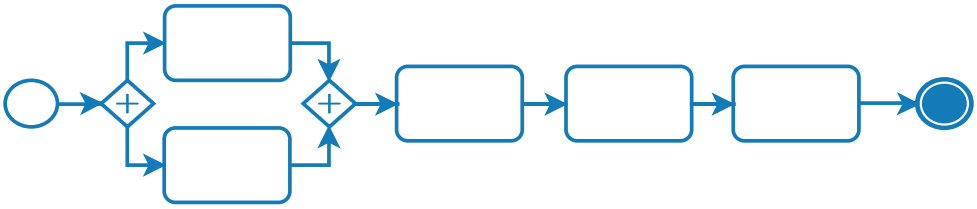
SE | RWTH AACHEN UNIVERSITY

Simon Hacks

Improving the Quality of Enterprise Architecture Models

-Processes and Techniques-



Aachener Informatik-Berichte,
Software Engineering

Band 43

Hrsg: Prof. Dr. rer. nat. Bernhard Rumpe
Prof. Dr. rer. nat. Horst Lichter

Improving the Quality of Enterprise Architecture Models -Processes and Techniques-

Von der Fakultät für Mathematik, Informatik und Naturwissenschaften der RWTH Aachen University zur Erlangung des akademischen Grades eines Doktors der Naturwissenschaften genehmigte Dissertation

vorgelegt von

Simon Hacks, M.Sc.
aus Tönisvorst

Berichter: Universitätsprofessor Dr. rer. nat. Horst Lichter
Professor Pontus Johnson, Ph.D.

Tag der mündlichen Prüfung: 12.09.2019

Aachener Informatik-Berichte, Software Engineering

herausgegeben von
Prof. Dr. rer. nat. Bernhard Rumpe
Software Engineering
RWTH Aachen University

Band 43

Simon Hacks
RWTH Aachen University

Improving the Quality of Enterprise Architecture Models

-Processes and Techniques-

Shaker Verlag
Düren 2019

Bibliographic information published by the Deutsche Nationalbibliothek
The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data are available in the Internet at http://dnb.d-nb.de.

Zugl.: D 82 (Diss. RWTH Aachen University, 2019)

Printed in Germany.

ISBN 978-3-8440-6977-8
ISSN 1869-9170

Shaker Verlag GmbH • Am Langen Graben 15a • 52353 Düren
Phone: 0049/2421/99011-0 • Telefax: 0049/2421/99011-9
Internet: www.shaker.de • e-mail: info@shaker.de

Kurzfassung

Information Technology (IT) durchdringt Organisationen immer mehr und wird immer wichtiger für deren Geschäftsmodelle. Dabei hat sich die IT von einer rein unterstützenden Rolle, hin zu einer wichtigen strategischen Säule in vielen Organisationen entwickelt. Umso wichtiger ist es, dass die IT die Anforderungen der Organisation umsetzt. Ansätze, die dies realisieren sollen, werden in der Forschung häufig unter dem Begriff "business-IT-alignment" subsummiert. Ein Instrument, mit dem ein "business-IT-alignment" erreicht werden soll, ist die Unternehmensarchitektur (engl.: "Enterprise Architecture"). Unternehmensarchitekturen erlauben eine holistische Perspektive auf die Struktur der Organisation und bieten eine Sammlung von Techniken, um die Entwicklung der Organisation zu einem gewünschten Zielzustand zu begleiten und zu lenken.

Ein zentrales Artefakt der Unternehmensarchitektur ist das Unternehmensarchitekturmodel. Es abstrahiert die Elemente und deren Beziehungen der Organisation auf ein verständliches und steuerbares Maß. Dabei werden üblicherweise von Geschäftsprozessen über Applikationen bis zu Hardwarekomponenten, auch Datenmodelle und Kundenbeziehungen modelliert. Basierend auf den im Unternehmensarchitekturmodel gespeicherten Informationen, trifft das Management der Organisation wichtige Entscheidungen, betreffend der zukünftigen Ausrichtung. Aber auch auf der operationalen Ebene, kann das Modell wichtige Informationen liefern, zum Beispiel welche Applikation in welchem Geschäftsumfeld eingesetzt wird und dabei mit anderen Applikationen Daten austauscht.

Um sinnvolle Entscheidungen aus dem Unternehmensarchitekturmodel ableiten zu können, ist deren Qualität von entscheidender Bedeutung. Daher beschäftigt sich diese Arbeit damit verschiedene Prozesse und Techniken zu entwickeln, die die Qualität des Unternehmensarchitekturmodells sicherstellen.

Zuerst wird ein Prozess präsentiert, der die Qualität des Unternehmensarchitekturmodels sicherstellen sollen, in dem die Modelpflege als eine kontinuierliche Evolution verstanden wird. Dafür werden verschiedene Schritte definiert, die in einem solchen Prozess durchgeführt werden müssen. Dieser Prozess dient im Folgenden als Grundlage für eine Continuous Delivery Pipeline, die dabei helfen soll möglichst viele dieser Schritte zu automatisieren. Im Anschluss daran wird ein Ansatz vorgestellt, der ermöglicht auch widersprüchliche Angabe in Unternehmensarchitekturmodellen zu speichern.

Neben den Prozessen, die die Qualität von Unternehmensarchitekturmodellen verbessern sollen, enthält diese Arbeit auch verschiedene Techniken zu diesem Zweck. Um die Qualität des Models zu verbessern, wird allerdings eine Methode benötigt, um die Qualität zu bewerten, die ebenfalls innerhalb dieser Arbeit eingeführt wird. Im Anschluss, werden Techniken des Maschinenlernens genutzt, um den Modellierenden dabei zu unterstützen, existierende Elemente des Models wiederzuverwenden. Zusätzlich wird die Performanz verschiedener Algorithmen verglichen, um den besten in einer bestimmten Situation bestimmen zu können. Darüber hinaus wird eine Methode vorgestellt, mit der unnötige Elemente im Model identifiziert werden können.

Abstract

Information technology (IT) pervades organizations more and more and becomes increasingly important for their business models. It has evolved from a purely supportive role to an important strategic pillar in many organizations. Even more, it is important that IT is aligned to the needs of the organization. Approaches that realize this are often subsumed under the term "business-IT-alignment". One instrument for achieving business-IT-alignment is Enterprise Architecture (EA). EAs provide a holistic perspective on the structure of the organization and provide a set of techniques to guide and steer the evolution of the organization to a desired goal state.

A key artifact of EA is the EA model. It abstracts the elements and their relationships to an understandable and manageable measure. Usually enterprise architects model business processes, applications, hardware components, data models and customer relationships. Based on the information stored in the EA model, the organization's management makes important decisions regarding future focus. Contrary, also on the operational level, the model can provide important information, for example which application is used in which business environment and exchanges data with other applications.

In order to be able to derive meaningful decisions from the EA model, their quality is of crucial importance. Therefore, this work elaborates on developing different processes and techniques that ensure the quality of the EA model.

First, we present a process to ensure the quality of the EA model, where model maintenance is understood as a continuous evolution. For this purpose, we define different steps, which have to be considered in such a process. This process will serve as foundation for a continuous delivery pipeline that will help automate as many of these steps as possible. Next, we present an approach that allows storing contrary information in EA models.

In addition to the aforementioned processes, we developed also several techniques to improve the quality of EA models. However, for improvement, we need a method to evaluate the quality, which we also introduce within this work. Subsequently, we facilitate machine-learning techniques to support the modeler reuse existing elements of the model. In addition, we compare the performance of different algorithms to determine the best on in a certain situation. Additionally, we present a method to identify unnecessary elements in the model.

Contents

Part I.

Foundations

Chapter 1.

Preface

1.1. Introduction

Information Technology (IT) pervades organizations more and more and becomes further important [170, 236]. Oliner and Sichel [170] have shown that IT accelerated the overall labour productivity growth beginning with the mid 1990's. Moreover, they expected that the influence of IT will further grow, which was proved by the past, as IT became ubiquitous [236]. Additionally, business models are changing dramatically, even driven by IT [37].

Accordingly, different authors [57, 89, 168] classify the operation mode of organization's IT with respect to its influence on the organization. Earl [57] defines four classes building up on each other: support mode, factory mode, turnaround mode, and strategic mode. Nolan and McFarlane [168] reorganize this classification by placing those classes into their strategic grid, which describes the need of the organization for reliable IT and new IT. A slightly different view on organization's IT is given by Hanschke [89]. She illustrates IT's percipience in four tiers building on top of each other: IT as cost driver, IT as financial asset, IT as business partner, and IT as enabler. All those classifications show that IT has developed from a supporting department to a important driver of changes within modern organizations.

Additionally to the rising importance of IT for the business, the integration of business requirements with implemented IT functionality becomes more important. Consequently, the business-IT alignment has been most important for CIOs of different industries and different company sizes for a long time [143, 144]. To promote the business-IT alignment, more and more Information System (IS) change and development projects focus the realization of technical solutions for local business needs. Enterprise Architecture (EA) is a widely accepted discipline to guide local IS endeavors through a holistic view on the fundamental structures, design, and evolution principles of the overall organization [31]. EA eases the alignment of IS projects with enterprise-wide objectives, which leads to reduced complexities as well as integration efforts in the overall corporate IS landscape [14, 183].

Since it beginnings in the 1980s [116], EA has developed to an established discipline in industry and research [193, 210]. A widely accepted definition of the term architecture [194] is given in the ISO 42010:2011 [99]:

Definition 1.1 (Architecture) *Architecture describes the fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution.*

As this definition implies, the EA model, comprised by the elements and relationships of the organization, is one central artifact of EA. The model provides a holistic view on the organization and, therefore, eases the value creation for EA's stakeholder [165].

Several EA modeling languages like ArchiMate exist, that are used in practice [127, 225]. The benefits of EA highly depend on the model quality [165]. So far, only a few researchers have published work that elaborates on the specification of EA quality attributes, but do not develop means to ensure the quality of EA models. Thus, we address with this work the goal to provide a set of tools that enable EA practitioners to assess and improve the quality of their EA models.

1.2. Research Questions

To motivate our research questions, we shed light on the maintenance process of EA models similar to a customer journey [133] in the domain of marketing. Therefore, we imagine a company with an established EA initiative. The EA department consists of ten enterprise architects who implement the IT-strategy by publishing guidelines, consulting the management, and steering projects towards the designated direction. Additionally, they provide actual information on the EA to their stakeholder and prepare decision documents for the higher management. Besides the enterprise architects, the company employs solution architects. A solution architect is an expert for a certain sub-domain and takes responsibility for all projects in this sub-domain. Therefore, their main tasks are to design a solution architecture which conforms to the big picture and to ensure that the projects adhere to the guidelines.

Our customer journey starts with Mary having her first day in the EA department. After having her first coffee with the new colleagues, Peter shows her the EA repository containing the current EA model. As she should get common with the model and its elements, she receives her first task to satisfy a request for information by a manager. Therefore, she asks herself if there are any differences between stakeholders on the expectation towards the EA. More concrete she wonders:

RQ 1 *What are the differences of stakeholder concerns on EA?*

As soon as she grasped the differences between the different stakeholders, she prepares a suitable report and sends it to the manager. Afterwards, she has a little free time as all colleagues are in meetings. Therefore, she discovers the EA model further and, soon, she thinks about the possibilities to do analysis on the model and searches for research elaborating on this. Unfortunately, there is already a lot of work around and she is not sure which approaches are suitable for her model and her purposes. She thinks that a classification scheme would be very helpful to find the best suiting approach and asks herself:

RQ 2 *How is the EA analysis research classified according to its analysis concerns, techniques, and modeling languages?*

She finds a lot of interesting articles and plans to apply some of the approaches. However, before she can go deeper into the topic, Peter comes back and gives her an introduction into the maintenance of the EA model. Basically, there is the central model which is maintained by

the enterprise architects. Erratically, they provide an export of the model which is provided to EA's stakeholders. One of those stakeholders are projects which facilitate the provided export to model their changes on the EA. There are a plenty of different projects in every shape, size, and duration. This results in something Peter calls a distributed EA evolution. Nonetheless, Mary perceives this process as unstructured and the way the results of the projects get back into the central model remains unclear to her. Consequently, she wonders:

RQ 3 *How does an effective architecture roundtrip process looks like supporting the distributed EA evolution?*

Furthermore, everything Peter tells her, reminds her on problems she knows from the domain on software engineering. During her studies, she was part in a lab where she built with other students an application. Every student had his own copy of the central code and their changes had to be integrated into the central code again. One means, which she perceived as valuable, were continuous delivery pipelines as they shorten feedback circles and help to uncover flaws early. Therefore, she likes to answer the question:

RQ 4 *How can continuous delivery help to overcome the challenges of manual EA model's maintenance?*

Whilst she is still thinking about this question, she recognizes a discussion of her colleagues. Steven and Cathy recently returned from a decision board and recapitulate the happenings. A project presented two different scenarios how the EA could evolve in the next step. There were a lot of pro's and con's so that the board members did not come to a conclusion. However, Steven and Cathy like to preserve both scenarios for a future analysis. As Mary is new and, therefore, open-minded, they ask her if she has an idea:

RQ 5 *How can evolutionary EA scenarios provided with uncertainty information be presented in an EA model?*

She promises to think about this and will find a solution for this issue. But, it is lunchtime and all colleagues got together to a restaurant. On their way, Peter asks her if she feels comfortable with her new colleagues and what her first impression is. Mary is satisfied so far. Though, her gut feeling tells her that some parts of the EA model could be improved. Peter agrees on her impression, but he is also unsure how to identify the weak parts within the model. He thinks that a framework would be useful to measure the quality. Accordingly, Mary wonders:

RQ 6 *What aspects need to be included in a framework for assessing the quality of EA models?*

Peter likes Mary's question and they decide to elaborate together on this soon. Meanwhile, they reached the restaurant. After a seating at a table, the waiter brings the menus in German and English as the enterprise architects are bilingual. This reminds Cathy on a situation where she had to import the new models of projects into the central EA model. Two projects modeled the same elements, but named them differently as they speak different languages. It took her quite a long time to recognize that the projects described the same things and to avoid unintentional redundancies. As Mary's university time is not that long ago, Cathy asks her:

RQ 7 *How can machine learning techniques help to avoid redundancies in EA models?*

Mary already has some ideas and plans to investigate if those techniques might be applicable within her company. But first, she enjoys her meal. Back in the office, Mary takes a closer look at the EA model with the aspect of redundancies in mind. Suddenly, she observes that the redundancies are not solely related to the model elements themselves, but also to the artifacts they present in reality. For example, several applications fulfill the same functionality and probably not every application is necessary. This raises the question:

RQ 8 *How can EA models serve as input to optimize the entire EA?*

Meanwhile, it has become evening and Mary's first day in the EA department comes to an end. During her first day, she found a lot of open questions which should be answered in the next time.

1.3. Structure of the Dissertation

This dissertation is structured as follows: After, we have presented the research questions in section 1.2, we will next present the related work in section 2. More concrete, we present existing research on model evolution in general and EA model evolution in particular. Additionally, we give insights into different approaches of EA evolution and work elaborating on the quality of EA and EA models.

Part II deals with EA in research and in practice. On the one hand, we investigate the different concerns of stakeholders towards EA, especially, how the concerns are related to the hierarchical position of the stakeholder in the organization. On the other hand, we develop a taxonomy to classify EA analysis research based on a Systematic Literature Review (SLR).

We focus on different processes to improve EA models in part III. First, we sketch the issue of a distributed EA evolution and develop a process to overcome this issue. In section 7.2.1, we extend this process and unite it with the idea of continuous delivery. This leads to a pipeline implementation which shortens the feedback cycles and, therefore, eases the improvement of the EA model. Last, we present how contradictory information can be preserved in an EA model.

In contrast, part IV focuses on different methods to improve EA models. Section 8 introduces a framework which enables EA practitioners to assess the quality of their EA model. As this relies more on the improvement of already existing EA models, section 9 provides methods to support the designer of EA models within the creation phase. A further facet is added in section 10 by not only solely considering the EA model, but also to facilitate it as input to optimize the entire EA.

In part V, we present the results of a quantitative evaluation on different EA maintenance processes, before we conclude our work by a summary and an outlook on future work.

As parts of this work got published beforehand, we like to clarify in Table 1.1 which parts of the published papers were produced by the author of this work.

Table 1.1.: Contribution to Pre-Published Works.

Part of this work	Publication	Idea	Doing	Writing	Supervision
Section 3	[79]	◕	◕	◑	-
Section 4	[23]	◑	◔	◔	◔
Section 5	[84]	●	●	◕	-
Section 6	[86]	◑	◑	◑	-
Section 7	[82]	●	◕	◕	-
Section 8	[227, 87]	◑	◑	◑	-
Section 9.1	[32]	◕	◔	◔	◕
Section 9.2	[185]	◕	◔	◔	◕
Section 10	[81, 83]	●	●	◕	-
Section 11	[85]	●	●	◕	-

1.4. Recurring Parts Within This Work

In our research, we facilitated certain parts several times. To ease the read of our work and not to repeat ourselves, we will present the effected parts following. In concrete, we will present, first, the case study environment that is utilized in sections 3 and 8. Second, a fictitious EA model is used in sections 9.2 and 6. Last, the Design Science Research (DSR) method is presented, which we utilized in Sections 5, 6, 7, and 9.1 to develop the required artifacts.

1.4.1. A Case Study Environment

Our research was mainly conducted in cooperation with an industrial partner. Therefore, some of our artifacts were partly or completely evaluated within the context of our cooperation partner. To not repeat ourselves, we will following present the case in detail. Additional information is given if necessary at the certain point of this work.

The case organization is one of the leading insurance providers in the German-speaking market. About 30,000 employees and 16,000 associated agents count toward the workforce of the company. Furthermore, the case organization has several subsidiaries: One of them is the internal IT service provider, in which we conducted our cases.

The IT service provider employs around 1,400 employees. These are responsible for operations and development of technological solutions for the whole organization, including all of its subsidiaries. The IT provider began establishing EA initiatives in 2008 and currently hosts two EA units: The first unit, “architecture management” is responsible for application development. The second unit, “infrastructure architecture management”, is responsible for infrastructure management (e.g., operations of servers). As regulatory instances, both units are responsible for all EA-related questions, ranging from EA development to EA implementation and EA maintenance.

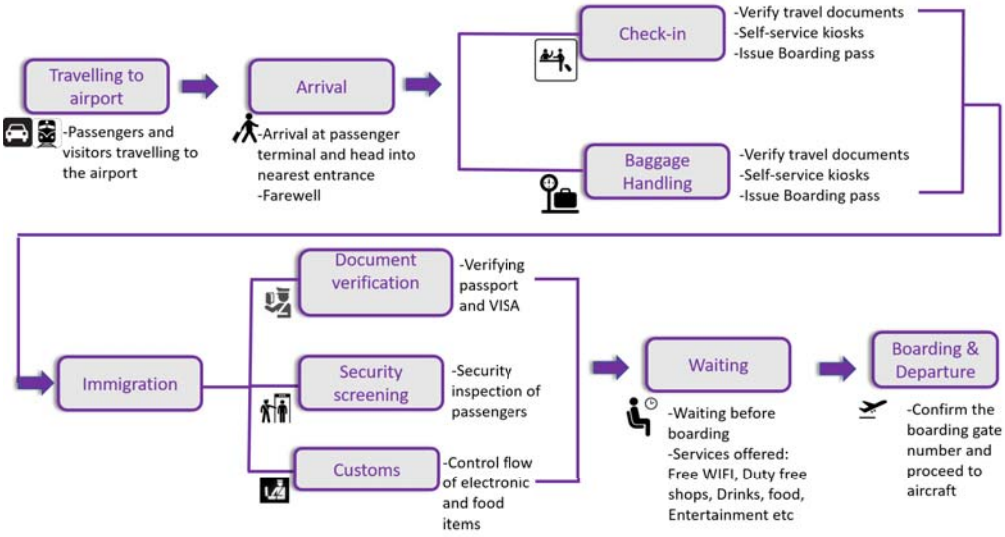


Figure 1.2.: Global Organization Structure of an Airport Departure System.

1.4.2. An Exemplary EA Model

The beforehand presented case study environment (cf. section 1.4.1) was not always suitable for our purposes. This is grounded in the fact that the provided model was to complex to capture it entirely. But sometimes it is essential to know how the optimal solution for a particular scenario looks like, e.g., to evaluate different algorithms. Therefore, we sketched an airport departure system for EA model analyzing purposes. This case study strategy provides a richer understanding of the context for the people and the researcher involved. We illustrate a scenario of airport departure system which depicts the functionality of the passengers before boarding to an aircraft. Figure 1.2 shows the overall organization structure and Figures 1.3, 1.4, and 1.5 an EA model representing each layer of the immigration process in ArchiMate notation.

To guide future changes in their business and information technology, we transformed our fictitious example of an airport departure system into an EA model based on ArchiMate 3.0.1. This case study is used as an example throughout different parts to evaluate and compare different graph analytic techniques to analyses the EA model in a better way and to select the best-performing algorithm (cf. Section 9.2) or to serve as means to demonstrate the application of our designed artifact (cf. Section 6).

The business layer depicts the business services offered to the customers, which is mainly used to build business architecture [78]. In this example, the active entities of the business layer are airline employees, passengers, and security guards. There are four functions that are provided by boarding and departure process. The function boarding to airplane is internally divided into three sub-processes.

The application layer includes for example the airline administration support, which is respon-

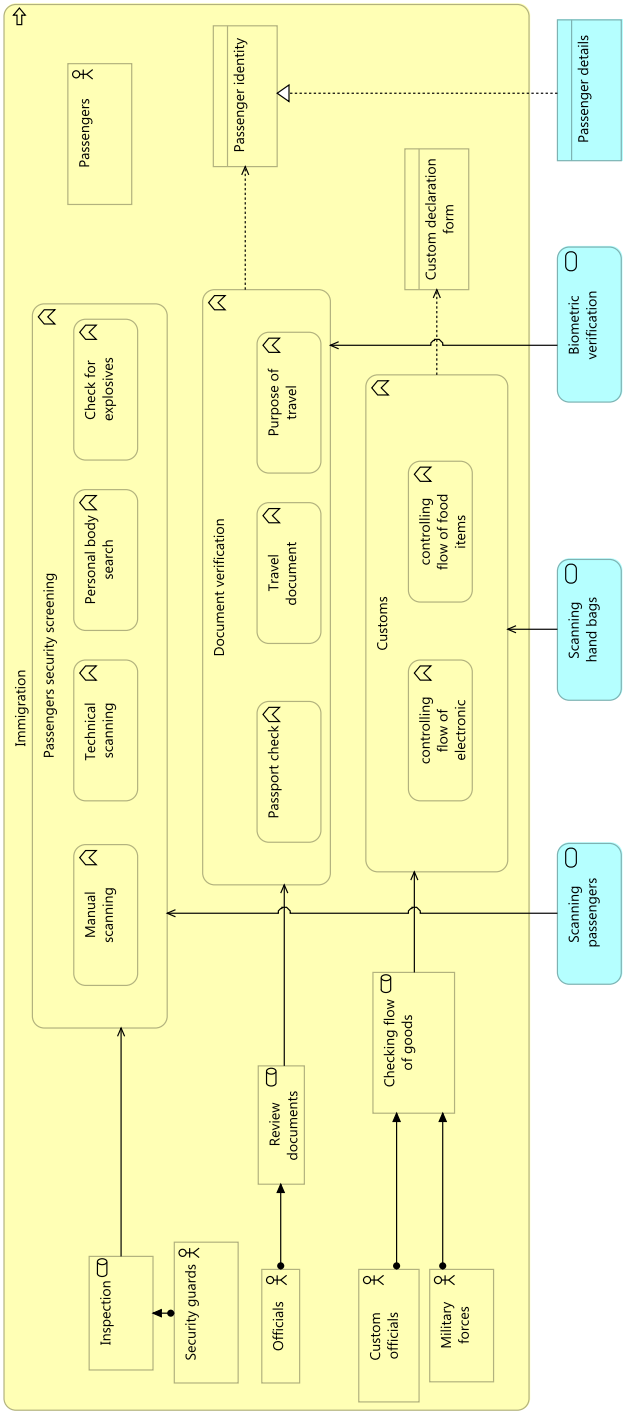


Figure 1.3.: An EA Model Representing the Business Layer of the Immigration Process at an Airport.

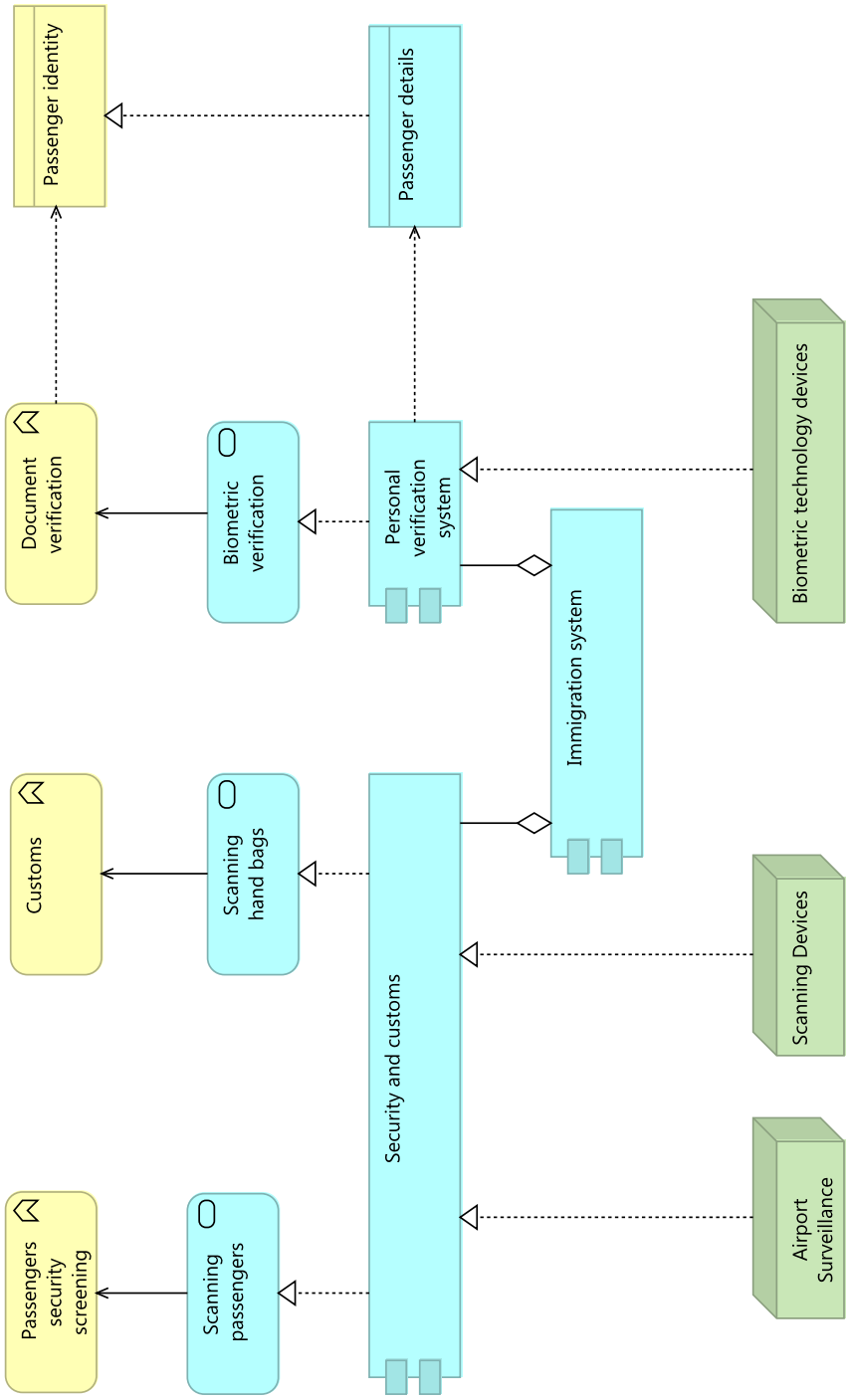


Figure 1.4.: An EA Model Representing the Application Layer of the Immigration Process at an Airport.

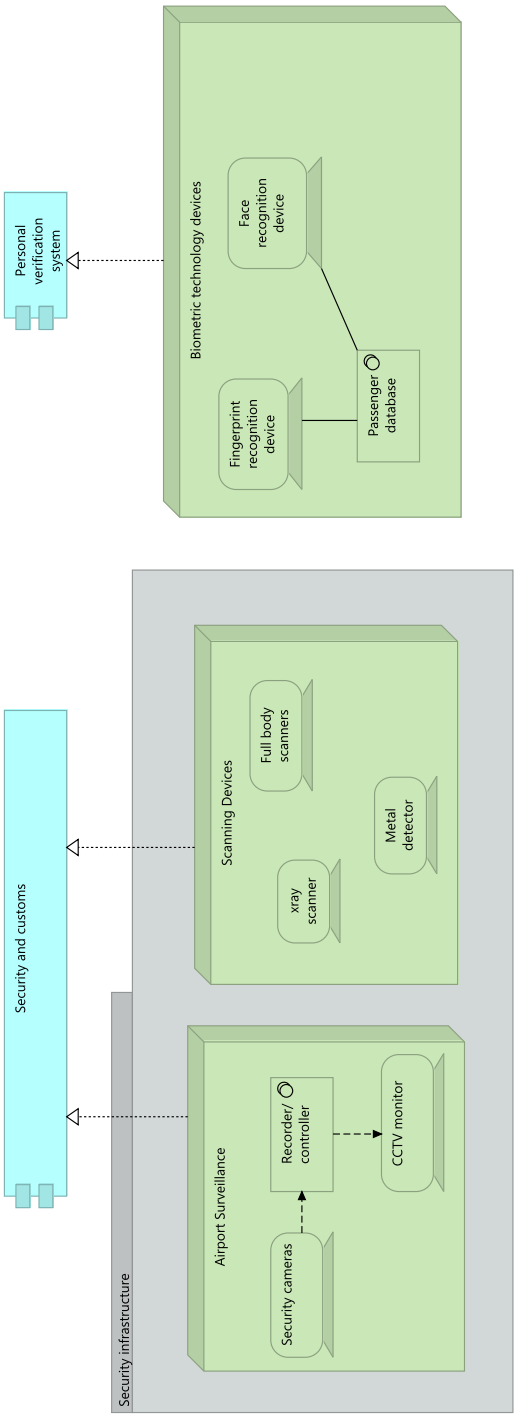


Figure 1.5.: An EA Model Representing the Technology Layer of the Immigration Process at an Airport.

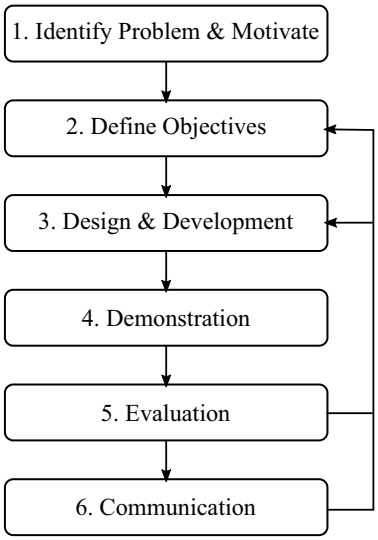


Figure 1.6.: Design Science Research.

sible for handling check-in process, and the boarding control, which handles boarding process. Furthermore, two application components collaborate in boarding and departure control system, i.e., the airline administration component and the boarding control to provide application level services like identifying boarding pass, security and navigation control support.

The technology layer offers several components to the application layer. E.g., there is a bar code system offering the needed means to validate the bar codes of the boarding tickets and a Global Positioning System (GPS) navigation system guiding the bus drivers to the right plane on the airfield.

1.4.3. Design Science Research

To tackle the different issues within this work, we opt several times (cf. Sections 5, 6, 7, and 9.1) for Design Science Research (DSR), which is a widespread means to develop artifacts. DSR offers a systematic structure (cf. Figure 1.6) for developing artifacts, such as constructs, models, methods, or instantiations [93]. As our research questions indicate the development of means, the application of DSR is appropriate. We stick to the approach of Peffers et al. [178], which is split up into six single steps and two possible feedback loops:

Activity 1: Identify Problem & Motivate A problem discovery activity is mandatory to identify the research domain and potential subjects to be addressed. The literature is reviewed to gain knowledge regarding the current state of problem for problem statement formation, which is then used to develop a viable artifact that provides a solution to the problem.

Activity 2: Define Objectives The research objectives and goals are formalized to solve the identified problem and, consequently, provide the intended direction for the research. The objectives can be either quantitative or qualitative [178].

Activity 3: Design & Development In this activity the artifact is created. Artifacts could be constructs, models, methods, instantiations, new properties of technical, social, or informational resources. Further, this activity includes determining the desired functionality and its architecture.

Activity 4: Demonstration It is important to demonstrate that the artifact can solve one or more instances of the problem identified before. This could involve its use in experimentation, simulation, case study, proof, or other appropriate activities.

Activity 5: Evaluation After demonstrating the capability of the artifact that it can solve an instance of the problem, the next step is to observe and measure how well the artifact solves the problem. This includes comparing the objectives of the solution to the observed results. Depending on the nature of the problem and the artifact, the evaluation could be manifold. For example, it could include a comparison of the artifact's functionality with the objectives from activity two, quantitative performance measures, the results of surveys, client feedback, or simulations. Conceptually, an evaluation could include any appropriate empirical evidence or logical proof. At the end of the evaluation, the researchers can decide whether to iterate back to step three to improve the effectiveness of the artifact or to continue.

Activity 6: Communication Towards the end of DSR, it is mandatory to document the problems, solution, objectives, description related to the developed artifact for communication with the relevant audience. The outcome of research is presented in the form of research papers, presentations, and this work. Additionally, insights are given to provide a direction for future research.

Chapter 2.

Related Work

Based on the presented problem description and the distributed EA evolution scenario that we want to support, two categories of scientific research can be identified. The one set of research counts towards information maintenance in EA, but does not necessarily elaborate on the EA model like we do. The other set concerns questions regarding models, but not in the domain of EA. Additionally, important parts of our work are related to the quality of EA models. Therefore, we present also related work to this topic.

2.1. Model Evolution

Focusing on the model facet of our work, one core challenge is to integrate different versions and scenarios of sub-models, i.e., the change sets, into one central EA model. As this problem has been researched for some years in the context of model driven software development, we can apply or adopt existing techniques to merge and integrate models. For instance, Gorek and Kelter [76] present an approach for matching sub-models in an early project phase. This approach can be extended and adjusted to distributed developed models (the change sets), as these change sets usually affect only a subset of the EA model. Hence, these sub-models have to be reintegrated into the central EA model.

Another problem to be solved is rooted in the different projects' time spans leading to scenarios where projects are using outdated EA information. To handle this problem, we investigate the approach published by Wenzel et al. [238], which traces model elements through different versions of models, and the one published by Schmidt et al. [202], that applies history-based merging of models.

Finally, the most general representation of an EA is a structured text file. Therefore, respective text merging techniques are of interest as well. According to Mens [149], two-way merging aims to figure out the differences between two documents, while three-way merging additionally includes the document from which both documents evolved. In the context of EA evolution, three-way merging might be promising (see Lindholm [137]), since the projects should use the EA as central input to develop their solution scenarios and change sets.

The beforehand presented approaches have the assumption in common that the models are exchanged on a traditional, file-based way. On the contrary, some approaches have proposed relying on alternative technologies. De Lucia et al. [50] developed a management system which uses dependency links to support traceability in context-aware change management. The system can be customized to handle different kinds of software artifacts, e.g., models being serialized

in the XML Metadata Interchange (XMI) format [24, 140]. Another approach which is based on the XMI standard is Odyssey-VCS [156, 171] focusing on software configuration management. Oda and Saeki [169] propose to use generative techniques in order to manage various types of visual models.

2.2. Enterprise Architecture Model Evolution

Zimmermann develops in his dissertation [251] a reference process model to guide the Business-IT-Management (BIM). IT governance, strategic management, multi-project management, and EA are concepts he reuses. Within the process of project mentoring of the EA, he outlines the added value of providing up-to-date information regarding the EA model to the projects [251, pp. 176-178]. Moreover, he highlights that the additional effort on keeping the EA up-to-date at the end of the project is worth it [251, pp. 176-178]. Unfortunately, Zimmermann neither illuminates how this update should work nor which problems may arise.

In contrast to Zimmermann [251], Wittenburg [244] appears in outlines an architecture roundtrip. Thus, he depicts the interchange between EA and projects still in a sequential way. But, there exist several interchanges, which can be understood as small roundtrips. Unfortunately, Wittenburg focuses on the representation of the application landscape and does not zoom into the issues we are facing.

Next to a roundtrip is the work of Moser et al. [155]. They model an architecture cycle which has a continuous interchange with a project cycle. They propose process patterns which deal with the elicitation of EA data and their quality assurance. Certainly, the authors stay on a management level and do not give advises how to overcome with issues on the model level.

Since, change sets evolve during the project's time span, the uncertainty regarding completeness and correctness of the models can be handled by the approach of Johnson et al. [104]. They observe a network and use a Bayesian network to predict the likeliest representation of the EAs technology layer. An extension of this approach could allow representing the different solution scenarios and different evolution steps of the EA. Furthermore, similarity measures applied to graphs could be valuable to identify elements which were introduced in a project's change set but already exist in the EA itself. For instance, Jeh and Widom [103] published SimRank which can be used to identify similar objects depending on their neighborhood.

EA is used in large organizations and information that is used within the EA is often owned by different departments. This makes it hard for a central EA team to gather all information and keep it up-to-date. Fischer et al. propose a federated approach for the maintenance of EA models [63]. The main idea is that the data is kept within specialized architectures and linked to a central EA repository. Because this is an organizational as well as technical challenge, the authors propose a maintenance process that involves different roles.

Farwick et al. elaborate in their work [62] on change events for EA models. Therefore, they identify, first, a list of EA neighboring processes and related tools which can cause EA change events. Second, they determine events caused by those processes and categorize them to different facets like which EA layer is impacted or the as-is or to-be state of the EA model is affected. Last, they propose a process to incorporate the results of the change events into the EA model.

Other inputs are presented, e.g., by Buschle et al. [39], who facilitate an Enterprise Service

Bus (ESB) to extract EA model elements for ArchiMate [224], CySeMol [96, 211], and planningIT. In contrast, Holm et al. [95] concentrate more on technically observable components as they map the output of a network scanner to ArchiMate. An extension of this work is presented by Johnson et al. [104], who incorporate uncertainty into the mapping as they observe a network and use a Bayesian network to predict the likeliest representation of EA model's technology layer. Landthaler et al. [124] conduct a SLR to identify all automated application landscape documentation approaches. Additionally, they propose a new Machine Learning (ML)-based technique. Kleehaus et al. [115] propose further different techniques to monitor business, application, and technology layer.

The work of Välja et al. [229, 228] focuses on uniting different information from contradictory sources. Hence, they try to estimate the trustworthiness of the sources by facilitating techniques from the human-computer interaction [175] and data fusion [135] domain. Therefore, they build their means upon a model of information processing automation. This model is comprised of the phases of data acquisition, data analysis, decision and action selection, and implementation. In contrast to Välja et al., Kirschner and Roth [113] rely on a human component to solve arising conflicts from different sources. To get to the point where conflicts need to be solved, the authors first define a meta-meta-model for EA model repositories to loosely couple all incoming references of federated tools to the EA meta-model.

2.3. Enterprise Architecture Evolution

EA related research did not only elaborate solely on the technical aspects of EA model maintenance. For example, Khosroshahi et al. [112] investigated the social factors influencing the success of federated EA model maintenance. They structure their research along the socio-technical system theory framework [33], especially in the areas of structure, people, technology, and task. Their results show that organizations prefer a strong business involvement, a lean role allocation, a standardized EA terminology is necessary, and an adequate tool support is needed.

A slightly different point of view is taken by Hauder et al. [92] as they focused on the challenges of a federated EA model maintenance. Therefore, they took model transformations from three information sources into account, conducted a survey among 123 EA practitioners, and performed a literature study. The identified challenges were grouped along four categories, namely data, transformation, business and organization, and tooling.

EA (model) evolution is not barely a technical issue. Usually, the acceptance of EA and its artifacts within the organization is challenging. Therefore, Brosius et al. [35] study the EA assimilation and elaborate on the influence of institutional pressures. Those pressures make EA part of the organization's work life and, thus, contribute to EA's intended outcomes. Brosius et al. can empirically confirm the influence of the institutional pressures on EA assimilation and EA outcomes. Additionally, the engagement of local organizational stakeholders significantly mediate the relation between institutional pressures and EA assimilation.

Winter [241] broadens the view on EA and argues that the focus on the traditional EA players should be widened to those who are not directly related to the IT function. He stresses that local stakeholders' acceptance of EA depends on certain preconditions: First, actors need to be convinced that they benefit of EA. Second, actors need to understand that they can be

more efficient. Third, actors need to perceive EA as something that is strategically important for the organization. Last, actors need to perceive EA deployment as transparent, useful, and professional.

Wißotzki et al. [243] analyze systematically manifold EA literature sources as well as Enterprise Architecture Framework (EAF)s in order to derive a set of roles in EA. The results were validated by dint of an expert interview. Their work presents a generalized overview of EA roles and relates them to certain EA tasks and required competencies.

Zimmermann et al. [250] aim to support flexibility and agile transformations for both business domains and related enterprise systems through adaptation and evolution of digital EA. Therefore, they investigate the digital transformations of business and IT and integrates fundamental mappings between adaptable digital enterprise architectures and service-oriented information systems.

2.4. Quality of Enterprise Architecture Models

First, we want to clarify the term of EA model quality. Regarding to ISO/IEC 25010 quality "is the degree to which a product or system can be used by specific users to meet their needs to achieve specific goals with effectiveness, efficiency, freedom from risk and satisfaction in specific contexts of use" [100]. In the context of EA research Ylimäki states that "a high-quality EA conforms to the agreed and fully understood business requirements, fits for its purpose [...] and satisfies the key stakeholder groups' [...] expectations in a cost-effective way understanding both their current needs and future requirements" [248, p. 30]. In general, research regarding EA quality agrees that it is defined by the ability to meet the EA users' requirements [165, 136, 222, 179]. Most of the related work divides quality aspects of EA into the quality of EA products (e.g., EA models of current state or future vision), its related services, and EA processes (e.g., management tasks like EA planning) [165, 136].

Since model quality is not a research topic solely related to the EA discipline, we also relate to relevant work from other IS disciplines. In the context of software engineering, the ISO/IEC organization defines five characteristics to assess a system's quality and further divides them into sub-characteristics, namely, effectiveness, efficiency, satisfaction, freedom from risk, and context coverage [100]. A well-known framework for determining the success of information systems is the IS success model by DeLone and McLean, last updated in 2003 [52]. Lange et al. adapted this model to the EA domain and depict EA product quality, EA function setup quality, EA service delivery and EA cultural aspects as the drivers that influence EA's user satisfaction and the intention to use it [125, p. 4234].

At this point, we want to emphasize that the quality framework presented in this work (cf. Section 8) focuses on assessing the quality of EA models. The EA model is related to the prior explained concept of EA product quality. We thus understand EA model quality as the degree of fulfillment towards a set of attributes a model has to fulfill regarding its purpose and requirements defined by its stakeholders.

In the discipline of enterprise modeling there are approaches that discuss model quality in general, without focusing on a certain modeling structure. Becker et al. define six principles that have to be considered when assessing an enterprise model's quality (e.g., business process

model, entity-relationship diagram). These principles are namely the principle of validity, the principle of relevance, the principle of economic efficiency, the principle of clarity, the principle of systematic model structure and the principle of comparability [27]. Although, these principles do not provide explicit measures, they offer a thorough quality frame from different perspectives regarding a certain model type, e.g. an EA model. Sandkuhl et al. also apply them to evaluate the quality of their modeling language 4EM and further depict concrete quality attributes: unambiguity, flexibility and stability, homogeneity, completeness, scope, integration and simplicity [196]. Moreover, Pitschke provides a list of quality attributes for IS models and discusses them [179]. This list is mainly related to a prior work by Rauh and Stickel from the data modeling domain [186]. Pitschke expands the quality attributes and explains them in relation to business process models.

Although literature identifies a lack of research in the topic of assessing EA quality (cf. [165, 122, 216]), some articles investigate EA quality related issues. Ylimäki defines twelve critical success factors for EA and relates them to maturity levels [248]. In addition, Ravazi et al. propose a quantitative approach to assess the maintainability and interoperability of a certain EA [188]. Another generic approach for EA quality assessment is proposed by Lakhrouit et al. who define a generic evaluation concept model that can be used for several metrics to assess an EA's quality [122].

As explained above, general EA quality does not necessarily directly relate to the EA model as an artifact, but also EA management processes or other services. In the majority of the related work only general statements on EA quality are made. Still, some articles focus on the investigation of certain attributes that can be used to assess the EA model's quality. Lim et al. provide a list of EA quality attributes, which were derived from six established EA frameworks [136]. Likewise, Niemi et al. provide a further list of EA quality attributes based on 14 interviews in [165]. The authors relate identified attributes to EA product and EA service quality (cf. [125]). Further, Davoudi and Aliee define measures to assess EA maintainability [48]. Next to their generic model, Lakhrouit et al. also discuss EA quality indicators they deem reasonable [122].

After analyzing the relevant literature, it becomes obvious that a thorough quality of EA models includes both quantitative and qualitative metrics. Khayami suggests a list of qualitative characteristics of EA models [111]. In contrast, Spence and Mitchell develop quantitative metrics for defining an EA models syntactic and semantic correctness as well as its completeness using insights from set theory [216].

As discussed earlier, the common sense of all articles is that the EA model's quality has to be evaluated regarding its purpose and the stakeholders' concerns [27]. Hence, Lankhorst et al. emphasize that the establishment of the EA's purpose and its stakeholders is a vital aspect, each EA model should follow [127]. As can be seen in this section, numerous research relates to the topic of EA quality. Still, most of the identified articles do not provide a holistic approach how to assess the quality of an EA model [165].

Part II.

Enterprise Architecture in Research and Practice

Chapter 3.

Concerns of EA Stakeholders

As it is important to get an overview over the environment EA takes place, next, we introduce our work, which elaborates on EA in research and practice. In chapter 4, we structure existing EA analysis research along common characteristics to enable researches classifying their research and enabling interested persons capture relevant works faster. But first, we present the results of conducted interviews resulting in classification of different EA stakeholder concerns, which has been initially presented in [79].

