Variability modelling in enterprise architecture management – survey on existing approaches

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

Managing and dealing with variability is a common challenge in the daily practice of many enterprises and organizations. Recent studies have observed that digital transformation and artificial intelligence solutions lead to changes in enterprises that simultaneously require variability on several levels of an enterprise (for example, 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. Still, there is not much work on managing variability in enterprise architectures. This paper aims to structure the existing research work and tribute to a better understanding of future investigation needs. 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 levels.

Keywords

Variability, Enterprise Architecture, Enterprise Architecture Management

1. Introduction

Managing and dealing with variability is a common challenge in the daily practice of many enterprises and organizations. Examples are business processes that exist in several variants, products with variations regarding components and configuration, services varying in their customer frontend, or business offers depending on their delivery context. Different strategies have been developed to deal with variation with the extreme positions of allowing for no variability at all (rigid standardization) and full flexibility (no limitation for variability). Often the approach is to control the number of variants as a compromise between limiting complexity and allowing for flexibility.

Recent studies have observed that digital transformation 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, for example, 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 as 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 in many organizations gaining importance due to the increasing number of data-driven products and services, and due to the use of AI solutions. Data engineering, from an EA perspective should be based on and tightly coupled to the data architecture of an organization 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 the use in data-driven applications

BIR-WS 2023: BIR 2023 Workshops and Doctoral Consortium, 22nd International Conference on Perspectives in Business Informatics Research (BIR 2023), September 13-15, 2023, Ascoli Piceno, Italy
ahmed.dehne@uni-rostock.de (A. 1);kurt.sandkuhl@uni-rostock.de (A. 2)
0000-0002-7431-8412 (A. 2)

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 levels. 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. The objective of this paper is to structure the existing research work and to contribute to a better understanding of future research needs.

The paper 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 literature search. Section 5 discusses the results. Section 6 gives an outlook on future work.

2. Research Approach

The main objective of this research 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 method of structured literature review according to Kitchenham [1]. We choose this approach because it was developed to evaluate what has been published in a specific research topic, to compare existing research, and to analyze potential research gaps. Based on [1], 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 (step 1):

- RQ: 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 take into account in light of the research questions. 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) has its focus on extracting the information relevant for 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 of relevant papers.

We finalize the paper with a research summary and explain possible future work (cf. section 6).

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". Thereby, EAM depicts a management philosophy, organizational function and a systematic approach with tools and practices to develop the EA [2].

TOGAF [3] is considered by many researchers as industry standard and defines three different architectural levels which are visible in many other frameworks: The Business Architecture defines the 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 their relationships to the core business processes of an organization. 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 Management

Variability modeling has its origin in software product line research and generative programming [4]. 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 at the same time restrict the systems' complexity in order 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 as in particular 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" [5]. 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 are the elements of the products that differentiate and show the configuration options, choices and variation points that are possible between variants of the product, aimed to satisfy 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 tree, in the feature model. The hierarchy and other properties of the feature model are visualized in a feature diagram. Feature diagrams express the relation 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 [6] and an overview to methods for feature model development is provided in [7]. Both papers show a focus on notations and approaches specialized for certain application fields, i.e., there is no generally accepted feature model development method.

4. Conducting a Structured Literature Review

We first developed several search strings (cf. section 4.1) and selected the papers according to inclusion and exclusion criteria (cf. section 0). We then collected the data and summarized the results in section 4.3.

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:

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. Alongside, we 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. For the search string development and the actual search, we used the database 'Scopus'. 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 a lot of 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 of the fourth, fifth and sixth search strings. We also tested different search strings, but we only share the most relevant ones here because of page restrictions. 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	[8–11]	Business Architecture, Software Architecture, Technology Architecture, Application Architecture
2	TITLE-ABS-KEY(variability) AND TITLE-ABS-KEY("application architecture")	14	[12–14]	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	[15, 16]	Business Architecture, Business Architecture
4	TITLE-ABS-KEY(variability) AND TITLE-ABS-KEY("business architecture")	5	[17]	Business Architecture
5	TITLE-ABS-KEY(variability)	8	[18]	Technology Architecture

	AND TITLE-ABS-KEY("technology architecture")			
6	TITLE-ABS-KEY(variability) AND TITLE-ABS-KEY("information architecture")	11	[19]	Information Architecture

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

4.2. Paper Selection & 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. We grouped the criteria as shown in Table 3.

