 825–845, 2023. Available: https://doi.org/10.15388/23-INFOR532

[13] M. Langermeier, P. Rosina, H. Oberkampf, T. Driessen, and B. Bauer, "Management of Variability in Modular Ontology Development," in *Service-Oriented Computing – ICSOC 2013 Workshops, Lecture Notes in Computer Science*, vol. 8377, 2014, pp. 225–239. Available: https://doi.org/10.1007/978-3-319-06859-6_20

[14] D. Wille, K. Wehling, C. Seidl, M. Pluchator, and I. Schaefer, "Variability Mining of Technical Architectures," in *Proceedings of the 21st International Systems and Software Product Line Conference – Volume A*, 2017, pp. 39–48. Available: https://doi.org/10.1145/3106195.3106202

[15] A. P. Allian, B. Sena, and E. Y. Nakagawa, “Evaluating variability at the software architecture level,” in *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, 2019, pp. 2354–2361. Available: https://doi.org/10.1145/3297280.3297511

[16] N. Rurua, R. Eshuis, and M. Razavian, “Representing Variability in Enterprise Architecture,” *Bus Inf Syst Eng*, vol. 61, no. 2, pp. 215–227, 2019. Available: https://doi.org/10.1007/s12599-017-0511-3

[17] T. Nerome and M. Numao, “A Product Domain Model Based Software Product Line Engineering for Web Application,” in *2014 Second International Symposium on Computing and Networking*, 2014, pp. 572–576. Available: https://doi.org/10.1109/CANDAR.2014.105

[18] M. Cohen *et al.*, *Proceedings of the 21st International Systems and Software Product Line Conference – Volume A*. ACM, 2017. Available: https://doi.org/10.1145/3106195

[19] K. Wehling, D. Wille, C. Seidl, and I. Schaefer, “Decision Support for Reducing Unnecessary IT Complexity of Application Architectures,” in *2017 IEEE International Conference on Software Architecture Workshops (ICSAW)*, 2017, pp. 161–168. Available: https://doi.org/10.1109/ICSAW.2017.47

[20] M. Asadi, B. Mohabbati, N. Kaviani, D. Gašević, M. Bošković, and M. Hatala, “Model-driven development of families of Service-Oriented Architectures,” in *Proceedings of the First International Workshop on Feature-Oriented Software Development*, 2009, pp. 95–102. Available: https://doi.org/10.1145/1629716.1629735

[21] N. Mani, M. Helfert, and C. Pahl, “A Domain-specific Rule Generation Using Model-Driven Architecture in Controlled Variability Model,” *Procedia Computer Science*, vol. 112, pp. 2354–2362, 2017. Available: https://doi.org/10.1016/j.procs.2017.08.206

[22] D. Benavides and J. A. Galindo, “Variability management in an unaware software product line company,” in *Proceedings of the Eighth International Workshop on Variability Modelling of Software-Intensive Systems*, 2014, pp. 1–6. Available: https://doi.org/10.1145/2556624.2556633

[23] K. Wehling, D. Wille, C. Seidl, and I. Schaefer, “Automated recommendations for reducing unnecessary variability of technology architectures,” in *Proceedings of the 8th ACM SIGPLAN International Workshop on Feature-Oriented Software Development*, 2017, pp. 1–10. Available: https://doi.org/10.1145/3141848.3141849

[24] S. Adjoyan and A. Seriai, “An Architecture Description Language for Dynamic Service-Oriented Product Lines,” in *Proceedings of the 27th International Conference on Software Engineering and Knowledge Engineering*, 2015, pp. 231–236. Available: https://doi.org/10.18293/SEKE2015-217

[25] J.-M. Horcas, M. Pinto, and L. Fuentes, “Product Line Architecture for Automatic Evolution of Multi-Tenant Applications,” in *2016 IEEE 20th International Enterprise Distributed Object Computing Conference (EDOC)*, 2016, pp. 1–10. Available: https://doi.org/10.1109/EDOC.2016.7579384