Motivation Standardized architecture models and frameworks, such as The Open Group Architecture Framework (TOGAF) [223], entail valuable EA deliverables. However, these can hardly be applied without greater adoption. A major reason lies in the high degree of standardization and comprehensiveness, by which EA deliverables are aimed to become applied to a wide range of stakeholders and use cases. Yet, leaving EA deliverables untailored to the concerns of stakeholders jeopardizes guidance effects on both the IT and business side [118].

A few publications cope with the question of how to tailor EA deliverables to different stakeholder concerns (e.g., [118, 139, 107, 28]). Despite their discussion around EA deliverables, the existing literature still lacks a mere concrete intuition to stakeholder concerns on EA. More specifically, there is a research gap about the hierarchical differences of stakeholder concerns and their implications to EA. Recognizing these shortcomings in the existing literature, we formulate our research question:

RQ 1 *What are the differences of stakeholder concerns on EA?*

In order to identify stakeholder concerns on EA along hierarchical differences, we opt for a case study to investigate our research objective in a real-life context. Our findings demonstrate different concerns on EA deliverables and EA, depending on the hierarchical level of the respective interviewee. Using TOGAF exemplary as standardized and highly comprehensive EA approach, we discuss the implications of our findings for future research.

The research method follows a two-step process, starting with the data collection, followed by a scheme-guided classification according to Miles and Huberman [150] as well as Eisenhardt [59] for presenting and discussing the data. The information processed in this work is collected within the environment already presented in Section 1.4.1. Our final classifications scheme is comprised of five major dimensions and twelve facilitating characteristics (Table 3.2). We differentiate between three various types of stakeholders, which are sketched in Figure 3.1. For more details on the creation process of our classification scheme, we refer to Appendix A.1.

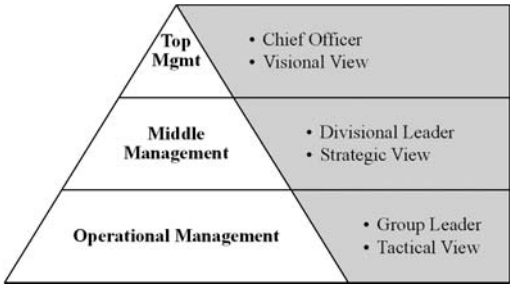


Figure 3.1.: EA's Different Groups of Stakeholder.

Table 3.2.: Stakeholder Concerns on EA.

Cluster	Dimension	Characteristic	Operational	Middle	Top
Deliverables	Type	Architectures	◑	◑	○
		Policies	○	○	○
	Quality	Actuality	○	○	○
		Simplicity	○	○	○
		Stability	○	○	○
	Abstraction Level	Low	●	◑	○
		High	◑	◑	◑
Organizational Anchoring	Context	Assertiveness	●	●	◑
		Integration between EA and other departments	◑	◑	◑
		Acceptance of other departments	●	●	◑
	Transparancy	EA deliverables	●	●	○
		EA function	●	●	●

Legend	rarely mentioned	○
	frequently mentioned	◑
	often mentioned	●

3.1. Stakeholder Concerns

In the following, we present the collected stakeholder concerns along the dimensions of the classification scheme. All concerns were classified by the percentage of stakeholders of each group reporting to them: Rarely (less than 33% of stakeholders), frequently (between 33% and 66%), and often (more than 66%).

3.1.1. Type

Architectures Interviewees from the operational and middle management level were similarly concerned with architectures (see Table 3.2). The most often reported deliverable was the as-is architecture. Thereby, concerns referred to infrastructure components on the technology layer as well as the application layer, i.e., what applications exist and how they are connected.

Members of the top management were less concerned than members of the operational and middle management. As a result of different use cases, interviewees of the top management mentioned as-is architectures solely as a means of steering.

Policies Policy concerns, discussed mainly in the context of project management and reporting activities, referred to complexity-related (e.g., "documents should be formulated in a simpler way.") and transparency-related (e.g., "the added value and the function of a policy should be clear.") aspects. A further concern, according to the interviewees, referred to guidelines (e.g., architecture principles): On the one hand, interviewees acknowledged the usefulness of architecture principles. On the other hand, usefulness of architecture principles appeared to be of limited value once generating too much effort for following them (e.g., "developers ignore the architecture principles if it is inconvenient."). Moreover, interviewees concerned missing assertiveness of the EA, thus, developers are not forced to comply with policies (e.g., "violation of the architecture principles will not be sanctioned.").

3.1.2. Quality

Actuality A rarely mentioned concern among all hierarchical levels of stakeholders was actuality. Interviewees described actuality as expectancy for the case of using the deliverables of EA (e.g., "... should be up-to-date"). Particularly, actuality has been emphasized for application layer (e.g., detailed information about applications, communication between applications) due to its frequency of usage as a basis for architecture design decisions (e.g., "the provided as-is application layer is not up-to-date. This is insufficient if we were about to use it.").

Simplicity Another mentioned characteristic of quality was simplicity. Participants stressed architecture principles to function as guidelines for application developers: Firstly, there should be a limited number of principles to enable affected persons to keep the overview. Secondly, principles should not be too complex (e.g., "a multi pages document comprises all information about the principle privacy and security."). Finally, deliverables should be easily accessible (e.g., simple search functionalities along all deliverables, no spread of information in different sources).

Stability Interviewees expected stability in the management of deliverables (i.e., design, maintenance, retirement). Stakeholders throughout all hierarchical levels differentiated two facets of this concern: First, names of individual deliverables had to be changed only on purpose. Second, policies should not be changed too often and not too fundamentally.

More generally, quality concerns were reported as important to avoid confusion (e.g., "if a certain product of EA is renamed, it will take some time to recognize whether the product is renamed or completely retired.") and to reduce unintentional effort (e.g., "changing policies leads to additional effort, because coaching is needed to internalize the changes.").

3.1.3. Abstraction Level

Low Concerns for a low abstraction were mentioned very divergently by the stakeholder levels. Apparently, this characteristic was most relevant for operational management stakeholders. These stakeholders were particularly concerned with concrete instructions and for operations usable information. In detail, coaching in projects was requested in order to learn how to build applications in line with existing policies. Additionally, detailed information on applications as well as specific information on business structures was concerned. Compared to operational management stakeholders, members from the middle and top management valued a low abstraction level less high.

High The need for a high abstraction level was reported throughout all stakeholder levels similarly. Concerns were related to the degree of details among steering decisions, which commonly necessitated a higher level of abstraction. Moreover, a high level of abstraction also favored better visualization of relevant information to be presented.

3.1.4. Context

Assertiveness Assertiveness played an important role for all interviewees. Concerns of assertiveness dealt with EA as a control and enforcement function in implementation processes. This control and enforcement function was particularly highlighted by the interviewees due to missing sanctions on non-architecture-compliant technological developments and non-principle conform behavior.

Integration between EA and other departments Like assertiveness, the integration of EA was stated as important from all stakeholder levels. Concerns referred to the necessity to integrate neighboring departments into the development of EA policies. Especially, the involvement of the strategy department was mentioned. Moreover, the representatives of business concerned further involvement in planning processes of EA business aspects (e.g., modeling business functions and assigning those to applications).

Acceptance of other departments Interviewees argued that IT departments do not follow the architecture policies for different reasons. Some are not aware of the guidelines. Others are aware, however, do not follow policies or do not have the resources to follow. On the one hand, interviewees stated the need for budget to cope with the additional efforts that are generated by the implementation of architectural principles. On the other hand, budget issues concerned the sufficient staffing of the EA function.

3.1.5. Transparency

EA deliverables EA deliverables were often mentioned by interviewees from the operational and middle management. In contrast, interviewees from the top management quoted deliverables occasionally. Concerns referring to transparency mainly resulted as a lack of knowing EA deliverables (e.g., “I did not know that there exists such a thing like an application portfolio.”).

Consequently, interviewees suggested a clearer communication of the existing deliverables to the stakeholders.

EA function The EA function is often acknowledged across all stakeholder groups. The interviewees often wondered about the process of generating EA deliverables. Most importantly, the question of how decisions regarding architectures are taken remains unclear (e.g., "what are the decision processes?"; "what are the inputs of the deliverables?"). Interviewees suggested a higher degree of transparency of the EA processes as well as their communication toward affected stakeholders.

3.2. Discussion on Stakeholder Concerns

The review of stakeholder concerns brings about two major distinctions: We found relatively homogeneous concerns among stakeholders on EA deliverables, such as type, quality, and abstraction level. In turn, heterogeneous concerns were found on the role of EA (i.e., context, transparency), depending on the hierarchical level of the interviewees. In the following, we discuss these two distinctions of responses along each of our five classification dimensions. For purpose of demonstrating the implications to EA approaches, we use TOGAF as illustrative example.

3.2.1. Type

TOGAF [223] differentiates *architectures* along three levels of granularity: Strategic, segment, and capability. The stakeholders of these levels -as entailed by TOGAF- correspond to the identified stakeholder groups in section A.1.2. Consequently, TOGAF states the concern to deliver in different granularity levels of architecture deliverables to different stakeholder. Unlike TOGAF, interviewees of the executive management hardly reported concerns on architectures, which indicate less need for such a granularity level.

Policies are used to ensure implementation governance of the architecture [223] as they set the frame to steer the application development and to describe the architecture compliance review process. Furthermore, TOGAF lacks a separation between different stakeholder groups, too. This lack of differentiation is in line with our case results, finding no differences with regards to the hierarchical level of stakeholders (such as on policies).

3.2.2. Quality

In TOGAF, *actuality* does not apply to all EA deliverables [223], which is primarily due to the high level of abstraction. In contrast, our case results promoted actuality with the same importance among all hierarchical levels of stakeholders.

Understandability is one of the quality concerns TOGAF defines for EA deliverables [223]. However, interviewees did not acknowledge the term understandability, but the term *simplicity*. Understandability selectively reflects simplicity. Interviewees brought up simplicity, though, in context of all deliverables. This is in contrast to the use of understandability in TOGAF.

Further, stakeholders concerned not only an understandable design of deliverables, but also their easy access. This is also reflected in simplicity, but not in understandability.

TOGAF defines *stability* as one of the quality criteria [223]. However, it does not differentiate a relevancy between different stakeholder groups. Within our interviews, we were not able to confirm such a differentiation either.

In general, the concern for quality characteristics was low among all stakeholder levels. This may stem from the lack of knowledge regarding EA deliverables among our interview participants.

3.2.3. Abstraction Level

TOGAF emphasizes the development of EA deliverables in a stakeholder-specific fashion [223]. While promoting a stakeholder focus, TOGAF does not entail a method to systematically identify stakeholders and their concerns. Moreover, TOGAF proposes a *low* level abstraction of architectures for operational managers [223]. In line with TOGAF, our interviewees concerned deliverables with low abstraction, too. As opposed to TOGAF, stakeholders who belong to the middle management concerned also deliverables with a low abstraction level.

Compared to low abstraction levels of EA deliverables, stakeholders also raised the need for *high* levels of abstraction. Particularly, members of the operational management concerned the need for a high abstraction of EA deliverables, yet also stated a high abstraction level to be more feasible for middle as well as top management.

3.2.4. Context

Interviewees often brought up *assertiveness* to concern controlling the implementation, enforcing policies, and committing adherence to authority structures established by the EA. Certainly, only the last aspect is reflected properly in TOGAF by its term discipline. TOGAF defines discipline as "a commitment to adhere to procedures, processes, and authority structures established by the organization" [223]. Therefore, we take over interviewees' assertiveness to reflect all pointed out aspects.

Interviewees across operational and middle management named assertiveness of the EA as an important characteristic. Only stakeholders of the top management were less concerned with assertiveness. Interviewees assumed that this may stem from the fact that assertiveness regarding policies is most important on non-strategic levels. Thus, on non-strategical levels, deliverables are produced which must comply with policies.

According to TOGAF [223], cross integration is an important success factor of architecture governance which is part of the organizational context. Similarly, interviewees stated the necessity to *integrate neighboring departments* into the development of EA policies. Contrary to TOGAF, our case analysis did not bring about any separation between the stakeholder groups with regards to the characteristic integration.

Acceptance is selectively reflected in TOGAF [223] as one of the cornerstones for realizing conformity to procedures, processes, and authority structures. Surprisingly, we observed only partial interest among interviewees of the top management. This correlates with the promoted need for a better staffing of the operational management, steered by the top management.

3.2.5. Transparency

According to TOGAF, transparency is the availability of all implemented actions and their decision support for authorized organizations and provider parties [223]. Moreover, TOGAF promotes the necessity of transparency also for understanding deliverables [223]. One facet of transparency mentioned by our interviewees dealt with the communication of existing *EA deliverables*. On the one hand, particular members of the operational and middle management, who are often guided by EA deliverables (e.g., complying with policies), inherently promoted transparency. On the other hand, members of the top management who are not guided by EA deliverables were less concerned by transparency of deliverables. In contrast, we could not find any separation for transparency regarding the *EA function*.

3.3. Implications

Discussing our findings on the illustrative example of TOGAF, we conclude four implications for EA approaches in general. Firstly, two rather than three different levels of abstraction for EA deliverables appear to be sufficient: On the one hand, interviewees stated only concerns regarding two different layers, namely low and high level. On the other hand, stakeholders of the top management throughout our interviews appeared not to be interested in architectural deliverables.

Secondly, some concerns are not entirely reflected in TOGAF. For example, the definition of quality concerns should be expanded to consider all EA's deliverables, not being limited to architecture principles. While interviewees were concerned with simplicity, simplicity is not referred to in TOGAF's terminology. The term understandability selectively reflects simplicity, but not in a fully comprehensive manner. Therefore, it may be helpful for TOGAF's completeness to either replace understandability with simplicity or to add simplicity to deliverable qualities.

Thirdly, we identified a transparency concern for the EA function and its deliverables. Future research should elaborate on strategies how EA departments could more effectively advertise their deliverables and the EA function itself.

Lastly, our results confirm TOGAF's standardization ambitions. However, our results also show the need for a stakeholder specific tailoring, following stakeholders who expect different abstraction levels of deliverables according to their hierarchical position in the organization.

Apart from the improvement potentials of TOGAF, our investigation delivered additional implications for EA research: Interviewees were just modestly interested in quality criteria of EA deliverables. Future research can elaborate on this in organizations where EA deliverables are better known. Moreover, the results may be transferable to other domains in organizations which have crossing functions, like IT-security or strategy, which can be evaluated in the future.

3.4. Limitations

This research has some limitations. Firstly, all interviewees belonged to the IT. Consequently, the results may be limitedly applicable to the business. As one of the main objectives of EA is to

incorporate the business side, future research may elaborate on their concerns in general as well as on distinctions of stakeholder on different hierarchical level among these concerns.

Secondly, while focusing exclusively hierarchical differences due to our research objective, there are further differences in stakeholder concerns within the same hierarchical level, e.g. Chief Executive Officer (CEO) versus Chief Information Officer (CIO), too. However, all interviewees were part of the IT side. Therefore, interviewees of a certain hierarchical level were relatively homogeneous in our study. For a study comprising interviewees from the IT side as well as the business side, this is not imperatively the case.

Finally, one limitation refers to the analysis of a single case study. There are still more concerns included in TOGAF, such as further quality criteria or architecture patterns, which have not been reflected in the case at hand. For future research, more detailed in-sights from multiple stakeholder groups will become necessary to strengthen and extend our findings and amplify the number of considered quality criteria.

Chapter 4.

A Taxonomy of EA Analysis Research

Beforehand, we have elaborated on the concerns of EA stakeholders in practice. Those stakeholders can also benefit from the taxonomy that we present next. This taxonomy allows practitioners to grasp research faster, which is relevant for their special purpose. Additionally, researchers can more easily find related work and determine the produced scientific contribution. The following results have initially been presented in [23].

Motivation The continuous establishment of EA techniques as a means to model a holistic representation of corporate structures, processes and IT infrastructure still attracts many researchers today [10, 193]. While themes like EA frameworks, modeling languages, and the management of EA are reasonably represented in meta-research, EA analysis, a fundamental practice in EA [147], has received much less attention from the research community.

As EA analysis is one of the most relevant functions in EA, it enables informed decision-making and plays a crucial role in projects [147]. Following, we will use the definition suggested by [197]:

Definition 4.1 (EA analysis) *EA analysis is the property assessment, based on models or other EA related data, to inform or bring rationality to decision support of stakeholders.*

The property is related to an analysis concern (e.g., risk, business-IT alignment, cost, etc.). Our definition of concern agrees with the Oxford Dictionary of English definition, which is "A matter of interest or importance to someone". We consider as an analysis concern the main objective of an analysis approach such as cost, risks, performance and so on.

EA analysis is based on the data collected from models and documents. EA modeling itself is a cost and time-consuming effort and, therefore, organizations expect to extract value from those EA models in return [128]. EA analysis enables informed decisions and plays a crucial role in projects because it manages the project's complexity and provides the possibility of comparing architecture alternatives [145].

To date, there are a plethora of analysis paradigms such as ontology-based [22], probabilistic network analysis [106] and network theory [197]; which use several types of EA model based on Web Ontology Language - Description Logic (OWL-DL), ArchiMate, Graphs and so on. Every analysis supports a different analysis concern and, thus, for a sound evaluation of the architecture different kinds of analyses are required [187].

Despite the importance of EA analysis, EA practitioners and researchers do not have an overall shared and acknowledged comprehension about EA analysis techniques. Little research about

mechanisms to classify, compare, or organize the existing EA analysis research can be found. As a consequence, the task of choosing an analysis approach might be challenging when little guidance is provided. Even worse, the design of analysis efforts might be redundant if there is no systematization of the analysis techniques due to the inefficient socialization of practices and results.

We contribute with one important step in that direction deriving a taxonomy to classify analysis research according to its layers, analysis concerns, analysis techniques, and modeling languages. We also evaluate the proposed taxonomy against recent EA analysis research. Doing so, we create foundational elements aiming to foster the development of this research field and also establishing alignment among researchers, tool designers, and EA subject matter experts. Therefore, we answer the following question:

RQ 2 *How is the EA analysis research classified according to its analysis concerns, techniques, and modeling languages?*

To tackle this research question, we apply qualitative and descriptive research, which is split up into four steps. First, we apply the SLR method according to Kitchenham [114] to gather a set of papers related to EA analysis research. Second, we perform a data categorization [45] to end up with a taxonomy answering the question: "How to classify EA analysis research according to its analysis concerns and modeling languages?". We gathered a second data set with papers published between 2016 and September 2018. Finally, we apply the created taxonomy created on the evaluation data set to evaluate and improve the taxonomy. More detailed information can be found in the appendix A.2.

4.1. EA Analysis Taxonomy

The taxonomy has four main dimensions: EA Scope, Analysis Concern, Analysis Technique, and Modeling Language, depicted in Figure 4.1.

4.1.1. EA Scope Dimension

By investigating the architecture models, we observed that plenty of papers operate their evaluation on rather specific components instead of looking at a whole model. Even if the authors introduce a case study with a comprehensive EA model, the evaluation considered very specific parts of it for example only the technical layer or process layer [235, 197]. In [215] the authors used an EA model's visions and goals hierarchy for their evaluation although the exemplary data-set consists of much more information. In this case, the relevant components of the EA model were the dependencies between visions and goals. In [246, 19, 174] even larger components were used spreading along multiple layers of the EA model.

Our analysis shows that the EA model related work sticks to the well-known layered structure, e.g. defined in TOGAF [223] or by Winter and Fischer [242]. Accordingly, our EA Model Scope dimension is composed by following well-known layers:

- **Motivation** - Since the publication of [242], recent frameworks offer the opportunity to model elements modeling the motivation or the purpose of the organization (cf. ArchiMate

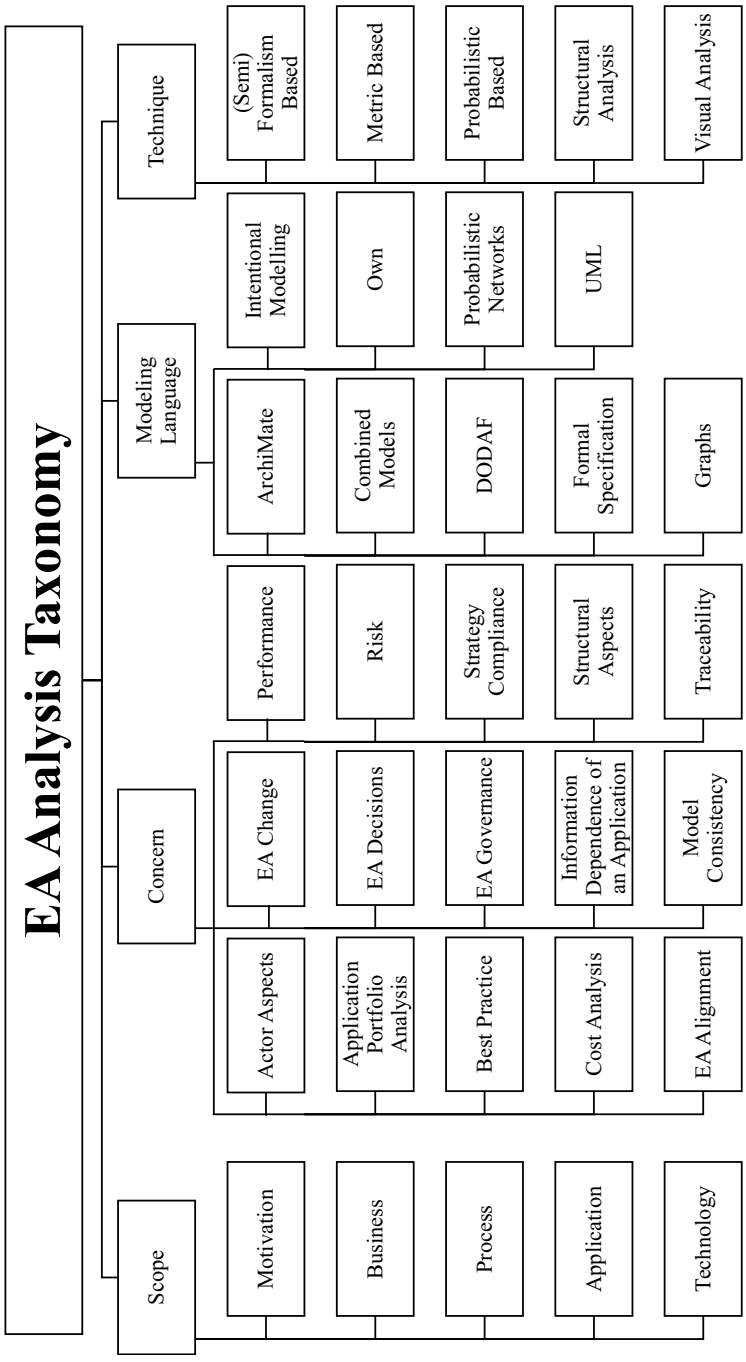


Figure 4.1.: Proposed Taxonomy.

3.0.1 [225]). Additionally, recent research stresses the need for modeling the business motivation [215, 227]. Therefore, we opt for a motivational scope, even if it is implicitly included in the business layer of Winter and Fischer [242].

- **Business** - This represents the fundamental corporate structure as well as any relationships between actors or processes of the business architecture [242].
- **Process** - This layer represents "the fundamental organization of service development, service creation, and service distribution in the relevant enterprise context" [242].
- **Application** - Since there was no observation of requirements for a deeper differentiation of business integration and software architecture, we merge the layer "Integration Architecture" and "Software Architecture" of Winter and Fischer [242]. Consequently, it represents an organization's enterprise services, application clusters, and software services.
- **Technology** - This layer represents the underlying IT infrastructure [242].

4.1.2. Analysis Concern Dimension

We define concerns as relevant interests that pertain to system development, its operation or other important aspects to stakeholders [99]. Since an approach may suit more than one concern at a time, several papers are classified with more than one concern (e.g., [210, 232]). According to our research results, the dimension Analysis Concern consists of 55 concerns, grouped in fifteen categories:

- **Actor Aspects** - This category covers papers dealing with actor's relations to business process, goals, and the impact on them of EA changes, e.g. the organization's impact on the motivation and learning of employees [159].
- **Application Portfolio Analysis** - It means to analyze why certain applications are well liked and widely used than others and what it means to the EA [157].
- **Best Practice** - Papers elaborating on the value of best practice analysis establish EA patterns or evaluate real world EAs with respect to EA patterns [60, 126, 173].
- **Cost Analysis** - Papers related to the value of cost analysis are manifold. For example, they estimate or assess the cost of the current IT architecture [68], or determine the Return on Investment (ROI) of EA [190]. Another facet is related to the costs of changing components of the EA [121, p. 440],[210, p. 25].
- **EA Alignment** - For instance, EA redundancy is contained within papers related to EA alignment. Those papers identify redundancies and eliminate unplanned redundancies [41, p. 118]. Additionally, there are papers promoting alignment between layers [34].
- **EA Change** - This value covers concerns related to modifications of the current EA. Scientific research related to this value elaborates, for example, the consequences of changes, scenarios' choices, or performs gap analysis.

- **EA Decisions** - This value covers approaches related to the decision-making process itself. Exemplary, it is related to the rationale behind decisions, stakeholders' influence on the decision-making process, or methods to evaluate alternatives [180, 181].

- **EA Governance** - Research related to EA Governance evaluates EA from a strategic viewpoint, comprehending the analysis of EA's overall quality and its function. This value includes works dealing with EA effectiveness, EA data quality, EA documentation, or metrics monitoring [48, 40].

- **Information Dependence of an Application** - This category aims to evaluate dependent applications on EA, helping CIOs to manage their application landscape and to eliminate redundancies [2].

- **Model Consistency** - This value aims to evaluate the integrity of EA models and its consistency through time and organizations' evolution [22, 64].

- **Performance** - This value is concerned with specific measures of performance, e.g., EA component performance, business performance, or system quality [72, 160].

- **Risk** - Papers related to the value of risk elaborate on different aspects: risk of component's failure and its consequences, information security as a whole, EA project risks, or EA implementation risks [72, 77].

- **Strategy Compliance** - Research on Strategy Compliance analyze if EA decisions, EA projects, models, and its structure are compliant with the organization's strategy [182, 219].

- **Structural Aspects** - This value covers analysis how components are organized, the relations among the components and their emergent complexity, possible ripple effects, clustering issues, and positional values in the structure [4, 132].

- **Traceability** It represents the need of querying or tracking components that are connected/linked to a particular component or have specific attributes values.

4.1.3. Modeling Language Dimension

In some papers, the proposed method relies on certain properties introduced by specific frameworks [246, 174]. Others require EA models where the actual meta-model was of less importance or they require models that follow either less formalized or more general meta-models [197]. Researchers, therefore, may require model data to follow a specific conceptual format which is captured by the third dimension *Modeling Language*. In this case, conceptual format serves as a generic term for meta-model or framework.

We identified several modeling approaches, some already existing, others created by the authors to suit their specific analysis approach. We categorized the modeling techniques into nine values of the dimension:

- **ArchiMate** - Obviously all research modeled with ArchiMate is classified within this value. Mainly, there can three subcategories be distinguished: Firstly, papers applying ArchiMate [182, 48]. Secondly, papers extending ArchiMate [77, 40]. Finally, papers that explicitly used the Archimate adapted or merged with other entities and attributes [182].

- **Combined models** - This category comprises papers that use more than one modeling language to perform their analysis, e.g., [221] which uses Business Motivation Model (BMM) and intentional modeling together with ArchiMate to evaluate if and how business rules and goals are compliant with the organization's directives.

- **DODAF** - Papers related to this category, obviously, use the incorporated model of Department of Defense Architecture Framework (DoDAF) [218].

- **Formal specification** - This category is characterized by the attempt to describe EA models with textual languages or mathematical specifications such as set theory [152], ontology [102, 20], or Domain Specific Language (DSL)s [29]. Usually, the related papers build their models aiming to take advantage of reasoning techniques to support EA analysis.

- **Graphs** - In this value, the EAs are modeled as graphs, with their components and relations being represented by nodes and edges, respectively. In addition, design structure matrix is included because they are structurally equivalent to graphs. Examples can be found in [72, 12].

  A special sub case of EA graph models are probabilistic relational models, influence diagrams, Bayesian networks, and fault tree analysis models. All those models work with uncertainty and probability principles in their modeling approaches [173, 106].

- **Intentional modeling** - This category covers papers concerned with goals, modeled with the I* framework [249] and related models [16, 146]. Commonly, these papers aim to analyze strategy related concerns.

- **Own** - In this value, we included papers that present their own EA modeling framework and it is not classifiable in none of the other categories [126, 97].

- **Probabilistic networks** - This category incorporates all models, which work with uncertainty and probability principles [173, 97]. Performance metrics along all EA layers are common concerns.

- **UML** - This category covers papers that use Unified Modeling Language (UML) [205, 158] or UML-based models [153, 159] to perform their analysis.

4.1.4. Analysis Technique Dimension

This dimension covers techniques and methods used to perform EA analysis. We identified a plurality of different approaches, as a large portion of the approaches was proprietary, and many were poorly detailed, focusing on the results rather than the analysis process. The results

were classified in 22 categories according to their main characteristics: (Semi) Formalism based, Analytic Hierarchy Process (AHP), Architecture Theory Diagram (ATD) based, Axiomatic Design, Best practice conformance, Business Intelligence (BI), Business-IT-Alignment (BITAM), Compliance analysis, Design Structured Matrix, EA Anamnesis, EA executable models, EA misalignment catalog, Fuzzy based, ML techniques, Mathematical functions, Metrics based, Multi-criteria analysis, Prescriptive models, Probabilistic based, Proprietary techniques, Structural analysis, and Visual analysis. About 70% of the studies corresponded to the following five values:

- **(Semi) Formalism Based** - It includes description languages, ontologies, set theory, and other formalisms. All those techniques try to take advantage of reasoning mechanisms to perform (semi-) automated analysis of the EA, through queries, model consistency, and restrictions checks, for example [64, 126].

- **Metric-based** - Analysis approaches including several punctual quantitative metrics to evaluate operational data from the components (e.g., performance, usage, workload) or from the overall EA (e.g., entropy) [235, 151].

- **Probabilistic-based** - Cause and effect, uncertainty and probabilistic events are concepts present in all variations of methods belonging to this category. Typical techniques are Bayesian networks, probabilistic Bayesian networks, extended influence diagrams, and fault-tree analysis. Those are frequently used to perform EA components' performance analysis [173, 97].

- **Structural analysis** - In this category, structural aspects of the overall EA or specific layers are analyzed. Methods and techniques based on network theory are employed to identify critical points, clusters or overall indexes for the EA structure [245, 55].

- **Visual analysis** - This category covers several techniques that use the power of visualization intrinsic to the models to extract valuable information for the experts. Typical concerns analyzed are alignment between layers, the impact of changes or failures in the overall structure [200, 132].

The previous dimensions were defined as a result of the SLR performed, as described in Appendix A.2. In order to assess the taxonomy, we updated the data through a new SLR (see Figure A.2, Step 3) addressing papers published after the first research's interval and applied the taxonomy to its final data-set, containing 46 articles.

The papers on the new data-set addressed 26 concerns classified in 13 categories already present on the taxonomy, which indicates its good coverage. From the 47 preexisting concerns, six were merged into three ones and eight new concerns were mapped on the update (into the categories of Actor aspects, Best practice analysis, Actor aspects, EA Alignment, EA Change, Model consistency, and Structural aspects).

Regarding modeling approaches, 89.1% of studies presented model-based analysis. Only two new values of modeling approaches were detected, one of them also resulting in one new category (DoDAF models). The papers from the data set were classified, according to the taxonomy,

into seven categories – i.e., only one paper was not covered by the taxonomy's preexisting values, which, again, indicates its good coverage.

Our first study resulted in a considerable number of different analysis techniques and methods, classified into 23 categories. When applying the results to the new data-set, we found 19 of those, and five new categories, determined by specific approaches.

4.2. Discussion

Following, we present existing research classified by our taxonomy and discuss the insights.

All the evaluated papers were covered by the five layers of the EA scope dimension (cf. Figure 4.2). Regarding the frequency of EA targeted scopes, most of the papers approached more than one layer. Business and Application are the layers that received more focus on the analysis in general - 77% and 83% of the total, respectively.

We identified about 22 different modeling approaches, divided into nine categories (ArchiMate-based, Combined models, DoDAF, Formal Specification-based, Graphs-based, Intentional Modeling, Own, Probabilistic networks-based, and UML-based). The distribution of the studies, from both SLRs, regarding their modeling approaches is depicted in Figure 4.3.

Even though ArchiMate-based and graphs-based represent a large part of the studies, 34.9% of the approaches used a proprietary model or a combined model to perform their analysis. The plurality of different modeling approaches reflects the lack of standardization regarding EA models and corroborates the affirmation from [105] that "there is no clear understanding of what information a good enterprise architectural model should contain".

Our taxonomy defined 52 concerns, classified into 15 categories: Actors aspects, Application Portfolio Analysis, Best practice analysis, Cost analysis, EA Alignment, EA Change, EA Decisions, EA Governance, Information dependence of an application, Model consistency, Performance, Risk, Strategy Compliance, Structural aspects, and Traceability. The amount of papers on each concern category is illustrated by Figure 4.4.

It is important to consider that some studies approached more than one concern on their analysis (e.g., [209] performs an analysis on eight different aspects of the Application scope). According to our research' results, the focus of EA analysis has been in five main categories: EA Change, EA Alignment, Strategy Compliance, Performance and Structural Aspects, as shown in Figure 4.4. Papers covering these concerns correspond to 64.7% of the whole final set.

We identified a plurality of different analysis approaches (i.e., techniques or methods), classified in 22 categories according to their main characteristics, as shown in Figure 4.5. A large portion of the approaches was proprietary, and some of them so specific that we gathered them resulting in a specific category. Many approaches were poorly detailed, focusing on the results rather than the analysis process.

In our present literature review about EA analysis, from both set of papers, 57.5% of the works presented empirical data, while 28.7% of them used simulated and 13.8% only theoretical data. Although several publications present empirical cases, some of them do not present enough information about how the study was conducted and the benefits obtained from the analysis approach (e.g., [75]). This lack of information leads, on the one hand, to the issue of the reproducibility of methods, as some techniques require a specific set of data. This set of data is

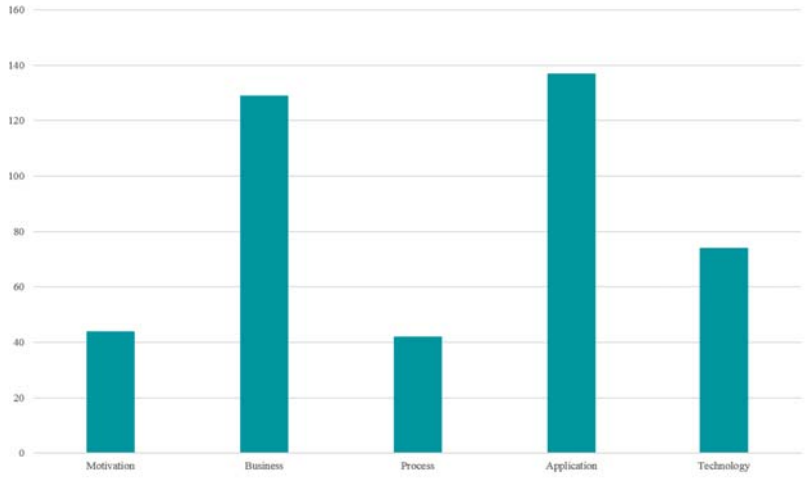


Figure 4.2.: Number of Studies Per Scope Category.

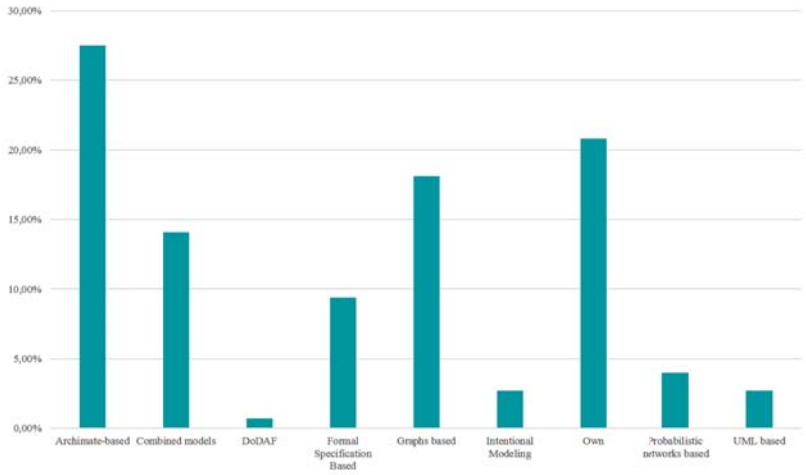


Figure 4.3.: Percentage of Studies on Each Modeling Language.

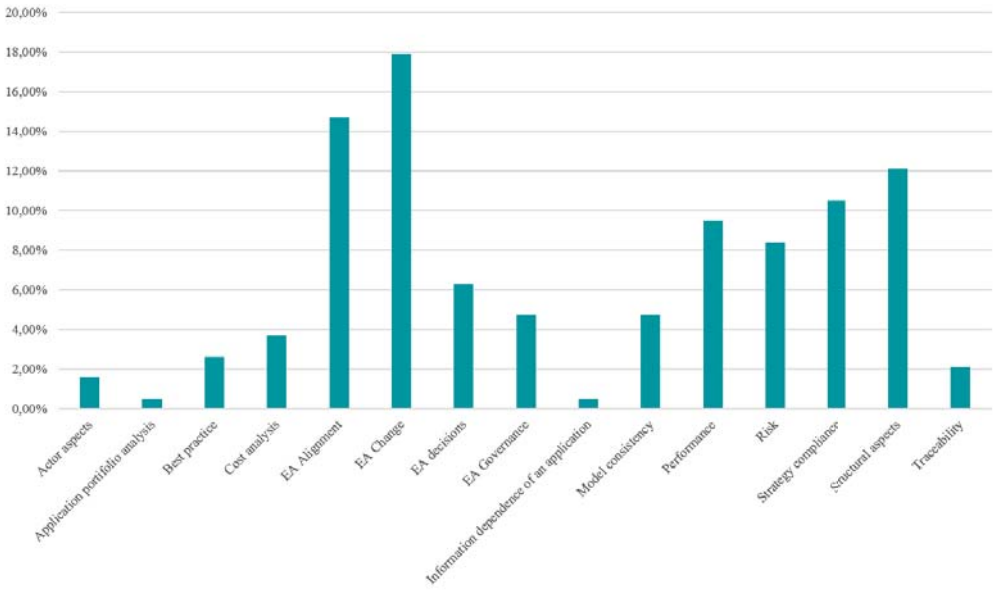


Figure 4.4.: Number of Studies Per Concern Category.

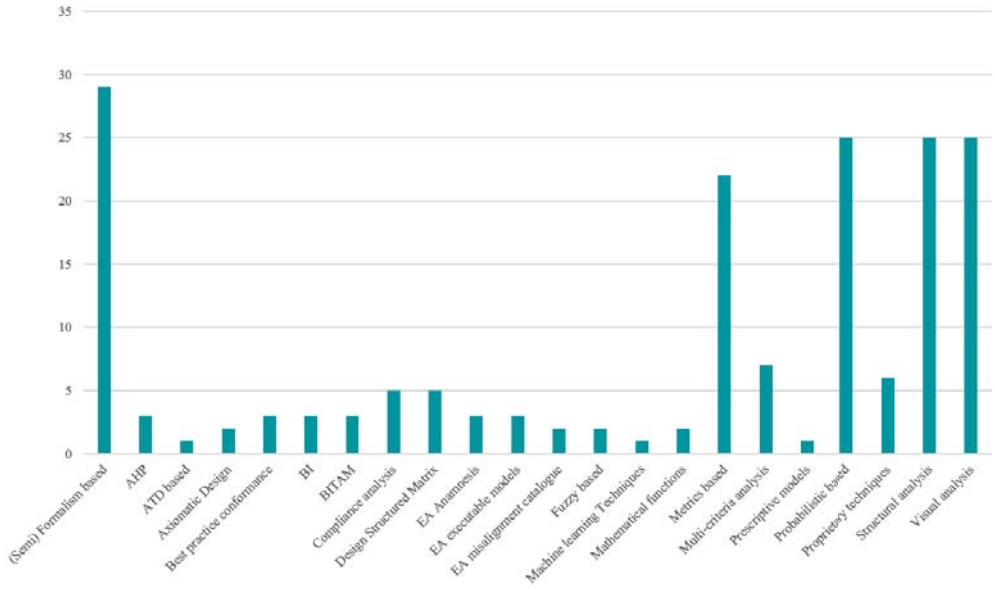


Figure 4.5.: Number of Studies Per Analysis Techniques Category.

not always available, due to classification as confidential by its owning organization [75]. On the other hand, the data set might be artificially created for a special purpose, because there was no data publicly available and, therefore, the created data set might not applicable to real world scenarios (e.g., [74, 220]). Despite no empirical evidence to which degree EA research faces those issues is found, many examples of a fallback to artificial evaluation by using exemplary data sets can be given [70, 215, 19, 246]. In those cases, the developed artifact normally undertakes an evaluation at non-realistic conditions and produces results which do not hold in a realistic setting [233].

4.3. Limitations

As for limitations, we did not perform backward and forward searches. However, because of the broad coverage of our search string, we are confident that the additional search would not uncover much more works. In our SLR, we did not perform a qualitative assessment of primary studies. We accepted intentionally all the works that aimed to perform EA analysis, without a very strict quality criteria, to be able to have a broad understanding of the field and the authors' purpose.

Part III.

Processes to Improve EA Models

Chapter 5.

An Enterprise Architecture Model Roundtrip

Hitherto, we have developed a taxonomy to classify EA analysis research and elaborated on the concerns of EA stakeholder. As our results indicate, architecture models in general and their actuality in special are relevant factors of EA stakeholder. Therefore, we will develop different processes to ensure the actuality and quality of EA models in the next chapters. First, we sketch a process, which enables enterprise architects to keep a central EA model up-to-date. Second, we derive a continuous delivery pipeline based on the previously mentioned process to automate as many steps as possible. Last, we develop an approach to preserve contradictory information of different sources in an EA model. The following results have initially been presented in [84].

Motivation EA is no end in itself but has to provide central, important, and up-to-date information of the organization (e.g., business processes, application and data architectures, or infrastructure components) to its clients, e.g., to all projects of an organization. The results produced by the projects may change the EA model, thus, contributing to a continuous evolution of the EA. But often, the interaction between the projects and the EA department is not coordinated systematically, leading to only a weak acceptance of EA, to unnecessary overhead, and to a state where the EA model and the results developed by the projects are not in sync, but inconsistent.

There are a lot of different drivers for changes of the EA model [62], which contribute to a continuous evolution of the EA. As our research is related to a project driven environment, we will following refer to projects as the main event of changes to the EA model as already identified by Sousa et al. [213] and confirmed by Farwick et al. [62]. Though, projects are just an example for changes and can be replaced by any other trigger.

A systematic EA model roundtrip process is one approach to overcome the problems of EA model consistency rooted in the continuous evolution of the model by different projects. Existing research focuses solely on single aspects of such a roundtrip process, like model merging issues [109] or quality related questions [165]. Since a holistic perspective on a roundtrip for EA is absent, we identified our research question associated with an architecture roundtrip approach:

RQ 3 *How does an effective architecture roundtrip process looks like supporting the distributed EA evolution?*

Obviously, this process has to define the roles and their responsibilities as well as the needed process steps. Knowing these steps, we can investigate which steps can be either automated to

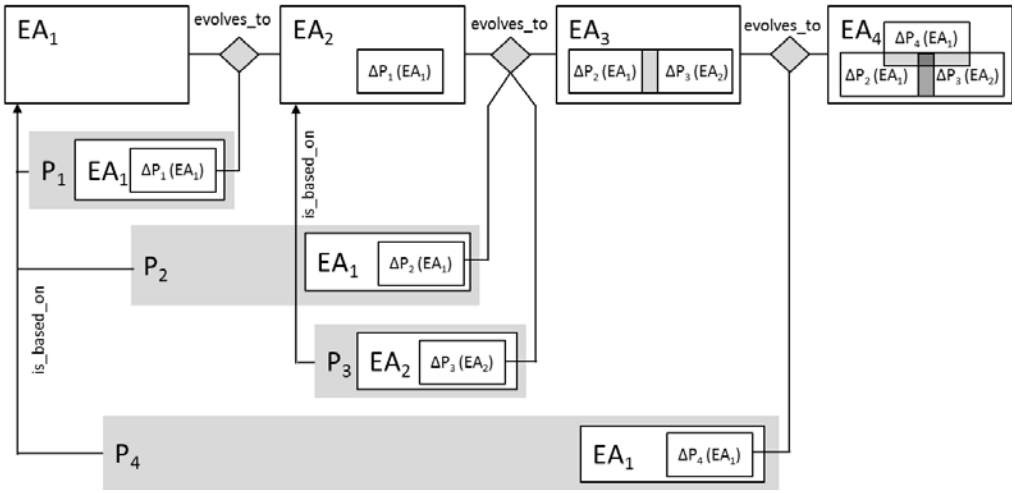


Figure 5.1.: Dependencies Between Different Projects Elaborating on the EA model.

avoid error prone manual work [155], like modeling components several times, or technically supported, like suggestion of possible duplicates. Moreover, existing research may be facilitated to implement certain process steps.

To tackle the issue of a distributed EA model evolution, we opt for DSR as presented in Section 1.4.3. For more details on the procedure of applying DSR in this case, we refer to Appendix A.3.

5.1. Research Problem

As already argued, EA has to provide central, important, and up-to-date information of the organization to its clients. Therefore, it is necessary to include the changes from all important sources of information as soon as possible. However, the input can be contradictory. Following, we will elaborate further, how these contradictions can arise.

The EA model and the distributed projects are related in different ways among each other, cf. figure 5.1. Each project receives all needed information and a copy of the central EA model based on the current version of the central EA model when it starts. All architectural changes performed by the project and affecting the EA model are collected in a so called EA change set (denoted $\Delta P_x(EA_y)$). For instance, project P_1 receives its needed information based on the first version EA_1 and creates the change set $\Delta P_1(EA_1)$. At the end of project P_1, its change set is returned to the EA model to be integrated and used to evolve EA_1 to the succeeding version EA_2.

Projects do not necessarily know about each other and the decisions they are taking. Therefore, we assume that they develop solution scenarios independently resulting in a distributed evolution of the EA. This distributed evolution leads to different challenges: For example, the change sets of projects P_1 and P_2 are not completely distinct; they share some common EA in-

formation. When both change sets are returned, the union of the change sets with the EA model can cause different conflicts [110]: On the one hand, there is no conflict between the change sets and the EA. Consequently, there is no need for action. On the other hand, if there are changes, four different types can be distinguished: structural changes, attribute changes, reference changes, and move changes.