Throughout the search process, we determined a big amount of research about capturing the variability in different levels of systems and enterprises. However, the objective of this research is to discover papers which can deliver methods of modeling variability in EAM. For more of the variability-related papers, the target group are the tool users. Hence, we decided to exclude papers that researched the variability without context to EAM.

Table 3.
Inclusion Criteria for the SLR

Variability and EAM	EAM Level
Var and EAM1. Is Variability in context with EAM investigated at all?	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, and introduction and summary. Table 4. Overview of inclusion criteria applied to the most relevant papers shows the abstracted form of the Excel sheet used to record the results. It summarizes the 13 papers that fulfilled the criteria and requirements for the research question. The results and the content of the identified papers will be described in the next section.

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

Paper Criteria	[8]	[9]	[10]	[11]	[12]	[14]	[17]	[15]	[16]	[18]	[19]
Var. and EAM1.	X	X	X	X	X	X	X	X	X	X	X
EA1.1.	X						X	X	X		
EA1.2.				X		X					

EA1.3	X									
EA1.4		X		X					X	
EA1.5										X
EA2										

4.3. Data Collection & 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 [20]. 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. Afterward we measured the maturity level of each paper according to the Design science research method. Hence, we investigated if the proposed approaches used the six process iteration steps in it. especially the demonstration and evaluation steps. We summarize the identified relevant papers in this section.

Rurua, N., Eshuis, R., Razavian, M. - Representing Variability in Enterprise Architecture: A Case Study [8]

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 in the mentioned level of architecture. Solution design is an extended enterprise architecture metamodel based on ArchiMate. For the extension some new elements were added to cover concepts like 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. All in all, 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 didn’t expose whether it’s possible to apply the same method on any other level of architecture.

Allian, A.P., Sena, B., Nakagawa, E.Y.- Evaluating variability at the software architecture level: An overview [9]

The evaluation of software architectures with variability information is an important task. In the current state of the art, various means including Product Line Architectures (PLAs), software architectures, reference architectures, and enterprise architectures are utilized for evaluation purposes. Therefore, the paper provides an overview and insight to practitioners about the most relevant techniques and methods developed for this evaluation. A Systematic Mapping Study (SMS) reviewed 1,457 studies about evaluation techniques and methods. After the initial exclusion process, 30 studies were selected to provide practitioners with an overview and insight on the most relevant ones. The SMS shows the following observation:

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

Wille, D., Wehling, K., Seidl, C., Pluchator, M., Schaefer, I.- Variability mining of technical architectures [10]

The paper discusses the problem of the increasing number and variety of Technical Architectures 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 potential for reuse and make maintenance decisions. 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 applied just on the level of technology architecture and didn’t expose if the method could be applied to other enterprise architectures.

Langermeier, M., Rosina, P., Oberkampf, H., Driessen, T., Bauer, B.- Management of variability in modular ontology development [11]

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 be stored 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 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 extension of stronger constraints depending on specific domain requirements. This form formalizes variables that have meaning, and which is 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 with the connection with the variability in the application architecture level, but the paper did not discuss the ability of using the same method on the entire Enterprise architecture.

Wehling, K., Wille, D., Seidl, C., Schaefer, I.- Decision support for reducing unnecessary IT complexity of application architectures [12]

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 consists of five phases and as shown in figure 3 will be offered to support experts in determining and reducing the unnecessary part of the application architecture's complexity by using the identified variability. Though the introduced approach was applied to as a preliminary case study considering data on AAs that obtained from one industry partner. An analysis of the usage of applications in four different company sites with focus on manufacturing of customer ordered products. To see the impact of different variability levels on the applied approach. However, the applied method has been tested in the level of application architecture, it does not apply to other levels of architecture.