The integration of change sets into an EA model is needed to keep it consistent and up-to-date. But this becomes more difficult if we consider the different time spans of projects. E.g., project P_3 starts when EA_1 is the current version and ends after version EA_2 already exist, integrating the change sets of projects P_1 and P_2. Hence, there could be data contained in the change set of project P_3, which is based on outdated EA information. In those cases, the projects P_1 and P_2 are already closed and, obviously, their output cannot be reworked if the issues is caused by one of them. Consequently, the enterprise architects have to correct the misleading information in the EA model by themselves.

To sum up, we aim to develop an EA model roundtrip process that guides the project-driven distributed evolution of an EA model. This helps to offer up-to-date information and models to EA's clients at any time, thus, increasing the acceptance and benefits of EA within an organization.

5.2. Roundtrip Process

The EA model roundtrip process depicted in figure 5.2 frames our approaches to tackle the issue of a distributed EA model evolution. As we introduced the problem to solve, we present the current version of our process after several iterations and introduce its mayor steps following. But at first, we briefly characterize the two roles that are mainly involved:

In TOGAF [223], the Open Group exemplary defines five categories of EA stakeholders: corporate functions, end-user organization, project organization, systems operation, and external. Shrinking it to the EA model roundtrip process, we differentiate two main groups of stakeholder: Those who are just interested in the results of the process, and those who are actively involved in the process itself. The most important stakeholders of the last category are:

Enterprise architects are responsible to maintain the EA model and keep it up-to-date. They have to ensure that the EA model evolves appropriately and have to provide EA information to all EA clients.

Solution architects are members of the project and responsible to create project specific solution scenarios based on the overall EA model. The developed solutions may affect the EA model and have to be integrated appropriately.

As an EA model is always just a representation of the EA of an organization, a revealed deviation between the EA and the EA model is the trigger for the enterprise architects to evolve the EA model. On the one hand, this can be a genuine change of the EA such as the introduction of an application, the retirement of a process. On the other hand, a deviation can arise from faulty or incomplete information in the EA model.

The enterprise architects process these revealed deviations, which includes *determining the change set*, *aligning the data* of the change set to a proper EA level, *assuring data's quality*, *updating the EA model*, and *offering updated information* to EA's stakeholders. To acknowledge

the two different stakeholder groups and their different doing, the EA model roundtrip process is composed of two different processes: The maintenance of the central EA model and the development within projects. In the following we will discuss the single steps of the EA model roundtrip process in more detail.

5.2.1. Process Trigger

Trigger for the EA model evolution process have been researched by the Farwick et al. [62] in the past. Therefore, we will present their results in the following.

Trigger for EA model changes can be classified by two different categories: the impact on the affected area of the EA and the origin they appear from. Not every change within the organization may cause a trigger for an EA model change. Farwick et al. [62] postulate according to Winter and Fischer [242] that changes beyond the scope of the EA model should not trigger the EA model change process. For instance, we assume a setting within the technology layer of the EA model. A solved service ticket regarding a non-running server does not affect the EA model as this information is on a very detailed level. In contrast, the introduction of a new server might have an impact, because a new technical platform could be used.

A further categorization for process triggers arise especially from projects and strategic planning as they might not only contribute to the as-is of the EA model, but also to the to-be and the will-be. The to-be refer to the planned evolution of the EA. Will-be states correspond to EA model states that are decided and will become as-is in the future. However, to-be and will-be are not necessarily identical.

Farwick et al. [62] identify also a technical perspective of process triggers. More concrete, they detail the operations that the EA model can change, as elements in the model can be added, updated, or removed. Additionally, they state a generic operation consisting of a combination of the beforehand mentioned operations.

Farwick et al. [62] gathered an non-exhaustive list of tool-supported EA neighbored processes:

- **Project Management:** Projects are fundamental drivers for EA and, consequently, EA model changes [213]. Farwick et al. [62] stress the challenge to identify which projects are architectural relevant and, thus, should be process triggers.
- **Release Management:** As release management serves as planning for new IS, respectively, their releases, this information can used as a trigger [6, 89].
- **License Management:** Newly acquired licenses indicate the introduction of a new IS. Contrary, a non-renewed license indicate the retirement of an IS. Additionally, new business functions could be introduced as an IS might be introduced to support them [214].
- **Change Management:** Depending on the detail level of IT Infrastructure Library (ITIL) [201] changes, those can be valuable information sources for EA model maintenance. In contrast to projects, those changes cover also small changes not worth it to set up a project.
- **Service Management:** Similar to change management many service processes are maintained according to ITIL [201]. Those processes can also serve as input for EA models.

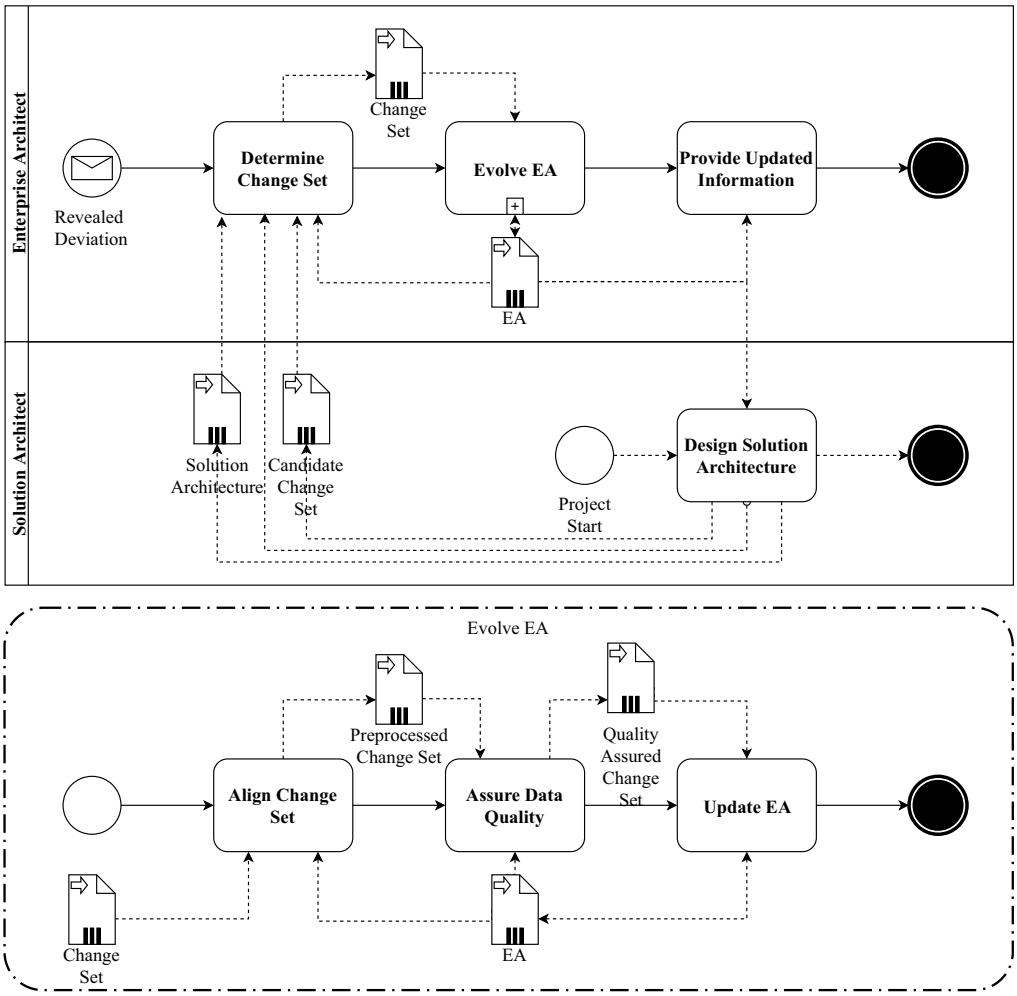


Figure 5.2.: BPMN Model of the EA Model Roundtrip Process.

- **Organizational Management:** Buckl et al. [36] discuss how organizational restructuring should be taken into account for EA. Consequently, changes in Human Resource Management (HRM) and Lightweight Directory Access Protocol (LDAP) can cause changes in EA models.
- **Service Registries:** Organizations register technical services in registries, e.g., Service Oriented Architecture (SOA) or micro-service registries. Additional, communication along such services is often steered along certain technologies, like an ESB [39].

5.2.2. Change Set Determination

Posterior the occurrence of a possible change event, the change set for the next roundtrip needs to be determined. As determining a change set is a known problem in model driven software development, some approaches solving this issue are investigated by, e.g., Gorek and Kelter [76], Wenzel et al. [238], and Schmidt et al. [202]. According to these authors, we define change set as follows:

Definition 5.1 (Change Set) *A change set is a collection of all necessary atomic changes (create, update, and delete) to transform a model from one state to another.*

Furthermore, the work of Kehrer [109] deserves more interest. Based on the results of the aforementioned works, he states the necessity to group atomic changes of a model to a single logical change group. For instance, if a method should be moved from one class to another class, there are two atomic changes: the deletion of the method in the one class and the addition of the method into the other class. This simple example shows that both atomic changes are only useful in combination. Otherwise, the model becomes inconsistent. Hence, a simple change set determination as proposed by Mens [149] or Lindholm [137] is not suitable, since it does not take into account the relations between atomic changes.

To enable the enterprise architects determining the change set correctly, the data provider, e.g., the solution architects, deliver a so called candidate change set besides the modeled (project) architecture. This candidate change set contains the atomic changes as well as the logical change groups. Furthermore, the enterprise architects have to consult with the data provider about the changes while determining the change set to ensure the consistency of the EA model.

5.2.3. EA Model Evolution

Change Set Alignment

After determining the change set, the EA model can be evolved. But first, the data of the change set may be aligned to yield the right abstraction level, as usually the architecture models provided by the solution architects are more detailed than needed for EA. The respective alignment consists of two steps: First, the identification and deletion of too detailed elements and, second, the insertion of derived relations.

Identifying too detailed elements is challenging, since the right level of abstraction depends on the organization and even in the research community there is no common agreement on the

right level of abstraction [11]. Indications can be aggregation or composition relations between elements where the aggregated/composed elements may be needless. Especially, if the aggregating/composing element is already part of the EA, the aggregated/composed elements may not be of interest for EA.

Furthermore, element types which are not part of the EA model are candidates to be removed, too. Another technique to determine unintended elements within EA models can be supervised machine learning. Enterprise architects could train algorithms, which propose in future iterations possible candidates that may are unintended. However, this list of techniques might be incomplete and further research is necessary.

If elements are removed, transitional relations may get lost, which should not happen as this data is still important and needed. Van Buuren et al. [231] suggest a method to derive a relation between two non-connected ArchiMate elements by analyzing the path between these elements and choosing the most general relation. This can be applied to ensure the consistency of the EA model.

Hence, if an element is removed, all connected elements and their relations have to be analyzed to choose the most general relation. This relation will be added to the model, whilst the analyzed relations and the corresponding element will be removed.

EA Model Quality Assurance

A quality assurance is necessary to ensure the consistency of the EA model, including, for example, the detection of duplicates, the correction of typing errors, or the amendment of semantic misuse of modeling elements.

In this context, we define duplicates as elements in a change set which are already modeled in the EA but cannot matched to each other. This can be rooted in the solution architect's lack of knowledge regarding the EA model. Therefore, they may introduce elements again. For instance, a solution architect models *Windows Server 2008 SP3*, but the EA model just contains the more general *Windows Server 2008*. This leads to a duplicate that needs to be identified.

To identify duplicates similarity measures can be applied. For example, the well-known Soundex algorithm by Russell [191] can be used, which also helps to overcome with typing errors. Furthermore, structural similarities can be found by using some techniques applied at the field of web search. E.g., Blondel et al. [30] propose a measure for similarity between vertices and apply it to synonym extraction.

The aforementioned semantic misuse can be caused by a wrong mapping or a wrong modeling by the solution architects. Whereas, a wrong mapping is negligible due to its one time correction effort, a wrong modeling is a main issue which should be avoided. For example, our experience shows that the element type *Application Component* of the application layer and the element type *System Software* of the technology layer often get inverted within an ArchiMate model.

The beforehand discussed methods of duplicate identification can be used as well as methods for anomaly detection to overcome with the semantic misuse. Approaches for anomaly detection are presented among others by Noble and Cook [167], or Eberle and Holder [58].

To ensure and assess the quality of a certain EA model, we did a first step by designing an EA Quality Framework (EAQF) (see section 8). Therefore, we adapt the work of Becker et al. [27] to our purpose and come up with a structure consisting of three parts. One part forms the

basis on which the other both parts are established. In this basis the purpose, objectives, and stakeholder are determined. The other parts are utilized to either rate the quality of the whole model or the quality of a certain view.

EA Model Update

There are two different origins of contradictory EA data input. First, there are technical sources like network scanners or databases, e.g., the Configuration Management Database (CMDB). Those sources, mostly contained in the technology layer [229, 228], may deliver data on different levels of details or even outdated data. Second, data provided by human EA suppliers may cause conflicts, because there are different ways to model the same aspect or they rely on wrong or outdated data.

To handle such contradictions, so far two different kinds of approaches are proposed: The first approach tries to resolve the contradictions, e.g., by estimating the trustworthiness of the sources [229, 228]. The second approach tries to prevent contradictions before they emerge, for example, by providing a holistic framework to assess the quality of EA models [227].

After determining the change sets, aligning the data, and assuring data's quality, the enterprise architects can import the processed change set into the EA model. This step needs to be done mostly manually, because conflict resolution cannot be solved fully automatic. Those conflicts arise from the problems stated in section 5.1. The enterprise architects have to keep the change sets in mind, which were defined by the solution architects. If a conflict arises, the enterprise architects and the modeler who have issued the conflict have to discuss these change sets.

An alternative is not trying to solve contradictions, but to keep them. Exemplary, we propose a method which creates a EA model based on probabilities in section 7. The basic idea is that projects plan competitive scenarios and estimate their probability to become true. Then, this information is stored within the EA model.

5.3. Provision of Updated EA Models

Providing the updated information to EA's stakeholders is straight forward. All reports have to be created newly and pushed into their communication channels. An area of special interest in the context of the architecture roundtrip is the project specific view on the EA model. This view is the basis on which the solution architects model the solution architecture and needed changes of the EA model.

We distinct two different ways, how the project specific view on the EA is handled. On the one hand, the solution architects receive the view at the project start and it is used as it is. On the other hand, the solution architects already worked on an ancestor version of the view and a merging of both views is necessary to avoid data loss.

The merging consists of three different cases. First, an element does not exist in the ancestor version and has to be added. Second, an element exists in both views, but it has been changed. In this case the element in the ancestor version will be overwritten. If this element is already used it gets marked to inform the solution architect. Third, an element is not contained in the

ancestor version. Therefore, it should be deleted. If the element is already in use, it should be marked instead of being deleted to enable the solution architect to correct the affected views.

Chapter 6.

A Continuous Delivery Pipeline for EA Model Evolution

Previously, we have sketched a process to keep a central EA model in an environment up-to-date, in which different projects refine the EA model in a decentral, uncoordinated manner. Following, we will present a continuous delivery pipeline to automate as many steps of the process as possible, which has initially been presented in [86].

Motivation EA are currently mostly modeled manually and changes require huge manual efforts. This is especially true when complex organizational structures need to be covered and the organization is constantly changing. The pace of changing structures and complexity is expected to increase and this makes it even more challenging [240]. In recent years, the field of enterprise architecture management already adopted techniques to reduce model maintenance effort. However, there are still challenges in regards to conflicting changes, different semantics and responsibilities [61].

In the field of software engineering, changing requirements are also very common. Software engineering deals with this by becoming as agile as possible and uses various social and technical techniques to improve towards this direction [90].

Examples for social techniques are the ongoing adoption of agile process frameworks like Scrum or Kanban and even techniques directly related to the development itself like pair programming. Technical examples are the rise of continuous integration and delivery. All of these techniques lead to the same shared goal: Shorten feedback loops [94]. Techniques used for software engineering are also being adopted for other parts of organizations: With the DevOps movement, which emphasis on the collaboration of development and operations, infrastructure is being covered using techniques typically used in the context of software engineering and processes are also adopted [51].

To overcome the beforehand stated problems of EA modeling, we proposed already an architecture roundtrip process in Section 5. However, this process is still abstract and needs to be instantiated. To do so, we facilitate the well-known technique of Continuous Delivery (CD) and realize the architecture roundtrip process. Accordingly, we formulate our research question:

RQ 4 *How can continuous delivery help to overcome the challenges of manual EA model's maintenance?*

To develop the pipeline, we opt for DSR as presented in Section 1.4.3. For more details on the procedure of applying DSR in this case, we refer to Appendix A.4.

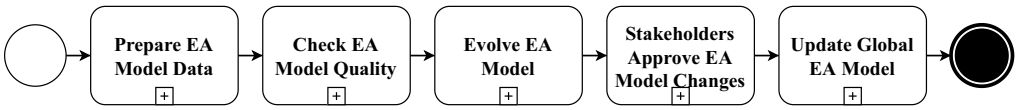


Figure 6.1.: EA Model Maintenance Deployment Pipeline.

6.1. A Pipeline for EA Maintenance

Before we present our pipeline for EA maintenance, we like to clarify the term CD. Humble and Farley [98] define CD as a set of practices, which enables to speed-up, automate and optimize the delivery of software artifacts to the customer with higher quality and lower risks in a continuous manner. Continuous delivery uses an automated development infrastructure, called deployment pipeline, which automates nearly every step of the delivery process. Each commit of a developer enters the deployment pipeline and an automated process is started, which produces a new software increment as a result artifact.

The deployment pipeline incorporates all activities known from continuous integration [56] as automatic build, unit testing, and static code analysis. In addition to these, the pipeline performs testing activities like integration, performance, and security testing. All these tasks are executed in a defined order of stages. After each stage, the test results are evaluated at a quality gate, which stops the processing if the quality conditions are not met. If all quality gates are passed, the software artifact is stored and can be accessed and used from external clients; it is released.

Next, we will sketch our pipeline for an EA model maintenance. Fischer et al. [63] contribute two main findings to our pipeline. First, they propose an EA model maintenance process, which we unite with our work from [84]. Second, they offer a fine-grained role concept, which we incorporate into the pipeline as well.

To implement our deployment pipeline for EA model maintenance, we opt for the prototype JARVIS presented by Steffens et al. [217]. JARVIS is the implementation of a conceptual model for a new generation of software delivery systems. It focuses on two specific challenges for adopting CD. The first is the lack of flexibility and maintainability of software delivery systems. The second is the insufficient user support to model and manage delivery processes and pipelines. Further, it allows integrating the proposed processes into a deployment pipeline. Therefore, we create a Business Process Model and Notation (BPMN) representation of the process as JARVIS is equipped to use BPMN as modeling language.

From the BPMN model, we derive the necessary activities, which needs to be implemented as microservices. During this, we transform the process model to reflect better the principles of JARVIS and continuous delivery in general. Figure 6.1 shows the resulting BPMN model for EA model maintenance and Figures 6.2 to 6.6 the concrete realization of each sub-task.

The first process steps from Fischer et al. and Hacks et al. of initializing and collecting the necessary data of the EA model evolution can be omitted. We assume, that in an environment following the principles of continuous delivery [98] all artifacts like the global and the special EA models are under version control and stored in an appropriate Software Configuration Management (SCM) system like subversion or git. Each change to one of these models needs to be committed to the repository. A change results in a new version of the model. Whenever a change

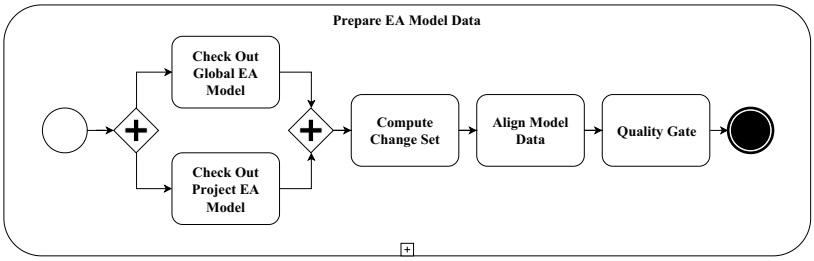


Figure 6.2.: Sub-Task Prepare EA Model Data.

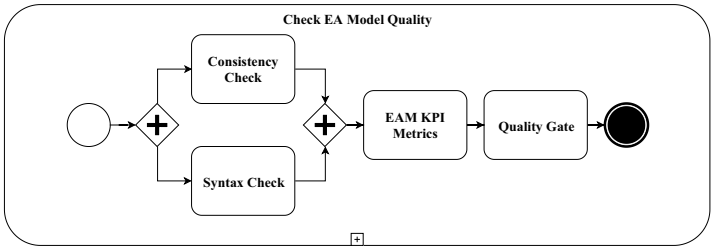


Figure 6.3.: Sub-Task Check EA Model Quality.

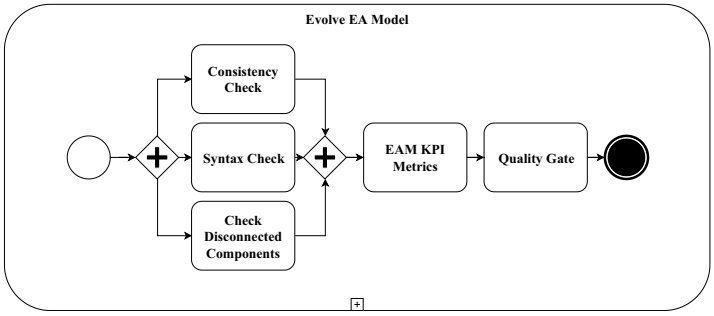


Figure 6.4.: Sub-Task Evolve EA Model.

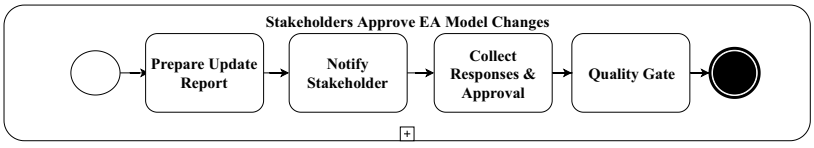


Figure 6.5.: Sub-Task Stakeholders Approve EA Model Changes.

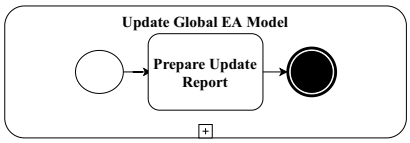


Figure 6.6.: Sub-Task Update Global EA Model.

is committed to the repository for the special architecture our deployment pipeline is triggered automatically.

The technical infrastructure of the SCM and the deployment pipeline ensure the automatic processing of the first process steps of both proposed maintenance processes. Necessary notifications can be sent by the system to check if the actual process is compliant to the overall process, but effectively we want the stakeholders only to be involved if really necessary.

The pipeline starts by first checking out the new models versions from the repository and provide both to the first transformation activity within JARVIS. This activity is called “Compute Change Set” and uses the provided input models to compute the existing deviations between both and provide these as a new artifact called “Change Set”. All existing artifacts are now processed in the next transformation activity “Align Model Data”. Hacks et al. [84] argue that a specific project may contain more detailed information than the more general global EA model. Therefore, the two models have to be aligned in order to be effectively compare- and merge-able. The following quality gates check the successful execution of the proceeded activities and the existence of the three artifacts. Afterwards, the first stage of our deployment pipeline is finished. This stage corresponds to the checkout and compile stages in classic software delivery pipelines.

Fischer et al. and Hacks et al. both incorporate steps to check the model quality like consistency or correctness of syntax. We follow their idea and model these as assessments to check models quality. The assessments are performed on singular artifacts of the preceding stage and produce a report each. This stage corresponds to static analysis for software source code. Duval et al. [56] incorporate an inspection phase into their continuous integration model in which relevant metrics for software quality are measured and evaluated. We adopt this by applying well-known EAM Key Performance Indicator (KPI)s [148] to models inside the pipeline.

In the next stage, the artifacts are integrated to produce a new and updated candidate for the EA model. This candidate is then examined by the same assessments as before. The modular architecture allows us to integrate even more sophisticated assessments, which can be performed on EA models. We integrated a check for disconnected components, which checks if parts of the resulting EA model candidate has components, which are not connected to the rest of the model. Based on the assessment reports, the quality gate decides if the pipeline should continue to the next stage where the candidate is presented to the stakeholders of the overall process.

Up to this point the pipeline is performing its tasks completely autonomous, so the stakeholder are only involved if the model candidate has reached a certain degree of quality due to the assessments performed before. The manual approval of the stakeholders corresponds to the User Acceptance Test (UAT) stage in classic pipelines. Bass et al. [25] define the UAT stage as the last one before going to production and are meant to ensure these aspects of the delivery process which cannot be automated.

Table 6.7.: Mapping From EA Model Roundtrip Process to the EA Model Maintenance Deployment Pipeline.

EA Model Roundtrip Process		**EA Model Maintenance Pipeline**	
Task	*Sub-Task*	*Task*	*Sub-Task*
Determine Change Set		Prepare EA Model Data	Compute Change Set
Evolve EA	Align Change Set	Prepare EA Model Data	Align Model Data
Evolve EA	Assure Data Quality	Check Model Quality	
Evolve EA	Update EA	Update Global EA Model	
Provide Updated Information		Update Global EA Model	
-		Stakeholder Approve EA Model Changes	
Design Solution Architecture		-	

If this stage is successfully executed, the EA model candidate is promoted to the final stage where it is deployed to the EA model repository. The next run of the pipeline will use this new version of the EA model and so the roundtrip is completed. Additionally, all reports build now on this updated data and EA's stakeholder can receive the latest information.

As already mentioned, our pipeline can understood as an implementation of the EA model roundtrip process presented in Section 5. Most mappings are intuitively, as we can recognize in Table 6.7. Table 6.7 sketches that some tasks are more detailed in the roundtrip process. E.g., we explicitly state that the updated EA model should be provided to the stakeholders, while in the pipeline, we subsume this in the update task. Furthermore, the design of the solution architecture is not modeled explicitly in the pipeline, since it does not take the same holistic perspective as the roundtrip. One thing, which is not reflected in the roundtrip, is the approval of the stakeholders. This should be added in a next iteration of the process.

6.2. Demonstration

To demonstrate and evaluate our artifact, we conduct a fictitious case study. We opt for a fictitious case study to create reproducible results as the facilitated example of an airport departure system presented in section 1.4.2 is freely available. Contrary, a fictitious case study lacks a real world evaluation. However, as most companies classify their EA models as confidential, results on those would not be reproducible.

6.2.1. Facilitated Metrics

To simulate the "Check Model Quality" step of the pipeline, we check the EA model against KPIs from the EAM KPI Catalog [148]. As we do not want to implement all KPIs of the catalog, we randomly chose three of them as representatives for all KPIs. Those KPIs are only exemplary and can be replaced by any other calculable metrics. Nonetheless, we have to keep in mind that it can be quite challenging to assess the necessary input parameters (e.g., if interviews have to be conducted).

PM guideline adherence checks if IT projects adhere to the stated PM guideline [148, p. 28]. As the information model of the KPI catalog is not directly reflected in ArchiMate, we identify work package as an IT project and business object with a property PMguideline as PM guideline. To compute this KPI, the project managers, first, answer the degree the project adheres to every guideline. Second, we compute the average for every project along all guidelines. The catalog defines three categories of adherence. If a project adheres to 100% to the guideline it is full adherence. Between 100% and 75% it is a minor deviation which will cause a warning in our pipeline. With less than 75% it is a major deviation causing a fail of the pipeline.

Application continuity plan availability [148, p. 19] measures the degree how completely IT continuity plans for business critical applications have been drawn and tested for the IT's application portfolio. To reflect the information model in ArchiMate, we map the application to application component and continuity plan to business object with the property ContinuityPlan. The responsible for the operation of the applications answer if there exists a continuity plan for a certain application and if it is tested. The KPI is then computed by the number of critical applications where a tested continuity plan is available divided by the total number of critical applications. The value is good above 80%. Normally, the value will between 60% and 80% resuming in a warning to related stakeholders. If the value drops beyond 60% the value is problematic and the pipeline fails.

IT process standard adherence [148, p. 33] checks if a certain application (application component in ArchiMate) adheres to the IT standard processes (application process in ArchiMate). This is answered by the process responsible and then calculated by the number of applications, which adhere to an IT standard process, divided by the total number of applications. The value is good at 100%. Normally, the value will between 80% and 100% resuming in a warning to related stakeholders. If the value drops beyond 80% the value is bad and the pipeline fails.

Besides the EAM KPI Catalog, we check also the *connectivity* of the graph representing the EA. A graph is connected if there is a path between every pair of nodes. We assume that the model of an EA should be always connected. If the model contains isolated elements or sub graphs, there are parts in the organization, which are not related to the other parts. So to say, there are parts in the EA pursuing different goals and, therefore, different organizations within the organization. Nevertheless, for two organizations there would be two EA's. Consequently, we expect the model of the EA to be connected. Otherwise, the pipeline should fail.

6.2.2. Implementation of the Pipeline

Our designed pipeline for EA model maintenance was implemented for the continuous software delivery system JARVIS [217] and each activity not natively supported by JARVIS was implemented as an independent microservice following the architectural framework of JARVIS (cf. table 6.8). From JARVIS, we reused the complete infrastructure and general activities like the git checkout activity and the quality gate activity.

To simulate the distributed character of our case with a centrally maintained EA model and several projects evolving this central model, we set up three git repositories. The first repository contains the central EA model. The other repositories simulate different projects elaborating on the central model. Therefore, they clone the central repository at the beginning of the project

Table 6.8.: Implemented Microservices Within JARVIS.

Activity	**Purpose**	**Microservice**	**JARVIS native**
CheckOut EA Model	Get the central model from git.	○	●
CheckOut Project EA Model	Get the decentral project model from git.	○	●
Compute Change Set	Compute the changes to the central EA model made by the project.	●	○
Align Model Data	Remove too detailed information from the model.	●	○
Quality Gate	Check if all preceding activities were successful.	○	●
Consistency Checks	Check the model for consistency flaws.	●	○
Syntax Check	Check if the model adheres to the ArchiMate Exchange Format syntax.	●	○
EAM KPI Metrics	Three different microservices checking the adherence to the EAM KPIs.	●	○
Evolve EA Model	Update the central EA model by the changes done by the project.	●	○
Check Disconnected Components	Check if the EA model as a graph got disaggregated.	●	○
Prepare Update Report	Collect all effective changes to the central EA model.	●	○
Notify Stakeholders	An email is sent to all stakeholders containing all changes.	○	●
Upload EA Model To Repository	Upload the new version to git.	○	●

Table 6.9.: Exemplary test cases.

Test case ID	Input				Expected output
	PM guideline adherence	Application continuity plan availability	IT process standard adherence	Connected	
1	100%	100%	100%	TRUE	success
6	100%	70%	100%	TRUE	warning
7	100%	70%	90%	TRUE	warning
8	100%	50%	100%	TRUE	fail

and conduct their changes on the model before they provide it back to the central repository by a pull request.

6.3. Evaluation

To evaluate our pipeline, we conduct an equivalence class test [38, p. 623] where the values of the metrics serve as input parameters (three respectively two classes per KPI) and the behavior of the pipeline (i.e., successful, warning, and fail) represents the output. To test the pipeline, we combine each possible input class and determine the expected output for each combination resulting in 54 test cases. We always choose the worst expected output (fail>warning>successful) if different outputs would be possible. An extract of four exemplary test cases is presented in Table 6.9.

For the execution, the presented example model from section 1.4.2 was stored in a git repository and used as the global EA model. A variant of the model was stored in a second repository and was used as the repository for a simulated specific project model. The variant introduced changes to the original model which had an negative impact on the model KPIs, especially to “Application continuity plan availability” and “IT process standard adherence”.

The execution of the test cases by triggering the pipeline with different inputs showed that our approach is feasible. The expected behavior of our pipeline could be observed. In case of a failing pipeline the execution always stopped at the first KPI assessment in the Model Quality stage. This is caused in the fact, that we already test the KPIs on each model in this stage, so the assessment of the project model results in a fail. By deactivating this assessment the pipeline performs the Model Execution stage and fails at this point. Both behaviors are correct. Due to this phenomenon we recognize, that our pipeline is already performing a simpler inspection process for the models provided by projects, it is embedded in the global EA model maintenance process.

6.4. Discussion

Before, we presented our pipeline and its application on a fictitious example. Our results show that the existing approaches are missing certain steps, which we incorporated into our pipeline.

For example, our roundtrip process lacks a step for an evaluation of EA KPIs, which are represented in the Model Quality stage and in the Model Evolution stage of our pipeline. Therefore, future evolutions of this process should include such mechanisms. As the KPIs can be easily computed and evaluated automatically, we can apply it inside in a continuous delivery pipeline. Furthermore, the pipeline incorporates a simple inspection process of models provided by projects. We think, it may be a good idea to extract this inspection process into its own independent pipeline and use it during the project solution development. This would lead to a similar result as with Continuous Integration and Continuous Delivery. Continuous Delivery can be seen as an extension of Continuous Integration as Fowler argued [67]. The project pipeline would only consider the single project model as our maintenance process also considers the global EA model and an EA model candidate which integrates changes from the project model into the EA model.

In addition, the roundtrip approach has a need for the incremental and iterative nature of an agile development process. The project solution delivers its model only one time to the maintenance process. With incorporating continuous delivery, the project can deliver the changed model every time to the overall maintenance process. So, the project will get feedback on the compatibility with the global EA model earlier and can adapt to this feedback more easily. The deviations between global and specific model are therefore minimized.

On the other hand, changes to the EA model are much earlier distributed to other projects in the organization, as their maintenance process will use the adapted EA model also for other active projects. So the deviations between the various projects are minimized. In result, the automation of the maintenance process may lead to more relevant EA model, which represents the current state of the organization and its enterprise architecture in a much more accurate way. Furthermore, the whole process is completely transparent and most important traceable, which supports further requirements regarding compliance and security.

The process of Fischer et al. [63] lacks the roundtrip approach. As we count on short feedback cycles as typical for agile development, we overcome this shortcoming. In addition, our proposed pipeline reduces the involvement of stakeholders and the necessary manual work to a minimum. Stakeholders only assess EA model candidates which has achieved a certain degree of quality.

Furthermore, we introduced a new metric in section 6.2.1 to measure the connectivity of the EA model represented by a graph. For our case study we assumed that the complete graph needs to be connected. However, depending on the needs of the organization under observation multiple connected components are desired. Another organization's need could be for a metric to assess the certain degree of connectivity for the whole EA or its sub graphs. As our case study is only fictitious, it does not offer further insights into these aspects and needs to be investigated in future research.

6.5. Limitations

However, our research includes still some limitations. First, we were not able to test our approach in a real world environment. Such a field evaluation may raise additional issues, especially related to the influence of our approach on the sociological environment. So far, we focused

only on technical aspects, but internal resistance might hinder our approach.

Second, we just took a single project as data provider for our pipeline into account. A plenty of distributed data provider might cause issues, we did not consider thus far. In particular, we encourage short feedback cycles, which might cause problems as well if the mindset of the involved employees is missing.

Third, we took a very technical view on the problem. For instance, we assumed for simplicity reasons that the needed input for the KPIs, which we facilitate for our quality gate, can be computed easily. However, the assessing of certain inputs for the KPIs can be quite challenging, which needs to be further evaluated in future research. Additionally, there might be not only one perception of a KPI as multiple stakeholders with a diverse background and possibly different expertise and expectations contribute to its assessment and interpretation, which has to be taken into account.

Chapter 7.

A Probabilistic Enterprise Architecture Model Evolution

Both artifacts presented before have in common that they include steps to resolve conflicts arising from contradictory information. However, there a situations in which we want to keep the contradictory information, for example, because the information describes future states of the EA model and we still not know which information will be the truth. Therefore, we present next an approach to tackle this issue, which has initially been presented in [82].

Motivation As already stressed, the EA model is one central artifact of EA and it supports the EA's stakeholders to create added value [165]. Consequently, EA has to provide central, important, and up-to-date information to its stakeholders. For this purpose, EA needs to collect data from several sources which may be contradictory.

There are two different origins of contradictory EA data input. First, there are technical sources like network scanners or databases, e.g., the CMDB. Those sources, mostly contained in the technology layer [229, 228], may deliver data on different levels of details or even outdated data. Second, data provided by human EA suppliers may cause conflicts, because there are different ways to model the same aspect or they rely on wrong or outdated data.

To handle such contradictions, so far two different kinds of approaches are proposed: The first approach tries to resolve the contradictions, e.g., by estimating the trustworthiness of the sources [229, 228]. The second approach tries to prevent contradictions before they emerge, for example, by providing a holistic framework to assess the quality of EA models [227].

Both approaches presented before, which try to solve the conflict, have in common that they try to determine one single truth for the model. However, in some cases it might be useful to keep the contradictory information. For example, there are two contradictory information from different sources and it is not possible to estimate which one is true. Therefore, we want to keep both information and do not want to abolish one. Another example could be that both information describe alternative future EA scenarios and we do not know which one will become true.

Consequently, we formulate our research question:

RQ 5 *How can evolutionary EA scenarios provided with uncertainty information be presented in an EA model?*

This research question encloses four facets:

1. The EA model needs to provide uncertainty regarding the existence of its entities.
2. The EA model needs to reflect different evolutionary scenarios along a certain time-span, because projects may deliver competing scenarios for different points in time.
3. As interaction with the EA model is needed, e.g., to integrate new scenarios, the proposed approach should define process guidelines, by means of a set of rules, to add and remove scenarios to an EA model as well as how to handle different versions along a certain time series.
4. A version of the EA model without uncertainty is needed, e.g., the management concerns an EA model with fewer details or the used EA tool cannot represent uncertainties.

In order to develop means to reflect uncertainty in EA model evolution, we opt for DSR in accordance to Peffers et al. [178]. Therefore, we generalize the Predictive, Probabilistic Architecture Modeling Framework (P^2AMF) [106] (cf. section 7.1) to a simple graph representation and add capabilities to represent uncertainty regarding future states of the model.

To discuss our work, we create a proof of concept prototype using a graph database containing a small EA model, transform our procedures to Cypher [162] to calculate different EA model states, and, finally, apply Cypher queries to the EA model. We do not rely on our established example of an airport departure system (cf. section 1.4.2) as it incorporates a high degree of complexity, which would raise the complexity of our prototype as well. However, this would complexity is not necessary for our purpose.

7.1. Architecture Modeling Framework for Probabilistic Prediction

In the following we present concepts to represent a probabilistic EA for solving quality issues. Our solution is based on P^2AMF, a framework to model uncertainty in class and object diagrams proposed by Johnson et al. [106]. Therefore, they facilitate Object Constraint Language (OCL) and add attributes storing existence probability values for objects or relations. Moreover, object attributes can be stochastic. However, we are only interested in objects and relations among them.

The following example illustrates P^2AMF. Assume there are two classes (Service and Cloud) connected by an association. As a service might be down or we are not absolutely certain about its presence, its existence is modeled with an uncertainty expressed by a Bernoulli distribution with a probability of 0.98. In other words, a service has a probability of 98% to be existent and 2% to be not existent. Each service is provided by a cloud. As we have no secure knowledge regarding the relation between the service and its respective cloud, the association between Service and Cloud is also fraught with uncertainty. This may lead to a situation in which the defined cardinalities get violated. For more details on this issue and how to solve it, we refer to Johnson et al. [106], especially Section 5.

Second, there is one central cloud instance. Since we definitively know about the presence of the cloud and due to the advent of modern technologies ensuring its up time, we assume

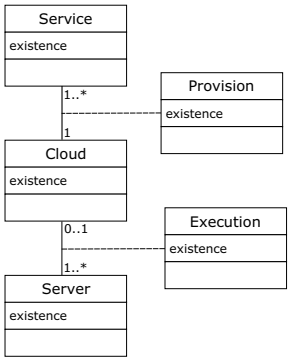


Figure 7.1.: P^{2}AMF Example Class Diagram.

the existence of the cloud equal to 1. One central part to ensure such a up time, is to execute the cloud on several servers. As not every server is utilized to execute the cloud, we expect a probability of 70% that a server executes a cloud (cf. line 8).

```
1  context Service::existence:Boolean
2     init: Bernoulli(0.98)
3  context Provision:existence:Boolean
4     init: Bernoulli(0.98)
5  context Cloud::existence:Boolean
6     init: Bernoulli(1.00)
7  context Execution:existence:Boolean
8     init: Bernoulli(0.70)
9  context Server::existence:Boolean
10    init: Bernoulli(0.97)
```

Listing 7.1: P^{2}AMF Expression Describing Figure 7.1.

Last, we assume that the existence of a server to be present as 97% (cf. line 10). This probability distribution expresses our belief that the server will be installed as planned. However, all those distribution estimations depend on the knowledge of the modeler and should be ideally supported by historical data.

7.2. A Probabilistic Enterprise Architecture

P^{2}AMF provides a framework to represent uncertainty in class and object diagrams. However, if we apply this approach to solve our research problem, two shortcomings occur: First, it is restricted to EAs modeled with UML class and object diagrams. Consequently, dedicated EA modeling languages like ArchiMate [225] are not supported. Therefore, a more general representation is needed. Second, the special needs of decentralized performed projects contributing to a distributed EA evolution are not sufficiently reflected neither. Those needs can be satisfied

by applying Dynamics Bayesian Networks [47] similar to the idea by Johnson et al. [104].

To get a general representation of EA models, we describe an EA model as a pair of a set of nodes and a set of relations: $EA = (N, R)$. A node, $n \in N$, represents an architectural element of the EA like a business process, an application, or a server. A relation, r, is a tuple of two architectural elements, which are linked somehow to each other:

$$r \in R \subseteq \{(u, v) : u, v \in N\}. \tag{7.1}$$

To model uncertainty, we introduce a probabilistic existence function

$$\mathfrak{p} : N \cup R \to \{x \in \mathbb{R} \,|\, 0 \leq x \leq 1\} \tag{7.2}$$

annotated as $\mathfrak{p}(e)$ which returns the probability of a node or a relation to be existent. Those probabilities can be determined in different ways. First, the value could be calculated similar to the approach presented in Johnson et al. [104]: The more often a certain element appears within a reported data set, the higher is its probability to be existent. Second, experts could estimate these probabilities.

Unfortunately, calculating the probability whether a certain scenario will be realized, falls short, because this needs a large amount of data, typically produced in an automated way. But, this is not the case in the EA domain due to the following reasons: First, project architects model a future state which could not be captured automatically. Second, projects architects would model only changes they plan to carry out. Consequently, changes would appear seldom. Third, all projects rely on the same data basis. If a project removes an element, other projects will not, because they necessarily do not know that this element does not exist anymore. Hence, the model would never change. Therefore, calculating the probabilities is not applicable in our case.

Expert estimation seems to be a more promising approach to determine existence probabilities of architecture elements. Obviously, estimation is biased and not perfect, but research has shown that even more formal approaches are not necessarily better [108]. Moreover, the estimation effort is low compared to the suggested calculations. Therefore, we opt for expert estimations.

Project architects or project managers are possible experts to estimate these probabilities. There are two different levels of granularity conceivable to estimate probabilities: First, experts can estimate the probability for each node and relation individually. Second, experts can estimate the probability for a complete scenario consisting of many nodes and relations to become existent.

Apart from the question of *who* should estimate, it is also important to answer the question *when* the estimations should be conducted. Obviously, there is the initial estimation at the beginning of a project when the experts know about the different possible scenarios. This estimation should be updated every time the probabilities of the scenarios change fundamentally, e.g., because one of the scenarios got discarded.

As estimating each node and relation individually is too fine-grained, we take estimating the probability of complete scenarios as more feasible. Let S be the set of all scenarios. To represent a scenario $s \in S$ we annotate it as a quadruple with $s = (N_s^+, R_s^+, N_s^-, R_s^-)$, $N_s^- \subseteq N$ and $R_s^- \subseteq R$. To differentiate between added and removed elements of the EA, we introduce $^+$ to describe that a node or relation is added and $^-$ to describe that a node or relation is removed.

Additionally, we extend the definition of $\mathfrak{p}$ so that it also returns the probability value of a scenario to be existent:

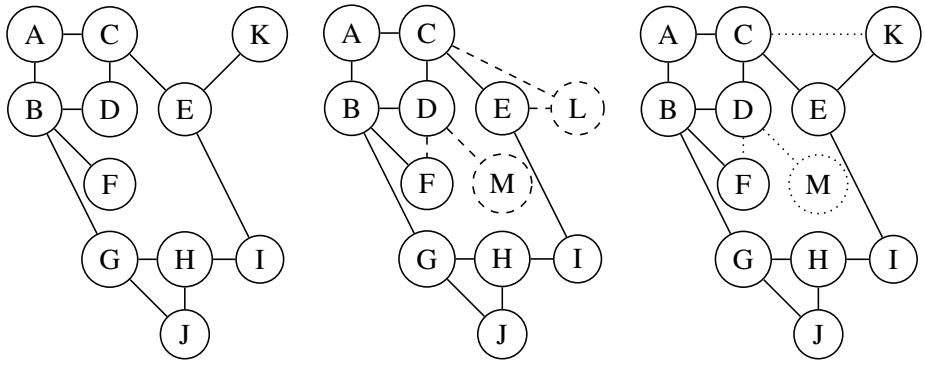
$$\mathfrak{p} : N \cup R \cup S \rightarrow \{x \in \mathbb{R} \,|\, 0 \leq x \leq 1\}. \tag{7.3}$$

Furthermore, it might occur that different projects deliver their results for the same point in time. Therefore, it is necessary to know which scenarios are competitive to each other. Thus, we group all competitive scenarios into one set: $S_C \subseteq S$, where the sum of all scenario probabilities has to be 1:

$$\sum_{s \in S_C} \mathfrak{p}(s) = 1. \tag{7.4}$$

In other words, we know every competitive scenario which delivers its results to a certain point in time.

The following short example clarifies this concept. We assume an EA model simplified to nodes and relations as depicted in Figure 7.2(a). Additionally, there is a project contributing to EA's evolution by proposing two competitive scenarios (cf. Figure 7.2(b) and 7.2(c)). Either scenario 1 or 2 will be implemented. An expert estimates the realization of scenario 1 with $\mathfrak{p} = 0.6$ and scenario 2 with $\mathfrak{p} = 0.4$. Within both scenarios the node *M* is added as well as the relation between *M* and *D*. Furthermore, a relation between *D* and *F* is added in both scenarios. Both scenarios differ in handling the node *K*. Scenario 1 replaces this node by node *L* and additionally links *L* to node *C*. In contrast, scenario 2 keeps node *K* and adds a relation to node *C* like in scenario 1.



(a) Origin EA (b) Scenario 1 (c) Scenario 2

Figure 7.2.: Possible Evolution Scenarios of an EA Model.

For example, scenario 1 and 2 could introduce an application and establish an interface between this application and another one. Moreover, they could introduce a connection between two existing applications. Furthermore, another connection between two applications is set up

in scenario 2. In contrast, scenario 1 retires this application and introduces a new one with the same connections.

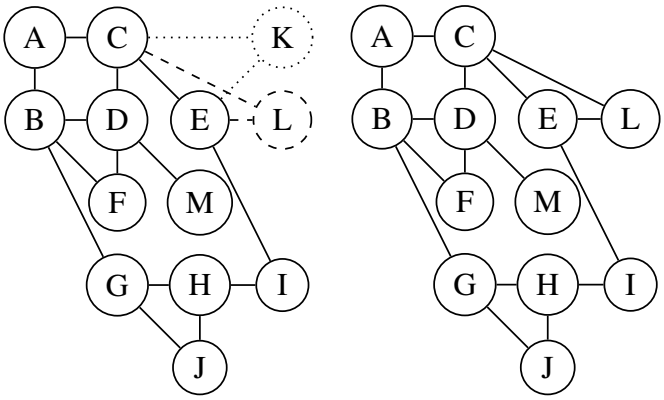
To merge an origin EA model with all competing scenarios (leading to EA'), the existence probability for each node and each relation has to be calculated. For added nodes and relations, we simply sum up the estimated probabilities along all scenarios $s \in S$. If nodes or relations are removed, we have to subtract the estimated probabilities from 1. If a node or relation is unchanged, it keeps its initial probability. This leads to the following equation to calculate the probability for a node $n \in N' = N \cup \bigcup_{s \in S} N_s^+$ of EA':

$$\mathfrak{p}(n) = \begin{cases} \sum_{s \in S} getP(n, N_s^+) & n \in \bigcup_{s \in S} N_s^+, \\ 1 - \sum_{s \in S} getP(n, N_s^-) & n \in \bigcup_{s \in S} N_s^-, \\ \mathfrak{p}(n) & else. \end{cases} \tag{7.5}$$

with

$$getP(n, N_o) = \begin{cases} \mathfrak{p}(n) & n \in N, \\ 0 & else. \end{cases} \quad N_o : set\ of\ nodes \tag{7.6}$$

The equation for a relation r of EA' looks similar. To explain and clarify these equations, we merge the origin EA model with scenarios 1 and 2 from Figure 7.2. The result is depicted in Figure 7.3(a). The solid lines represent nodes and relations with $\mathfrak{p} = 1.0$, the dashed lines with $\mathfrak{p} = 0.6$, and the dotted lines with $\mathfrak{p} = 0.4$. Two observations can be made. First, nodes and relations occurring in both scenarios get a value of 1. Second, node *K* and its relation to *E* have a probability value of 0.4, because both are removed in scenario 1 and, therefore, the probability value has to be inverted.



(a) Merged Version (b) Reported Version

Figure 7.3.: Merging the Origin EA Model with Two Different Scenarios.

So far, our solution to represent probabilities does not take the time dimension into account. However, this is necessary since projects usually do not deliver their results synchronously. Therefore, we introduce points in time, $t_i \in T$, and link each point with the current valid EA model.

Figure 7.4 shows an example. The EA model M contains only entities with $\mathfrak{p} = 1.0$ and, thus, is considered to have no uncertainty. Starting with M as a baseline at t_0, we introduce project A delivering two competitive scenarios A_1 and A_2 at t_1 and project B delivering one non-competitive scenario B_1 at t_2. To keep track of the changes of the competitive scenarios, we trace A_1 and A_2, until we know which scenario will be realized. At this point in time, we integrate the chosen scenario into the current EA model and discard the others. E.g., at t_3 we decide to realize scenario A_2, thus, we incorporate it into M leading to M'. A_1 is discarded and B_1 remains unchanged as it is not competitive to the other scenarios.

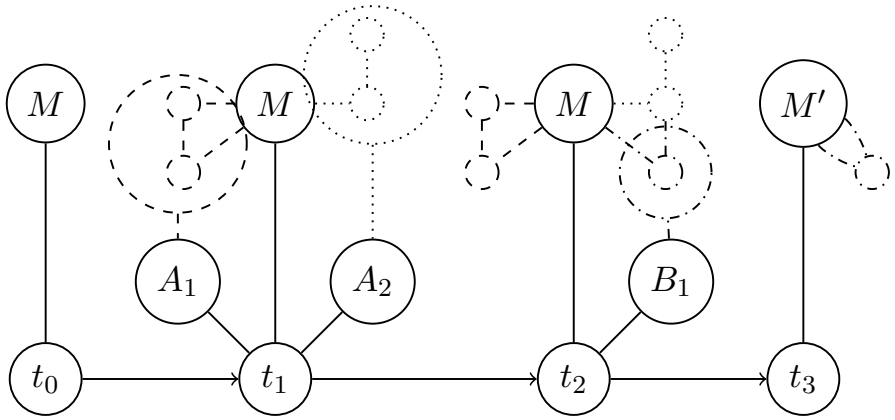


Figure 7.4.: Evolution of an EA Model M over Time.

If an EA model representation without probabilities is needed, a report can be generated containing the most likely EA model. This report can either base on elements with a probability value greater than a threshold or on the most likely scenarios of each project. Assuming a threshold of $\mathfrak{p} \geq 0.6$, the EA model presented in Figure 7.3(b) is created consisting of the origin EA model and the changes introduced by scenario 1. If we choose to create the EA model based on the most likely scenario, the resulting one will be the same.

7.2.1. Demonstration

To demonstrate the proposed approach, we implemented the aforementioned examples in the graph database Neo4j[1]. Additionally, we created the needed reports utilizing Cypher [162] as query language on the database.

Representing the origin EA model depicted in Figure 7.2(a) in a graph database was straight forward. We simply added all nodes and edges and initialized their `existence` property with

[1] `https://neo4j.com`

1.0. To represent points in time, as a project delivers two competitive scenarios, we added two special nodes. The first node expresses the time value itself (e.g., t_0 in Figure 7.4). The second node, linked to the introduced time node via the `AT`-relation, represents the current EA model (cf., M in Figure 7.4). As we also wanted to relate edges to the current EA model, we introduced additional nodes for each edge, because it is not possible to create an edge between an edge and a node. Each special "edge-node" inherits the `existence` property from its respective edge.

Based on this initial EA model representation, we introduced the two competitive scenarios from Figures 7.2(b) and 7.2(c) named A_1 and A_2 in Figure 7.4. First, we added a time node representing $t = 1$ (cf., t_1 in Figure 7.4) and linked it to the already present time node ($t = 0$). Second, we copied the entire EA model related to the current EA node (i.e., M in Figure 7.4) and linked it to time node $t = 1$. Third, we added two nodes representing the two scenarios with the `existence` property equal to 0.6 respectively 0.4. This is necessary to allow for a rollback if a scenario gets dismissed, since we store just the computed `existence` value at each node and not how it was computed.

Last, we added the edges and nodes introduced by each scenario with their probability to the database, linked them to the related scenario node via the `BELONGS`-relation, and recorded that they are added at the relation to ease a possible rollback. If a scenario retires a node or a relation, we remove it from M and relate it to the appropriate scenario with a note that the entity should be retired. Furthermore, we reduce the `existence` property by the probability of the scenario to become existent.

Adding further projects and their scenarios to the database works the same way. More interesting is the rollback of a scenario, because the complementary scenario has to be realized. First, we remove all entities related to the scenario which have been added. Second, we move the entities, which should have been retired, back to the current EA model and restore their `existence` property. Third, we do the opposite with the realized scenario: move the added entities to the current EA model, set their `existence` property to 1.0, and remove the retired entities.

So far, we have presented a concept to enhance an EA model with uncertainty and how to maintain such a model over time. But, how to generate a report representing the EA model in a "classical" way, e.g., to import the model into existing EA tools? This can be achieved by evaluating a Cypher query [162], shown in Listing 7.2, which generates a report incorporating no respectively a certain degree of uncertainty.

```
1 MATCH
2   (:TIME {time: 1})
3   -[:AT]-
4   ()
5   -[:BELONGS]-
6   (n)
7 WHERE n.existence >= 0.5
8 RETURN (n);
```

Listing 7.2: Querying for EA entities at $t = 1$ with $existence \geq 0.5$.

To ensure that we only get EA entities to a specified point in time, the query asks for nodes

labeled with TIME and a property time equal to 1 (cf. line 2). From this start point, we follow the relation AT (cf. line 3) to all nodes representing the current EA model and all scenarios (cf. line 4). From these nodes, we follow the BELONGS relation (cf. line 5) and end up at all entities of the EA model at the chosen point in time and store them into a variable n (cf. line 6).

Now, we apply a filter to ensure that we select only entities we are interested in, i.e., entities with a probability of existence greater or equal than the defined threshold 0.5 (cf. line 7). Last, we return the collected and filtered entities (cf. line 8).

Another and more sophisticated procedure to create such a report is not to rely on the existence probability of each entity, but on the most probable scenario (see Listing 7.3). The query is composed of three parts. The first part (cf. lines 1 to 6) retrieves the current EA model at the time point 1. Line 7 passes the results of the first part to the second part (cf. lines 8 to 11), which collects all scenarios at time point 1. Next, the maximum value of all existence values along all scenarios is calculated. This value together with the current EA model is handed over to the last part in line 12. Last, all entities related to the most probable scenario (cf. lines 13 to 18) are collected and returned together with the current EA model (cf. line 19).

```
1  MATCH
2    (:TIME {time: 1})
3    -[:AT]-
4    (:CURRENT)
5    -[:BELONGS]-
6    (c)
7  WITH c
8  MATCH
9    (:TIME {time: 1})
10   -[:AT]-
11   (s:SCENARIO)
12 W\ac{it}H c, max(s.existence) as max
13 MATCH
14   (:TIME {time: 1})
15   -[:AT]-
16   (:SCENARIO {existence: max})
17   -[:BELONGS]-
18   (e)
19 RETURN (c),(e);
```

Listing 7.3: Querying for EA Entities at $t = 1$ With Most Probable Scenario.

7.2.2. Discussion

According to Shaw [207] examples are a proper technique to discuss artifacts produced in software engineering. Consequently, we discuss to what extent our proposed approach answers the stated research question represented by its four facets compared to P^{2}AMF.

The first facet of our research question covers the need to represent uncertainty regarding the entities within the EA model itself. In P^{2}AMF this is considered by adding an existence property

to the model elements and assigning it a probability. Further, in an instantiation of a P^{2}AMF model there is still no uncertainty regarding the existence anymore. Similarly, we have added an existence property to edges and nodes, but we still have uncertainty in our EA model instances.

The second facet expresses the need for a representation of the evolution of the EA model over time. This is not present in P^{2}AMF, since one model only represents one state. However, the probability may be facilitated to express the behavior of a certain entity along a time series. In contrast, our approach links different evolution alternatives of an EA model along the time to each other. Therefore, we create an actual representation of the EA model at each point in time when it is somehow altered.

The third facet requires the ability to manipulate the model. It covers, on the one hand, adding and removing scenarios to an EA model and, on the other hand, the handling of different versions along a certain time series. The beforehand stated requirement is not explicitly covered within P^{2}AMF, since it neither contains competing scenarios nor different versions. However, there is no need to cover this requirement in P^{2}AMF, because there are no dependencies among the included entities, which would raise the need for such a rule set. Whereas, we described beforehand how to fulfill this requirement both in theory and in practice.

The fourth facet incorporates the necessity to create a report without uncertainty. As the instantiation of P^{2}AMF does not contain any uncertainty regarding the existence of its entities, this instantiation can be utilized as the required report. In opposition, we suggest two different ways to extract a representation of the EA model at an arbitrary point in time. This is more sophisticated, but it is more flexible regarding to the stakeholders' needs.

7.3. Limitations

Our research encloses still some limitation: We implicitly assumed a continuous evolution of the EA model. In other words, we expect that a new state added to our time series belongs to a point in time which is more in the future than all thus far persisted ones. Consequently, there are still no mechanisms which could handle a change of an included state or the addition of a state between two existing states. A possible option could be not to copy the current EA each time but to use symbolic links or to keep only the changes to the model and not the whole model.

Part IV.

Techniques to Improve EA Models

Chapter 8.

Assessing EA Model Quality

Up to now, we have elaborated on the concerns of EA stakeholders and derived from those the necessity to ensure the quality of EA models. In the previous part, we have developed different processes which support enterprise architects to achieve a higher level of EA model quality. In the next chapters, we introduce different methods, which focus on EA models quality solely by proposing concrete actions for improvement. Therefore, we facilitate ML techniques in Chapter 9 to identify possible duplicates in the EA model and develop a Linear Integer Program (LIP) to determine improvement potentials due to functional overlapping in Chapter 10.

However, before we can improve the quality of EA models, we need, first, a deeper understanding of EA quality in general (cf. Section 2.4) and, second, a method to assess the quality, which we present in the following. This method has initially been presented in [227].

Motivation To raise the quality of EA models, it is necessary to ensure the consistency of it, including, the detection of duplicates, the correction of typing errors, or the amendment of semantic misuse of modeling elements. Nevertheless, to our knowledge no widely accepted approach exists, that enables stakeholders of EA to completely assess the EA model's quality [122]. Still, the benefits of EA management highly depend on the model quality [165]. As we found out during our research, only a few articles address this research gap with the specification of EA quality attributes, but without providing a holistic framework how to actually use them in an EA context. Thus, we address with this work the following research goal: to develop a holistic framework that reveals what EA practitioners have to consider when assessing their EA model's quality. In contrast to other works (cf. [165]), we therefore solely focus on the EA model and define the following research question:

RQ 6 *What aspects need to be included in a framework for assessing the quality of EA models?*

From our point of view this includes the analysis of related work, which also may root in other domains than EA modeling, the structure of the framework and guidelines for the framework's application to real-world contexts.

To develop a method for assessing the quality of EA models, we opt for SLR. For more details on the procedure of applying the SLR in this case, we refer to Appendix A.5.

8.1. A Framework for Assessing the Quality of EA Models

In contrast to the other research on EA quality, we propose a framework that aims to guide architects how to assess their EA's quality. Thereby, the framework solely focuses on the EA

model as one of the EA products. Thus, we do not address EA management processes or services related to an EA product. The framework was built by (i) identifying and using an appropriate conceptual framework categorizing different quality aspects of EA models and (ii) identifying relevant EA model attributes within these categories. In contrast to related work (cf. [122, 165, 111, 136, 179]) this goes beyond merely measure quality attributes and guides architects how they should apply them to their concrete EA context.

For (i) identifying a conceptual quality framework approaches from disciplines beyond EA research can be facilitated. Semiotics theory provides a general framework for assessing model quality [154]. It addresses model quality from the perspectives of syntax, semantic and pragmatism and is applied in research related to enterprise reference model quality [230] or conceptual modeling [138]. While the syntactic quality discusses the alignment between an IS model and the modeling language it uses, semantic quality refers to the similarity between the IS model and the domain it reflects on. Further, the pragmatic quality aspect considers choices made during modeling in terms of comprehensibility [138].

We understand this threefold conceptualization of model quality sufficient when it comes to detailed IS models like entity relation graphs or business process models. EA models, on the other hand, are means for decisions regarding business-IT alignment or other strategic issues related to IT. They focus on a broader and more aggregated extract of reality. In order to support EA related decisions, architects may also include economic aspects in their EA models. Further, due to their complexity and the heterogeneity of their addressees, the structure and the documentation of EA models seem to have a distinctive influence on their quality. Therefore, we assess the framework from semiotics theory as too narrow and aim to conceptualize EA model quality from a more holistic point of view. Therefore, we apply the framework of modeling principles proposed by Becker et al. Besides using the dimensions of semiotics theory they explicitly define quality aspects addressing economic efficiency and the structure of enterprise models. They name six principles for proper modeling [27] as explained in the following:

- **Principle of validity:** Does the model match the segment of reality? Syntactic and semantic correctness are the most important criteria for this principle.
- **Principle of relevance**: It says that it is not necessary to model all elements from the real world, just the ones that are needed for the modeling purpose. The decision for relevance has to be made with aim and purpose of the model.
- **Principle of clarity:** All stakeholders of the EA model have to comprehend the model, even without being involved in the modeling process itself.
- **Principle of economic efficiency**: Modeling should follow a clear purpose or aim. Even the cost-effectiveness should be taken into account.
- **Principle of systematic model construction:** All model parts should follow a general documented structure and should be held consistent.
- **Principle of comparison:** The model should be comparable in semantic and syntactic against others, even with different model notations. This includes possible transformations into different modeling languages.

These six principles of model quality form the basis we used to fill with quality attributes from the literature analysis. Therefore, we analyzed relevant research we found for EA quality attributes that focus on the EA model in concrete and related them to the appropriate quality principle. Differently named attributes were aggregated, when they addressed the same aspect of EA model quality. The following articles were identified: [128, 122, 165, 111, 136, 27, 179, 196, 216]. For each attribute we defined a concrete description and identified assessment methods that help architects to measure them. Thereby, we used both qualitative and quantitative as metric types. Although they relate to quantitative metrics, we also defined yes/no questions as a separate type, as well as assessment methods that could be performed by dint of modeling tools.

Before presenting our framework to assess the quality of EA models (EAQF) in detail, we explain how to use it. According to Lankhorst et al. modeling is goal-driven and, thus, highly depends on its purpose, its different stakeholders, and their concerns towards the EA model. Therefore, an EA model repository stores all model elements and their relationships among each other. In order to address the manifold stakeholder concerns, different views on this complete EA model exist, that address different aspects [128]. We deem it vital to align our EAQF with this approach. Thus, we structure it by three dimensions: (i) EA purpose, objective, stakeholders, (ii) EA model as a whole, and (iii) certain EA model views. Statements should be made regarding these dimensions. Thus, for each of these dimensions we identify relating quality attributes from the different principles from [27].

As shown in the illustration, the EA purpose, its objectives and the stakeholders' concerns form the basis to assess the actual EA model's quality. After clarifying this, the EAQF defines quality attributes addressing the two EA artifacts of interest: the EA model as a whole and each EA model view in particular. While the attributes related to the basis help to assess whether the EA's scope is sufficiently determined, the architect can use the results from the whole assessment of the EA model and each EA model view to balance them against the determined EA scope, if e.g. the level of detail is appropriate. This supports the idea that different EA models focus on different aspects. For example, if an enterprise uses EA models to analyze the complexity of their IS landscape, a highly detailed business architecture layer does not seem appropriate. Although the quality attribute "level of detail" (see Table 8.5) for the business architecture model view in this case would be rated as "bad" in isolated consideration, in balance against the EA's purpose it is appropriate.

The assessment of EA model quality is guided by the simple process depicted in Figure 8.1. First, the assessor has to determine the EA models purpose by answering the questions of Table 8.3. If she is aware of the purpose, she can tailor the questions regarding the EA model and its views by removing unintended questions. If the questions of Table 8.4 and 8.5 are aligned to the identified purpose, she can collect all necessary information e.g. by conducting interviews or performing calculations. Based on this information, she can identify possible weaknesses, which should be addressed until the next assessment of EA model's quality.

Additionally, EAQF can be utilized within existing EA model maintenance processes. For example, EAQF can serve in the process of Hacks and Lichter [84] as means to assess the model quality within the process step "Assure Data Quality" or as a further input to the pipeline of Hacks et al. [86] in "Check Model Quality" and "Model Evolution". Furthermore, the step

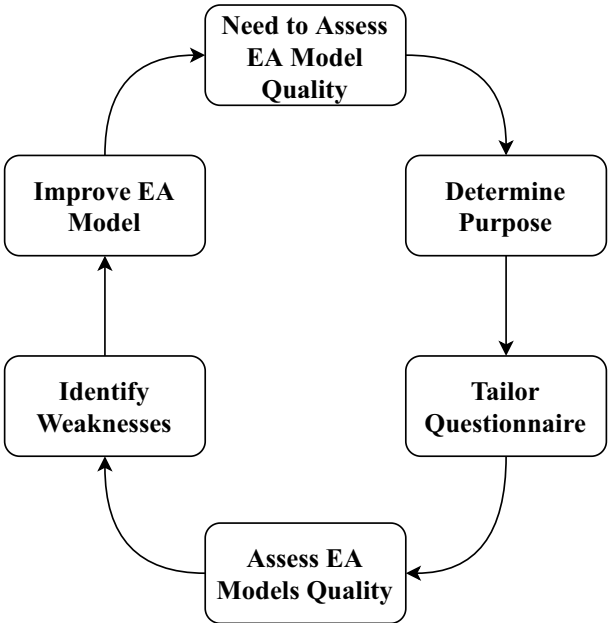


Figure 8.1.: The EAQF Assessment Process.

"Check Consistency" in the process of Fischer et al. [63] can benefit from the input EAQF can provide. Lastly, EAQF can also be integrated into existing EA frameworks like TOGAF [223]. Within TOGAFs Architecture Development Method (ADM) process, EAQF can enrich the architecture creation steps. More concrete, these creation steps imply a gap analysis were the enterprise architects identify areas of the current and target system for which provision has not been made. This analysis the gap for real world systems. However, we can link EAQF into this analysis as well, because issues on the EA model may arise from issues in the real world systems, too.

The complete EAQF is shown by dint of several tables that are described in the following. Table 8.2 gives an overview of all EA quality attributes that were identified in literature. They are described and classified towards the respective quality principle from [27]. Further, the table reveals from what source the attributes were taken. After relating the attributes to the quality principles we decided whether an attribute is to be assessed regarding the EA purpose, the complete EA model or for each single EA view. We therefore initially clarified for each attribute, if it addresses contextual characteristics (e.g., a clear EA purpose, stakeholders) or the EA model in particular (e.g., semantic and syntactical correctness). To distinguish between the assessment of the whole model and the assessment of each view it was clarified whether the attribute can only be assessed on global model level or depends on a single EA view. For instance, while statements for the attribute "completeness vs. conciseness" can only be made on global level, the attributes "Level of Detail" should be discusses for each EA view. Here, some attributes may also be assessed on both level (e.g., "comprehensibility"). The resulting

Table 8.2.: EA Quality Principles and Their Related Quality Attributes.

Quality Principle	Quality Attribute	Description	Source
Validity	Syntactical Properness	Does the model follow the specifications of the chosen modelling language(s)? EA model may integrate different modelling languages.	[122, 27, 179, 216]
	Semantical Properness	Is the EA model correct in terms of representing the reality in relation to the EA's purpose?	[27, 179, 216]
	Up-To-Dateness	Does the EA model represent the current situation?	[165]
	Quality of Information Sources	Can the information source, on which the EA model view relies be considered correct?	[165]
	Uniformity and Cohesion	The EA model should follow a certain framework behind it. Further, it should represent a coherent aggregate, where all parts of it are integrated with each other.	[122, 165, 111]
	Model Reliability	Does the EA model what it is supposed to do and what is expected from it?	[122, 111]
	Reduction of Redundancy	A beneficial EA model does not hold any duplicates of a model or model elements that only seem to be different but are the same.	[136, 179]
Relevance	EA Purpose and Objectives	In order to properly develop a beneficial EA model a purpose and objectives clearly have to be defined.	[165, 179, 196]
	EA Stakeholders Concerns	Next to the EA purpose in general all involved stakeholders have to be defined and their concerns towards the EA model have to need to be defined.	[129]
	Usefulness	Each EA model should be relevant and beneficial for its user. Every part of the EA model has to be developed for a certain purpose and addresses a set of stakeholders.	[122, 111]
	Level of Detail	An EA model should provide both a holistic view and sufficient level of detail in the relevant areas.	[165, 27]
	Completeness vs. Conciseness	Does the EA model represent on the one hand all necessary information and on the other hand not too much information is provided regarding its purpose.	[165, 136, 179, 216]
Economic	Reusability	For efficient development the reusability of model components has to be enabled. Business of Software Reference Model can help in this regard.	[122, 136]
Efficiency	Flexibility	In order to support the organization's adaptations to environmental changes, the EA model should be able to be highly flexible.	[122, 111, 136]
	Model Maintenance	In the light of continuous model quality, the EA model should be maintained.	[111, 136, 179]
Clarity	Comprehensibility	All aspects represented in the EA model have to be easily understandable for the model user.	[165, 111]
	Layout Design	Each EA model view should provide a clear layout of elements.	[136, 27]
	Complexity	The model complexity should be appropriate regarding its level of detail and purpose.	[165, 179]
	Documentation	The EA model should be sufficiently documented.	[165, 27]

Table 8.2.: EA Quality Principles and its Related Quality Attributes

Quality Principle	Quality Attribute	Description	Source
	Communication	The EA model has to be communicated by the right means. Therefore, it has to be clear who needs what information in what detail at what time.	[165, 136]
Systematic Model Structure	EA Model Structure	Every EA model should ground on a thoroughly predefined model structure. The structure should be aligned with the model's purpose and guide both the modeller and user.	[165, 27, 179]
	Model View Specification	In order to properly maintain an EA model, the intention of each model view has to be made transparent.	[129]
	Model View Linkage	In order to prevent isolated modeling parts, each model view that is created should be clearly related to the EA model structure.	[179]
Comparability	Model Interoperability	The EA model should be exchangeable e.g. to other modelling tools.	[122, 136]
	Inter-Model Relations	Many organizations work with different modelling standards (e.g. BPMN for processes) they should be comparable to the EA model, even when they represent higher levels of detail.	[129, 27]

Table 8.3.: Quality Attributes Addressing the EA Model's Purpose.

Quality Principle	Quality Attribute	Attribute Assessment	Assessment Method
Relevance	EA Purpose and Objectives	Is there a clear purpose for the EA defined?	Yes/No
		Does the EA team define objectives to fulfill the EA purpose	Yes/No
		Are purpose and related objectives regularly revisited?	Yes/No
	EA Stakeholders Concerns	Is there a thorough assessment of stakeholders involved?	Yes/No
		Are the concerns of the stakeholders determined?	Yes/No

Table 8.4.: Quality Attributes Addressing EA Models.

Quality Principle	Quality Attribute	Attribute Assessment	Assessment Method
Validity	Syntactical	Calculate the ratio according to [216]	quantitative
		Validate towards Language Syntax	Yes/No
	Properness Uniformity and Cohesion	Is the EA model based on a EA framework?	Yes/No
		Is a EA development method used?	Yes/No
		Does the EA model conform to a predefined model structure?	Yes/No
		Does the EA model hold any duplicates	Yes/No
	Reduction of Redundancy	Is the EA model repository free from duplicates?	Yes/No
		Conduct Expert Interviews for EA model to identify implicit duplicates	qualitative

Table 8.4.: Quality Attributes Addressing the Whole EA Model

Quality Principle	**Quality Attribute**	**Attribute Assessment**	**Assessment Method**
Relevance	EA Purpose and Objectives	Is there a clear purpose for the EA defined?	Yes/No
		Does the EA team define objectives to fulfill the EA purpose	Yes/No
		Are purpose and related objectives regularly revisited?	Yes/No
	EA Stakeholders Concerns	Is there a thorough assessment of stakeholders involved?	Yes/No
		Are the concerns of the stakeholders determined?	Yes/No
	Completeness vs. Conciseness	Are there modeling guidelines defined addressing what (not) to model?	Yes/No
		Does the EA repository only store used elements?	Yes/No
Economic Efficiency	Reusability	Are reoccurring phenomena reused in the model?	Yes/No
		Does the EA team agree on and use reference models?	Yes/No
	Flexibility	Does the EA model show alternative paths for organizational development?	Yes/No
		Is a process implemented how EA models support decisions?	Yes/No
	Model Maintenance	Does the model conform to the most current version of the modeling language?	Yes/No
		Are outdated parts of the model extracted or deleted?	Yes/No
		Is there a maintenance plan defined?	Yes/No
Clarity	Comprehensibility	Are the EA model elements clearly named?	Yes/No
	Communication	Is there a communication/reporting strategy of the EA model defined?	Yes/No
		Conduct Interviews with stakeholders of EA to reveal whether communication strategy realizes goals.	qualitative
Systematic Model Structure	EA Model Structure	Does an EA model structure definition exist?	Yes/No
		Does the EA structure follow a top-down design?	Yes/No
Comparability	Model Interoperability	Is there an exchange format available for the modeling language in use?	Yes/No
	Inter-Model Relations	Does the EA model use information of other existing models in the organization?	Yes/No
		Are relations between parts of the EA models and other existing models made explicit?	Tool Support

Table 8.5.: Quality Attributes Addressing EA Model Views.

Quality Principle	Quality Attribute	Attribute Assessment	Assessment Method
Validity	Semantical Properness	Conduct Expert Interviews	qualitative
		Conduct Validation Workshops with relevant Stakeholders	qualitative
	Up-To-Dateness	Date of Last Change	quantitative
		Frequency of Change	quantitative
	Quality of Information Sources	Conduct Expert Interviews	qualitative
		Conduct Validation Workshops with relevant Stakeholders	qualitative
	Model Reliability	Conduct Expert Interviews	qualitative
		Conduct Validation Workshops with relevant Stakeholders	qualitative
	Reduction of Redundancy	Is the EA model repository free from duplicates?	Yes/No
		Conduct Expert Interviews for EA model to identify implicit duplicates	qualitative
Relevance	Usefulness	Are the goals of the EA model view clearly defined?	Yes/No
		Is there a ĖA supply chainppresent?	Yes/No
	Level of Detail	How many levels of detail are used by the model view?	quantitative
		Are the different levels of detail made transparent?	Yes/No
		Conduct Feedback Interviews with Stakeholders	qualitative
Clarity	Comprehensibility	How many elements are documented/explained in ratio to all elements.	quantitative
		Does the EA model vocabulary follow a clear taxonomy (e.g. of a certain domain)?	Yes/No
	Layout Design	Does each model view follow a clear layout?	Yes/No
		Are modeling convention developed for certain situations to ensure consistent layout?	Yes/No
		Do view templates exist that can be used for certain views?	Yes/No
	Complexity	Show number of model view elements.	quantitative
		Depending on the view's purpose, is the amount of elements reasonable?	Yes/No
	Documentation	Is the structure of the EA model made transparent?	Yes/No
		Is further material attached that explains ambiguous elements?	Yes/No
		Is external material referenced?	Yes/No
Systematic Model Structure	Model View Specification	Is the intention of each model view explicitly documented?	Yes/No
	Model View Linkage	Does every model view relate to the model structure?	Yes/No
		Are interrelations among model views made transparent?	Yes/No

allocation of attributes can be seen in Table 8.3 (attributes addressing EA purpose), Table 8.4 (attributes addressing EA models) and Table 8.5 (attributes addressing EA model views). Each attribute can consist of several assessment methods, that can be either of qualitative, quantitative, or Yes/No nature. For each model view a separate assessment should be done. Furthermore, we want to emphasize that these attributes offer a guideline to the architect, who is conducting

the EA model quality assessment. Thus, depending on the model's purpose, he may exclude irrelevant attributes.

8.2. Applying the Framework

8.2.1. Case Environment

The case is conducted within the environment already presented in Section 1.4.1. Following, we will present only necessary additional information:

There exist mainly two processes to maintain the elements of the EA model. First, the EA model is used to execute an Application Lifecycle Management (ALM). Second, the EA model is used to execute an Infrastructure Lifecycle Management (ILM), whose results are incorporated into the ALM.

The EA department utilizes the ALM process to calculate the technical fit and the conformance to business demands once a year. Depending on these results the EA department either determines areas of activity to improve technical fit and business conformance or decides in accordance with the business to shut down the application. Moreover, the ALM process is employed to trigger the responsible persons to update all information to applications which are not relevant for the ALM process.

The ILM process is quite similar utilized compared to the ALM process. It is also executed once a year and employed to trigger the responsible persons to update infrastructure information. Especially, the status of the infrastructure is emphasized, i.e. planned, phase in, active, phase out, or end of life. Since several infrastructure components are exploited to realize applications, this status is also essential for the ALM process. Therefore, the latest status of all exploited infrastructure components is included into the ALM rating.

8.2.2. Exemplary Framework Application

We applied EAQF at the beforehand presented case. To answer the questions regarding EA's purpose (cf. Table 8.3), we reused the results of the stakeholder interviews in [79]. Following Patton [176] we conducted a series of open-ended interviews, using a fixed set of questions for all interviewees. These questions dealt with stakeholder concerns. However, questions regarding EA products in general and the EA model in particular were also taken into account. Moreover, the results of [79] could also be used to answer several qualitative questions of the other EAQF parts (Table 8.4 and Table 8.5). Where the results of [79] were not sufficient to answer EAQF questions, we conducted deepening expert interviews with members of the EA unit.

Answering questions regarding the whole EA model (Table 8.4), we took the EA model of the organization into account as well as the results of the interviews. The EA unit notates their model using ArchiMate 2.1 [224] with slight changes. For example, they introduced so called work areas which are used to determine operation costs on the host system and to assign them to certain applications. Last, we answered EAQF's questions regarding a specific EA model view (Table 8.5). Therefore, we chose the application systems portfolio, which illustrates all business domains and which applications are used to realize this business domain. The business domains are modeled as business functions according to ArchiMate 2.1 within the repository.

The applications are modeled as application collaborations. However, both elements are represented as stacked boxes in the view which is not in conformance to ArchiMate 2.1, since the Anwendungssystemportfolio (AWP) view is older than the decision for ArchiMate 2.1.

Quality Attributes addressing the EA Model's Purpose

The EA model's purpose is defined at our case. The overall assessment can be found in Table 8.6. On the one hand, it is used to do a guided life cycle management, i.e., ALM and ILM. On the other hand, it is used for information and decision demands, e.g., to spread who is in response for a certain application or to offer needed data for an informed decision of the management.

According to EAQF the EA model's purpose quality lacks only in one point: the purpose is not regularly revised. The EA unit performed stakeholder interviews in which, inter alia, stakeholders' demands towards the EA model were inquired. However, it is not planned to perform such interviews regularly and to revise the purpose meanwhile.

Quality Attributes addressing the EA Model

The results of assessing the EA model can be found in Table 8.7. As already mentioned, the EA model is notated in ArchiMate 2.1 with slight changes. Consequently, the validity is not perfect, which is also reflected in Spence and Michel's ratios [216]: Q_s = 53,6%, Q_a = 71,4%, and Q_c = 100%. The value of Q_s indicates that nearly the half of the elements of ArchiMate 2.1 is not used within the repository. Moreover, almost one third of the contained element types in the repository is not actively used (Q_a).

Another shortcoming of the model identified by EAQF can be situated in the principle of relevance. EAQF asks for modeling guidelines which lay down what (not) to model. Those guidelines are not explicitly formulated in our case. Rather, there exists some kind of oral tradition to pass on what (not) to model to new architects.

Further quality flaws can be found in the context of economic efficiency. For example, the repository contains, apart from unused element types, elements which have not been updated for a long time, even though, their representation has been changed. However, those elements were not considered for reports or the like.

Regarding clarity, the stakeholder concerned especially more up-to-date information. They stated that the used communication channels could be useful, but as long as the information is not up-to-date the communication channels are useless.

In the cases of systematic model structure and comparability we could not uncover any issues related to EA model's quality. This is grounded in the fact that ArchiMate 2.1 is used as modeling language and, consequently, ArchiMate supports all requested quality attributes for these two principles.

Quality Attributes Addressing a Specific EA Model View

Stakeholders perceive the validity of the considered view as sufficient, which can be seen in Table 8.8 summarizing the results of the assessment. They remarked only a lack of up-to-dateness

Table 8.6.: Results of Assessing the EA Model's Purpose.

Quality Principle	Quality Attribute	Attribute Assessment	Assessment Result
Relevance	EA Purpose and Objectives	Is there a clear purpose for the EA defined?	Yes
		Does the EA team define objectives to fulfill the EA purpose	Yes
		Are purpose and related objectives regularly revisited?	No
	EA Stakeholders Concerns	Is there a thorough assessment of stakeholders involved?	Yes
		Are the concerns of the stakeholders determined?	Yes

Table 8.7.: Quality Attributes Addressing EA Models.

Quality Principle	Quality Attribute	Attribute Assessment	Assessment Result
Validity	Syntactical Properness	Calculate the ratio according to [216]	$Q_s = .536$ $Q_a = .714$ $Q_c = 1.00$
		Validate towards Language Syntax	No
	Uniformity and Cohesion	Is the EA model based on a EA framework?	Yes
		Is a EA development method used?	Yes
		Does the EA model conform to a predefined model structure?	Yes
		Does the EA model hold any duplicates	No
	Reduction of Redundancy	Is the EA model repository free from duplicates?	No
		Conduct Expert Interviews for EA model to identify implicit duplicates	No duplicates
Relevance	EA Purpose and Objectives	Is there a clear purpose for the EA defined?	Yes
		Does the EA team define objectives to fulfill the EA purpose	Yes
		Are purpose and related objectives regularly revisited?	No
	EA Stakeholders Concerns	Is there a thorough assessment of stakeholders involved?	Yes
		Are the concerns of the stakeholders determined?	Yes
	Completeness vs. Conciseness	Are there modeling guidelines defined addressing what (not) to model?	No
		Does the EA repository only store used elements?	No
Economic Efficiency	Reusability	Are reoccurring phenomena reused in the model?	No
		Does the EA team agree on and use reference models?	No
	Flexibility	Does the EA model show alternative paths for organizational development?	No
		Is a process implemented how EA models support decisions?	Yes
	Model Maintenance	Does the model conform to the most current version of the modeling language?	No
		Are outdated parts of the model extracted or deleted?	No
		Is there a maintenance plan defined?	Yes

Table 8.7.: Quality Attributes Addressing the Whole EA Model

Quality Principle	Quality Attribute	Attribute Assessment	Assessment Result
Clarity	Comprehensibility	Are the EA model elements clearly named?	Yes
	Communication	Is there a communication/reporting strategy of the EA model defined?	Yes
		Conduct Interviews with stakeholders of EA to reveal whether communication strategy realizes goals.	No interviews
Systematic Model Structure	EA Model Structure	Does an EA model structure definition exist?	Yes
		Does the EA structure follow a top-down design?	Yes
Comparability	Model Interoperability	Is there an exchange format available for the modeling language in use?	Yes
	Inter-Model Relations	Does the EA model use information of other existing models in the organization?	Yes
		Are relations between parts of the EA models and other existing models made explicit?	Yes

as a negative characteristic. Updating the view up to three times a year is not satisfactory to stakeholders' needs. The stakeholder concern as short update cycles as possible.

Relevance and Clarity are two principles which evoke no issues related to quality. According to EAQF, references to external material are needed. However, the stakeholder interviews have shown that contained information is sufficient and external references for the purpose of the view are not necessary.

In the field of systematic model structure, the relations between various views are not made explicit. Moreover, the intention of the view is not explicitly formulated.

Improvement Potentials of the EA Model

After conducting the assessment of the EA model, the results can be interpreted to improve the quality. Generally, result that is not as wanted indicates improvement potentials. For example, one usually wants for Yes/No assessments a Yes.

In the case of our exemplary application, we observe that nearly the half of the elements of ArchiMate 2.1 is not used within the repository and almost one third of the contained element types in the repository is not actively used. This reveals potential improvements of the model. It is arguable if it is necessary to use all elements of ArchiMate 2.1, but at least the number of not used element types can be reduced. This would lead to a clearer structure of the repository and would, consequently, heighten its quality.

Further, we notice that a couple of elements within the repository have not been updated for a long time, even though, their representation has been changed. Additionally, those elements are not incorporated in any reports. Consequently, EA model's quality could be raised by removing those.

The investigated view offers improvement potentials. In the field of systematic model structure, the relations between various views are not made explicit. This is an existing flaw of this view and should be corrected soon consonant with stakeholders' opinion. Moreover, the intention of the view is not explicitly formulated. This should be aligned as well.

Table 8.8.: Quality Attributes Addressing EA Model Views.

Quality Principle	**Quality Attribute**	**Attribute Assessment**	**Assessment Result**
Validity	Semantical Properness	Conduct Expert Interviews	Yes
		Conduct Validation Workshops with relevant Stakeholders	Yes
	Up-To-Dateness	Date of Last Change	17.05.2017
		Frequency of Change	several times a year
	Quality of Information Sources	Conduct Expert Interviews	Yes
		Conduct Validation Workshops with relevant Stakeholders	Yes
	Model Reliability	Conduct Expert Interviews	Yes
		Conduct Validation Workshops with relevant Stakeholders	No
	Reduction of Redundancy	Is the EA model repository free from duplicates?	Yes
		Conduct Expert Interviews for EA model to identify implicit duplicates	Yes
Relevance	Usefulness	Are the goals of the EA model view clearly defined?	Yes
		Is there a "EA supply chain" present?	Yes
	Level of Detail	How many levels of detail are used by the model view?	2
		Are the different levels of detail made transparent?	Yes
		Conduct Feedback Interviews with Stakeholders	Yes
Clarity	Comprehensibility	How many elements are documented/explained in ratio to all elements.	100%
		Does the EA model vocabulary follow a clear taxonomy (e.g. of a certain domain)?	Yes
	Layout Design	Does each model view follow a clear layout?	Yes
		Are modeling convention developed for certain situations to ensure consistent layout?	Yes
		Do view templates exist that can be used for certain views?	Yes
	Complexity	Show number of model view elements.	203
		Depending on the view's purpose, is the amount of elements reasonable?	Yes
	Documentation	Is the structure of the EA model made transparent?	Yes
		Is further material attached that explains ambiguous elements?	Yes
		Is external material referenced?	Yes
Systematic Model Structure	Model View Specification	Is the intention of each model view explicitly documented?	No
	Model View Linkage	Does every model view relate to the model structure?	Yes
		Are interrelations among model views made transparent?	No

Chapter 9.

Improving the Design of EA Models

Before, we have presented a means to assess the quality of EA models. This assessment can also be utilized to identify quality weaknesses of the model after the model has been created. Another approach, which we like to present next, is to support the modeler in a way that she does not introduce quality flaws into the EA model.

Motivation One of the major problems in EA is to keep the EA model up-to-date and keeping on an adequate level of quality [141, 61]. However, this maintenance is still dominated by error-prone manual work [61, 240]. As the projects are one of the main drivers for changes of the EA [213], we elaborated in our research on means to support solution architects in their work modeling changes in the EA. The results of this research are presented in the following sections and have initially been presented in [32] (section 9.1) and [185] (section 9.2).

9.1. Avoiding Redundancies in EA Models

Enterprise architects, are inter alia responsible for modeling the EA using architecture models. The fundamental building blocks of the models are the components and the relations between them. For easier understanding and communicating of the architecture to the stakeholders, the architects can develop a set of representations of the overall architecture called views.

We investigate the scenario of a company (cf. Section 1.4.1) that uses a repository of all accepted models. In time, the repository can grow into one complex structure of components and relations. Adding a new model can cause unnecessary expansion in the repository if they are not checked beforehand. Components with same attributes, or components used in the same context but with different names will be treated as newly introduced components and added again in the repository. We take a simple illustration to depict the problem. The models provided in Figure 9.1(a) and Figure 9.1(b) are developed independently. After the acceptance of both, the two models form the initial state of the repository, depicted in Figure 9.2.

In the repository, *Transaction Administration* aggregates both *Accounting* and *Billing* components from different models, while retaining only one instance of *Transaction Administration*. An architect decides to implement the similar financial scenario in a different model (Figure 9.3). The system proceeds to integrate this model in the repository, which results in several replicated components, introducing three new components (see Figure 9.4):

– *Billing*: The architect supplied a shorter name for the component *Billing Component*.

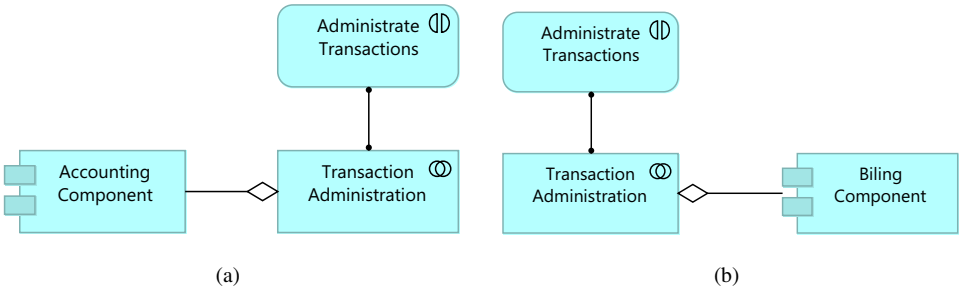


Figure 9.1.: Different EA Models Example.

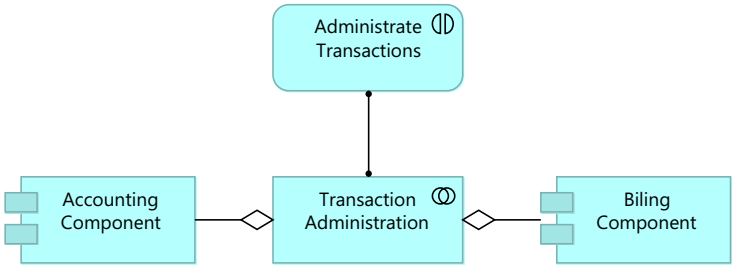


Figure 9.2.: Simplified Scenario of the State of the Repository Formed from the Models Described in Figure 9.1.

- *Transactions Administration*: Providing behavior for administrating several transactions can be replaced by the already existing *Transaction Administration*.
- *Manage Transactions*: Although the name differs from the *Administrate Transactions*, they provide the same functionality.

Currently, there is no mechanism for redundancy check. Components with similar attributes and purpose can be stored multiple times, which in turn introduces management issues due to the increased complexity of the repository data. This also makes it difficult for architects to reuse certain components.

To elaborate on this issue, we opt for DSR as presented in Section 1.4.3. For more details on the procedure of applying DSR in this case, we refer to Appendix A.6.

9.1.1. Theoretical Background

This section describes the theory behind our approach. First, we define similarity and distance in the scope of our research. Then, we present the fundamental aspects of graph theory used in our proposed solution. Finally, we formalize association rules and a set of parameters used for association analysis.