Horcas, J.-M., Pinto, M., Fuentes, L.- Product line architecture for automatic evolution of multi-tenant applications [20]

The approach for multi-tenant apps 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 us to validate the generated configuration models automatically and efficiently. Furthermore, the objective function helps us 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 are recommending investigating the generalizability of the approach to other applications and cloud platforms. Additionally, it's 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. All in all, the applied approach was tested in the case study and proved to be successful in working in the defined environment. However, it didn’t apply to any other architecture layers.

Nerome, T., Numao, M.- A product domain model based software product line engineering for web application [14]

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 evolving needs of the banking industry and clients. As it’s mentioned the study was applied in one industry field which is banking and covered the variation in the application layer. Nevertheless, the study didn’t deal with other architecture layers.

Benavides, D., Galindo, J.A.- Variability management in an unaware software product line company. An experience report [17]

This paper presents an experience report on a software development company that was unaware of software product line approaches. The author spent two months gathering information and observing variability management practices and identified opportunities for improvement. Despite being unfamiliar with product line engineering concepts, the company already had some variability management practices in place. The report briefly covers how variability management is performed in different areas of the company, including business architecture and software assets management. The authors conclude that companies with positive potential characteristics for transition could benefit from adopting software product line engineering practices. The case study company practices Enterprise Architecture management with the norms of TOGAF. The variability in Business architecture is essential due to changes in taxes laws and procedures. Business rules are used to handle the variability in processes, which can change without affecting the entire system. However, explicit identification of business rules is necessary, and business process families are not yet identified. Variability management and product line engineering are crucial in Infrastructure Architecture, and the Case Study Company has around 100 applications that could be considered for several product lines. TOGAF does not explore software product line engineering view in the product portfolio level.

This experience report outlines the use of variability management techniques in a company that was not familiar with software product line concepts. The study found that variability management was performed throughout the organization and was seen as a necessity in most areas. The report offers insight into opportunities for research and recommends more structured empirical research in similar companies to determine how to improve their practices. The report suggests improving the methodology description and learning from past research for a better approach to adopting software product line concepts. Regardless, the experience report didn't mention how to apply the used variability approach in another architectural layer.

Mani, N., Helfert, M., Pahl, C.- A Domain-specific Rule Generation Using Model-Driven Architecture in Controlled Variability Model [15]

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. By using a Model Driven Architecture (MDA) with a four level of architecture to develop the prototyping of the application. The four levels would be representing in figure 7:

After the development of the MDA the paper proposes the using of 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. At third a Domain- specific language will be developed to describe the environment to develop 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 believe that this approach has the potential to be applied in various domains and can be further extended to support more complex scenarios. However, and as it mentioned the approach was applied in a specific domain and its main purpose was to use in the relation with business processes. On the other side the authors didn't expose if this approach is able to apply in other architecture levels.

Asadi, M., Mohabbati, B., Kaviani, N., Gašević, D., Bošković, M., Hatala, M.- Model-driven development of families of service-oriented architectures [16]

Developing SOA in a coherent process is important and Software Product Line Engineering (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 phase involves implementing the business process model based on unit service models, whilst the Service Interface Implementation phase 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: s 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, we can reduce development costs, improve time-to-market, and increase product quality and customer satisfaction. Our future work will focus on further refinement and evaluation of our methodology, as well as developing tools and frameworks to support its implementation in real-world settings.

Nevertheless, the paper didn’t expose a practical example of the proposed methodology and didn’t mention whether this method can be applied to other layers of architecture.

Wehling, K., Wille, D., Seidl, C., Schaefer, I.- Automated recommendations for reducing unnecessary variability of technology architectures [18]

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 like following:

1. Rule-based analysis
2. Pattern analysis
3. 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 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 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 can 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. All in all, the proposed method was applied in specific industry domains and on the level of technology architecture.