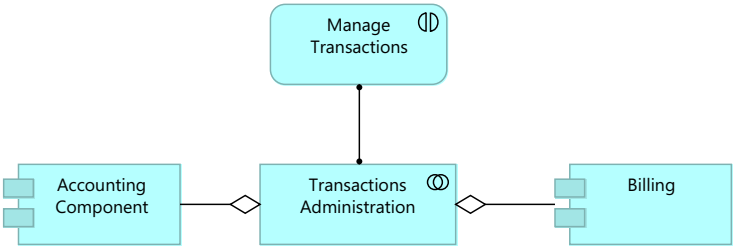


Figure 9.3.: Model Containing Redundant Components.

Similarity and Distance

With the use of complex objects, we identify the need of fundamental operation for similarity assessment between two objects. If we consider the following spaces: $\mathbb{F}$, which denotes the feature space of an object and $\mathbb{R}^{\mathbb{F}}$ - the space of all feature representations, then such function will map the future space to a score $s : \mathbb{R}^{\mathbb{F}} \times \mathbb{R}^{\mathbb{F}} \to \mathbb{R}$.

Similarity function is the measure which determines how closely related two objects are based on their representation, following the given properties[198]:

- Symmetry: $\forall x, y \in \mathbb{R}^{\mathbb{F}} : s(x, y) = s(y, x)$ - the order of the objects in the input should not affect the output score
- Maximum self-similarity: $\forall x, y \in \mathbb{R}^{\mathbb{F}} : s(x, x) \geq s(x, y)$ - nothing can be more similar than the object itself

If two objects are similar, then the similarity function will have a high positive score.

In contrast to similarity, the *dissimilarity* is a measure defined by a distance function that quantifies how different two objects are. For function $d : \mathbb{X} \times \mathbb{X} \to \mathbb{R}$ to qualify as distance, in needs to fulfill the following constraints [53]:

- Non-negativity: $\forall x, y \in \mathbb{X} : d(x, y) \leq 0$
- Reflexivity: $\forall x \in \mathbb{X} : d(x, x) = 0$
- Symmetry: $\forall x, y \in \mathbb{X} : d(x, y) = d(y, x)$

Both dissimilarity and similarity models express the closeness between objects. The conversion from the first to latter is essentially converting from distance to similarity function. Since they are negatively correlated, any monotonically decreasing transformation can be applied to convert similarity measures into dissimilarity. Consequently, any monotonically increasing transformation can be applied to convert the similarity to distance. If the similarity values are normalized in the range from 0 to 1, then the corresponding dissimilarity (distance) can be expressed as:

$$d(x, y) = 1 - s(x, y) \tag{9.1}$$

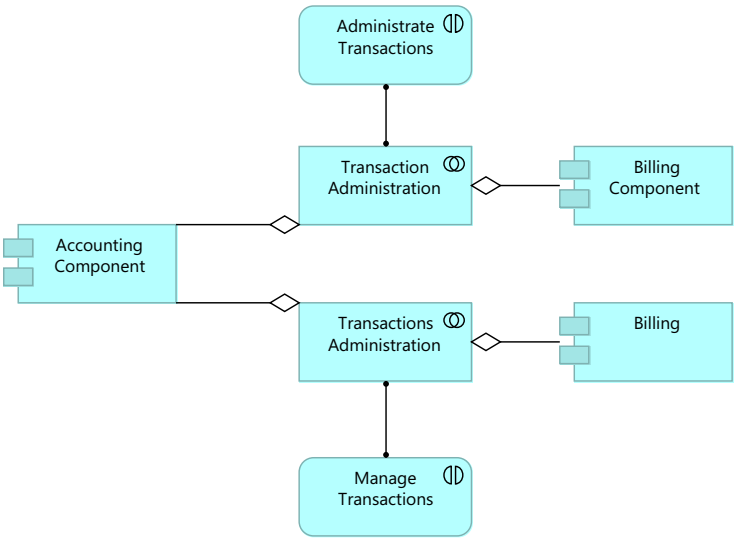


Figure 9.4.: Simplified Scenario of the State of the Repository Formed by Adding the Model in Figure 9.3 to the Repository in the State as in Figure 9.2.

Graph Theory

In order to search for similarities between the EA models, a proper representation is needed. We find that representing the models as labeled graphs is the most acceptable solution [42].

A **labeled graph** is defined by a tuple $G = (V, E, r_V, r_E)$ such that:

- V is a finite set of vertices,
- E is a finite set of edges between the vertices,
- $r_V \subseteq V \times L_V$ is the function that assigns labels to vertices,
- $r_E \subseteq V \times V \times L_E$ is the function that assigns labels to edges.

In this manner, we can represent the EA components as vertices and the connections between them as graph edges.

To calculate the similarity between vertices in a graph, both the labels of the vertices and the edges in the graph need to be considered. The SimRank [103] algorithm takes this into consideration. It accepts a labeled graph *G* as input and compares each vertex of the graph with the rest. Two vertices are considered similar if they are referenced by other similar vertices in the graph, or formally expressed with by Eq. 9.2:

$$s^{SR}(p,q) = \frac{C}{|I(p)||I(q)|} \sum_{i=1}^{|I(p)|} \sum_{j=1}^{|I(q)|} s^{SR}(I_i(p), I_j(q)) \tag{9.2}$$

The constant $C \in \mathbb{R}$ is a user given value from 0 to 1 called the decay factor, $I(p)$ and $I(q)$ are the set of all predecessor vertices of p and q(other nodes who point to p and q) with the total count of $|I(p)|$ and $|I(1)|$ accordingly, and $I_i(p)$ and $I_j(q)$ are the i-th and j-th predecessor of the nodes p and q. Dividing by the total number of predecessors pairs allows us to obtain normalized value: a range between 0 (maximum dissimilarity) to 1 (same pair of nodes). For any vertex v that has no predecessors ($I(v) = \emptyset$), the similarity is set to zero. Alternatively, the equation 9.2 can be expressed using the set of successors $O(p)$ and $O(q)$of the nodes p and q (vertices pointed by p and q), or (as shown later in our approach) both $I(p)$ and $O(p)$ combined.

SimRank is calculated recursively: two score between two vertices is dependent of the pre-calculated similarity of their neighbors. The initial similarity score is calculated using the binary similarity:

$$s_0^{SR}(p,q) = \begin{cases} 1 & \text{if } p = q \\ 0 & \text{else} \end{cases} \tag{9.3}$$

Association Analysis

Association rule mining [88] searches for recurring relationships in a given data set. More specific, it discovers the associations and correlations between two item sets.

An association rule is indicated as $\{A_1, A_2, ..., A_m\} \Rightarrow \{B_1, B_2, ..., B_n\}$, where $\forall i, j \mid i \leq m, j \leq n$, A_i and B_j are non-empty *item sets*. In order to select only the rules that are interesting for evaluation, we use the *support* and *confidence* functions. Support *supp* for the rule $A \Rightarrow B$ indicates the fraction of transactions that contain both *A* and *B*:

$$supp(A \Rightarrow B) = P(A \cup B) \tag{9.4}$$

$supp$ filters out rules that occurred by chance. A rule with support above a given *minimum support* threshold ($minsup$) is called *frequent item set*. Confidence $conf$ for the rule $A \Rightarrow B$ indicates the fraction of items contained in B that are also contained in A. It measures the reliability made by a rule:

$$conf(A \Rightarrow B) = P(B|A) \tag{9.5}$$

Rules with $supp \geq minsup$ and $conf \geq minconf$ are called *strong rules*. For generating rules that satisfy the minimal support and confidence, we use the *Apriori* algorithm [3]. This approach considers only the frequent items as basis for generating candidates and extends them to larger item sets with other frequent items.

9.1.2. Research Proposal

In this section we give an overview how the similarity models were applied to suit our needs.

Prerequisites

For a given EA model that contains the intended changes conducted by a project (following referred to as project model), we make the following assumptions:

– The architecture model is complete: the calculation of the candidate set of components, which might have already a representation in the architecture model, is based on the assumption that all the necessary components and relations are there.

– The views in the architecture projects are clearly defined: this allows successful application of the association mining. Since there is no explicit notion for a transaction, we rely on the views - every component that is a part of the view belongs in the same transaction.

To categorize a component from the model as a newly introduced, we check if a component with the exact name and type does not exist in the repository. We ignore the description attribute since this field is used very extensively and already small discrepancies in the description would lead to a false negative.

Feature Extraction and Similarity Models

The underlying representation of the given EA model provided by the project is a labeled directed graph. Each node of the graph presents a component with the features: *name* (or title) and *type*, whereas the relations between them are modeled as edges. The same representation applies to the central EA model. For two labeled directed graphs, we combine **attribute based** and **structure based** similarities.

Attribute-based similarities calculate the similarity score between two features while ignoring the graph structure. For evaluating the similarity between the names of the components, we apply the *String Edit Distance* (d^{EDIT}) [161] as a basis with regularization. The edit distance is very robust when it comes to comparing single words, but results in large distance score if the strings contain words in different order. To overcome this, we apply the distance measure on pairs of words [119]. We introduce word tokenization, remove any digit characters from the words and discard the words that do not contribute to semantic meaning, such as personal pronouns and definite or indefinite articles.

Let *p* and *q* be two components and *p_words* and *q_words* be the tokenized and processed titles of the first and second component respectively. For each word from p_words, we apply the String Edit distance to measure the difference with every word from q_words, convert it to a similarity score and record only the highest word–wise similarity that occurred for that given word. In the case of difference in the number of words, we compare each word from the input string that has more words to avoid any loss of information. To achieve normalization between 0 and 1, we divide the similarity value by the larger number of words for the given titles. This is shown in Listing 1.

For evaluating the similarity between the types of components, we apply the binary similarity. This is a very restrictive similarity that suits the assumption that the EA models are correct and the components have the right type. For two given components p and q with their respective types t_p and t_q, the type similarity is:

$$s^{TYPE}(p,q) = \begin{cases} 1 & \text{if } t_p = t_q \\ 0 & \text{else} \end{cases} \tag{9.6}$$

For calculating the similarity based on the description of components, we rely on the semantic meaning of the text. The description of the text is converted to a Term Frequency - Inverse

Algorithm 1 Name-based similarity

```
1: procedure s^NAME(p,q)
2: p_words = tokenize(p)
3: q_words = tokenize(q)
4: less_words = min(p_words, q_words)
5: more_words = max(p_words, q_words)
6: total_similarity ← 0
7: for each m in more_words:
8:     word_similarity ← 0
9:     for each l in less_words do:
10:        current_similarity = 1 − d^EDIT(m,l) / max(size(m),size(l))
11:        if current_similarity > word_similarity then
12:            word_similarity = current_similarity
13:    total_similarity = total_similarity + word_similarity
14: return total_similarity / size(more_words)
```

Document Frequency (TF-IDF) vector[163]. This metric was motivated by the length of the text: the String Edit Distance will not be able to perform well since it relies on the syntactic matching of the words. TF-IDF emphasizes the importance of a word based on how frequent they appear for the given component's description (term frequency) and how rarely they appear throughout other component's description (document frequency).

Let *t* be a word from the description *d*, *N* the total number of documents and *D* the number of documents where *t* appears. The TF-IDF score for *t* is calculated as:

$$tf\_idf(t, d, D) = word\_count(t, d) log \frac{N}{1 + D}, \tag{9.7}$$

where *word_count(t,d)* returns the number how many times the word *t* has appeared in the document *d*.

After obtaining the TF-IDF vector for each word in the description $desc_p$ and $desc_q$ of the components *p* and *q*, the description similarity is calculated using the cosine distance:

$$s^{DESC}(p, q) = 1 - d^{COS}(tf\_idf(desc_p, D), tf\_idf(desc_q, D)) \tag{9.8}$$

The function $tf\_idf(desc_p, D)$ returns a TF-IDF vector for every word from the description $desc_p$ given the previously obtained corpus *D* from every description in the repository.

To combine the different similarities into one, we use a weighted average function from the similarity models:

$$s^{ATTR}(p, q) = \frac{w_1 s^{NAME}(p, q) + w_2 s^{TYPE}(p, q) + w_3 s^{DESC}(p, q)}{w_1 + w_2 + w_3} \tag{9.9}$$

The **context-based** similarity is calculated based on the graph structure of the EA models. For this, we change the input of the original SimRank approach $s^{SR}(p, q)$. For the initial cases

$s_0^{SR}(p,q)$, we assign maximal similarity if the titles are a complete match and the components have the same type.

To come up with combined context–based and attribute–based similarity, we integrate the attribute similarity in the SimRank approach. We call this model *Extended SimRank* (Eq. 9.10). To achieve this, we relax the initial similarity s_0^{ESR} by assigning a score calculated from the attribute similarity model, if the score is above a given threshold *t* (Eq. 9.11). We also check both the predecessors and the successors of any node *v* for the context-based similarity: $D(v) = I(v) \cup O(v)$.

$$s^{ESR}(p,q) = \frac{C}{|D(p)||D(q)|} \sum_{i=1}^{|D(p)|} \sum_{j=1}^{|D(q)|} s^{ESR}(D_i(p), D_j(q)) \tag{9.10}$$

$$s_0^{ESR}(p,q) = \begin{cases} s^{ATTR}(p,q) & \text{if } s^{ATTR}(p,q) > t \\ 0 & \text{else} \end{cases} \tag{9.11}$$

Inspecting every pair of vertices between graphs with *n* and *m* nodes leads into generating $n * m$ candidates. To speed up the process of evaluation, we skip the nodes which do not have any incoming and outgoing edges, since such nodes cannot contribute to the structural similarity when using s^{ESR}. For such nodes, we rely only on the s^{ATTR}.

9.1.3. Implementation

The implementation of our solution is dependent on several technologies. Figure 9.5 gives an overview of the architecture of our solution. The **server side** is built using the Java technology. The server returns for a POST request a list of candidates for duplicates. The request accepts the following parameters:

- file: mandatory field which contains the ArchiMate (Extensible Markup Language (XML)) file that needs to be evaluated.
- k: a number of maximum returned components for a single query component (optional).

The ArchiMate files and the repository are read and converted to directed labeled graphs using the JGraphT library [208] and its *DirectedGraph* class. Each XML node represents either an ArchiMate component, if located under the "Components" section, or an ArchiMate relation, if located under the "Relations" section.

To get a better understanding of the structure of the repository as a graph, we use *Gephi* - a tool for analytics and detailed visualization of graphs [26]. The total size of the graph is 3,922 nodes with 9,657 edges. Out of those, 1,147 are isolated nodes (no incoming and outgoing edges). The diameter is 7, and the average path length is 4.79. The average degree per node (both incoming and outgoing considered) is 4.93. Given graph size, we consider the repository graph as a weakly connected. This is also confirmed by the low value of 0.001 for the density.

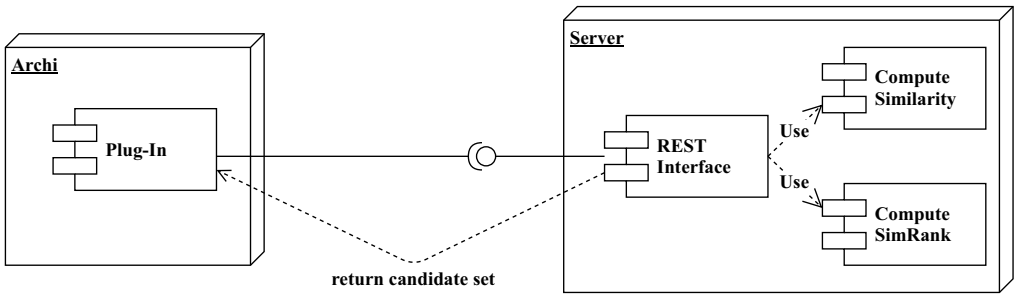


Figure 9.5.: Architecture of Our Solution Calculating a Candidate Set of Duplicates.

Before calculating any similarity, we filter out any unnecessary replicated information which might affect the prediction outcome at the end. The repository is cleaned up by discarding all the replicated components, i.e., components with the same *name* and *type* (the *description* feature is not mandatory, therefore not considered a factor). This results in a reduction of 28 components.

The description of every component is converted into a TF-IDF vector. For this, a corpus needs to be built where each description of a component is considered as a document. Every word (term) gets evaluated using equation 9.7. For tokenization of the title of the components, we use the *WordTokenizer* class from the WEKA library for Java. For the association mining we use the R package *arules*. This package provides functions to read a Comma Seperated Values (CSV) file as transactions and to generate the frequent item sets using the Apriori algorithm. The chosen value for minimal support was 0.17 was the largest value where rules were still generated. Combined with the minimum confidence value of 0.75, the algorithm resulted in the generation of 77 rules.

The **client side** was realized as a plug-in for the Archi tool[1], which is an Eclipse-based Integrated Development Environment (IDE). The plug-in provides a button on the toolbar of the IDE, which allows the user to select the desired ArchiMate model file. Afterwards, the file is uploaded to our server and the module waits for a response back. The response contains a JavaScript Object Notation (JSON) list of components that are not part of the repository and their most similar components from the repository. The result is presented as a dialog with a tabular view inside Archi.

9.1.4. Evaluation

The similarity models we applied are unsupervised, meaning we do not have the true output available. To successfully evaluate them, we manually created a simulation and architecture model. The repository data consisted of 327 nodes and 275 edges and the model of 35 nodes and 23 edges. The model components were provided from two sources chosen pseudo-randomly from the repository and inserted without any relations to the repository.

A subset of 16 model components were subject to manual change of the attributes, so that different scenarios for similarity can be tested based on a title, description, type, structural simi-

[1] https://www.archimatetool.com/

larity and combination of all. Only the nodes that did not appear in the repository were tested. There were 20 unidentified nodes in total, out of which four did not have any substitution. For every evaluation test, the k value was set to the lowest value of 1, which evaluates the shortest result list.

We also set up a simulation of an ArchiMate model for creating models like projects would do. The simulation file consisted of the same number of components and edges as the previous simulation model file, distributed in the same number of views. We replaced each unidentified component with its counterpart from the repository if such existed. Using the approach of creating transaction per view level, we created a set of five transactions with 18 items. Finally, we discarded newly introduced components without substitution as well as the components that did not belong to any view.

For evaluating the correctness of the similarity models, we compared three different metrics: accuracy, precision, and recall[206]. For the association rule mining, we focused only on the accuracy, since the results were returned in the form of "if the components on the left–side of the rule exist, then the components on the right–side of the rule might be of interest". Therefore we could not make the connection which result components belong to which query components. All metrics range from 0 (the lowest value) to 1 (highest value).

We evaluated the similarity models each one separately, as well as the combination between them. The overview is given in figure 9.6.

The first evaluation was performed on each feature similarity model separately. We set the number of returned components to one ($k = 1$). The title and type similarity showed poor performance in every metric. The description similarity model showed maximum precision value and better values for accuracy and recall. However, since the description is optional for the components, it cannot be taken as a single metric. The usage of a single feature similarity model is not recommended as they do not provide an effective recommendation service.

Next, for the evaluation of the weighted similarity combination s_{ISO} we configured the similarity using the values 0.5, 0.1 and 0.4 for the weights w_1, w_2, and w_3 respectively. The weights were provided from a domain expert and reflected the importance of each feature. To avoid results with components with low similarity score, we introduced a threshold $t = 0.5$. The weighted combination performed better than any separate feature similarity model if the number of suggested components is taken into account as well, which was 17 for the given threshold.

Next, we evaluated the SimRank approach, using the combined in- and out-degree. We noticed the improvement in the score compared to the weighted combination, so we incorporated the two methods together as the *SimRank Extended* s^{ESR} with a threshold of 0.5. The *SimRank Extended* showed the highest accuracy, recall, and the highest F1 score. Increasing the number of returned components k to higher number did not affect the score in our simulation.

We performed the association mining with the following input: minimal support of 0.2 (the highest support value that still returns results), and confidence of 1 (the maximal possible value). This generated a set of 293 rules. Out of those, for the model that we evaluated, 13 components were returned. We identified a total of 7 correctly suggested components that were a suitable replacement, with an accuracy of 0.54. The rules had a low support value, which means that the component sets did not often appear in transactions. However, confidence had the highest value, thus giving a high level of certainty concerning the truth of the association rules. However, this

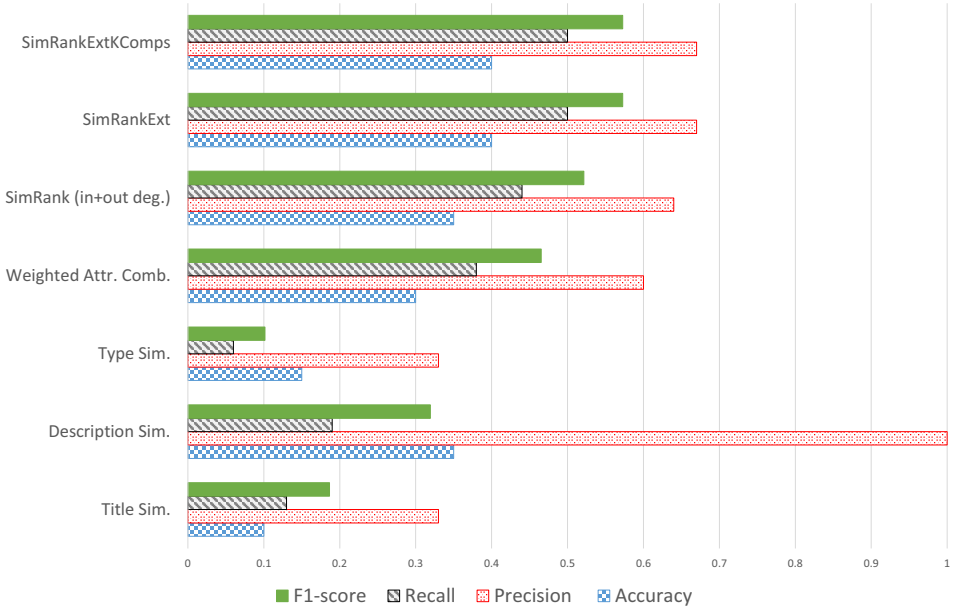


Figure 9.6.: Comparison of all Similarity Models.

method is highly dependent on the data set provided by the architects.

9.1.5. Limitations

As we rely on a graph-like presentation of the EA model, our approach can be generalized and applied also to other models which can be presented as graphs. We assume that, for instance, our approach might also work for UML models as ArchiMate and UML class diagrams are quite similar.

The current limitations are the constraints we impose before evaluating the EA models. The EA models have to be complete and correct, which means that the tool cannot be used as a recommendation system (suggesting components as the model is in the process of creation). This limitation may be solved in the future by incorporating techniques from the model recommendation domain. Also the approaches are not optimized: as the size of the repository increases, recommending a list of components takes more time. Consequently, we will elaborate on this point in the future. Lastly, our approach cannot be fully automatized since we rely on a human expert to confirm that the recommended components are the right substitution. We do not see any possibilities to overcome with this issue.

9.2. A Performance Comparison of Graph Analytic Methods

In large organizations, the nature of EA data makes it difficult to quickly select, tune and apply graph analytic and ML algorithms to provide support for maintaining the EA model. Thus, much time is spent in searching and selecting the best performing model. To cope with these challenges, we set up a ML study addressing the following research questions:

RQ 9 *Which specific ML algorithms are suited to support EA model maintenance?*

RQ 10 *What kind of decision can be made out of different graph analytic techniques?*

RQ 11 *Which algorithm or metric fits best to a particular use case?*

To tackle the challenges mentioned above, we provide generic approaches which address a specific kind of challenges in EA model maintenance; including the identification of duplicate components in the EA repository. To address this issue, we investigate different similarity techniques to find a match between EA models.

We also propose and evaluate a nearest neighbor approach, an alternate method for SimRank [103]. This approach helps to analyze the underlying structures of complex EAs by providing insights from the processes and functions used in different layers of the EA.

Motivated by the life-cycle model for ML frameworks provided by Sapp et al. [199], we propose a general work-flow framework to analyze EA models by comparing different graph analytic and ML algorithms and then selecting the best-performing approach based on suitable performance metrics. More details on the work-flow can be found in Appendix A.7.

9.2.1. Graph Analytic Approaches

Previous research has identified the potential of network science applied to the context of EA analysis [204]. We depict the EA model as a network-graph representing ArchiMate components as nodes and relationships between the components as edges in the graph. Thus, analyzing EA model as network-graph provides a deeper understanding of the relation between the components. As there are several graph analytic techniques, we focus on graph-based similarity and community detection (unsupervised learning).

Similarity Between Enterprise Architecture Models

Considering EA model as a graph like structure [197], we apply graph similarity techniques to assess the match between EA models. The graph similarity involves determining a degree of similarity between the two graphs [117] quantified by a similarity score between 0 (no similarity) and 1 (complete similarity). We investigate the similarity between EA models based on the node and edge information. The experiment results provide a way to compare the relatedness between two EA models. We then compare and evaluate different similarity metrics to select the best-performing metric.

Computing node similarity

The first similarity measure we study, namely node matching similarity, is based on examining the content of the elements/components name of EA models. Figure 9.7 shows two models (Business-Application alignment model) representing two simple instances for boarding and departure process of an airport departure system. Although the naming of components for Model m_2 (Figure 9.7(b)) is slightly different from the Model m_1 (Figure 9.7(a)), both the models illustrate the same boarding process of an airport departure system.

We consider two ways of measuring the similarity between elements of different EA models: syntactic similarity and Latent Semantic Analysis (LSA).

Syntactic Similarity The term syntactic refers to the structure of words and phrases. The syntactic approach is related to both, the occurrence of terms and the number of words in the text/sentence. In this work, we use the Cosine similarity index [17] and Jaccard similarity index [166] as a syntactic approach to detect the match between components of two EA models. To find the Cosine and Jaccard similarity score between the phrases (component names) present in two EA models, phrases are turned into words, words are then converted to vectors with 0s and 1s.

Definition 9.1 (Cosine similarity) *Given two set of words A and B represented as vectors, Cosine similarity is measured by using the word vectors as in below equation:*

$$Cosine(A, B) = \frac{A.B}{||A||||B||} \tag{9.12}$$

where $A.B$ is the intersection (dot product) between the word vectors A and B, $||A||$ and $||B||$ represents the vector length of A and B respectively and is calculated as such:

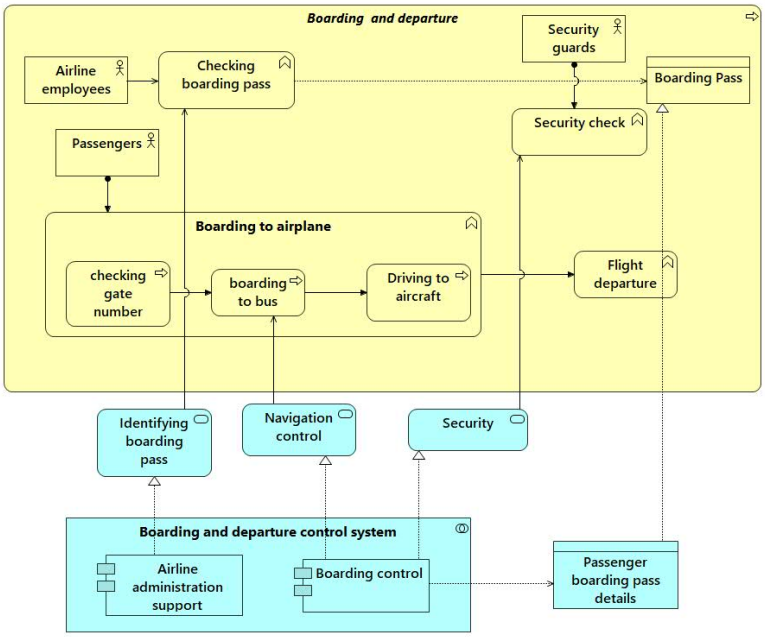
$$||A|| = \sqrt{\sum_{i=1}^{n} A_i^2} \quad and \quad ||B|| = \sqrt{\sum_{i=1}^{n} B_i^2}$$

Apart from the cosine similarity, another well-known measure for determining the degree of similarity is the Jaccard similarity index. The Jaccard similarity measures the similarity between finite sample sets and is defined as the cardinality of the intersection of sets divided by the cardinality of the union of the sample sets [101].

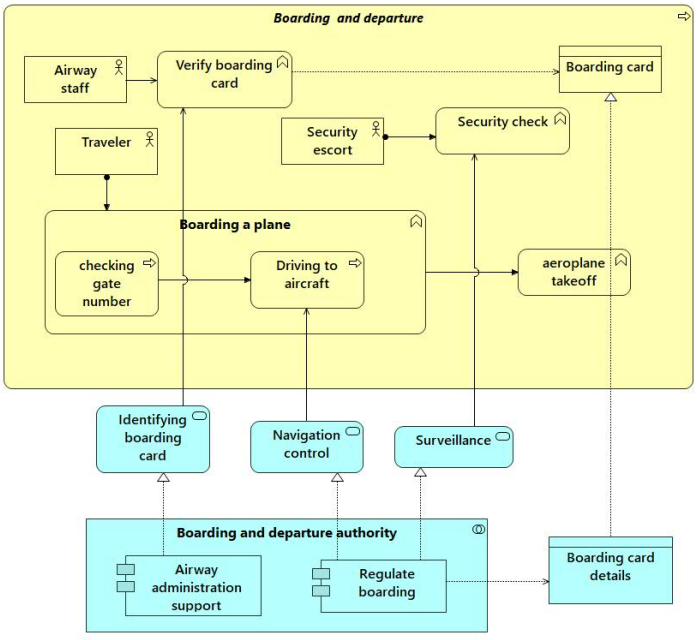
Definition 9.2 (Jaccard similarity) *Given two sets of words, A and B, the Jaccard similarity is computed using the following equation:*

$$Jaccard(A, B) = \frac{A \cap B}{A \cup B} \tag{9.13}$$

where $A \cap B$ represents the intersection of sample sets and $A \cup B$ represents the union of the sample sets.



(a) Model m_1



(b) Model m_2

Figure 9.7.: EA Models Representing Two Simple Instances of an Airport Departure System to Illustrate Syntactic Similarity.

Latent Semantic Analysis An alternative method to compare the similarity between EA models is based on fetching the relations between words in the texts. LSA [123] is a corpus based method which does not use the semantic network, grammars, syntactic and dictionaries. The main idea behind LSA was to overcome techniques that exclusively try to match search queries with only the presence of words in a document as discussed in the previous section. The intuitive idea behind search is based on the related concept of the documents. The difficulty arises when we want to compare concepts or meaning behind the words. LSA tries to overcome this problem with a statistical analysis of the latent structures of the documents by finding the underlying meaning or concepts between the documents. It tries to map the words in a document into a concept space and then comparing in the space.

9.2.2. Computing edge similarity

The second similarity metric we study, is a similarity metric over the structure of the EA model, by considering EA model as a network graph. This method is applicable when one knows the node correspondence for, e.g., the number of nodes present in the two graphs is equal, and edges do not vary much across two graphs. The main intuitive idea behind this approach is, by knowing the node correspondence, a node in one graph is said to be similar to a node in another graph if they share a standard set of neighborhoods. Again, its neighbors are similar if their neighbors are similar and so on [192]. This method helps enterprise architects to keep track the changes between similar kind of EA models.

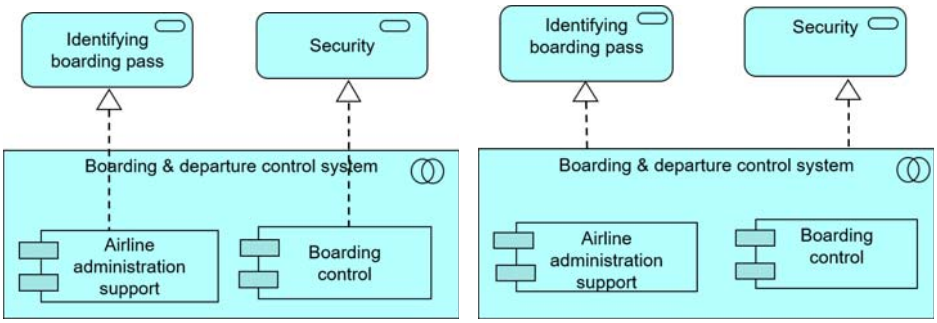
Figure 9.8 illustrates the scenario of the airport departure system. We assume, Architect 1 models the application layer as shown in Figure 9.8(a), which shows the presence of realization relationship between "Airline administration support" (Application Component) and "Identifying boarding pass" (Application Service) and between "Boarding control" (Application Component) and "Security" (Application Service). Architect 2 models the similar kind of application layer as shown in Figure 9.8(b). Both the models have an equal number of components with varying edge connection. Since both the application components have collaborated within "Boarding & departure control system" (Application Collaboration), there exists a similarity between the models generated by both the architects irrespective of edge connection.

Let $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$ be the directed graphs that represent the EA models. For a vertex $u \in V_1$ and $u \in V_2$, we define $N(u)$ as the set of neighbors of u. Similarly, for vertex $v \in V_1$ and $v \in V_2$, we define $N(v)$ as the set of neighbors of v. Degree of vertex u and v is represented as k_u and k_v.

Motivated from the work of [134], we selected the most widely used structural similarity measures based on edges namely Jaccard, Dice, and Adamic-Adar similarity indexes.

Definition 9.3 (Jaccard index) *The Jaccard index between two vertices is the number of common neighbors divided by the number of vertices that are neighbors of at least one of the two vertices being considered [134]. The similarity score ranges between* 0 *and* 1*, where score* 0 *represents no similarity and score* 1 *indicates complete similarity. The Jaccard index between vertices* u *and* v *for graph* G_1 *is computed as shown below:*

$$Jaccard(u, v) = \frac{|N(u) \cap N(v)|}{|N(u) \cup N(v)|} \tag{9.14}$$



(a) Application View of Model m_1 (b) Application View of Model m_2

Figure 9.8.: Two Simple Instances Illustrating Structural Similarity.

Definition 9.4 (Dice index) *The Dice index (Sørensen–Dice index) is twice the number of common neighbors divided by the sum of degrees of two vertices [212]. Similar to Jaccard, the function ranges between* 0 *and* 1. *The Dice index between vertices* u *and* v *for graph* G_1 *is computed as shown below 9.15:*

$$Dice(u,v) = \frac{2|N(u) \cap N(v)|}{|k_u + k_v|} \tag{9.15}$$

Definition 9.5 (Adamic-Adar index) *The inverse log-weighted similarity or Adamic-Adar index between two vertices is the number of their common neighbors, weighted by the inverse logarithm of their degrees. It is based on the assumption that two vertices should be considered more similar if they share a low-degree common neighbor [1]. The inverted log-weighted index between vertices* u *and* v *for graph* G_1 *is computed as shown below:*

$$Inverted\ log = \sum_{z \in N(u) \cap N(v)} \frac{1}{log\ k_z}, \tag{9.16}$$

where z *is a common neighbor to both nodes* u *and* v *and* k *is the degree of node* z.

9.2.3. Nearest Neighbor

To analyze the underlying structure of a complex EA, it is important to find relevant services, functionality, and processes and appropriate granularity of the services used in different layers of the EA model [4]. Therefore, it is necessary to analyze existing business processes, organizational structures and their relationship with the different elements or components of an organization's IT landscape.

Nearest neighbor approach [192] can provide a graph-based recommendation for enterprise architects in order to avoid modeling already existing elements. Early work on this domain was

attempted by [13] in the field of domain engineering supporting the design of a SOA. Thus, finding communities on a network helps to investigate the roles in an EA model where components in a single cluster will have similar kind of roles compared to other clusters [184].

We present a technique based on unsupervised learning method, i.e., community detection or clustering to determine the communities [66]. Considering EA model as a network [197], we apply different community detection algorithms and analyze the results [172]. The community detection method tries to group a set of vertices having a higher probability of being interconnected than being connected to the members of other groups. In this way, it is possible to investigate similar kinds of grouped components. Thus, in a given network, members within clusters are highly similar compared to members outside the cluster.

Here, similarity refers to the components having a common subset of neighbors. The resulting clusters or communities results in generating different viewpoints while minimizing the complex view of the different layers of EA models. Identifying communities in an EA model also provides decision support for the enterprise architects to place new component into the model repository based on identified roles and position of the components. When the size of the model is complex enough to analyze EA model, community detection technique becomes robust regarding execution speed and outperforms SimRank approach.

We do not know in advance which community detection algorithm can perform well on EA data. One of the objectives of our work is to estimate and select the best possible community detection algorithm which performs well on partitioning EA model. This, in-turn help to reduce the search time of a best-performing algorithm in the future. We have evaluated the most widely used community detection algorithms supported by the igraph library [46].

Once a community detection algorithm is implemented and the network is partitioned into communities, it is important to interpret the results to know which algorithm performs well and detect meaningful communities. Since we do not know the gold standard to which the communities should belong, highly effective approach to evaluate communities formed by using internal criteria such as "modularity" [164].

Definition 9.6 (Modularity) *The modularity, Q, of a community structure can be defined as:*

$$Q = \frac{1}{2m} \sum_{ij} \left[A_{ij} - \frac{k_i k_j}{2m} \right], \tag{9.17}$$

where m is the number of edges, A_{ij} is the element of the adjacency matrix, and k_i is the degree of node k. The value of Q ranges between -1 and +1. Higher the modularity score, better is the structure of the communities found.

9.2.4. Results

In this section, we outline the evaluation results to select the best performing similarity method to investigate the similarity between the EA models and to select the best performing community detection algorithm, which provides the best support of EA model maintenance.

Similarity Between Enterprise Architecture Models

Node Similarity Evaluation

Table 9.9.: Term Document Matrix.

	Documents	
Terms	doc1	doc2
airline	2	0
boarding	10	7
security	3	2
navigation	1	1
surveillance	0	1
traveler	0	2
⋮	⋮	⋮

Table 9.10.: LSA Semantic Space.

	Documents	
Terms	doc1	doc2
airline	2.20	0.00
boarding	2.40	2.08
security	1.39	1.10
navigation	0.69	0.69
surveillance	0.00	1.39
traveler	0.00	2.20
⋮	⋮	⋮

Syntactic Similarity. Concerning Model m_1 and Model m_2 (see figure 9.7), Cosine and Jaccard similarity measures are applied to assess the syntactic similarity between component names of the EA models. Considering a target similarity score as 1 (whole similarity) and knowing that Model m_1 (Figure 9.7(a)) and Model m_2 (Figure 9.7(b)) almost convey the similar meaning. The comparison results for our case shows that cosine similarity (0.700) has a better similarity score which is closer to value 1 than Jaccard Index (0.537). Thus, we conclude cosine similarity performs better than Jaccard index in capturing the similarity, at least in our example.

Latent Semantic Analysis. Considering a corpus of documents, the first step in the LSA is to create a term-document matrix. The term-document matrix is a two-dimensional matrix whose rows represent the terms and columns are the documents, so each entry (i, j) represents the frequency of term i in document j.

Regarding Figure 9.7, we created a term-document matrix as shown in Table 9.9. For example, the word "airline" appears twice in doc1 and zero time in doc1, whereas "boarding" appears ten times in doc1 and seven times in doc2 and so on.

The next step is to apply Singular Value Decomposition (SVD), a matrix decomposition algorithm to the term-document matrix [123]. This allows factorizing an original matrix M as a product of three matrices: term vector matrix T, diagonal matrix S and the document vector matrix D:

$$M = TSD^T \tag{9.18}$$

The matrix is then reduced to the specified number of dimensions $k = dims$ which produces a representation in a new space, called the latent semantic space.

In order to create LSA space for our example (refer Figure 9.7), we used "lsa" [2] package

[2] https://cran.r-project.org/web/packages/lsa/lsa.pdf

from R library. Table 9.10 summarizes the LSA space, which represents a two-dimensional reconstruction of the original matrix (see table 9.9) based on SVD.

The new space (Table 9.10) is then used to find similarity between different words or phrases in the component names present in the two EA models (see Figure 9.7(a) & 9.7(b)) by considering the words that are co-occurring in similar contexts may be considered to be semantically related.

Then we compute the similarity between two documents consisting of multiple words by using $Costring()$ function available in the "LSAfun" [3] package. This function calculates cosine values between two sentences or documents. The LSA similarity computes a score ranging from 0 (no similarity) to 1 (complete similarity).

Knowing that the Model m_1 (Figure 9.7(a)) and Model m_2 (Figure 9.7(b)) convey similar meaning and from the above results, we conclude that by using LSA (0.832) we got the highest score closer to 1. Thus, LSA outperforms Cosine and Jaccard similarity measure in capturing relatedness between EA models. Therefore, any one of the models can be added to the EA repository. Thus, LSA technique is more efficient in finding similarity between EA models compared to simple cosine similarity (0.700) based on the type of content present between the component names. For a certain threshold (similarity score) closer to 1, enterprise architects can decide upon adding a new model into the repository.

Structural similarity evaluation In order to assess the structural similarity between two graphs, we first calculate the similarity between the pair of vertices by constructing similarity matrices for the graphs G_1 and G_2 respectively using Jaccard (equation 9.14), Dice (equation 9.15) and Adamic-Adar (equation 9.16) indexes. Let $Sim(G_1)$ and $Sim(G_2)$ represent similarity matrices computed for the directed graphs G_1 and G_2 using the similarity index.

Association or correlation between two similarity matrices is later computed by testing correlation to compare whether two graphs are similar based on its structure. The correlation coefficient can range in value from -1 to $+1$. The larger the absolute value of the coefficient, the stronger the relationship between the variables. We used the Pearson correlation coefficient to compute the association between two similarity matrices (equation 9.19):

$$\rho(X,Y)\& = \frac{cov(X,Y)}{\sigma_X, \sigma_Y} \tag{9.19}$$

Our results show that best associations were obtained using the Jaccard similarity coefficient (0.968) whose correlation score is closer to 1 than Dice similarity coefficient (0.956) and Adamic-Adar similarity coefficient (0.853). Thus, we choose the Jaccard similarity index as the best performing metric in finding the structural similarity.

Comparison Between Community Detection Algorithms

In this section, we discuss the approach to find the best-performing community detection algorithm. Since we do not know to which the communities should belong, we use modularity score

[3] https://cran.r-project.org/web/packages/LSAfun/LSAfun.pdf

(internal measure) as a general criterion to find the best-performing algorithm. We consider two different cases in determining the communities of similar kind of EA components. Thus, for each case, we evaluate the performance of different community detection algorithms.

Ignoring Isolated Nodes In this case, we consider the graph by ignoring all the isolated nodes present in the graph including the title node. Here, the title node refers to the title element of an EA model. Hence, this case will be applicable when there are no isolated EA components present in the model. The algorithms were executed using the igraph library available in R software. Table 9.11 summarizes the resulting community size, modularity score, execution time and ranking of different community detection algorithms. Based on modularity scores and running time, the algorithms are ranked from the highest performing to the lowest performing.

Table 9.11.: Modularity Evaluation Result for Case 1.

Algorithm	Size	Modularity	Running time(in sec)	Ranking
Edge-betweenness	27	0.579	≈0.05	6
Infomap	28	0.694	≈0.08	4
Label propagation	26	0.601	≈50	5
Leading eigenvector	14	0.749	≈0.11	3
Spinglass	14	0.786	≈7.82	1
walktrap	18	0.746	≈0.009	2

As shown in Table 9.11, spinglass takes a top position. Higher the modularity score indicates better is the network partition. Leading eigenvector and walktrap returns the almost equal result with modularity score of 0.75. In this situation, running time, t, is considered as a tie-break criterion. Since walktrap computes faster than leading eigenvector, the user can decide between the selection of an algorithm based on running time of the specific algorithm. Edge-betweenness algorithm performed worst in this case with the lowest modularity score of about 0.579. The lesser the modularity score indicates larger the number of nodes in a single community and makes it difficult to understand the distribution of communities formed within a network.

Including Isolated Nodes The case 2 is applicable when the EA model contains isolated components including title component. Thus, we consider the graph with isolated nodes. From the experiment, we observe that isolated nodes are assigned a separate community. Table 9.12 summarizes the experimental result of the airport departure system model consisting of isolated components (including title node).

From Table 9.12, we observe walktrap and leading eigenvector algorithm perform well on the data-set containing an isolated node with modularity score of about 0.75, considered as a sound algorithm. Since walktrap computes faster than leading eigenvector, walktrap got the top position. Label propagation algorithm took a longer time to execute with modularity score of 0.612. Edge-betweenness algorithm performed worst in this case with lowest modularity score

Table 9.12.: Modularity Evaluation Result for Case 2.

Algorithm	Size	Modularity	Running time(in sec)	Ranking
Edge-betweenness	28	0.579	≈0.031	5
Infomap	29	0.690	≈0.05	3
Label propagation	27	0.612	≈8.41	4
Leading eigenvector	15	0.749	≈0.09	2
Spinglass	-	-	-	6
walktrap	19	0.746	≈0.009	1

of about 0.579, which makes it difficult to understand the distribution of communities formed within a network. The spinglass algorithm did not work in this case, accusing the network to have unconnected nodes (e.g. title node). Hence, it got the last position.

9.2.5. Limitations

An obvious limitation of finding similarity method is that computation time is linear concerning the product of graph sizes due to the size of the similarity matrix. One of the challenges in determining the similarity between networks is defining a measure of similarity. For instance, it would be difficult for a domain expert to choose a threshold on the similarity measures to decide whether two models are similar. Another limitation is that the igraph community detection algorithm fails to handle overlapping communities. Thus, the overlapping community algorithm is necessary when a large set of components are used in different business processes or functions.

Chapter 10.

Optimizing EA Models

So far, we have concentrated on the EA model as such, meaning that we tried to assess its overall quality and to improve it. However, the EA model reflects real world assets like applications or processes. Therefore, we like to improve the quality of the EA model with respect to its real world properties in the following. The following results have initially been presented in [81, 83].

Motivation The up-to-date EA model can serve as input for EA itself. For example, the model can be facilitated to identify potential optimizations to achieve a better IT/business alignment. Typically, EAFs include the management of business capabilities, the information system architecture, infrastructure components, and information structures [242]. Obviously, the beforehand stated properties of EA allow to interpret the EA model as a graph, where EA's elements are vertices and their relationships are edges. The different layers of EA can be represented by sets of vertices.

One way to achieve the IT/business alignment is EA [14, 183]. Corresponding to the maturity of the EA [15] different goals are aimed. For example, the IT landscape should be consolidated to reduce costs or to identify functional redundancies between business and IT. To support those goals different techniques are feasible.

In the following, we propose a graph-based technique to improve EAs with respect to lose coupling, minimal amount of elements, and minimal operation costs, using LIP. We use the metaphor of triangles to build up our LIP, which is more intuitively compared to other techniques (e.g., [71, 74]). Those techniques optimize EAs using graphs and LIP.

10.1. Foundations

Since, we can represent an EA as a graph, we can apply existing graph algorithms to solve problems within the domain of EA. For instance, the well-known Traveling Salesperson Problem (TSP) can be formulated as a LIP on a graph [21, pp. 1-5]. This universal formulation allows to apply this solution to other domains like genome sequencing, drilling problems, or data clustering [21, pp. 59-70].

As presented by [242], an EA is structured in a layered manner. Each layer contains different architectural elements, which have relations to other elements of the same layer as well as relations to architectural elements of adjacent layers. We assume that each layer offers capabilities to the layer that is defined on top of it to realize its needed functions and behavior. In the following, we will refer to the layer offering capabilities as the "lower layer" comparable to a server and

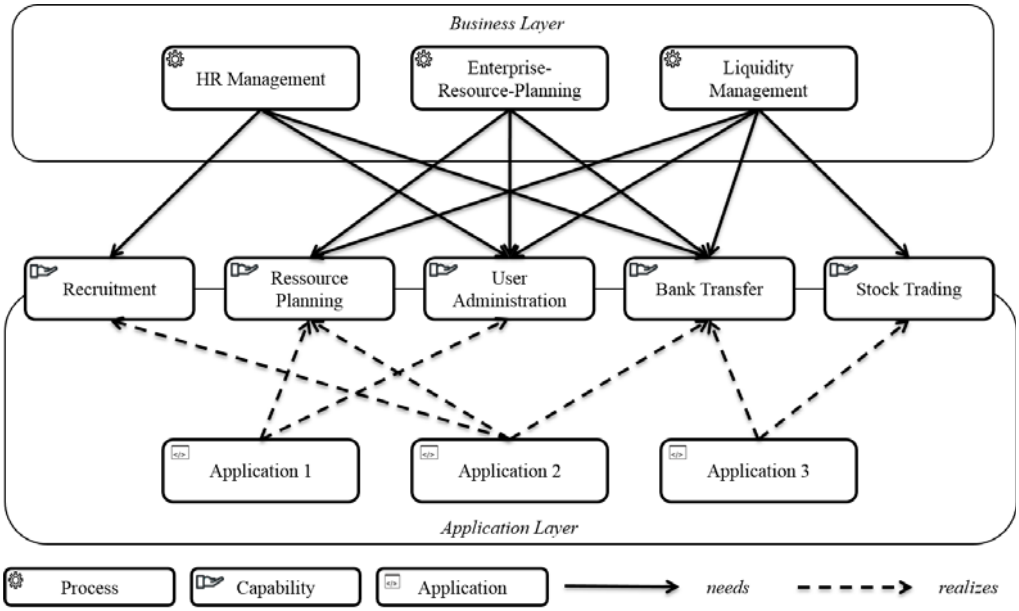


Figure 10.1.: Simple EA Model, Which Serves as Input for the Optimization.

call the layer using these capabilities the "upper layer" comparable to a client. Hereafter, we will describe these layers and elements more vividly:

We want to optimize the relations between the elements of two adjacent layers. For instance, taking an ArchiMate notated EA in account, we may want to optimize the relations between the business layer and the application layer. The business layer contains some business functions, which should be realized by several application components. Therefore, each business function is connected to all necessary capabilities and each application component is connected to all capabilities it realizes. Based on the relations between the architectural elements and the capabilities, we suggest a technique to find the "optimal" relations between the elements of the adjacent layers. In our case, "optimal" can stand for e.g. a minimal amount of elements or minimal costs.

To be more concrete, assume that the business layer contains the business function *HR Management*, which requires e.g. the capabilities *Recruitment*, *User Administration*, and *Bank Transfer* as presented in Figure 10.1. These capabilities can be realized either by two different application components or one single application component. Depending on the understanding of "optimal", different solution scenarios are feasible.

Following, we will use a simplified EA model sketched in Figure 10.2. It contains two adjacent layers and a set of capabilities. Each architectural element of the upper layer, ul_i, is related to several capabilities, c_j, which are needed to realize ul_i.

Having a look from the lower layer, there are several architectural elements, ll_k, which realize different capabilities, c_j, defined by the relation ll_k to c_j. Based on this structures, we want to optimize the relations between the upper and the lower layer elements to different subjects.

Before we dive into the optimization, we define the used EA model more formal:

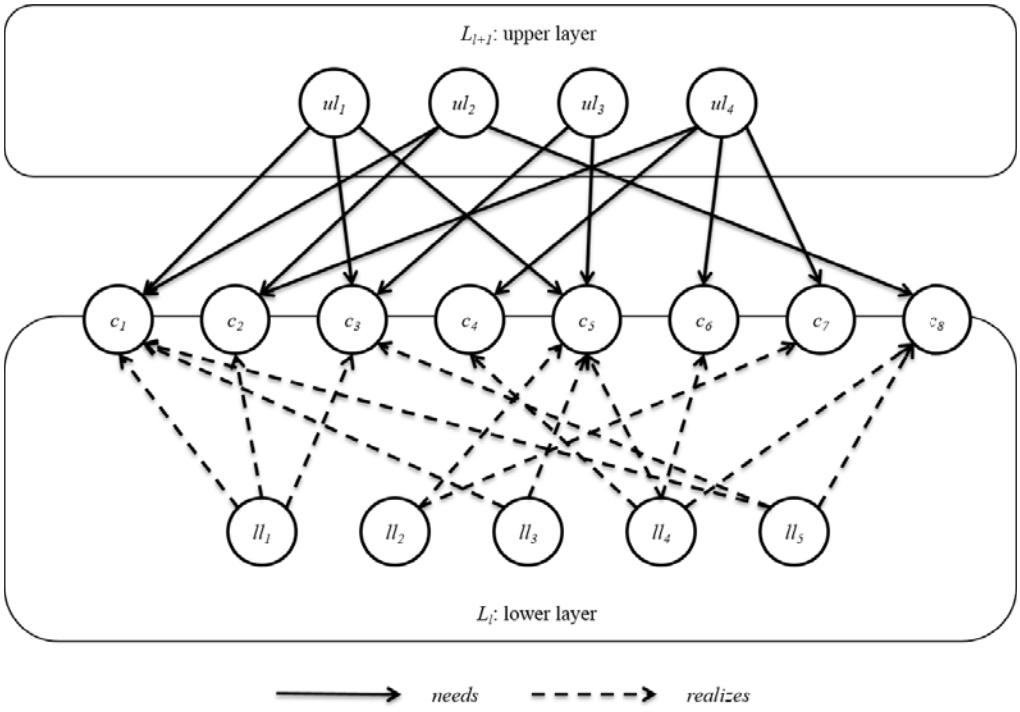


Figure 10.2.: Example Enterprise Architecture.

An enterprise architecture is a quadruple EA $= (\mathcal{L}, \mathcal{C}, E, R)$ comprising an ordered set $\mathcal{L}$ of layers, a set $\mathcal{C}$ of capability sets, a set E of architectural elements, and a set R of relations.

Each layer $L \in \mathcal{L}$ consists of architectural elements and layers are disjunct:

$$L_i \cap L_j = \emptyset \quad | \quad \forall L_i, L_j \in \mathcal{L}, i \neq j \tag{10.1}$$

For each two adjacent layers, L_l and L_{l+1}, there is a set of capabilities $C_{L_l} \in \mathcal{C}$. Each capability describes some specific behavior, which can be only offered by a certain layer; therefore, all these capability sets are disjunct:

$$C_{L_i} \cap C_{L_j} = \emptyset \quad | \quad \forall C_{L_i}, C_{L_j} \in \mathcal{C}, i \neq j \tag{10.2}$$

A relation is a tuple of an architectural element and a capability:

$$r \in R \subseteq \{(e, c) : e \in L_l \cup L_{l+1}, c \in C_{L_l}\} \tag{10.3}$$

We want to find the "optimal" set of relations between the elements of the upper and the lower layer using beforehand stated foundations. Therefore, we create a complete bipartite

graph between the layers L_l and L_{l+1}:

$$r_L^I \in R_l^I \equiv \{(ul, ll) : ul \in L_{l+1}, ll \in L_l\}. \tag{10.4}$$

This intermediately introduced relations represent all possible connections between the two adjacent layers with no respect to constraints given by the relations between architectural elements and capabilities.

10.2. Applying the Modeling Approach to ArchiMate

To illustrate our EA modeling approach based on layers and capabilities, we applied our approach for ArchiMate modeled EAs, as ArchiMate [128] is a widely accepted and used EA modeling language.

For the sake of simplicity, we considered only the element types of three ArchiMate layer: business layer, application layer, and technology layer. Those layers contain element types like processes or software, which are related to each other. Within the chosen layers, we disregarded the passive structure element types, since they all represent some kind of information objects in different ways, which cannot be linked to some kind of capabilities.

Table 10.3 shows the mapping of ArchiMate modeling element types to the three considered layers and the respective capabilities.

Table 10.3.: Suggested Mapping for ArchiMate Element Types to Proposed Sets.

	ArchiMate Element Type											
	Business Process	Business Function	Application Component	Application Collaboration	Application Function	Application Process	Node	Device	System Software	Technology Collaboration	Technology Function	Technology Process
Business Layer	•	•										
Capability					•							
Application Layer			•	•		•						
Capability											•	
Technology Layer							•	•	•	•		•

We mapped the Application Function element type of ArchiMate to a capability, because it can describe a requirement the business has for an application. Business Process and Business Function, modeling the overall structure of an organization and how the organization realizes the value chain, can be obviously mapped to element types of the upper layer. These element types

may need some application support from the Application Layer, which can be represented by the element types Application Component, Application Collaboration, or Application Process, realizing the needed capabilities, i.e. Application Functions.

Similarly to Application Function, we mapped the Technology Function element type to a capability describing the needs of the element types of the Application Layer. Furthermore, Technology Collaboration and Technology Process can be mapped to element types of the lower layer. Unfortunately, there is no element type called "Technology Component". Instead, the element type Node can be used to offer certain functionalities to the Application Layer. Furthermore, it aggregates Device, which represents hardware resources, and System Software, which represents software resources. Moreover, these aggregated element types are also needed to realize the functionality of the Application Layer and, consequently, are mapped to elements of the lower layer.

10.3. Different Optimization Subjects

Beforehand, we formulated a formal description of an EA model. Based on this formal description, we postulated a mapping for ArchiMate. Next, we will define several objectives to optimize the EA model.

10.3.1. Optimizing with Respect to Minimal Coupling

Loose coupling between two components eases the interchangeability [195]. Hence, the exchange of a single component will not be as expensive as in a highly coupled system. This holds also for EAs. Consequently, managers might be interested to find the minimal coupling between EAs elements.

Optimizing the EA with respect to a minimal coupling between two adjacent layers L_l and L_{l+1}, we have to minimize the amount of used intermediate relations, r_l^I. Consequently, those r_l^I represent the optimization variables of our LIP:

$$min \quad \sum x_{r_l^I}, \tag{10.5}$$

where $x_{r_l^I} \in \{0,1\}$ and $x_{r_l^I} = 1$ means that r_l^I is in the optimal solution and $x_{r_l^I} = 0$ means that it is not in the optimal solution.

To guarantee the restrictions sketched in Figure 10.2 we have to define several constraints. To ensure for each upper element that every capability which is needed will be served in the solution, there has to be at least one relation between an upper layer element and a lower layer element which supports this capability:

$$\sum_{r^I \in S^R_{ul_i,c_j}} x_{r_l^I} \geq 1 \quad | \, \forall S^R_{ul_i,c_j} \in \mathcal{S}^R, \tag{10.6}$$

with $\mathcal{S}^R \ni S^R_{ul_i,c_j} \subseteq R^I_l$. Where $\mathcal{S}^R$ contains all intermediate relations which are taken into account for the optimization and $S^R_{ul_i,c_j}$ contains all intermediate relations between ul_i and ll_k, where ll_k has a relation to c_j:

$$S^R_{ul_i,c_j} = \{(ul_i, ll_k) \in R^I_l\} \quad | \forall (ul_i, c_j) \in R, \forall (ll_k, c_j) \in R \tag{10.7}$$

To easily create all $S^R_{ul_i,c_j}$ we can create a maximum flow problem [91] for each connection between the upper layer elements and the related capabilities. Therefore, we introduce additional nodes: a source s, and a sink t. We connect s to ul_i and replace all upper layer elements in intermediate relations, r^I_l, by t. Furthermore, we introduce the capacity of a relation as a mapping $\mathfrak{c} : R \cup R^I_l \cup (s, ul_i) \rightarrow \mathbb{R}^+$ denoted by $\mathfrak{c}(u, v)$. The capacity defines the maximum amount of flow which can pass a relation.

A flow of a relation is a mapping $\mathfrak{f} : R \cup R_v \cup (s, ul_i) \rightarrow \mathbb{R}^+$ denoted by $\mathfrak{f}(u, v)$ with subject to two constraints. First, the flow cannot exceed its capacity:

$$\mathfrak{f}(u, v) \leq \mathfrak{c}(u, v) \quad | \forall (u, v) \in R \cup R_v \cup (s, ul_i) \tag{10.8}$$

Second, the sum of the entering flows must equal the sum of the leaving flows:

$$\sum_{u:(u,v)\in R} \mathfrak{f}(u, v) = \sum_{u:(u,v)\in R} \mathfrak{f}(v, u) \mid \forall v \in L_l \cup L_{l+1} \cup C_m \tag{10.9}$$

Before we apply e.g. the algorithm of [65] or [54] to solve the maximum flow problem, we set the capacity of all intermediate relations, r^I_l, to 1 and the capacity of all other relations to infinity. After the application, we add all intermediate relations whose flow is 1 to $S^R_{ul_i,c_j}$.

Figure 10.4 shows a subset of the EA in Figure 10.2. It contains ul_2 and all related capabilities, all ll_k which offer those capabilities, and the intermediate relations represented by the dashed lines. For instance, solving the maximum flow problem and creating the necessary set for the relation between ul_2 and c_1 leads to

$$S^R_{ul_2,c_1} = \{(ul_2, ll_1), (ul_2, ll_3), (ul_2, ll_5)\}, \tag{10.10}$$

which, consequently, creates the following constraint in our LIP:

$$x_{(ul_2,ll_1)} + x_{(ul_2,ll_3)} + x_{(ul_2,ll_5)} \geqslant 1. \tag{10.11}$$

Those constraints ensure that in the final solution, proposed by the LIP solver, at least one intermediate relation between the upper and the lower layer element is used, which are both related to the same capability. This leads to a distinctive structure stressed out in Figure 10.4 by the thicker lines: a triangle. These triangles describe the constraints which have to be included in a later solution.

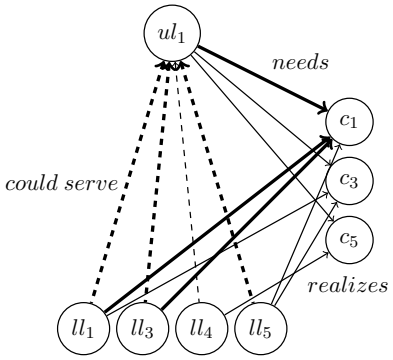


Figure 10.4.: Visualization of $S^R_{ul_2,c_j}$.

10.3.2. Optimizing the Amount of Needed Lower Layer Elements

The beforehand stated foundations can also be used to optimize the amount of needed lower layer architectural elements. Managers might want to have the minimal amount of elements in a certain layer to reduce the needed knowledge to maintain those elements.

To achieve the minimal amount of elements, we have to adjust the optimization function (10.5) and the constraints (10.6) from relations to lower layer elements. This leads to following optimization function:

$$min \quad \sum_{ll \in L_l} x_{ll}. \tag{10.12}$$

The constraints are constructed by using the lower layer elements as well:

$$\sum_{ll \in S^E_{ul_i,c_j}} x_{ll} \geq 1 \quad | \forall S^L_{ul_i,c_j} \in \mathcal{S}^L, \tag{10.13}$$

with $\mathcal{S}^E \ni S^E_{ul_i,c_j} \subseteq L_l$ containing all lower layer elements which are part of the intermediate relations between ul_i and ll_k, where ll_k has a relation to c_j:

$$S^E_{ul_i,c_j} = \{ll_k : (ul_i, ll_k) \in R^I_l\} \quad | \forall (ul_i, c_j) \in R, \forall (ll_k, c_j) \in R \tag{10.14}$$

According to the aforementioned example, solving the maximum flow problem creates

$$S^E_{ul_2,c_1} = \{ll_1, ll_3, ll_5\}, \tag{10.15}$$

which leads to following constraint:

$$x_{ll_1} + x_{ll_3} + x_{ll_5} \geqslant 1. \tag{10.16}$$

10.3.3. Optimizing Operational Costs

Managers are typically assessed by the costs which occur in their area of response. Consequently, they are often interested in reducing cost without losing functionality.

To optimize the operational costs, we have to apply slightly changes to (10.12) by introducing an operational cost function $\mathfrak{C}_O : E \rightarrow \mathbb{R}^+$ denoted by $\mathfrak{C}_O\,(e)$:

$$min \quad \sum_{ll \in L_l} \mathfrak{C}_O(ll) x_{ll}. \tag{10.17}$$

For simplicity reasons we assume that the operational costs are constant and not affected neither by lower layer elements attached to the considered element nor by relations between elements within the same layer.

10.4. Applying the Optimization Model

Following, we examine our model to check two aspects. First, we check if the proposed solutions are optimal. Second, we like to ensure that our approach is solvable for realistic problems in adequate time.

10.4.1. Exemplary Application

To test our previous formulated LIPs we translated the EA in Figure 10.2 to Java code and transformed it into a LIP which was solved by LPsolve[1]. Following, we will present the results of the optimization. For reasons of simplicity, we split up the graph in four sub graphs. Each sub graph represents the proposed solution for one upper layer element including the assigned capabilities and lower layer elements.

The results regarding the minimal coupling optimization are presented in Figure 10.5. The result set contains ten relations between the upper and the lower layer elements, which create the enforced twelve triangles. Furthermore, all five lower layer elements are used.

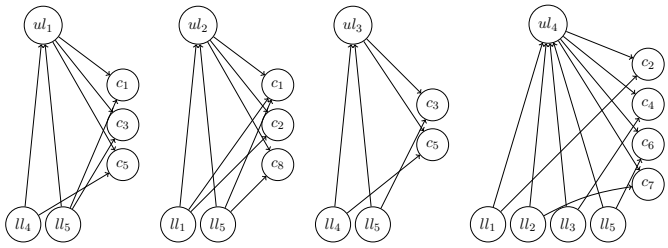


Figure 10.5.: Solution With Respect to a Minimal Coupling.

[1] https://sourceforge.net/projects/lpsolve/

Figure 10.6 sketches the results of the optimization regarding the minimal amount of lower layer elements. An optimal solution contains four elements as LPsolve suggests excluding ll_5. Since the LIP optimizes only the amount of lower layer elements, it tells nothing about the concrete assignment of upper to lower layer elements. Therefore, we suggest relations using solid lines as well as all other possible relations using dashed lines.

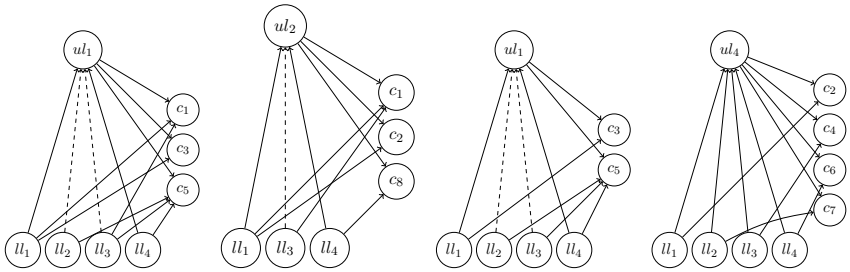


Figure 10.6.: Solution With Respect to Minimal Lower Layer Element Amount.

To process the optimization with respect to minimal lower layer element costs, we assigned costs stated in Table 10.8 to the lower layer elements. In Figure 10.7 we visualized the results. Compared to Figure 10.6, LPsolve suggests excluding ll_4 instead of ll_5 what leads to total costs of 17.

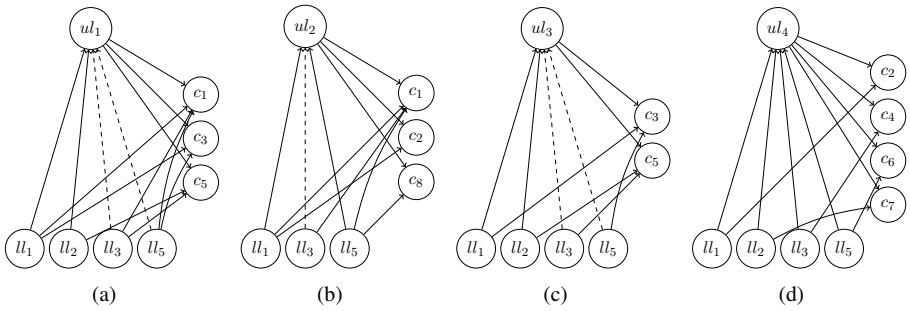


Figure 10.7.: Solution With Respect to Minimal Lower Layer Element Costs.

Table 10.8.: To Lower Layer Elements Assigned Costs.

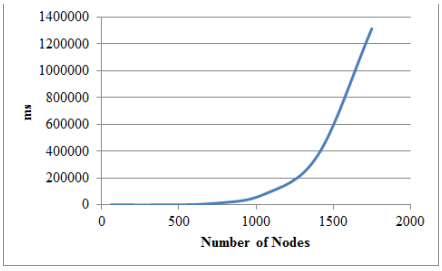
	ll_1	ll_2	ll_3	ll_4	ll_5
Assigned Cost	5	3	8	4	1

10.4.2. Does the Approach Scale?

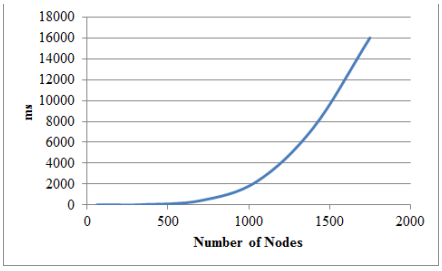
We performed a series of experiments to evaluate if the proposed optimization approach is applicable for realistic industry-sized EAs. To this end, we randomly generated graphs consisting of 60 to 1750 nodes and measured the execution time needed to propose a solution for three optimization scenarios. Each generated graph is split up into two layers and the linking capabilities.

The obtained results are depicted in Figure 10.9. The maximum execution time to compute a solution for the minimal coupling optimization scenario applied on a 1750 node graph was nearly 22 minutes (cf. Figure 10.9 (a)). In contrast, applied on the same graph the optimizations regarding the minimal amount of lower layer elements and lower layer elements minimal costs took only 16 (cf. Figure 10.9 (b)) respectively 5 seconds the longest (cf. Figure 10.9 (c)). These remarkable differences can be explained by the fact that the constraints in the minimal coupling optimization scenario are strongly based on the relations between the layer elements and the capabilities. As each layer element is linked to several capabilities, the number of constraints grows faster compared to the other both optimizations scenarios.

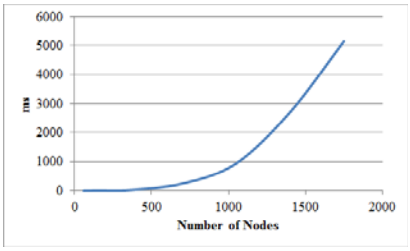
The three scenarios have in common that the solution space is based on the cross product between the nodes of the two layers. Therefore, the execution time is growing exponentially in all three scenarios.



(a) Minimal Coupling



(b) Minimal Lower Layer Element Amount



(c) Minimal Lower Layer Element Costs

Figure 10.9.: Execution Time Consumption.

Unfortunately, only fewer data regarding the size of realistic EAs is available (Schoonjans reports on an EA consisting of 108 nodes [204], Lagerström on one consisting of 407 nodes [120]). Although the EA of one of our cooperation partners (cf. Section 1.4.1) contains approximately

6000 nodes, the maximum number of elements which can be taken in account for a specified layer is below 500. Unfortunately, as no elements of this EA can be assigned to capabilities, we could not run our experiments on this EA as well.

Assuming that ordinary EAs are not significantly bigger, the execution time needed for all three optimization scenarios will be acceptable, especially because these optimizations need not be applied frequently.

10.5. Extension of the Optimization by Transition Costs

So far, our approach does not differentiate whether the elements belong to the actual state of the EA or their use is only planned so far. Accordingly, the necessary transition costs to get from the actual state to the optimal state are not taken into account. However, these transition costs can be so high that a transition to the optimal state does not make sense in an economic manner. This can be grounded in the fact that the transition costs from the actual state to a state that is functional optimal exceed the costs saved. Therefore, we extend our approach to the effect that the transition costs for an optimization of an EA are taken into account, in order to determine not only an optimal, but also an economically desirable state of the EA.

10.5.1. Formalism

But what influences the transition costs significantly? We assume that the transition costs are essentially due to the introduction or removal of elements of the lower layer. Additionally, we assume that the transition costs are not significantly influenced by changes of the relationships between elements of two adjacent layers. As a result, an optimization that properly accounts the transition costs must consider changes of lower-layer elements.

Therefore, we introduce a transition cost function $\mathfrak{C}_T : E \rightarrow \mathbb{R}^+$, annotated as $\mathfrak{C}_T(e)$, which returns the transition costs for an element e. For the sake of simplicity, we assume that the same costs are incurred for insertion and removal. Expanding the optimization function for the number of elements on the lower layer by this transition cost function results in the following function:

$$min \quad \sum_{ll \in L_l} \mathfrak{C}_T(ll) x_{ll}. \tag{10.18}$$

Obviously, this does not eliminate the weakness in the modeling of the initial optimization problem. Therefore, we introduce another function, the state function $\mathfrak{S} : E \rightarrow 0, 1$, annotated as $\mathfrak{S}(e)$. This determines whether an element is present in the actual state of the EA ($\mathfrak{S}(e) = 1$) or not ($\mathfrak{S}(e) = 0$). However, transition costs should only be taken into account if changes are needed in the transition from the actual state to the optimal state. Therefore, we subtract the value of the optimization variable from the result of the state function and determine the amount from it. In other words, in the event of a change the value of the transition cost function is multiplied by 1 and thus the transition costs are included in the optimization function. If there is no change, the amount is 0 and the costs are not taken into account in the optimization function.

Table 10.10.: Properties of Lower Layer Elements.

Lower Layer Element, u_i	**Part of as-is State**	**Transition Costs**	**Operational Costs**			
			$t=0$	$t=1$	$t=2$	Σ
u_1	Yes	3	2	3	5	10
u_2	No	5	5	5	5	15
u_3	Yes	2	4	5	11	20
u_4	No	4	4	5	9	18
u_5	Yes	8	2	2	3	7

Accordingly, the optimization function is:

$$min \quad \sum_{ll \in L_l} \mathfrak{C}_T(ll) x_{ll} |\mathfrak{S}(ll) - x_{ll}|. \tag{10.19}$$

We would like to show the effect of this optimization function using the example below. For this, we apply them to the EA shown in Figure 10.2. In addition, we use the additional information for lower layer elements contained in Table 10.10.

From a functional point of view, the elements u_1, u_2, and u_4 are included in a result, which yields the minimum number of elements on the lower layer. In addition, considering the actual state and the transition costs all elements of the lower layer (u_1 to u_5) are included in a functionally optimal solution and transition costs of 9 are needed to introduce u_2 and u_4 at costs of 5 and 4, respectively. The remaining elements are already part of the actual state, which is why no transition costs are incurred for them. Elements u_3 and u_5 are not removed from the lower layer because this would cause additional costs of 2 and 8 that are not compensated. Accordingly, this optimization function is only suitable for finding the most favorable state in which all the missing capabilities of the lower layer are present. In other words, unneeded elements of the lower layer are not removed.

In order to take this aspect into account and to achieve the most adequate model of reality, the transition costs must be balanced against the operating costs incurred. To do so, we define an operating cost function $\mathfrak{C}_O^t : E \to \mathbb{R}^+$, annotated as $\mathfrak{C}_O^t(e)$ analogous to the transition cost function, which provides the operating costs for an element e. In order to obtain the operating costs at a certain time, we index the operation cost function with a time $t \in T$, where T is the set of times considered. The optimization function takes this into account by the fact that not only the sum over the operating costs of the elements is formed, but that it is now formed over several points in time. Now the sum of both functions can be optimized towards the minimum:

$$min \quad \left(\sum_{ll \in L_l} \mathfrak{C}_T(ll) x_{ll} |\mathfrak{S}(ll) - x_{ll}| + \sum_{t \in T} \sum_{ll \in L_l} \mathfrak{C}_O^t(ll) x_{ll} \right). \tag{10.20}$$

Since the transition costs are incurred only once, but the operating costs accumulate over time, the results of the optimization also depend on the period considered. This is shown by the following example: We apply the optimization function to our example EA, whereby the

operating costs listed in Table 10.10 are now also considered over a period of 0 to 2 (see column 4). In the optimal state determined on this basis, the elements u_1, u_2, u_4, and u_5 are included with transition costs of 11 and operating costs of 50. Element u_3 has been removed because its capabilities are already provided by other elements. This also applies to element u_5, but it is not removed because its transition costs (8) are higher than the operating costs (7) over the considered period. In contrast to the previous approach, where only the transition costs were taken into account, unnecessary elements are removed if the expense for the transition does not exceed the operating costs over the considered period.

10.5.2. Application

To demonstrate the applicability of the proposed optimization approach, we generate a random EA consisting of 500 business processes, 750 capabilities, and 1,000 applications. A business process requires an average of 15 capabilities; An application offers an average of 8 capabilities. The application costs are between 1 and 100, with a median of 49, an average of 48, and a standard deviation of 29. Transition costs for an application range from 1 to 50, with median and average at 25 and standard deviation at 14. Using 515 applications in the actual EA, their total cost of ownership is 25,240.

In order to determine an optimal solution for this EA without considering any costs, we transform it into the LP format and have it solved in LPSolve. The optimal functional solution has operating costs of 2,672 and uses 161 applications. If this solution were implemented, 13,223 transition costs would arise.

If the transition costs are also considered, LPSolve needs 16.4% more time to determine the appropriate solution. This solution has an operating cost of 2,674 for 161 applications used. On the other hand, there are transition costs of 13,097, which corresponds to a savings of 128 over the considered period compared to the functionally optimal solution and 37.5% compared to the original solution.

Part V.

Evaluation and Summary

Chapter 11.

Evaluation

As most of our artifacts are outputs of conducting DSR or similar methods, according to Hevner et al. [93], five methods for evaluation are possible: observations, analysis, experiments, tests, and descriptions. Usually, we showed the applicability of our created artifacts by conducting case studies or interviews to evaluate our results (cf. Table 11.1). However, case studies grant just a first in-depth reflection in a real life scenario [247] and further evaluation is needed. Though, a further evaluation often requires substantial resources or access to a lot of organizations and their confidential information.

For example, our results regarding EA stakeholder concerns presented in section 3 could be evaluated further. First, we can conduct the case study in several organizations. Unfortunately, we do not have access to other organizations than the considered. Furthermore, the needed resources to carry all the necessary interviews out is not neglectable and exceeds our capabilities. Second, we can evaluate the results of the case study by enrolling a quantitative study. But, we wanted to focus on discovering new insights of stakeholder concerns and a questionnaire with closed questions is not able to fulfill this need, while one with open questions do not allow us to steer the interviewee like in an interview.

For other research, we just showed that the artifact that we have created solved the identified issue (cf. sections 6 and 7). According to DSR [93] and Software Engineering (SE) research [207, 131] this is a feasible approach to evaluate artifacts. Additionally, we applied tests to ensure the expected functionality (cf. sections 6).

There are also parts which are adequately evaluated. To evaluate our classification scheme presented in section 4, we applied a wide-spread and accepted means by splitting our input set into two parts, where the first part is facilitated to build the taxonomy and the second part to challenge it. Same holds for the artifacts of section 9 as we use our well documented EA model described in section 1.4.2 to prove our results. If other approaches arise, we can reuse the EA model and compare them directly to each other.

Lastly, there is our research on the optimization of EAs. We conducted no evaluation of the optimization itself as it is either correct or not. Contrary, we evaluated the applicability of the optimization regarding its calculation time and showed that a solution is computed in a reasonable time.

Taking a closer look to our evaluations, we perceive for the most of our created artifacts a satisfying degree of evaluation. Solely, the proposed EA model evolution process in section 5 falls short in regards to its evaluation. Therefore, we will elaborate on a further evaluation of this process in more detail in the following, which has initially been presented in [85].

Table 11.1.: Applied Evaluation Methods.

Part of this work	Applied Evaluation Method				
	Case Study	Interviews	ML Evaluation	SLR	Testing
Section 3	●	●	○	○	○
Section 4	○	○	○	●	○
Section 5	○	○	○	○	○
Section 6	●	○	○	○	●
Section 7	●	○	○	○	○
Section 8	●	○	○	○	○
Section 9.1	●	○	○	○	○
Section 9.2	○	○	●	○	○
Section 10	●	○	○	○	○

Motivation So far, different researchers have elaborated on processes to ensure a (semi-)automated EA model maintenance (see [63, 155, 84]). For practitioners this raises the question of how the processes can be compared to each other. We focus on quality aspects and, accordingly, we wonder first:

RQ 12 *What are important quality criteria of EA model maintenance processes?*

After answering this question, we can move forward to our research question:

RQ 13 *How differ certain EA model maintenance processes with respect to these quality criteria?*

To answer this question, we identified five quality criteria and asked EA practitioners and EA researcher to rate three different EA model maintenance processes [63, 155, 84] based on this quality criteria.

11.1. Research Method

Within this evaluation, we like to compare different EA model maintenance processes which are the resulting artifact of the application of DSR [84] or can be interpreted as the resulting artifact of the application of DSR [63, 155]. Further, we conduct a quantitative analysis of the proposed processes, which can be seen as an additional evaluation of the processes. Cleven et al. [44] provide a set of twelve variables quantified by two to seven values to classify DSR artifact evaluation. Accordingly, we classify our research according to Cleven et al. [44] as presented in Figure 11.2.

11.1.1. Tested Quality Criteria

Assessing the quality of a process is quite challenging. Especially, if the process cannot be applied in reality but has to be assessed based on its description only, as in our case. Therefore,

Variable	Value				
Approach	Qualitative		Quantitative		
Artifact Focus	Technical		Organizational		Strategic
Artifact Type	Construct	Model	Method	Instantiation	Theory
Epistemology	Positivism		Interpretivism		
Function	Knowledge function	Control function	Development function	Legitimization function	
Method	Action research	Case study	Field experiment	Formal proofs	
	Controlled experiment	Prototype	Survey		
Object	Artifact		Artifact construction		
Ontology	Realism		Nominalism		
Perspective	Economic	Deployment	Engineering	Epistemological	
Position	Externally		Internally		
Reference Point	Artifact against research gap	Artifact against real world	Research gap against real world		
Time	Ex ante		Ex post		

Figure 11.2.: Configuration of our Evaluation According to Cleven et al. [44].

we wanted to create a set of quality criteria which captures a broad range of process facets. Additionally, the set should not be too big so that the participant can keep the different criteria in mind. First, we evaluated our own quality framework to rate EA models [227]. Unfortunately, our framework assesses only the model quality and no deeper aspects of the process.

Next, we searched for a small set of quality criteria and ended up with the well-known eight dimensions of quality [73]. However, those quality criteria are related to products and, therefore, some criteria are hardly applicable to our issue. For example, aesthetics is a criterion which does not really matter for processes.

Last, we evaluated quality criteria arising from the domain of software engineering [142, p. 66]. We reduced the extensive set of quality criteria by removing those which does not suit the issue of rating process quality. For example, we neglect modularity as we do not want to assess the reusability of the processes. Following, we present the five quality criteria, we use to compare the three processes to each other:

- **Comprehensibility** describes how easy the process model can be understood.
- **Effectiveness** relates to the likelihood how easy the process under inspection can keep an enterprise architecture model up-to-date.
- **Completeness** assesses if the process contains all the necessary process steps to maintain an enterprise architecture model.
- **Minimality** reflects if the process contains only those process steps that are necessary to maintain an enterprise architecture model.
- **Efficiency** presents how efficient the process is perceived –in terms of time– to maintain changes into an enterprise architecture model.

We expect that the beforehand introduced quality criteria are not completely independent of each other as sketched in Figure 11.3. First, we anticipate a strong mutual correlation between

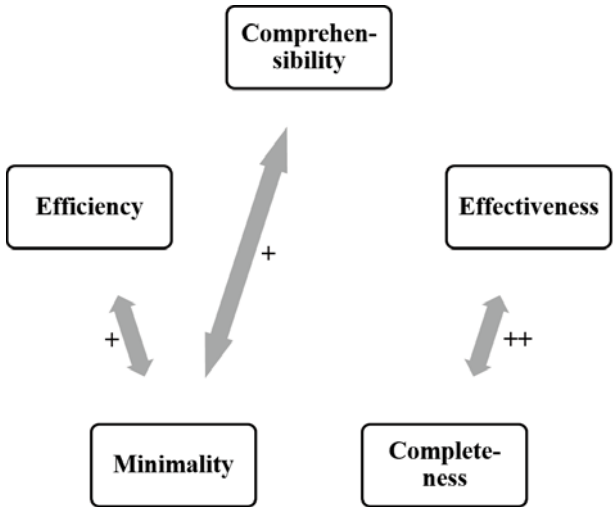


Figure 11.3.: Expected Dependencies Between Quality Criteria.

completeness and effectiveness. This is grounded in the fact that a process, which is not complete, can be hardly effective as if something is missing effectiveness cannot be guaranteed. Same holds vice versa, because if a process is not effective, it is likely that something is missing within the process. Nonetheless, –up to our mind– effectiveness and completeness are not the same. We argue that, for example, just small parts of the process could be missing which have a strong influence on the perceived effectiveness.

Second, we expect a mutual correlation between minimality and efficiency. If a process is minimal, there are no additional parts which could slow down the process and, therefore, decrease its efficiency. Same holds vice versa as the most efficient process does not contain unnecessary process steps. However, we do not expect the correlation between efficiency and minimiality as strong as between completeness and effectiveness, because efficiency of the whole process is strongly related to the efficiency of each process step which is not the case for minimality. Last, we await a positive correlation between minimality and comprehensibility, as we think that it is easier to understand a process which is minimal than a process which is not.

We will check our beforehand stated assumptions in Section 11.3.1 and if necessary, we will discard a criterion.

11.1.2. Questionnaire Design

To collect the necessary data, we created a seven-page questionnaire in German and English language. The first page gives a introduction to the research topic, and defines the targeted group of participants.

The second page explains the expectations towards the participant and introduces the five quality criteria to be assessed (cf. Section 11.1.1). Both pages have in common that they contain a not negligible amount of text to create a *high hurdle* [189]. Doing so, we want to sort out par-

ticipants which are not motivated to answer our questionnaire and reward the other participants as it is easier to capture the whole content on the following pages.

The next three pages present in each case one of the processes in a unified representation (see Figure 11.4, 11.5, and 11.6). Therefore, we sketched the processes in a "box-and-lines" notation, because we want the participant to focus on the process itself and not on notations. Additionally, we give a short description of the process' aim, followed by a characterization of all included roles. We unified also the role names and their description to ease the understanding of the different processes. Additionally, we provide an abstract of the process itself.

As the participant got all information she needs to assess process' quality, we ask her to rate every quality criterion on a five-point Likert-scale [49] to which degree the process suits the criterion from 1 (not) to 5 (perfect). If she is not able to assess a certain criterion, she can also indicate this. After rating the criteria, we offer the participant to give qualitative feedback on each process, too.

The sixth page is facilitated to collect demographic information on the EA experience of the participant and her organization. Further, she can also provide feedback on the questionnaire or give other comments.

On the last page, we ask the participant on her seriousness and consent to use her data. This is a common technique to exclude questionnaires which were not seriously filled [226, p. 114f].

To ensure usability of our questionnaire, we conducted a three stage development. In the first step the first author created the questionnaire and the second author checked it for any flaws. In the second stage, the questionnaire was distributed within the research group of the authors to check for flaws and to determine the necessary time to answer it. In the last stage the questionnaire was spread throughout several EA practitioners for a last check.

11.1.3. Data Collection

We spread the questionnaire among different channels to reach as much EA practitioners and researcher as possible. Therefore, we asked our industrial cooperation partners to answer the questionnaire and to spread it also to other EA practitioners. Additionally, we asked the participants of three regional EA related meetings to answer the questionnaire. To get responses of the scientific EA community, we send the questionnaire to our research network and distributed it through several EA related e-mail-lists.

In total, we received 123 questionnaires which finished at least the questions related to the processes. First, we removed all questionnaires where the participant stated that she did not answer the questionnaire seriously ending up with 100 questionnaires. Second, we checked the demographic answers of the questionnaires where the participant did not answer the question regarding conscientiousness. As those answers seemed to be very randomly (e.g., 999 years of experience in EA or 1000 architects employed in a medium-sized organization), we removed further 20 questionnaires.

The most of the left 80 participants are employed within the IT sector (27.5%), followed by the insurance sector (16.3%). 18.8% are working in an organization with more than 10,000 employees, followed by 15% working in an organization with 1,000 to 2,500 employees. However, the participants are likely distributed along all organization sizes and gained a experience of 3.8

years in average. The predominant part of the participants works as an employee without personnel responsibility (60%), followed by 16.7% working in the operational management (e.g., team or group leader). In average, the companies employed around 18 enterprise architects and the median is at 5. The companies have in average an EA initiative since 6.6 years in place.

11.2. EA Model Maintenance Processes

Following, we present the three processes we evaluate. We restrict us to three processes to stick in a time frame of maximum 15 minutes to answer the questionnaire. The first process [63] focuses on the integration of information distributed in different systems and is highly cited. The second process [155] focuses on the integration of information from different sources and is also highly cited. The third process [84] is designed by the authors and focuses on integrating information generated by projects.

11.2.1. Process 1: A Federated Approach to EA Model Maintenance

The process presented at [63] focuses on the integration of information distributed in different systems. Essentially, four different roles are assigned to the process:

- **EA Coordinator:** EA coordinator is part of the EA team and reports to the chief architect. Her main tasks include improving the EA meta-model, maintaining the EA model, and designing EA reports.
- **EA Repository Manager:** The focus of the EA repository manager is more technical. She is responsible for user administration, software updates, and the repository update.
- **EA Stakeholders:** EA stakeholders are business and IT departments that use EA information, e.g., to implement the strategy or security management.
- **Data Owner:** A data owner is responsible for a system whose data is to be transferred to the central EA model.

The process (see Figure 11.4) starts with the EA coordinators wanting to update the EA model and, for that reason, requesting up-to-date information from the appropriate data owner. Once she has delivered the information, the EA coordinators check the information for consistency. If inconsistencies persist, the data owner is notified and revises the information accordingly.

If the consistency check was successful, all changes to the EA model are identified and made available to all affected stakeholders. They review the changes and, if vetoed by a stakeholder, the EA coordinators coordinate a discussion to resolve the differences between stakeholders and data owners. After everyone agrees to the changes, the EA model can be updated and the changes communicated.

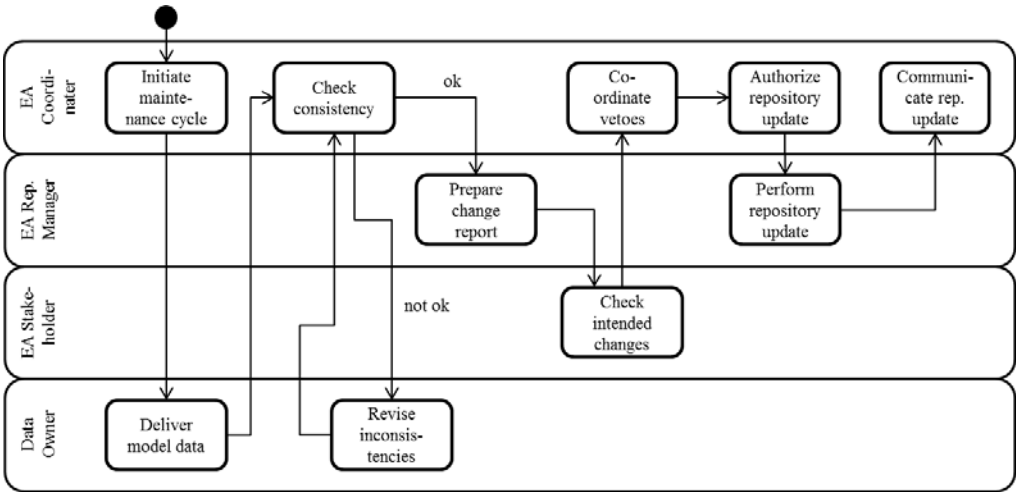


Figure 11.4.: A Federated Approach to EA Model Maintenance [63].

11.2.2. Process 2: Process Patterns for EA Management

The process of Moser et al. [155] focuses on the integration of information from different sources. Essentially, three different roles are assigned to the process:

- **Domain Expert:** Domain experts are a subset of EA's stakeholders. They formulate information requirements to the EA and are recipients of the information.
- **Enterprise Architect:** Enterprise architects are responsible for maintaining and keeping the EA model up-to-date.
- **Data Owner:** A data owner is responsible for a system whose data is to be transferred to the central EA model.

The process (see Figure 11.5) starts when a domain expert notices that she does not have all the information she needs for a particular task. So she asks the enterprise architects for this information. These check the request and contact the data owner who holds the corresponding information. She provides the information to the enterprise architects, whereupon they check its quality. If the result of the check is negative, the data owner improves the information.

Once the quality check has been successfully completed, the information is transformed and prepared for import. Before that, the changes will be checked by domain experts and enterprise architects. If there is no objection from any side, the information is imported and the updated EA model is made available.