Adjoyan, S., Seriai, A.- An architecture description language for dynamic service-oriented product lines [19]

The paper offers an approach to describe the changing architecture of Dynamic Service-Oriented Product Lines (DSOPL). By introducing an Architecture Description Language (ADL) to describe three types of information: architecture's structural elements, variability elements and system's configuration. 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 will have its own Metamodel which will be used to describe the specific section.

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 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. Nevertheless, the authors used an illustrative example for their approach. In addition, the proposed language didn't deal with the variability in architecture at any level.

The following table gives a summary of the outcomes of the utilized search phrases and the respective layer architecture that each phrase pertains to. The table also shows the limitation of the used approaches regarding the architecture layers.

Table5
summary of the outcomes of the utilized search phrases and the respective layer architecture

Search String Nr.	Paper name	Architecture Layer
1	Rurua, N., Eshuis, R., Razavian, M. - Representing Variability in Enterprise Architecture: A Case Study [8]	Business Architecture
1	Allian, A.P., Sena, B., Nakagawa, E.Y.- Evaluating variability at the software architecture level: An overview [9]	Software Architecture
1	Wille, D., Wehling, K., Seidl, C., Pluchator, M., Schaefer, I.- Variability mining of technical architectures [10]	Technology Architecture
1	Langermeier, M., Rosina, P., Oberkampf, H., Driessen, T., Bauer, B.- Management of variability in modular ontology development [11]	Application Architecture
2	Wehling, K., Wille, D., Seidl, C., Schaefer, I.- Decision support for reducing unnecessary IT complexity of application architectures [12]	Technology Architecture
2	Nerome, T., Numao, M.- A product domain model based software product line engineering for web application [14]	Application Architecture

3	Mani, N., Helfert, M., Pahl, C.- A Domain-specific Rule Generation Using Model-Driven Architecture in Controlled Variability Model [15]	Business Architecture
3	Asadi, M., Mohabbati, B., Kaviani, N., Gašević, D., Bošković, M., Hatala, M.- Model-driven development of families of service-oriented architectures [16]	Business Architecture
4	Benavides, D., Galindo, J.A.- Variability management in an unaware software product line company. An experience report [17]	Business Architecture
5	Wehling, K., Wille, D., Seidl, C., Schaefer, I.- Automated recommendations for reducing unnecessary variability of technology architectures [18]	Technology Architecture
6	Adjoyan, S., Seriai, A.- An architecture description language for dynamic service-oriented product lines [19]	Information Architecture

5. Interpretation and Discussion

This section covers the last step of the SLR, the interpretation of 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 the variability management in EAs.

The results show that not much research on the variability management in EAs exists. Only a small number of 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. [8] gave a solution design of an extended enterprise architecture metamodel based on ArchiMate to be identify the variation on the business process level, whereas the paper of Allian, Sena et al. [9] provides an overview of the most relevant techniques and methods used for evaluating variability in the software architecture level. Wille, Wehling et al. [10] used the software product line and the automatic mining algorithm as an approach of variability mining in the technology architecture level. The paper of Langemeier, Rosina et al. [11] uses the concept of variability management with having a knowledge engineer (KE) to 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 variable as a feature to make normalization between the variability models. Wehling, Wille et al. [12] 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. Similar to that and using the software product line approach obtain Nerome, Numao [14] in web application using UML based metamodel with notation of variability. Horcas, Pinto et al. [20] paper try to solve the problem of variability in multi-tenant application environment using Product Line Architecture (PLA)-based approach consist of a cardinality-based variability models of the Common Variability Language (CVL) to model each tenant as a cloneable feature. Benavides, Galindo [17] obtains in their paper the usage of variation points to identify configuration process out of variation of business processes in a specific domain. The papers of Asadi, Mohabbati at el. [16] and Mani, Helfert at el. [15] show the usage of the software product line (SPL) technique and the Model Driven Architecture(MDA) in the application architecture. Wehling, Wille et al. [18] proposing the usage of 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 [19] introduced the DSOPL-ADL (Dynamic Service-Oriented Product Lines- Architecture Description Language). The

DSOPL-ADL is structured and composed of four sections: structural, variability, context, and configuration.