11.2.3. Process 3: A Roundtrip Based EA Model Evolution

The process presented in [84] focuses on integrating information generated by projects. However, the projects can also be replaced by any other source of information. Essentially, two

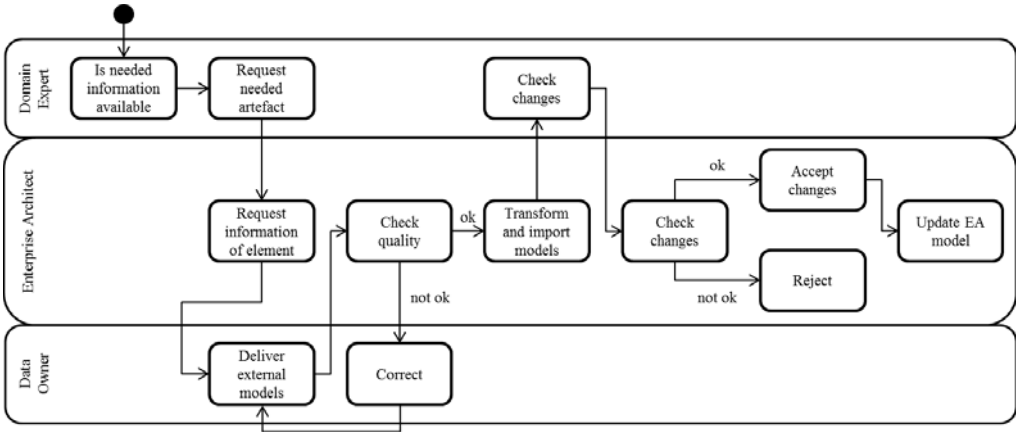


Figure 11.5.: Process Patterns for EA Management [155].

different roles are assigned to the process:

- **Enterprise Architect:** Enterprise architects are responsible for maintaining and keeping the EA model up to date.
- **Solution Architect:** The solution architects develop a solution for the project that evolves the EA and, thus, the EA model. Therefore these changes have to be included in the central model.

The process (see Figure 11.6) starts when enterprise architects identify changes in the EA and want to incorporate these changes into their core EA model. First of all, all changes that should be included in the next evolution of the EA model are captured. Subsequently, the data is quality-assured and aggregated to the necessary abstraction level of the EA model. Afterwards, the changes can be incorporated into the central EA model and the updated model distributed to the EA stakeholders.

For example, new projects receive this information and model the changes they make. These changes are then made available to enterprise architects and are the starting point for the next evolutionary step.

11.2.4. Preliminary Assumptions

Bringing the quality criteria and the processes together, we can formulate some assumptions how the processes are related to each other. First, we expect that process one and two are less comprehensible than process three as they include significant more process steps and roles. The plenty of process steps and roles in process one and two causes also our expectation that the third process gets the best rating for minimality.

As the third process includes no explicit negotiation between the different roles, we expect that the participants perceive this one as the most incomplete. Towards effectiveness and efficiency, we have no concrete expectations.

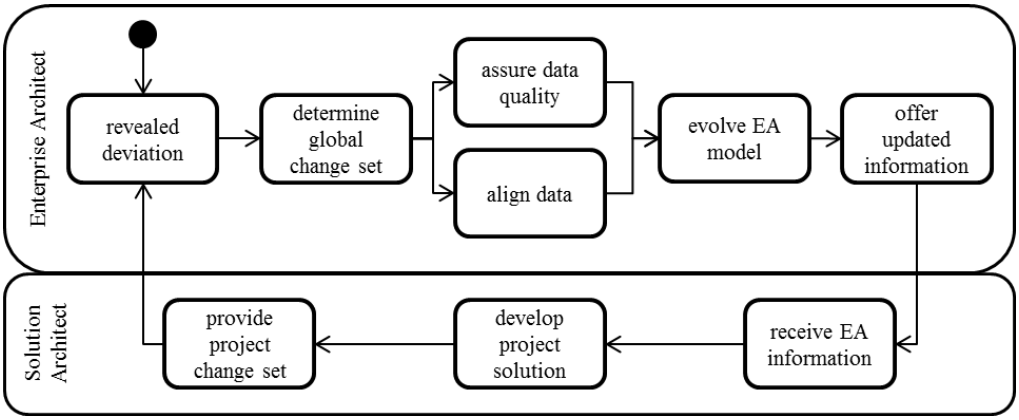


Figure 11.6.: An EA Model Evolution [84].

11.3. Results and Discussion

Following, we will present the results of our survey. As the participants could choose not to rate a certain criterion, we like to mention that only the comprehensibility was always rated. The efficiency of the processes was answered scarcest with a ratio of 93% (see Table 11.7).

11.3.1. Dependencies Between Quality Criteria

To test if our questionnaire might contain criteria which are coupled too close to each other, we calculated ρ_T according to [43]. Commonly, $\rho_T > 0.7$ means that the conducted items are in an acceptable matter linked to each other. As we calculate a value of 0.697, we can assume that we measure different concepts in our questionnaire. However, the value is close to 0.7 and, therefore, we calculate for each pair of our criteria ρ_T. As a result, we recognize for effectiveness and efficiency a value of 0.76. All other values are lower than 0.6.

To test the expected dependencies between our criteria, we calculated the Pearson correlation [177] for each pair of criteria. First, we can confirm a correlation between completeness and effectiveness. However, the correlation is not as strong as expected with a value of 0.42. Second, we found also a correlation between efficiency and minimality (0.39). Third, we could not uncover a correlation between minimality and comprehensibility (0.27).

Apart from the expected correlations, we notice a strong correlation between effectiveness and efficiency (0.62) and weak correlations between comprehensibility and effectiveness (0.34) as well as between completeness and efficiency (0.34). We assume that the correlation between effectiveness and efficiency can be explained by the fact that people often struggle to differentiate between both terms. This could also explain the unexpected correlations between and completeness and efficiency as we expected a correlation between completeness and effectiveness. The correlation between comprehensibility and effectiveness can also be explained by the confusion of efficiency and effectiveness and a transitive relation along minimality.

Table 11.7.: Descriptive Analysis of the Given Ratings.

	Comprehensibility			Effectiveness			Completeness			Minimality			Efficiency		
Answering Ratio	1.00			0.98			0.95			0.98			0.93		
Process	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3
Median	4	4	4	3	4	3	4	4	3	3	3	4	3	3	3
Mean	3.65	3.84	3.80	3.33	3.44	3.34	3.68	3.49	3.33	2.94	3.32	3.58	3.16	3.24	3.35
SD	0.85	0.74	0.71	0.77	0.76	0.75	0.70	0.83	0.85	0.74	0.71	0.79	0.89	0.79	0.70

11.3.2. Process Comparison

We show the results of the descriptive analysis of all responses in Table 11.7. Every quality criterion contains three values per measurement where the first value represents the first process, the second value represents the second process, and the third value represents the third process.

Following, we will discuss the insights we gathered from the survey:

- **Comprehensibility:** The participants could understand all process models more or less likely. However, the first process achieved a smaller mean as the other processes as well as the Standard Deviation (SD) is higher, indicating a bigger uncertainty. On the one hand, this is surprising as the second process contains more process steps and more decisions. On the other hand, process one comprises the biggest set of roles. Therefore, we conclude that the amount of roles is much more important for the perceived comprehensibility of a process model than the number of elements and decisions within a process.

  Additionally, we expected the third process to be most comprehensible as it is the simplest model. But, this is not the case. Consequently, a too abstract description –resulting in less process steps– does not necessarily lead to a better comprehensibility.

- **Effectiveness:** The participants perceived the effectiveness of the processes similarly according to the mean and SD. Only process two seems to be little more effective as the median is higher than for the other processes. This may stem from the fact that process two lasts of the most process steps and, therefore, the participants assume that it is most effective.

- **Completeness:** In accordance to our expectations, the third process got the lowest scores for completeness, because it is the simplest process and the participants are missing certain steps (e.g., “The process is missing certain information”, “There are no binding criteria for reporting deviations or need for action.”, or “Does the enterprise architect get all needed information”). Additionally, the participants are very uncertain about the completeness as the SD is 0.85 which is the second highest score in general.

- **Minimality:** The minimality scores are as expected. In accordance to our observations at comprehensibility, we can recognize that the plenty of roles in process one have a bigger effect on the minimality than the plenty of process steps and decisions in process two. Furthermore, we can appreciate a higher influence of this fact. This is also stressed by the participants as they state that “the plenty of roles lead to a communicative overhead” and the necessity to coordinate vetoes every time is questioned.

- **Efficiency:** All processes have in common that they are not perceived as very efficient. Again, process one gains a very low score and the participants are very uncertain. The communication between the different roles seems to be the main driver for this low score. This is in line with the feedback of the participants as they think that "reconciliation with so many parties may lead to efficiency losses" and that "there are too many communication channels".

Apart from the feedback directly related to the quality criteria, we can differentiate two divergent groups related to the complexity of the processes. The first group advocates for a "lean" maintenance process and claim a reduction of process steps and roles in processes one and two. The second group demands not only more roles and steps in process three but also further reconciliations in process two.

A further point, which should be considered, concerns all processes: Several participants remark a neglect of business stakeholder in the processes (e.g., "The maintenance of EA artifacts must also be handled by the business side"). They should not be "demoted to be only the auditor", but "actively involved into the [EA model] maintenance".

To summarize, if the organization tends to be more "agile" or "lean" the third process would be the best guess. If the organization tends to be more "classical" with strict hierarchies and lots of different stakeholders the second process suits best. The first process was never able to outperform the other processes in a significant manner.

11.4. Limitations

Our survey incorporates some limitations. First, the 80 answered questionnaires are not representative for the complete population of all EA experts. Additionally, the main part of the participants came from Europe in general and German-speaking countries in special. However, we believe that our questionnaire gives a good insight into the perception of maintenance process' quality. Second, we focused on a set of five quality criteria to keep the questionnaire lean. Obviously, there are other quality criteria which could be assessed, too. Nonetheless, none of the participants remarked that an important criterion is missing.

Chapter 12.

Summary

EA is a means to steer business-IT alignment. One of its central artifacts are EA models which can be facilitated to communicate the as-is architecture, plan the future to-be architecture, and control the progression by comparing will-be architectures to to-be architectures. Within this work, we have presented different processes and methods to support the quality of EA models.

First, we discussed the expectations of EA's stakeholders towards EA in general and the EA model in special. Therefore, our conducted study contributes to the existing literature on stakeholder concerns by introducing a differentiation of hierarchical stakeholder levels. In the study, we focused stakeholder groups from the operational management, middle management, and top management. For most concerns, the needs of the different stakeholder groups were found to be rather homogeneous. However, concerns on architecture, abstraction level, assertiveness, acceptance of other departments, and transparency of EA deliverables were discovered to be rather heterogeneous.

Second, we developed a taxonomy to classify EA analysis research. Accordingly, we performed a SLR on EA analysis research, its adopted models, analysis techniques and concerns analyzed. Grounded in those findings, we derived an initial taxonomy for EA analysis research to help researchers classify their work according to the analysis scope, technique, concern and modeling language. We validate the taxonomy's coverage with a second data-set of 46 papers. We consider the 46 papers in the final data-set give a good perspective regarding the coverage of our taxonomy. Therefore, we present the state of art of EA analysis research initiatives. We believe that tool modelers can also take this study as a conceptual reference to design EA analysis functionalities. The findings also show that EA analysis research presents very diverse EA models and concerns. Nevertheless, cases where most EA layers were analyzed rarely appeared.

After exploring the EA in research and practice, we presented different processes to improve EA models. The first process helps to overcome the problems related to a distributed EA model evolution. This process is comprised by a set of different process steps which are mainly performed by enterprise architects and solution architects.

Next, we provide a concrete implementation of the proposed process. Therefore, we implemented it within our tool JARVIS. Our first evaluation shows that our process benefit from the ideas of the agile domain leading from a model maintenance to a model evolution perspective. Additionally, we could show that the interaction between stakeholder and enterprise architects can be further reduced. Consequently, both can concentrate more on the essential parts of EA than on technically related issues.

Further, we propose an approach to keep contradictory information within EA models. Our approach refines P^2AMF [106], which already incorporates a way to represent uncertainty re-

garding the existence of modeled entities. To ease the use of our technique, we generalized P^{2}AMF from its UML/OCL notation to a graph presentation. Therefore, it can also be applied to P^{2}AMF models notated in arbitrary formats like ArchiMate [225]. Furthermore, we added competing scenarios and different versions along a time series to meet the requirements of a distributed P^{2}AMF evolution. To show the applicability of our approach, we utilized the theoretical described calculations and guidelines on a Neo4j graph database. Following, we argued that our realization meets the stated requirements of a distributed EA evolution.

After exploring different processes to improve EA models, we elaborate next on different methods for the same purpose. First, we designed a framework to assess and improve the quality of EA models called EAQF. To create the framework, we conducted a SLR, facilitated the framework of Becker et al. [27] and adapted it to our purpose. We came up with a structure consisting of three parts. One part forms the basis on which the other both parts are established. In this basis the purpose, objectives, and stakeholder are determined. The other parts are utilized to either rate the quality of the whole model or the quality of a certain view. This framework puts existing attributes into context and provides a means to assess EA model quality depending on its purpose and stakeholders' concerns.

Second, we discussed ML related techniques to support solution architects in their modeling of EA models. The first approach relied on finding patterns between two EA models. For this, we defined the term similarity between two EA components. We presented several models and also showed how to combine them. The second approach adopted a collaborative way of recommending components that might be of interest. Given a set of EA models from the same domain, we suggested how to convert architecture models to transactions in order to successfully apply association rule mining.

Further, we researched which ML method suits best a certain scenario. The first method is based on syntactic and LSA technique. Our results show that LSA outperforms traditional cosine similarity measure to capture similarity between EA models. The second method is based on edge information where a node in one graph is said to be similar to a node in another graph if they share a common set of neighborhoods. We have compared the outcomes of the three structural similarity metrics (Jaccard, Dice, and Adamic Adar) by computing the correlation between similarity matrices generated from different metrics. Thus, different similarity measures will show different performance in different applications.

Additionally, we addressed the issue of how to analyze the underlying structures of complex EAs. We investigated the performances of different community detection algorithms. The algorithms are compared by considering a set of different scenarios and evaluated based on performance metrics like modularity to select a well-performing algorithm for EA data. This methodology provides decision support for the enterprise architects by answering which parts of the EA model belong together and, therefore, suggesting probably related components.

Another investigated approach helps enterprise architects optimizing the EA itself. Therefore, we presented a technique to optimize the EA with focus on the relations between two adjacent layers. Thus, we constructed a LIP and searched for the optimal assignment between the elements of both layers and take different objectives into account.

As we use the metaphor of triangles to describe the constraints of the LIP, it makes it easier to give e.g. managers an insight into the LIP. This metaphor makes the constraints become more

vivid and, consequently, easier to understand. This leads to a higher acceptance of the solution compared to existing solutions and, consequently, rises the chance to apply it.

Moreover, we present a mapping between the elements of our technique and the widely accepted ArchiMate notation to enable organizations applying our approach. This mapping enables an optimization between the business and the application layer as well as between the application and the technology layer. Lastly, we could show that our approach solves problems of a realistic size in appropriate time and, thus, is applicable to real world problems.

For evaluation purposes, we check for every artifact of our work the appropriateness of its evaluation and recognize that a further evaluation for the proposed EA model evolution process is needed. Therefore, we compared our process to different EA model maintenance processes based on their quality. We identified a set of five quality criteria and asked EA researchers and practitioners to rate those for each process. Facilitating the outcome of this questionnaire, we cannot answer the question which process is qualitatively best in general. In point of fact, the answer is related to the setting the process should be deployed: Our process suits best in an “agile” environment while another process suits better in a “classical” environment.

Chapter 13.

Outlook

Our research still offers improvements and implications for future research. For instance, our research on EA stakeholders concerns potential improvements of EA approaches, as it has been reflected on the example of TOGAF: The described three different granularity levels of architectures may be sufficiently covered by two, while some stakeholders were not interested in the granularity level at all. The quality term should be applied not only to architectural principles, but also to all deliverables. Moreover, different quality concerns need further refinement. Lastly, investigation for EA advertising strategies will become a necessary focus for future research.

Our research on a taxonomy for EA analysis incorporates future works in three main directions. First, because the taxonomy is not exhaustive, we may need to look especially to the work of [130] to align all categories created. Then, the taxonomy's dimensions may be further validated and refined with experts (e.g., by conducting a survey to enclose more real-world examples). Second, based on our systematized set of analysis initiatives, a web catalog may be designed to share past results and to stimulate the reuse of EA models (EA data) among researchers. This could boost the EA empirical analysis research, as occurred in areas such as machine learning UC Irvine[1] which was supported by standard shared databases on which researchers apply their analysis approaches. For example, the open models initiative[2] [69] goes on that direction, offering a collection of models and also a rough classification of them.

At the same time, further work is needed to investigate technical aspects like model anonymization or model portability to lower the barriers for EA model sharing. Since existing analysis specifications usually presuppose a specific structure of meta-models and models, it is very difficult to reuse them with organizational models that do not conform to the respective assumptions. They required a high effort to transform the actual EA model in a manner, that the analysis can be executed. Additionally, the respective meta model does not make any statements about what concepts are actually used [126]. A generic meta-model could help in that as the one studied in [187]. Another option would be focusing in ArchiMate-based models, the *de facto* market standard for EA modeling.

Further work still remains on our EAQF. First, our conducted SLR covers limited number of search terms. For example, a further review should contain ancillary phrases like synonyms for the used terms. This could identify further quality attributes EAQF may include. Second, the external validity of EAQF needs further investigations. Therefore, supplementary DSR loops in other contexts should be executed. E.g., EAQF should be applied in different organizations from

[1] https://archive.ics.uci.edu/ml/index.php
[2] http://www.openmodels.org/

different industries or with different maturity grades. Third, a case study does not ensure that the quality attributes are sound and complete. Consequently, other evaluation methods should be applied in future work as well.

The maturity grade of the EA unit may be an important point, since for organizations with a low grade other quality attributes can be interesting compared to those with a higher grade. Consequently, EAQF should be aligned according to the maturity grade of the unit under inspection.

This stresses also another aspect for future research: the configurationally of EAQF. Every organization has special demands towards EA. Therefore, the demands of each organization should be reflected in EAQF properly. Nevertheless, organizations from equal industries may have similar demands which can represent as standard configurations within EAQF.

Last, executing EAQF has shown that questions are interrelated with each other. Though, these relations are not made explicit. This should be explored in future work, since this can reduce the needed effort to execute EAQF significantly.

For keeping contradictory information in EA models, we have shown the applicability of our approach. Next, the approach should be included into existing tools for EA. If the tool allows defining attributes on model elements and their relations, the existence probability can be depicted easily. Same holds for modeling the competing scenarios and the different versions along a time series if the tool allows altering its meta-model freely, i.e., add new model element types and relationship types. To create the needed reports, the tool needs to support free formulated queries. Especially, the last aspect is not easy to accomplish using the tools we know, since often some kind of scripting would be necessary.

Another way to enrich existing tools by the advantages of our approach is to patch a graph database in before the EA tool. The graph database would handle the uncertainty as presented in section 7.2.1 and an export would be generated which can be handled by the EA tool. In this case the database becomes the data master for the EA model. Therefore, all sources for the EA model have to be linked to the database which processes the data and delivers the results to the EA tool.

The proposed techniques to support the solution architects modeling the changes of the EA produces relevant results and can easily handle hundreds of nodes. However, more investigation is needed to determine whether it can scale extremely large graphs, those containing tens of thousands of nodes. So far, we focused on finding a match between two EA models based on the component names. As a future work, additional similarity approach can be extended to find similarity between EA models based on the description information of an EA component.

Our proposed approach to optimize the EA offers several possible extensions. First, we can assume different points in time with different costs and profit to predict the optimal time for each change. Second, we take no dependencies between our elements within one layer into account. But there are such dependencies in reality. Therefore, our model should be extended in this direction. Last, the model need to be evaluated in existing organization to test if the suggested optimizations really lead to savings.

Our evaluation offers insights for future EA model maintenance designs: All processes lacked the integration of the business side. The participants stressed that it is necessary to involve the business stakeholder actively into the maintenance. This would result in a shift from a centrally maintained EA model to a more locally maintained model.

Bibliography

[1] L. A. Adamic and E. Adar. Friends and neighbors on the web. *Social networks*, 25(3):211–230, 2003.

[2] J. S. Addicks. Enterprise architecture dependent application evaluations. In *Digital Ecosystems and Technologies, 2009. DEST'09. 3rd IEEE International Conference on*, pages 594–599, 2009.

[3] R. Agrawal and R. Srikant. Fast Algorithms for Mining Association Rules in Large Databases. In *Proceedings of the 20th International Conference on Very Large Data Bases*, VLDB '94, pages 487–499, San Francisco, CA, USA, 1994. Morgan Kaufmann Publishers Inc.

[4] S. Aier. How Clustering Enterprise Architectures helps to Design Service Oriented Architectures. In *IEEE International Conference on Services Computing SCC'06*, 2006.

[5] S. Aier. Understanding the Role of Organizational Culture for Design and Success of Enterprise Architecture Management. In R. Alt and B. Franczyk, editors, *Proceedings of the 11th International Conference on Wirtschaftsinformatik (WI2013)*, pages 879–894, Leipzig, 2013. Universität Leipzig.

[6] S. Aier, S. Buckl, B. Gleichauf, F. Matthes, C. M. Schweda, and R. Winter. Towards a More Integrated EA Planning: Linking Transformation Planning with Evolutionary Change. In M. Nuettgens, O. Thomas, and B. Weber, editors, *Enterprise Modelling and Information Systems Architectures (EMISA 2011)*, volume 190, pages 23–36, Bonn, 2011. GI.

[7] S. Aier, B. Gleichauf, and R. Winter. Understanding Enterprise Architecture Management Design: An Empirical Analysis. In *The 10th International Conference on Wirtschaftsinformatik WI 2.011 (Zurich)*, pages 645–654.

[8] S. Aier, S. Kurpjuweit, C. Riege, and J. Saat. Stakeholderorientierte Dokumentation und Analyse der Unternehmensarchitektur. *GI Jahrestagung (2)*, 134:559–565, 2008.

[9] S. Aier, N. Labusch, and P. Pähler. Implementing Architectural Thinking. In A. Persson and J. Stirna, editors, *Advanced Information Systems Engineering Workshops*, volume 215 of *Lecture Notes in Business Information Processing*, pages 389–400. Springer International Publishing, 2015.

[10] S. Aier, C. Riege, and R. Winter. Classification of Enterprise Architecture Scenarios-An Exploratory Analysis. *Enterprise Modelling and Information Systems Architectures*, 3(1):14–23, 2008.

[11] S. Aier, C. Riege, and R. Winter. Unternehmensarchitektur: Literaturüberblick und Stand der Praxis. *Wirtschaftsinformatik*, 50(4):292–304, 2008.

[12] S. Aier and M. Schönherr. Process orientied architecture integration with EAI. *Wirtschaftsinformatik*, 48(3):188–197, 2006.

[13] S. Aier and M. Schönherr. Integrating an enterprise architecture using domain clustering. *Journal of Enterprise Architecture*, 3(4):25–32, 2007.

[14] S. Aier and R. Winter. Virtual Decoupling for IT/Business Alignment – Conceptual Foundations, Architecture Design and Implementation Example. *Business & Information Systems Engineering*, 1(2):150–163, 2009.

[15] S. Aier, R. Winter, and F. Wortmann. Entwicklungsstufen des Unternehmensarchitekturmanagements. *HMD - Praxis der Wirtschaftsinformatik*, 284(49):15–23, 2012.

[16] O. Akhigbe, D. Amyot, and G. Richards. A Framework for a Business Intelligence-Enabled Adaptive Enterprise Architecture. In E. Yu, G. Dobbie, M. Jarke, and S. Purao, editors, *Conceptual Modeling*, pages 393–406, Cham, 2014. Springer International Publishing.

[17] A. C. Alhadi, A. Deraman, Yussof, Wan Nural Jawahir Wan, A. A. Mohamed, et al. An Ensemble Similarity Model for Short Text Retrieval. In *International Conference on Computational Science and Its Applications*, pages 20–29, 2017.

[18] P. Andersen and A. Carugati. Enterprise Architecture Evaluation: A Systematic Literature Review. In L. Mola, A. Carugati, A. Kokkinaki, and N. Pouloudi, editors, *Proceedings of the 8th Mediterranean Conference on Information Systems*, 2014.

[19] G. Antunes, J. Barateiro, A. Caetano, and J. L. Borbinha. Analysis of Federated Enterprise Architecture Models. In *ECIS*, 2015.

[20] G. Antunes, J. Borbinha, and A. Caetano. An Application of Semantic Techniques to the Analysis of Enterprise Architecture Models. In *2016 49th Hawaii International Conference on System Sciences (HICSS)*, pages 4536–4545, 2016.

[21] D. L. Applegate, R. E. Bixby, V. Chvatal, and W. J. Cook. *The traveling salesman problem: a computational study*. Princeton university press, 2011.

[22] M. Bakhshadeh, A. Morais, A. Caetano, and J. Borbinha. Ontology Transformation of Enterprise Architecture Models. In L. M. Camarinha-Matos, N. S. Barrento, and R. Mendonça, editors, *Technological Innovation for Collective Awareness Systems*, pages 55–62, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

[23] A. Barbosa, A. Santana, S. Hacks, and N. v. Stein. A Taxonomy for Enterprise Architecture Analysis Research. In *Proceedings of the 21st International Conference on Enterprise Information Systems*, volume 2, pages 493–504. SciTePress, 2019.

[24] I. Barone, A. D. Lucia, F. Fasano, E. Rullo, G. Scanniello, and G. Tortora. COMOVER: Concurrent model versioning. In *IEEE International Conference on Software Maintenance, 2008*, pages 462–463, Piscataway, NJ, 2008. IEEE.

[25] L. Bass, I. Weber, and L. Zhu. *DevOps: A Software Architect's Perspective*. Addison-Wesley Professional, 1st edition, 2015.

[26] M. Bastian, S. Heymann, and M. Jacomy. Gephi: An Open Source Software for Exploring and Manipulating Networks. 2009.

[27] J. Becker, W. Probandt, and O. Vering. *Grundsätze ordnungsmäßiger Modellierung: Konzeption und Praxisbeispiel für ein effizientes Prozessmanagement*. BPM kompetent. Springer Berlin Heidelberg, Berlin Heidelberg, 2012.

[28] S. A. Bernard. *An Introduction to Enterprise Architecture*. AuthorHouse, Bloomington, IN, 2 edition, 2005.

[29] M. Bhat, T. Reschenhofer, and F. Matthes. Tool Support for Analyzing the Evolution of Enterprise Architecture Metrics. In *ICEIS*, 2015.

[30] V. Blondel, A. Gajardo, M. Heymans, P. Senellart, and P. van Dooren. A Measure of Similarity between Graph Vertices: Applications to Synonym Extraction and Web Searching. *Society for Industrial and Applied Mathematics Review*, 46(4):647–666, 2004.

[31] W. F. Boh and D. Yellin. Using Enterprise Architecture Standards in Managing Information Technology. *Journal of Management Information Systems*, 23(3):163–207, 2006.

[32] V. Borozanov, S. Hacks, and N. Silva. Using Machine Learning Techniques for Evaluating the Similarity of Enterprise Architecture Models. In P. Giorgini and B. Weber, editors, *Advanced Information Systems Engineering*, pages 563–578. Springer International Publishing, 2019.

[33] R. P. Bostrom and J. S. Heinen. MIS Problems and Failures: A Socio-Technical Perspective. Part I: The Causes. *MIS Q*, 1(3):17–32, 1977.

[34] X. Boucher, J. Chapron, P. Burlat, and P. Lebrun. Process clusters for information system diagnostics: An approach by Organisational Urbanism. *Production Planning & Control*, 22(1):91–106, 2011.

[35] M. Brosius, S. Aier, K. Haki, and R. Winter. Enterprise Architecture Assimilation: An Institutional Perspective. In *Thirty Ninth International Conference on Information Systems (ICIS 2018)*, pages 1–16, San Francisco, CA, 2018. Association for Information Systems.

[36] S. Buckl, A. M. Ernst, H. Kopper, R. Marliani, F. Matthes, P. Petschownik, and C. M. Schweda. EA Management Patterns for Consolidations after Mergers. In *Software Engineering*, 2009.

[37] D. Buhalis and R. Law. Progress in information technology and tourism management: 20 years on and 10 years after the Internet: The state of eTourism research. *Tourism Management*, 29(4):609–623, 2008.

[38] I. Burnstein. *Practical software testing: a process-oriented approach*. Springer Science & Business Media, 2006.

[39] M. Buschle, M. Ekstedt, S. Grunow, M. Hauder, F. Matthes, and S. Roth. Automating enterprise architecture documentation using an enterprise service bus. *18th Americas Conference on Information Systems*, 2012.

[40] J. Capirossi and P. Rabier. An Enterprise Architecture and Data Quality Framework. In P.-J. Benghozi, D. Krob, and F. Rowe, editors, *Digital Enterprise Design and Management 2013*, pages 67–79, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[41] C. Castellanos, D. Correal, and F. Murcia. An Ontology-Matching Based Proposal to Detect Potential Redundancies on Enterprise Architectures. In *2011 30th International Conference of the Chilean Computer Science Society*, pages 118–126, 2011.

[42] P.-A. Champin and C. Solnon. Measuring the Similarity of Labeled Graphs. In *Proceedings of the 5th International Conference on Case-based Reasoning: Research and Development*, ICCBR'03, pages 80–95, Berlin, Heidelberg, 2003. Springer-Verlag.

[43] E. Cho. Making Reliability Reliable. *Organizational Research Methods*, 19(4):651–682, 2016.

[44] A. Cleven, P. Gubler, and K. M. Hüner. Design Alternatives for the Evaluation of Design Science Research Artifacts. In *Proceedings of the 4th International Conference on Design Science Research in Information Systems and Technology*, DESRIST '09, pages 19:1–19:8, New York, NY, USA, 2009. ACM.

[45] D. S. Cruzes and T. Dyba. Recommended Steps for Thematic Synthesis in Software Engineering. In *2011 International Symposium on Empirical Software Engineering and Measurement*, pages 275–284, 2011.

[46] G. Csardi and T. Nepusz. The igraph software package for complex network research. *InterJournal, Complex Systems*, 1695(5):1–9, 2006.

[47] P. Dagum, A. Galper, and E. Horvitz. Dynamic network models for forecasting. In *Proceedings of the eighth international conference on uncertainty in artificial intelligence*, pages 41–48. Morgan Kaufmann Publishers Inc, 1992.

[48] M. R. Davoudi and F. S. Aliee. Characterization of Enterprise Architecture quality attributes. In *2009 13th Enterprise Distributed Object Computing Conference Workshops*, pages 131–137, 2009.

[49] J. Dawes. Do data characteristics change according to the number of scale points used? An experiment using 5-point, 7-point and 10-point scales. *International journal of market research*, 50(1):61–104, 2008.

[50] A. de Lucia, F. Fasano, R. Oliveto, and G. Tortora. ADAMS: advanced artefact management system. In *Conference on Software Maintenance and Reengineering (CSMR'06)*, pages 349–350, 2006.

[51] P. Debois. Agile Infrastructure and Operations: How Infra-gile are You? *Agile 2008 Conference*, pages 202–207, 2008.

[52] W. H. DeLone and E. R. McLean. The DeLone and McLean model of information systems success: A ten-year update. *J. Manage. Inf. Syst.*, 19(4):9–30, 2003.

[53] M. M. Deza and E. Deza. *Encyclopedia of Distances*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[54] E. A. Dinic. Algorithm for solution of a problem of maximum flow in a network with power estimation. *Soviet Math. Doll.*, 11(5):1277–1280, 1970.

[55] D. Dreyfus and B. Iyer. Enterprise architecture: A social network perspective. In *HICSS'06. Proceedings of the 39th Annual Hawaii International Conference on System Sciences, 2006*, volume 8, 2006.

[56] P. Duvall, S. M. Matyas, and A. Glover. *Continuous Integration: Improving Software Quality and Reducing Risk (The Addison-Wesley Signature Series)*. Addison-Wesley Professional, 2007.

[57] M. J. Earl. *Management Strategies for Information Technology*. Prentice-Hall, Inc, Upper Saddle River, NJ, USA, 1989.

[58] W. Eberle and L. Holder. Discovering Structural Anomalies in Graph-Based Data. In *Seventh IEEE International Conference on Data Mining Workshops*, pages 393–398, New York, NY, USA, 2007. IEEE.

[59] K. M. Eisenhardt. Building theories from case study research. *Academy of management Review*, 14(4):532–550, 1989.

[60] A. M. Ernst. Enterprise Architecture Management Patterns. In *Proceedings of the 15th Conference on Pattern Languages of Programs*, PLoP '08, pages 7:1–7:20, New York, NY, USA, 2008. ACM.

[61] M. Farwick, B. Agreiter, R. Breu, S. Ryll, K. Voges, and I. Hanschke. Requirements for automated enterprise architecture model maintenance. In *13th International Conference on Enterprise Information Systems (ICEIS), Beijing*, 2011.

[62] M. Farwick, C. M. Schweda, R. Breu, K. Voges, and I. Hanschke. On Enterprise Architecture Change Events. In S. Aier, M. Ekstedt, F. Matthes, E. Proper, and J. L. Sanz, editors, *Trends in Enterprise Architecture Research and Practice*, pages 129–145, Berlin, Heidelberg, 2012. Springer.

[63] R. Fischer, S. Aier, and R. Winter. A Federated Approach to Enterprise Architecture Model Maintenance. In *EMISA*, 2007.

[64] H. Florez, M. Sánchez, and J. Villalobos. Extensible Model-Based Approach for Supporting Automatic Enterprise Analysis. In *2014 IEEE 18th International Enterprise Distributed Object Computing Conference*, pages 32–41, 2014.

[65] L. R. Ford and D. R. Fulkerson. Maximal flow through a network. *Canadian journal of Mathematics*, 8(3):399–404, 1956.

[66] S. Fortunato. Community detection in graphs. *Physics reports*, 486(3-5):75–174, 2010.

[67] M. Fowler. Continuous integration, 2006.

[68] C. Francalanci and V. Piuri. Designing information technology architectures: A cost-oriented methodology. *Journal of Information Technology*, 14(2):181–192, 1999.

[69] U. Frank, S. Strecker, and S. Koch. Open Model: Ein Vorschlag für ein Forschungsprogramm der Wirtschaftsinformatik. In *Wirtschaftsinformatik*, 2007.

[70] U. Franke. Enterprise Architecture Analysis with Production Functions. In *Enterprise Distributed Object Computing Conference (EDOC), 2014 IEEE 18th International*, pages 52–60, 2014.

[71] U. Franke, O. Holschke, M. Buschle, J. Rake-Revelant, and P. Närman. IT Consolidation: An Optimization Approach. In *14th IEEE International Enterprise Distributed Object Computing Conference Workshops*, 2010.

[72] A. Garg, R. Kazman, and H.-M. Chen. Interface descriptions for enterprise architecture. *Science of Computer Programming*, 61(1):4–15, 2006.

[73] D. Garvin. Competing on the eight dimensions of quality. *Harv. Bus. Rev.*, pages 101–109, 1987.

[74] V. Giakoumakis, D. Krob, L. Liberti, and F. Roda. Technological architecture evolutions of information systems: Trade-off and optimization. *Concurrent Engineering*, 20(2):127–147, 2012.

[75] I. Gmati, I. Rychkova, and S. Nurcan. On the Way from Research Innovations to Practical Utility in Enterprise Architecture: The Build-Up Process. *International Journal of Information System Modeling and Design (IJISMD)*, 1(3):20–44, 2010.

[76] G. Gorek and U. Kelter. Abgleich von Teilmodellen in den frühen Entwicklungsphasen. In *Proceedings Software Engineering*, Lecture Notes in Informatics, pages 123–134. GI, 2011.

[77] E. Grandry, C. Feltus, and E. Dubois. Conceptual Integration of Enterprise Architecture Management and Security Risk Management. In *17th IEEE International Enterprise Distributed Object Computing Conference Workshops*, pages 114–123, 2013.

[78] B. A. Guild. *A Guide to the Business Architecture Body of Knowledge (BIZBOK Guide)*, volume V04. 2014.

[79] S. Hacks, M. Brosius, and S. Aier. A Case Study of Stakeholder Concerns on EAM. In U. Franke, S. Aier, and M. Mocker, editors, *21st International Enterprise Distributed Object Computing Workshop (EDOCW)*, 2017.

[80] S. Hacks and H. Lichter. Distributed Enterprise Architecture Evolution–A Roundtrip Approach. In *Doctoral Consortium Wirtschaftsinformatik 2017 (unpublished)*, 2017.

[81] S. Hacks and H. Lichter. Optimizing Enterprise Architectures Using Linear Integer Programming Techniques. In M. Eibl and M. Gaedke, editors, *INFORMATIK 2017*, pages 623–636, Bonn, 2017. Gesellschaft für Informatik e.V.

[82] S. Hacks and H. Lichter. A Probabilistic Enterprise Architecture Model Evolution. In *2018 IEEE 22nd International Enterprise Distributed Object Computing Conference (EDOC)*, pages 51–57, 2018.

[83] S. Hacks and H. Lichter. Optimierung von Unternehmensarchitekturen unter Berücksichtigung von Transitionskosten. *HMD Praxis der Wirtschaftsinformatik*, 55:928–941, 2018.

[84] S. Hacks and H. Lichter. Towards an Enterprise Architecture Model Evolution. In C. Czarnecki, E. Sultanow, and C. Brockmann, editors, *Workshops der Informatik 2018*, Lecture Notes in Informatics, Bonn, 2018. Gesellschaft für Informatik e.V.

[85] S. Hacks and H. Lichter. Qualitative Comparison of Enterprise Architecture Model Maintenance Processes. In *EMISA 2019 conference (to be published)*, 2019.

[86] S. Hacks, A. Steffens, P. Hansen, and N. Rajashekar. A Continuous Delivery Pipeline for EA Model Evolution. In I. Reinhartz-Berger, J. Zdravkovic, J. Gulden, and R. Schmidt, editors, *Enterprise, Business-Process and Information Systems Modeling. BPMDS 2019, EMMSAD 2019*, pages 141–155. Springer International Publishing, 2019.

[87] S. Hacks and F. Timm. Towards a Quality Framework for Enterprise Architecture Models (Extended Abstract). *EMISA Forum*, 38(1):32–33, 2018.

[88] J. Han. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc, San Francisco, CA, USA, 2005.

[89] I. Hanschke. *Strategisches Management der IT-Landschaft: Ein praktischer Leitfaden für das Enterprise Architecture Management*. Carl Hanser Verlag GmbH Co KG, 2013.

[90] G. K. Hanssen, D. Šmite, and N. B. Moe. Signs of Agile Trends in Global Software Engineering Research: A Tertiary Study. In *2011 IEEE Sixth International Conference on Global Software Engineering Workshop*, pages 17–23, 2011.

[91] T. E. Harris and F. S. Ross. Fundamentals of a method for evaluating rail net capacities. *DTIC Document*, 1955.

[92] M. Hauder, F. Matthes, and S. Roth. Challenges for Automated Enterprise Architecture Documentation. In *TEAR/PRET*, 2012.

[93] A. R. Hevner, S. T. March, J. Park, and S. Ram. Design science in information systems research. *MIS quarterly*, 28(1):75–105, 2004.

[94] J. Highsmith and A. Cockburn. Agile Software Development: The Business of Innovation. *IEEE Computer*, 34:120–122, 2001.

[95] H. Holm, M. Buschle, R. Lagerström, and M. Ekstedt. Automatic data collection for enterprise architecture models. *Software & Systems Modeling*, 13(2):825–841, 2014.

[96] H. Holm, K. Shahzad, M. Buschle, and M. Ekstedt. P^2CySeMoL: Predictive, Probabilistic Cyber Security Modeling Language. *IEEE Transactions on Dependable and Secure Computing*, 12(6):626–639, 2015.

[97] O. Holschke, P. Närman, W. R. Flores, E. Eriksson, and M. Schönherr. Using enterprise architecture models and bayesian belief networks for failure impact analysis. In *International Conference on Service-Oriented Computing*, pages 339–350, 2008.

[98] J. Humble and D. Farley. *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation*. Addison-Wesley Professional, 1st edition, 2010.

[99] ISO, IEC, and IEEE. Systems and software engineering – Architecture description, 01.12.2011.

[100] ISO/IEC 25010. *Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models*, volume 25010 of *ISO/IEC*. ISO, Geneva, 2011.

[101] A. Jain, A. Jain, N. Chauhan, V. Singh, and N. Thakur. Information Retrieval using Cosine and Jaccard Similarity Measures in Vector Space Model. *International Journal of Computer Applications*, 164(6), 2017.

[102] J. Janulevičius, L. Marozas, A. Čenys, N. Goranin, and S. Ramanauskaitė. Enterprise architecture modeling based on cloud computing security ontology as a reference model. In *2017 Open Conference of Electrical, Electronic and Information Sciences (eStream)*, pages 1–6, 2017.

[103] G. Jeh and J. Widom. SimRank: A Measure of Structural-context Similarity. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 538–543, New York, NY, USA, 2002. ACM.

[104] P. Johnson, M. Ekstedt, and R. Lagerström. Automatic Probabilistic Enterprise IT Architecture Modeling: A Dynamic Bayesian Networks Approach. In U. Franke, J. Lapalme, and P. Johnson, editors, *20th International Enterprise Distributed Object Computing Workshop (EDOCW)*, pages 123–129, 2016.

[105] P. Johnson, L. Nordström, and R. Lagerström. Formalizing analysis of enterprise architecture. In *Enterprise Interoperability*, pages 35–44. Springer, 2007.

[106] P. Johnson, J. Ullberg, M. Buschle, U. Franke, and K. Shahzad. An architecture modeling framework for probabilistic prediction. *Information Systems and e-Business Management*, 12(4):595–622, 2014.

[107] H. Jonkers, M. M. Lankhorst, R. van Buuren, S. Hoppenbrouwers, M. Bosangue, and L. van der Torre. Concepts for Modelling Enterprise Architectures. *International Journal of Cooperative Information Systems*, 13(3):257–287, 2004.

[108] M. Jørgensen. A review of studies on expert estimation of software development effort. *Journal of Systems and Software*, 70(1):37–60, 2004.

[109] T. Kehrer. *Calculation and Propagation of Model Changes Based on User-level Edit Operations: A Foundation for Version and Variant Management in Model-driven Engineering*. Dissertation, University of Siegen, Siegen, Germany, 2015.

[110] U. Kelter, J. Wehren, and J. Niere. A Generic Difference Algorithm for UML Models. *Software Engineering*, 64(105):4–9, 2005.

[111] R. Khayami. Qualitative characteristics of enterprise architecture. *Procedia Computer Science*, 3:1277–1282, 2011.

[112] P. A. Khosroshahi, S. Aier, M. Hauder, S. Roth, F. Matthes, and R. Winter. Success Factors for Federated Enterprise Architecture Model Management. In A. Persson and J. Stirna, editors, *Advanced Information Systems Engineering Workshops*, Lecture Notes in Business Information Processing, pages 413–425. Springer International Publishing, 2015.

[113] B. Kirschner and S. Roth. Federated Enterprise Architecture Model Management: Collaborative Model Merging for Repositories with Loosely Coupled Schema and Data. In *Multikonferenz Wirtschaftsinformatik 2014*, 2014.

[114] B. Kitchenham. Procedures for Performing Systematic Reviews, Version 2.3: Technical Report EBSE-2007-01, Department of Computer Science, Keele University and National ICT, Australa Ltd.

[115] M. Kleehaus, Ö. Uludag, and F. Matthes. Towards a Multi-Layer IT Infrastructure Monitoring Approach based on Enterprise Architecture Information. In *CSE@ SE*, pages 12–17, 2017.

[116] S. Kotusev. The History of Enterprise Architecture: An Evidence-Based Review. *Journal of Enterprise Architecture*, 12(1):31–37, 2016.

[117] D. Koutra, A. Parikh, A. Ramdas, and J. Xiang. Algorithms for graph similarity and subgraph matching. In *Proc. Ecol. Inference Conf*, 2011.

[118] S. Kurpjuweit and R. Winter. Concern-oriented business architecture engineering. In *SAC*, 2009.

[119] M. La Rosa, M. Dumas, R. Uba, and R. Dijkman. Business Process Model Merging: An Approach to Business Process Consolidation. *ACM Trans. Softw. Eng. Methodol.*, 22(2):11:1–11:42, 2013.

[120] R. Lagerström, C. Baldwin, A. MacCormack, and D. Dreyfus. Visualizing and Measuring Enterprise Architecture: An Exploratory BioPharma Case. In J. Grabis, M. Kirikova, J. Zdravkovic, and J. Stirna, editors, *The Practice of Enterprise Modeling: 6th IFIP WG 8.1 Working Conference, PoEM 2013, Riga, Latvia, November 6-7, 2013, Proceedings*, pages 9–23. Springer, Berlin, Heidelberg, 2013.

[121] R. Lagerström, P. Johnson, and M. Ekstedt. Architecture analysis of enterprise systems modifiability: A metamodel for software change cost estimation. *Software Quality Journal*, 18(4):437–468, 2010.

[122] J. Lakhrouit, K. Baïna, and K. Benali. Model and Application Architecture Indicators of Evaluation the Enterprise Architecture. In Á. Rocha, A. M. Correia, F. . B. Tan, and K. . A. Stroetmann, editors, *New Perspectives in Information Systems and Technologies, Volume 2*, volume 276 of *Advances in Intelligent Systems and Computing*, pages 63–71. Springer International Publishing, Cham, 2014.

[123] T. K. Landauer, P. W. Foltz, and D. Laham. An introduction to latent semantic analysis. *Discourse processes*, 25(2-3):259–284, 1998.

[124] J. Landthaler, Ö. Uludağ, G. Bondel, A. Elnaggar, S. Nair, and F. Matthes. A Machine Learning Based Approach to Application Landscape Documentation. In *IFIP Working Conference on The Practice of Enterprise Modeling*, pages 71–85, 2018.

[125] M. Lange, J. Mendling, and J. Recker. A Comprehensive EA Benefit Realization Model–An Exploratory Study. In *2012 45th Hawaii International Conference on System Sciences*, pages 4230–4239, 2012.

[126] M. Langermeier, C. Saad, and B. Bauer. Adaptive Approach for Impact Analysis in Enterprise Architectures. In B. Shishkov, editor, *Business Modeling and Software Design*, pages 22–42, Cham, 2015. Springer International Publishing.

[127] M. Lankhorst. *Enterprise Architecture at Work: Modelling, Communication and Analysis*. The Enterprise Engineering Series. Springer Berlin Heidelberg, Berlin, Heidelberg and s.l., 2017.

[128] M. M. Lankhorst. Enterprise Architecture Modelling - The Issue of Integration: Enterprise Modelling and System Support. *Advanced Engineering Informatics*, 18(4):205–216, 2004.

[129] M. M. Lankhorst. *Enterprise Architecture at Work: Modelling, Communication and Analysis*. Springer, Berlin, 2005.

[130] B. Lantow, D. Jugel, M. Wißotzki, B. Lehmann, O. Zimmermann, and K. Sandkuhl. Towards a Classification Framework for Approaches to Enterprise Architecture Analysis. In *The Practice of Enterprise Modeling - 9th IFIP WG 8.1. Working Conference, PoEM 2016, Skövde, Sweden, November 8-10, 2016, Proceedings*, pages 335–343, 2016.