In the end we found that each suggested approach may be able to obtain an idea of capturing the variability in one or two architecture levels but nevertheless, we didn't find any papers that propose a method to deal with variability in the all EAM levels.

Finally, we would like to point out some limitations of this research project. First, we declared only one RQ that drastically narrowed the SLR's scope. Still, many papers had to be reviewed that deal with the overall area of variability and EAM. In addition, the inclusion criteria were carefully chosen to identify papers that treat topics relevant for the RQ. Also, only the impact of variability on EAs was studied.

6. Summary and Future Work

The work in this paper focused on the research question "What is the state of research on managing variability in enterprise architectures?". As discussed in section 5, there is a substantial amount of research, but the existing work focuses on treating variability on one architecture level (e.g., variability in the business architecture) with a few approaches also tackling the effects of this variability on one of the neighboring levels. However, how to control variability in business architecture and at the same time capture the induced variability of applications and variability in the data architecture has not been 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 introduction of artificial intelligence, which motivates further research in this field.

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. Our view is that feature modeling would be a promising way to integrate these different streams of research.

References

1. 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)
2. Ahlemann, F., Stettiner, E., Messerschmidt, M., Legner, C.: Strategic enterprise architecture management: challenges, best practices, and future developments. Springer Science & Business Media (2012)
3. Josey, A.: TOGAF\textregistered version 9.1-A pocket guide. Van Haren (2016)
4. Barth, B., Butler, G., Czarnecki, K., Eisenecker, U.: Generative programming. In: Object-Oriented Technology: ECOOP 2001 Workshop Reader: ECOOP 2001 Workshops, Panel, and Posters Budapest, Hungary, June 18-22, 2001 Proceedings 15, pp. 135–149 (2002)
5. Kang, K.C., Cohen, S.G., Hess, J.A., Novak, W.E., Peterson, A.S.: Feature-oriented domain analysis (FODA) feasibility study (1990)
6. Thörn, C., Sandkuhl, K.: Feature modeling: managing variability in complex systems. Complex Systems in Knowledge-based Environments: Theory, Models and Applications, vol. , 129–162 (2009)
7. Li, L., Zheng, Y., Yang, M., Leng, J., Cheng, Z., Xie, Y., Jiang, P., Ma, Y.: A survey of feature modeling methods: Historical evolution and new development. Robotics and Computer-Integrated Manufacturing, vol. 61, 101851 (2020)
8. 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
9. 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

10. 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
11. 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
12. Wehling, K., Wille, D., Seidl, C., Schaefer, I.: Decision Support for Reducing Unnecessary IT Complexity of Application Architectures. In: 2017 IEEE International Conference on Software Architecture Workshops (ICSAW), pp. 161–168. IEEE (2017). doi: 10.1109/ICSAW.2017.47
13. Cohen, M., Acher, M., Fuentes, L., Schall, D., Bosch, J., Capilla, R., Bagheri, E., Xiong, Y., Troya, J., Ruiz-Cortés, A., Benavides, D. (eds.): Proceedings of the 21st International Systems and Software Product Line Conference - Volume A, vol. . ACM, New York, NY, USA (2017). doi: 10.1145/3106195
14. 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
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. Wehling, K., Wille, D., Seidl, C., Schaefer, I.: Automated recommendations for reducing unnecessary variability of technology architectures. In: Simmonds, J., Walkingshaw, E. (eds.) Proceedings of the 8th ACM SIGPLAN International Workshop on Feature-Oriented Software Development, pp. 1–10. ACM, New York, NY, USA (2017). doi: 10.1145/3141848.3141849
19. 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
20. Horcas, J.-M., Pinto, M., Fuentes, L.: Product Line Architecture for Automatic Evolution of Multi-Tenant Applications. In: 2016 IEEE 20th International Enterprise Distributed Object Computing Conference (EDOC), pp. 1–10. IEEE (2016). doi: 10.1109/EDOC.2016.7579384