[131] M. Lázaro and E. Marcos. Research in Software Engineering: Paradigms and Methods. In *CAiSE Workshops*, pages 517–522, 2005.

[132] H. Lee, J. Ramanathan, Z. Hossain, P. Kumar, B. Weirwille, and R. Ramnath. Enterprise Architecture Content Model Applied to Complexity Management While Delivering IT Services. In *2014 IEEE International Conference on Services Computing*, pages 408–415, 2014.

[133] K. N. Lemon and P. C. Verhoef. Understanding customer experience throughout the customer journey. *Journal of Marketing*, 80(6):69–96, 2016.

[134] D. Liben-Nowell and J. Kleinberg. The link-prediction problem for social networks. *journal of the Association for Information Science and Technology*, 58(7):1019–1031, 2007.

[135] M. Liggins II, D. Hall, and J. Llinas. *Handbook of multisensor data fusion: Theory and practice*. CRC press, 2017.

[136] N. Lim, T.-g. Lee, and S.-g. Park. A Comparative Analysis of Enterprise Architecture Frameworks Based on EA Quality Attributes. *2009 10th ACIS International Conference on Software Engineering, Artificial Intelligences, Networking and Parallel/Distributed Computing*, pages 283–288, 2009.

[137] T. Lindholm. A Three-way Merge for XML Documents. In *Proceedings of the 2004 ACM Symposium on Document Engineering*, pages 1–10, New York, NY, 2004. ACM.

[138] O. I. Lindland, G. Sindre, and A. Solvberg. Understanding quality in conceptual modeling. *IEEE Software*, 11(2):42–49, 1994.

[139] Å. Lindström, P. Johnson, E. Johansson, M. Ekstedt, and M. Simonsson. A survey on CIO concerns: Do enterprise architecture frameworks support them? *Information Systems Frontiers*, 8(2):81–90, 2006.

[140] A. D. Lucia, F. Fasano, G. Scanniello, and G. Tortora. Concurrent Fine-Grained Versioning of UML Models. In R. Ferenc, editor, *13th European Conference on Software Maintenance and Reengineering, 2009*, pages 89–98, Piscataway, NJ, 2009. IEEE.

[141] C. Lucke, S. Krell, and U. Lechner. Critical Issues in Enterprise Architecting - A Literature Review. In *16th Americas Conference on Information Systems*, 2010.

[142] J. Ludewig and H. Lichter. *Software Engineering: Grundlagen, Menschen, Prozesse, Techniken*. dpunkt. verlag, 2013.

[143] J. N. Luftman. Key Issues for IT Executives 2004. *MIS Quarterly Executive*, 7, 2005.

[144] J. N. Luftman and T. Ben-Zvi. Key Issues for IT Executives 2010: Judicious IT Investments Continue Post-Recession. *MIS Quarterly Executive*, 9, 2010.

[145] L. Manzur, J. M. Ulloa, M. Sánchez, and J. Villalobos. xArchiMate: Enterprise Architecture simulation, experimentation and analysis. *SIMULATION*, 91(3):276–301, 2015.

[146] D. Marosin and S. Ghanavati. Measuring and managing the design restriction of enterprise architecture (EA) principles on EA models. In *2015 IEEE Eighth International Workshop on Requirements Engineering and Law (RELAW)*, pages 37–46, 2015.

[147] F. Matthes, S. Buckl, J. Leitel, and C. M. Schweda. *Enterprise Architecture Management Tool Survey 2008*. TU München, Chair for Informatics 19, Prof. Matthes (sebis), München, 2008.

[148] F. Matthes, I. Monahov, A. W. Schneider, and C. Schulz. *EAM KPI Catalog v1.0*. Garching, 2011.

[149] T. Mens. A State-of-the-art Survey on Software Merging. *IEEE Transactions on Software Engineering*, 28(5):449–462, 2002.

[150] M. B. Miles and A. M. Huberman. *Qualitative data analysis: An expanded sourcebook*. Sage Publications, Thousand Oaks, 1994.

[151] R. Montino, M. Fathi, A. Holland, T. Schmidt, and H. Peuser. Calculating risk of integration relations in application landscapes. In *Electro/Information Technology, 2007 IEEE International Conference on*, pages 210–214, 2007.

[152] S. Morimoto. Encouragement of Defining Moderate Semantics for Artifacts of Enterprise Architecture. In F. L. Gaol and Q. V. Nguyen, editors, *Proceedings of the 2011 2nd International Congress on Computer Applications and Computational Science*, pages 141–149, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[153] A. Morkevicius, S. Gudas, and D. Silingas. Model-driven quantitative performance analysis of UPDM-based enterprise architecture. In *Proceedings of the 16th International Conference on Information and Software Technologies*, pages 218–223, 2010.

[154] C. W. Morris. Foundations of the Theory of Signs. In *International encyclopedia of unified science*, pages 1–59. Chicago University Press, 1938.

[155] C. Moser, S. Junginger, M. Brückmann, and K.-M. Schöne. Some Process Patterns for Enterprise Architecture Management. In J. Münch and P. Liggesmeyer, editors, *Software Engineering 2009 - Workshopband*, pages 19–30, Bonn, 2009. Gesellschaft für Informatik e.V.

[156] L. Murta, C. Corrêa, J. G. Prudêncio, and C. Werner. Towards odyssey-VCS 2: Improvements over a UML-based Version Control System. In *Proceedings of the 2008 International Workshop on Comparison and Versioning of Software Models*, CVSM '08, pages 25–30, New York, NY, USA, 2008. ACM.

[157] P. Närman, H. Holm, D. Höök, N. Honeth, and P. Johnson. Using enterprise architecture and technology adoption models to predict application usage. *Journal of Systems and Software*, 85(8):1953–1967, 2012.

[158] P. Narman and P. Johnson. Analyzing Coordination and Flexibility in Organizations Using Enterprise Architecture. In *2016 IEEE 20th International Enterprise Distributed Object Computing Workshop (EDOCW)*, pages 1–8, 2016.

[159] P. Närman, P. Johnson, and L. Gingnell. Using enterprise architecture to analyse how organisational structure impact motivation and learning. *Enterprise Information Systems*, 10(5):523–562, 2016.

[160] P. Närman, M. Schönherr, P. Johnson, M. Ekstedt, and M. Chenine. Using Enterprise Architecture Models for System Quality Analysis. In *Enterprise Distributed Object Computing Conference, 2008. EDOC '08. 12th International IEEE*, pages 14–23. 2008.

[161] G. Navarro. A Guided Tour to Approximate String Matching. *ACM Comput. Surv.*, 33(1):31–88, 2001.

[162] Neo4j Inc. Cypher Query Language: Version 9.

[163] J. L. Neto, A. D. Santos, C. A. A. Kaestner, N. Alexandre, D. Santos, et al. Document clustering and text summarization. *Proc. of 4th Int. Conf. Practical Applications of KnowledgeDiscovery and Data Mining (PADD-2000)*, pages 41–55, 2000.

[164] M. E. J. Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23):8577–8582, 2006.

[165] E. Niemi and S. Pekkola. Enterprise Architecture Quality Attributes: A Case Study. In *2013 46th Hawaii International Conference on System Sciences*, pages 3878–3887. IEEE, 2013.

[166] S. Niwattanakul, J. Singthongchai, E. Naenudorn, and S. Wanapu. Using of Jaccard coefficient for keywords similarity. In *Proceedings of the International MultiConference of Engineers and Computer Scientists*, volume 1, 2013.

[167] C. C. Noble and D. J. Cook. Graph-based Anomaly Detection. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 631–636, New York, NY, USA, 2003. ACM.

[168] R. Nolan and F. W. McFarlan. Information technology and the board of directors. *Harvard business review*, 83(10):96, 2005.

[169] T. Oda and M. Saeki. Generative technique of version control systems for software diagrams. In *ICSM 2005*, pages 515–524, Los Alamitos, Calif, 2005. IEEE Computer Society.

[170] S. D. Oliner and D. E. Sichel. The Resurgence of Growth in the Late 1990s: Is Information Technology the Story? *FEDS Working Paper*, (20), 2000.

[171] H. Oliveira, L. Murta, and C. Werner. Odyssey-VCS: A Flexible Version Control System for UML Model Elements. In *Proceedings of the 12th International Workshop on Software Configuration Management*, SCM '05, pages 1–16, New York, NY, USA, 2005. ACM.

[172] G. K. Orman, V. Labatut, and H. Cherifi. Comparative evaluation of community detection algorithms: a topological approach. *Journal of Statistical Mechanics: Theory and Experiment*, 2012(08), 2012.

[173] M. Österlind, R. Lagerström, and P. Rosell. Assessing Modifiability in Application Services Using Enterprise Architecture Models – A Case Study. In S. Aier, M. Ekstedt, F. Matthes, E. Proper, and J. L. Sanz, editors, *Trends in Enterprise Architecture Research and Practice-Driven Research on Enterprise Transformation*, pages 162–181, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[174] S. Oussena and J. Essien. Validating enterprise architecture using ontology-based approach: A case study of student internship programme. In *Proceedings of the 15th International Conference on Enterprise Information Systems - ICEIS*, pages 302–309, 2013.

[175] R. Parasuraman, T. B. Sheridan, and C. D. Wickens. A model for types and levels of human interaction with automation. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 30(3):286–297, 2000.

[176] M. Q. Patton. *Qualitative Research & Evaluation Methods*. Sage Publications, Thousand Oaks, 3 edition, 2002.

[177] K. Pearson. Notes on regression and inheritance in the case of two parents. *Proceedings of the Royal Society of London*, 58:240–242, 1895.

[178] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee. A Design Science Research Methodology for Information Systems Research. *Journal of Management Information Systems*, 24(3):45–77, 2007.

[179] J. Pitschke. Gute Modelle–Wie die Qualität von Unternehmensmodellen definiert und gemessen werden kann, 2011.

[180] G. Plataniotis, S. de Kinderen, and H. A. Proper. Capturing Decision Making Strategies in Enterprise Architecture – A Viewpoint. In S. Nurcan, H. A. Proper, P. Soffer, J. Krogstie, R. Schmidt, T. Halpin, and I. Bider, editors, *Enterprise, Business-Process and Information Systems Modeling*, pages 339–353, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[181] G. Plataniotis, S. de Kinderen, and H. A. Proper. EA Anamnesis: An Approach for Decision Making Analysis in Enterprise Architecture. *International Journal of Information System Modeling and Design (IJISMD)*, 5(3):75–95, 2014.

[182] G. Plataniotis, S. d. Kinderen, Q. Ma, and E. Proper. A Conceptual Model for Compliance Checking Support of Enterprise Architecture Decisions. In *2015 IEEE 17th Conference on Business Informatics*, volume 1, pages 191–198, 2015.

[183] L. Plazaola, J. Flores, E. Silva, N. Vargas, and M. Ekstedt. An approach to associate strategic business-IT alignment assessment to enterprise architecture. In *Fifth Conference on Systems Engineering*, 2007.

[184] M. A. Porter, J.-P. Onnela, and P. J. Mucha. Communities in networks. *Notices of the AMS*, 56(9):1082–1097, 2009.

[185] N. Rajashekar, S. Hacks, and N. Silva. A Performance Comparison of Graph Analytic Methods for Supporting Enterprise Architecture Model Maintenance. In *Practice of Enterprise Modelling 2019 Forum (to be published)*. Springer, 2019.

[186] O. Rauh and E. Stickel. *Konzeptuelle Datenmodellierung*. Teubner-Reihe Wirtschaftsinformatik. Vieweg+Teubner Verlag and Imprint, Wiesbaden, 1997.

[187] J. Rauscher, M. Langermeier, and B. Bauer. Characteristics of Enterprise Architecture Analyses. In *Proceedings of the Sixth International Symposium on Business Modeling and Software Design (BMSD 2016)*, pages 104–113, 2017.

[188] M. Razavi, F. Shams Aliee, and K. Badie. An AHP-based approach toward enterprise architecture analysis based on enterprise architecture quality attributes. *Knowledge and Information Systems*, 28(2):449–472, 2011.

[189] U.-D. Reips. Standards for Internet-based experimenting. *Experimental psychology*, 49(4):243–256, 2002.

[190] D. F. Rico. A framework for measuring ROI of enterprise architecture. *Journal of Organizational and End User Computing*, 18(2):I, 2006.

[191] R. C. Russell. Soundex, 1918.

[192] S. Saha and S. P. Ghrera. Nearest Neighbor search in Complex Network for Community Detection. *arXiv preprint arXiv:1511.07210*, 2015.

[193] P. Saint-Louis and J. Lapalme. Investigation of the lack of common understanding in the discipline of enterprise architecture: A systematic mapping study. In U. Franke, J. Lapalme, and P. Johnson, editors, *20th International Enterprise Distributed Object Computing Workshop (EDOCW)*, 2016.

[194] P. Saint-Louis, M. C. Morency, and J. Lapalme. Defining Enterprise Architecture: A Systematic Literature Review. In U. Franke, S. Aier, and M. Mocker, editors, *21st International Enterprise Distributed Object Computing Workshop (EDOCW)*, 2017.

[195] R. Sanchez and J. T. Mahoney. Modularity, flexibility, and knowledge management in product and organization design. *Strategic Management Journal*, 17(S2):63–76, 1996.

[196] K. Sandkuhl, J. Stirna, A. Persson, and M. Wißotzki. Enterprise modeling. *Tackling Business Challenges with the 4EM Method. Springer*, 309, 2014.

[197] A. Santana, A. Souza, D. Simon, K. Fischbach, and H. de Moura. Network Science Applied to Enterprise Architecture Analysis: Towards the Foundational Concepts. In *2017 IEEE 21st International Enterprise Distributed Object Computing Conference (EDOC)*, pages 10–19, 2017.

[198] S. Santini and R. Jain. Similarity Measures. *IEEE Trans. Pattern Anal. Mach. Intell.*, 21(9):871–883, 1999.

[199] C. E. Sapp. Preparing and Architecting for Machine Learning, 2017.

[200] A. Šaša and M. Krisper. Enterprise architecture patterns for business process support analysis. *Journal of Systems and Software*, 84(9):1480–1506, 2011.

[201] F. Scarff, editor. *ITIL*. TSO, London, 2013.

[202] M. Schmidt, S. Wenzel, T. Kehrer, and U. Kelter. History-based Merging of Models. In *Proceedings of the 2009 ICSE Workshop on Comparison and Versioning of Software Models*, pages 13–18, Washington, DC, USA, 2009. IEEE Computer Society.

[203] A. K. Schnackenberg and E. C. Tomlinson. Organizational Transparency: A New Perspective in Managing Trust in Organization-Stakeholder Relationships. *Journal of Management*, 42(7):1784–1810, 2014.

[204] A. Schoonjans. *Social Network Analysis techniques in Enterprise Architecture Management*. PhD thesis, Ghent University, Ghent, 2016.

[205] R. Seghiri, F. Boulanger, C. Lecocq, and V. Godefroy. An Executable Model Driven Framework for Enterprise Architecture Application to the Smart Grids Context. In *2016 49th Hawaii International Conference on System Sciences (HICSS)*, pages 4546–4555, 2016.

[206] G. Shani and A. Gunawardana. Evaluating recommendation systems. In *Recommender systems handbook*, pages 257–297. Springer, 2011.

[207] M. Shaw. What Makes Good Research in Software Engineering? *International Journal on Software Tools for Technology Transfer*, 4(1):1–7, 2002.

[208] J. Sichi, J. Kinable, D. Michail, B. Naveh, and Contributors. JGraphT - Graph Algorithms and Data Structures in Java (Version 1.1.0), 2017.

[209] D. Simon and K. Fischbach. IT landscape management using network analysis. In *Enterprise Information Systems of the Future*, pages 18–34. Springer, 2013.

[210] D. Simon, K. Fischbach, and D. Schoder. An Exploration of Enterprise Architecture Research. *Communications of the Association for Information Systems*, 32(1):1–72, 2013.

[211] T. Sommestad, M. Ekstedt, and H. Holm. The cyber security modeling language: A tool for assessing the vulnerability of enterprise system architectures. *IEEE Systems Journal*, 7(3):363–373, 2013.

[212] T. Sørensen. A method of establishing groups of equal amplitude in plant sociology based on similarity of species and its application to analyses of the vegetation on Danish commons. *Biol. Skr.*, 5:1–34, 1948.

[213] P. Sousa, J. Lima, A. Sampaio, and C. Pereira. An Approach for Creating and Managing Enterprise Blueprints: A Case for IT Blueprints. In A. Albani, J. Barjis, and J. L. G. Dietz, editors, *Advances in Enterprise Engineering III*, pages 70–84, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[214] P. Sousa, R. Martins, and A. Sampaio. A Clarification of the Application Concept: The Caixa Geral de Depósitos Case. In E. Proper, K. Gaaloul, F. Harmsen, and S. Wrycza, editors, *Practice-Driven Research on Enterprise Transformation*, pages 1–17, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[215] S. Sousa, D. Marosin, K. Gaaloul, and N. Mayer. Assessing risks and opportunities in enterprise architecture using an extended ADT approach. In *Enterprise Distributed Object Computing Conference (EDOC), 2013 17th IEEE International*, pages 81–90, 2013.

[216] C. Spence and V. Michell. Measuring the Quality of Enterprise Architecture Models. *Journal of Enterprise Architecture*, 12(3):64–74, 2016.

[217] A. Steffens, H. Lichter, and J. S. Döring. Designing a Next-Generation Continuous Software Delivery System: Concepts and Architecture. In *2018 IEEE/ACM 4th International Workshop on Rapid Continuous Software Engineering (RCoSE)*, volume 00, pages 1–7, 2018.

[218] B. Stroud and A. Ertas. Enterprise cyclomatic complexity. In *2016 Annual IEEE Systems Conference (SysCon)*, pages 1–7, 2016.

[219] N. Subramanian, L. Chung, and Y.-t. Song. An NFR-Based Framework for Establishing Traceability between Enterprise Architectures and System Architectures. In *Seventh ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD'06)*, pages 21–28, 2006.

[220] R. Sundarraj and S. Talluri. A multi-period optimization model for the procurement of component-based enterprise information technologies. *European Journal of Operational Research*, 146(2):339–351, 2003.

[221] S. Sunkle, D. Kholkar, H. Rathod, and V. Kulkarni. Incorporating directives into enterprise TO-BE architecture. In *Enterprise Distributed Object Computing Conference*

Workshops and Demonstrations (EDOCW), 2014 IEEE 18th International, pages 57–66, 2014.

[222] T. Tamm, P. B. Seddon, G. Shanks, and P. Reynolds. How does enterprise architecture add value to organisations? 28:141–168, 2011.

[223] The Open Group. *TOGAF Version 9.1*. Van Haren Publishing, Zaltbommel, 1 edition, 2011.

[224] The Open Group. *ArchiMate 2.1 Specification*. 2013.

[225] The Open Group. *ArchiMate 3.0.1 Specification*. 2017.

[226] M. T. Thielsch and S. Weltzin. Online-Umfragen und Online-Mitarbeiterbefragungen. *Praxis der Wirtschaftspsychologie II*, pages 109–127, 2012.

[227] F. Timm, S. Hacks, F. Thiede, and D. Hintzpeter. Towards a Quality Framework for Enterprise Architecture Models. In H. Lichter, T. Anwar, and T. Sunetnanta, editors, *Proceedings of the 5th International Workshop on Quantitative Approaches to Software Quality (QuASoQ 2017) co-located with APSEC 2017*, pages 10–17. CEUR-WS.org, 2017.

[228] M. Välja, M. Korman, R. Lagerström, U. Franke, and M. Ekstedt. Automated architecture modeling for enterprise technology manageme using principles from data fusion: A security analysis case. In *Portland International Conference on Management of Engineering and Technology (PICMET)*, 2016.

[229] M. Välja, R. Lagerström, M. Ekstedt, and M. Korman. A Requirements Based Approach for Automating Enterprise IT Architecture Modeling Using Multiple Data Sources. In *19th International Enterprise Distributed Object Computing Workshop*, 2015.

[230] J.-P. van Belle. Evaluation of selected enterprise reference models. In *Reference Modeling for Business Systems Analysis*, pages 266–287. IGI Global, 2007.

[231] R. van Buuren, H. Jonkers, M.-E. Iacob, and P. Strating. Composition of Relations in Enterprise Architecture Models. In H. Ehrig, G. Engels, F. Parisi-Presicce, and G. Rozenberg, editors, *Graph Transformations Second International Conference*, 2004.

[232] A. Vasconcelos, C. M. Pereira, P. M. A. Sousa, and J. M. Tribolet. Open Issues on Information System Architecture Research Domain: The Vision. In *ICEIS*, pages 273–282, 2004.

[233] J. Venable. A framework for design science research activities. In *Emerging Trends and Challenges in Information Technology Management: Proceedings of the 2006 Information Resource Management Association Conference*, pages 184–187, 2006.

[234] J. Venable, J. Pries-Heje, and R. Baskerville. FEDS: a framework for evaluation in design science research. *European Journal of Information Systems*, 25(1):77–89, 2016.

[235] R. K. Veneberg, M.-E. Iacob, M. J. van Sinderen, and L. Bodenstaff. Enterprise architecture intelligence: combining enterprise architecture and operational data. In *Enterprise Distributed Object Computing Conference (EDOC), 2014 IEEE 18th International*, pages 22–31, 2014.

[236] S. Vodanovich, D. Sundaram, and M. Myers. Research Commentary—Digital Natives and Ubiquitous Information Systems. *Information Systems Research*, 21(4):711–723, 2010.

[237] J. Webster and R. T. Watson. Analyzing the Past to Prepare for the Future: Writing a Literature Review. *MIS Q*, 26(2):xiii–xxiii, 2002.

[238] S. Wenzel, H. Hutter, and U. Kelter. Tracing Model Elements. In *International Conference on Software Maintenance*, pages 104–113, New York, NY, USA, 2007. IEEE.

[239] S. A. White. *BPMN modeling and reference guide: understanding and using BPMN*. Future Strategies Inc, 2008.

[240] K. Winter, S. Buckl, F. Matthes, and C. M. Schweda. Investigating the state-of-the-art in enterprise architecture management method in literature and practice. In *5th Mediterranean Conference on Information Systems*. AIS, 2010.

[241] R. Winter. Establishing Architectural Thinking in Organizations. In J. Horkoff, M. A. Jeusfeld, and A. Persson, editors, *The Practice of Enterprise Modeling : 9th IFIP WG 8.1. Working Conference, PoEM 2016, Skövde, Sweden, November 8-10, 2016, Proceedings*, volume 267 of *Lecture Notes in Business Information Processing*, pages 3–8. Springer International Publishing, 2016.

[242] R. Winter and R. Fischer. Essential Layers, Artifacts, and Dependencies of Enterprise Architecture. In *10th IEEE International Enterprise Distributed Object Computing Conference Workshops*, pages 30–38, New York, NY, USA, 2006. IEEE.

[243] M. Wißotzki, F. Timm, and P. Stelzer. Current State of Governance Roles in Enterprise Architecture Management Frameworks. In *International Conference on Business Informatics Research*, pages 3–15, 2017.

[244] A. Wittenburg. *Softwarekartographie: Modelle und Methoden zur systematischen Visualisierung von Anwendungslandschaften*. Dissertation, TU München, München, Germany, 2007.

[245] J. Wood, S. Sarkani, T. Mazzuchi, and T. Eveleigh. A framework for capturing the hidden stakeholder system. *Systems Engineering*, 16(3):251–266, 2012.

[246] A. Xavier, A. Vasconcelos, and P. Sousa, editors. *Rules for Validation of Models of Enterprise Architecture - Rules of Checking and Correction of Temporal Inconsistencies among Elements of the Enterprise Architecture*. SciTePress, 2017.

[247] R. K. Yin. *Case Study Research: Design and Methods*. Sage Publications, Thousand Oaks and London and New Delhi, 5 edition, 2013.

[248] T. Ylimäki. Potential Critical Success Factors for Enterprise Architecture. *Journal of Enterprise Architecture*, 2(4):29–40, 2006.

[249] E. Yu, M. Strohmaier, and X. Deng. Exploring Intentional Modeling and Analysis for Enterprise Architecture. In *2006 10th IEEE International Enterprise Distributed Object Computing Conference Workshops (EDOCW'06)*, page 32, 2006.

[250] A. Zimmermann, R. Schmidt, D. Jugel, and M. Möhring. Evolving enterprise architectures for digital transformations. In A. Zimmermann and A. Rossmann, editors, *Digital Enterprise Computing (DEC 2015)*, pages 183–194, Bonn, 2015. Gesellschaft für Informatik e.V.

[251] K. Zimmermann. *Referenzprozessmodell für das Business-IT-Management: Vorgehen, Erstellung und Einsatz auf Basis qualitativer Forschungsmethoden*. Dissertation, University of Hamburg, Hamburg, Germany, 2013.

List of Publications

[1] A. Barbosa, A. Santana, S. Hacks, and N. v. Stein. A Taxonomy for Enterprise Architecture Analysis Research. In *Proceedings of the 21st International Conference on Enterprise Information Systems*, volume 2, pages 493–504. SciTePress, 2019.

[2] B. Bebensee and S. Hacks. Applying Dynamic Bayesian Networks for Automated Modeling in ArchiMate: A Realization Study. In *Proceedings of the 2019 IEEE 23rd International Enterprise Distributed Object Computing Conference Workshops and Demonstrations (to be published)*, 2019.

[3] V. Borozanov, S. Hacks, and N. Silva. Using Machine Learning Techniques for Evaluating the Similarity of Enterprise Architecture Models. In P. Giorgini and B. Weber, editors, *Advanced Information Systems Engineering*, pages 563–578. Springer International Publishing, 2019.

[4] N. Dohmen, K. Koopmann, and S. Hacks. Optimizing Enterprise Architecture Considering Different Budgets. In C. Czarnecki, E. Sultanow, and C. Brockmann, editors, *Workshops der Informatik 2019 (to be published)*, Lecture Notes in Informatics, Bonn, 2019. Gesellschaft für Informatik e.V.

[5] S. Hacks, M. Brosius, and S. Aier. A Case Study of Stakeholder Concerns on EAM. In U. Franke, S. Aier, and M. Mocker, editors, *21st International Enterprise Distributed Object Computing Workshop (EDOCW)*, 2017.

[6] S. Hacks, A. Hacks, S. Katsikeas, B. Klaer, and R. Lagerström. Creating Meta Attack Language Instances using ArchiMate: Applied to Electric Power and Energy System Cases. In *Proceedings of the 2019 IEEE 23rd International Enterprise Distributed Object Computing Conference (to be published)*. IEEE, 2019.

[7] S. Hacks, H. Höfert, J. Salentin, Y. C. Yeong, and H. Lichter. Towards the Definition of Enterprise Architecture Debts. In *Proceedings of the 2019 IEEE 23rd International Enterprise Distributed Object Computing Conference Workshops and Demonstrations (to be published)*, 2019.

[8] S. Hacks and H. Lichter. Distributed Enterprise Architecture Evolution–A Roundtrip Approach. In *Doctoral Consortium Wirtschaftsinformatik 2017 (unpublished)*, 2017.

[9] S. Hacks and H. Lichter. Optimizing Enterprise Architectures Using Linear Integer Programming Techniques. In M. Eibl and M. Gaedke, editors, *INFORMATIK 2017*, pages 623–636, Bonn, 2017. Gesellschaft für Informatik e.V.

[10] S. Hacks and H. Lichter. A Probabilistic Enterprise Architecture Model Evolution. In *2018 IEEE 22nd International Enterprise Distributed Object Computing Conference (EDOC)*, pages 51–57, 2018.

[11] S. Hacks and H. Lichter. Optimierung von Unternehmensarchitekturen unter Berücksichtigung von Transitionskosten. *HMD Praxis der Wirtschaftsinformatik*, 55:928–941, 2018.

[12] S. Hacks and H. Lichter. Towards an Enterprise Architecture Model Evolution. In C. Czarnecki, E. Sultanow, and C. Brockmann, editors, *Workshops der Informatik 2018*, Lecture Notes in Informatics, Bonn, 2018. Gesellschaft für Informatik e.V.

[13] S. Hacks and H. Lichter. Qualitative Comparison of Enterprise Architecture Model Maintenance Processes. In *EMISA 2019 conference (to be published)*, 2019.

[14] S. Hacks, A. Steffens, P. Hansen, and N. Rajashekar. A Continuous Delivery Pipeline for EA Model Evolution. In I. Reinhartz-Berger, J. Zdravkovic, J. Gulden, and R. Schmidt, editors, *Enterprise, Business-Process and Information Systems Modeling. BPMDS 2019, EMMSAD 2019*, pages 141–155. Springer International Publishing, 2019.

[15] S. Hacks and F. Timm. Towards a Quality Framework for Enterprise Architecture Models (Extended Abstract). *EMISA Forum*, 38(1):32–33, 2018.

[16] S. Katsikeas, P. Johnson, S. Hacks, and R. Lagerström. Probabilistic Modeling and Simulation of Vehicular Cyber Attacks: An Application of the Meta Attack Language. In *Proceedings of the 5th International Conference on Information Systems Security and Privacy*, pages 175–182. SCITEPRESS - Science and Technology Publications, 2019.

[17] D. Mathew, S. Hacks, and H. Lichter. Developing a Semantic Mapping betwen TOGAF and BSI-IT-Grundschutz. In P. Drews, B. Funk, P. Niemeyer, and L. Xie, editors, *Multikonferenz Wirtschaftsinformatik (MKWI) 2018*, volume 5, pages 1971–1982, 2018.

[18] N. Rajashekar, S. Hacks, and N. Silva. A Performance Comparison of Graph Analytic Methods for Supporting Enterprise Architecture Model Maintenance. In *Practice of Enterprise Modelling 2019 Forum (to be published)*. Springer, 2019.

[19] F. Timm, S. Hacks, F. Thiede, and D. Hintzpeter. Towards a Quality Framework for Enterprise Architecture Models. In H. Lichter, T. Anwar, and T. Sunetnanta, editors, *Proceedings of the 5th International Workshop on Quantitative Approaches to Software Quality (QuASoQ 2017) co-located with APSEC 2017*, pages 10–17. CEUR-WS.org, 2017.

List of Figures

Listings

List of Tables

Part VI.

Appendix

Appendix A.

Research Method Details

A.1. Concerns of EA Stakeholders – Creation Process

To come to the EA stakeholder concerns, presented in Section 3, we follow a two-step process, starting with the data collection, followed by a scheme-guided classification for presenting and discussing the data. The information processed in this work is collected within the environment already presented in Section 1.4.1.

A.1.1. Data Collection

A single case fits our purpose of gaining a first, in-depth reflection of hierarchical differences of different stakeholder concerns in a real life scenario [247]. A case study is further suitable to shed light on the phenomenon of interest from different perspectives [247], which fitted our research objective of tackling stakeholders from different hierarchical levels with diverse concerns (Figure A.1). Following Patton [176], we decided to conduct our case study by a series of open-ended interviews, using a fixed set of questions for all interviewees. Choosing this method over a fixed set of questions thereby contributed to a high degree of comparability of our respondents' answers.

In line with our research objectives, interview questions were focused on EA deliverables as well as stakeholder concerns on EA. In total, 36 questions were developed. In order to guide our participants structured through the interviews, the developed questions were assigned into seven sequential blocks of questions (i.e., introduction, perception, points of contact, strategic role, architectures, policies, and solution architectures).

The interviews lasted up to 60 minutes. In total, 38 stakeholders participated in the interviews. The chosen stakeholders represented different hierarchical levels (see Figure A.1): Operational management (e.g., group leaders, solution architects), middle management (e.g., division leaders, project leads), and top management (i.e. management board). The operational management was represented by 18 interviewees, the middle management by 15 interviewees, and the top management by five interviewees.

All interviews were recorded and transcribed. Finally, the transcripts were synthesized into a comprehensive summary of 1,042 stakeholder concerns for further classification (see scheme-guided classification).

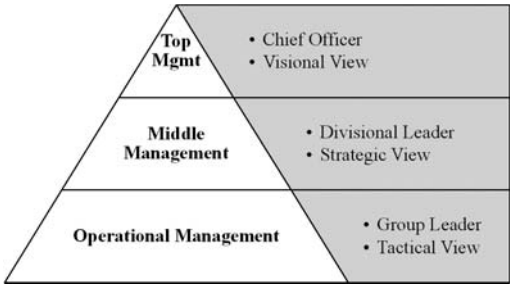


Figure A.1.: EA's Different Groups of Stakeholder.

A.1.2. Scheme-guided classification

In order to synthesize and present the collected 1,042 stakeholder concerns, we used a scheme-guided classification. Following the two steps by Miles and Huberman [150] as well as Eisenhardt [59], two of the three involved researchers accounted responsible for the early analysis and coding.

Early analysis At the outset, the consolidated transcript was screened, encompassing 1,042 synthesized statements of stakeholder concerns. These statements were studied for the identification of common characteristics: Statements referring to the same concern —or to similar characteristics of one and the same concern— were assigned to one dimension. Finally, the early analysis resulted in five dimensions of stakeholder concerns (Table 3.2). Consistent with our research objectives, these two dimensions differentiate EA deliverables and organizational anchoring.

Coding Counting toward the most comprehensive and widest implemented EA approaches [223], we applied TOGAF for developing the terminology of dimensions and for facilitating these dimensions with illustrative characteristics (Table 3.2). The first dimension of stakeholder concerns gave particular rise to the differences among EA deliverables:

Type [223]. Type refers to EA artefacts, being contractually specified and formally reviewed. In line with TOGAF, we differentiated two main groups of deliverables that interviewees reported, namely, architectures (e.g., as-is architectures, business domain model, infrastructure blueprint) and policies (e.g., principles, documentation rules, decision boards, standards).

The second dimension introduces *quality* to EA deliverables. Following TOGAF [223], three quality criteria were added to consolidate the respondents' concerns: Actuality, stability, and simplicity. While EA needs to deliver in quality to a wide range of diverse stakeholders, deliverables are required to maintain a certain degree of *abstraction* [223], which represents the third dimension of the classification scheme. More general, we differentiate high and low levels of abstraction. The fourth dimensions, *context*, describes the environmental setting and specificities, in which EA operates [223]. Context spans the set of expectations toward and acceptances of the EA function, which prevail in the organizational environment, as well as the structural

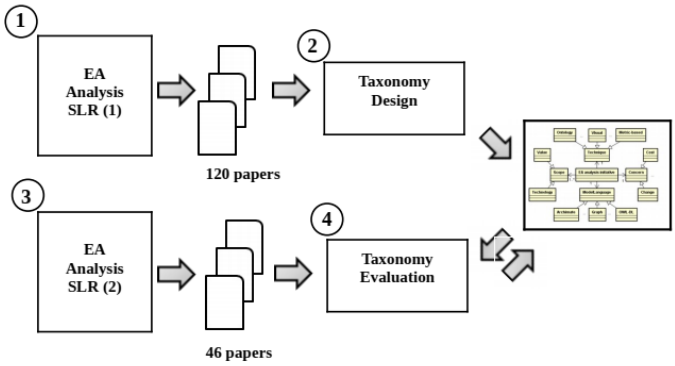


Figure A.2.: Research Design.

arrangement of organizational units. This includes, among others, the sufficient staffing of architecture relevant tasks as well as an organizational culture centered on architectural policies. Finally, *transparency* accounts as fifth dimension of the classification scheme [223], focusing on "the perceived quality of intentionally shared information" in the context of architectural guidance [203]. Transparency applies to EA deliverables as well as the organizational anchoring of the EA function.

Classification scheme Our final classifications scheme is comprised of five major dimensions and twelve facilitating characteristics (Table 3.2).

A.2. A Taxonomy of EA Analysis Research – SLR

Within this section, we like to present the details of the SLR grounding the taxonomy in Section 4. This is a qualitative and descriptive research split up into four steps. First, we apply the SLR method according to Kitchenham [114] to gather a set of papers related to EA analysis research (Step 1 in Figure A.2). Second, we perform a data categorization [45] to end up with a taxonomy answering the question: "How to classify EA analysis research according to its analysis concerns and modeling languages?" (Step 2 in Figure A.2). We gathered a second dataset with papers published between 2016 and September 2018 (Step 3 in Figure A.2). Finally, we apply the taxonomy created in Step 2 in the evaluation dataset (from Step 3) to evaluate and improve the taxonomy. Our research design is depicted in Figure A.2. The SLR steps are detailed in the next sections.

Research Query The keyword design was intentionally generic as it aimed for wide coverage of publications in the EA analysis field. The final string combined the terms related to EA and its subsets, as used in the work of [210]; and terms related to "analysis" such as goals, metrics, and evaluation, as listed by [18]. Thus, our final string was:

("Enterprise architecture" OR "business architecture" OR "process architecture" OR "information systems architecture" OR "IT architecture" OR "IT landscape" OR "information architecture" OR "data architecture" OR "application architecture" OR "application landscape" OR "integration architecture" OR "technology architecture" OR "infrastructure architecture") AND (Goals OR concerns OR methods OR procedures OR approaches OR analysis OR evaluate OR assess* OR indicator OR method OR measur* OR metric)*

Inclusion and Exclusion Criteria The inclusion criteria consisted of papers containing techniques, methods or any initiative to evaluate EA, e.g., papers which use EA as input for taking decision or papers that analyze EA itself, its changes and evolution. Papers in any language but English, related to product architecture analysis or internal architecture of software, containing only modeling approaches or that do not analyze EA itself but instead they describe the EA as a whole organizational function to an organizational variable (e.g., organizational performance) were not included. Literature reviews about EA (secondary studies) and papers dealing with the discussion of analysis approaches, but not performing any, were also excluded from the study.

Used Engines We selected the main engines/databases accessed in the information system community as our data-sources for primary studies: Scopus, IEEE, ScienceDirect, ISI Web of knowledge and AIS electronic library. Duplicates were removed. Table A.3 presents the results returned by each engine.

Table A.3.: Results by engine for the two time intervals of our SLR.

Engine	Time interval 1	Time interval 2
IEEE	1,762	358
ScienceDirect	832	623
Scopus	3439	949
AISEL	25	0
ISI	1,162	no access
Total (duplicates removed)	5174	1076

Screening Phases The SLR was performed considering two intervals. The first one (Step 1 of our research design) covers papers published until 2015. Then, using the data extracted from those papers, we applied the data categorization to derive our taxonomy's constructs. The second interval (related to Step 3 of our research design) encompasses papers published from 2015 to September 2018.

Considering the previous inclusion and exclusion criteria, our screening process was divided into three rounds for each one of the two-time intervals. For the first interval, during our first round, we read 7,220 abstracts and titles of primary studies returned by the engines. In the next round, the reading focus was on the introduction and conclusion sections of 803 remaining papers. Finally, the 183 resulted papers were completely read, forming a set of 120 final papers.

For the second interval (2016 to September 2018), we performed the same previous screening strategy: the first round had 1,076 titles and abstracts to be read, the second had 168 introductions and conclusions, 65 full paper readings in the third and 46 final papers as final dataset. We took the papers from this second set to validate the produced taxonomy.

Data Categorization We screened the 120 papers from the first data-set for the identification of common dimensions related to the EA analysis. Considering our research goals, this ended up in the four dimensions: EA Scope, Analysis Technique, Analysis Concern, and Modeling Language.

To bring the *coding* into practice, we follow an inductive approach of Cruzes and Dyba [45]. We reviewed the data line by line in detail and as a value becomes apparent, a code is assigned. To ascertain whether a code is appropriately assigned, we compare text segments to segments that have been previously assigned the same code and decide whether they reflect the same value. This leads to continuous refinement of the dimensions of existing codes and identification of new ones [45]. This process does not necessarily take a linear order rather an iterative and dynamic one. In the next section, we present the proposed taxonomy.

A.3. Integrated Enterprise Architecture Roundtrip – DSR

Following, we like to give a short overview of the DSR process applied in Section 5.

Problem Identification and Motivation To keep EA models up-to-date is a well-known issue in research (e.g., [62, 63, 155, 112, 113, 228]). However, none of those existing approaches tries to tackle the problem from an holistic point of view. Therefore, we elaborate on the technical issues of this problem (cf. Section 5.1). The problem itself occurs first time at the EA unit of one of our cooperation partners described in Section 1.4.1). Projects document which changes they plan to apply on the EA and the EA unit desires to integrate those changes into the central EA model.

Define Objectives The objectives of the artefact to be created get specified by the research question in Section 5. Especially, we focus on technical facets of this research question and neglect socio-technical problems as exemplary elaborated by Aier et al. [5, 7, 8, 9]. As we understand the sketched process as framework to guide our research, the research question gets detailed in the further parts of this work.

Design and Development We developed a process depicted in Section 5.2. This process is already a result of several iterations applying the DSR methodology. The process was presented and discussed in several different environments, e.g., at a doctoral consortium [80], at a workshop on EA [84], internally at our research group, and externally with our cooperation partner.

Demonstration The demonstratoin of our process is put into practice by applying it to a single case study (cf. Section 6). Single case studies gain a first, in-depth reflection on means in real life scenarios [247]. Moreover, single case studies are a feasible instrument to show applicability.

Evaluation The proposed architecture roundtrip process itself is evaluated in Section 11. We follow the approach presented by Venable et al. [234], classify our evaluation along Cleven et al. [44], and conduct a quantitative study to compare different EA model maintenance processes.

Communication The communication step of DSR is carried out by publishing parts of the work and the work itself. Based on the results of our work, i.e., the architecture roundtrip process, future research can elaborate on different result's facets. First, the whole process can be implemented in different organizations to evaluate and improve the process. Second, single steps of the process deserve more attentiveness, since existing approaches need to be adapted to the domain of EA or existing research seems to be still scarce. Last, the influence of the process on the organization and its employees shall be investigated as well. Therefore, possible drawbacks of an alignment of projects to a roundtrip could be interesting. For example, it could be explored if projects get slowed down by the alignment or the additional communication effort reduces project's agility.

A.4. A Continuous Delivery Pipeline for EA Model Evolution – DSR

Following, we like to give a short overview of the DSR process applied in Section 6.

Identify Problem & Motivate As previous research has shown, reasons to change the EA model are manifold [62] and raise many different challenges [92]. One of them is to handle different sources and how to design a suitable process for EA model maintenance. We believe that the principle of continuous delivery offer efficient means to support the EA model maintenance process.

Define Objectives Based on our research problem stated before, we identified mainly three sources for objectives: First, Farwick et al. [61] identified several requirements on automated EA model maintenance which should be incorporated into a feasible solution. Second, Fischer et al. [63] describe a EA model maintenance process and related roles which needs to be inherited into our resulting artifact. Last, we presented a process for a distributed EA model evolution [84] describing different tasks and their sequence which should serve as conceptual framework for our pipeline.

Design & Development To realize an artifact in accordance to the beforehand identified objectives, first, we align the input of the three objectives' sources. Then, we design an abstract process model using BPMN [239] and implement it using JARVIS [217]. Our derived integrated

EA maintenance process consists of activities, which will be implemented as microservices following JARVIS' architectural framework. In addition to the activities defined in our objectives, we include additional steps inspired by principles found in the continuous delivery domain.

Demonstration The demonstration is put into practice by applying the proposed means to a single fictitious case study. Single case studies gain a first, in-depth reflection on means in real life scenarios [247]. Moreover, single case studies are a feasible instrument to show applicability. Our case study is based on an EA model illustrating an airport.

Evaluation We identified 54 equivalence classes of possible actions which should be considered in our pipeline. Therefore, we created for each class an exemplary test case as a representative for this class [38, p. 623].

Communication The communication is done with this work itself and its presentation on a conference.

A.5. Assessing EA Model Quality – SLR

Following, we present the details on our conducted SLR, which we utilized to develop ourEAQF presented in 8. For the development of means to solve our research question, we conducted a SLR by combining the approaches by Kitchenham et al. [114] and Webster and Watson [237]. After defining the SLR scope, which is in line with our research question, we searched for the combination of the terms "enterprise architecture", "model" and "quality" in abstracts of articles on the Scopus[1] and AISeL[2] databases from 2007 to the present. After analysing the titles and abstracts of the 209 results, we gathered a first pool of four directly relevant articles, that discussed the quality of EA models [122, 48, 111, 136]. In a next step we searched back- and forward [165] with this basis and completed the literature base with further related work known to us [129, 27].

To demonstrate and evaluate our produced artifact, we apply it to a single case study. Our case study does not ensure that our quality attributes are sound and complete. Consequently, future feedback loops have to take this into account.

A.6. Avoiding Redundancies in EA Models – DSR

Following, we like to give a short overview of the DSR process applied in Section 9.1.

Identify Problem & Motivate We have identified the issue that due to different reasons enterprise architects may introduce new components into the EA repository, which already have a representation within the repository. This leads to an unintended pollution of the repository and might cause misled decisions based on faulty reports.

[1] https://www.scopus.com
[2] http://aisel.aisnet.org

Objectives and Solution The objective was to develop a solution that will identify all new components, evaluate the model against the repository, and return a list of components already stored in the repository that can be reused in the given project. The final decision whether the suggested components will be incorporated or not is left to the architect. This ensures that the architecture model retains its correctness.

Design and Development The solution we proposed was a machine learning model. The data on our disposal was unlabeled – we did not have any information on what the correct substitution for the specific component should be. Therefore, we focused on the unsupervised approaches, combining them into one suitable model.

Demonstration and Evaluation We tested our solution with a simulated repository and architecture models where we knew in advance the correct substitutions for every newly introduced component. This allowed us to evaluate a labeled data so that we could see how correct the model recommends the components.

Communication The solution was distributed to the architects as a software service. It consisted of two parts: a server-side module where we performed the evaluation, and a client-side module which allowed architects to select the model that they wanted to evaluate.

A.7. A Performance Comparison of Graph Analytic Methods – ML Evaluation

Following, we present the details on our conducted comparison of different algorithms presented in 9.2. Motivated by the life-cycle model for ML frameworks provided by [199], we propose a general work-flow framework (cf. Figure A.4) to analyze EA models by comparing different graph analytic and ML algorithms and then selecting the best-performing model based on suitable performance metrics. The main idea is to improve the performance of a model by comparing different algorithms and metrics.

1. **Raw data:** XML schema is a widely used file exchange format in the context of modeling EA. Any architecture models can be easily expressed in XML, which is mainly used for storage and communication purposes. Thus, we first acquire our EA data in XML format. For further analysis, XML data is converted to data table format with three different files consisting of elements, relations, and properties tables. All the elements are identified by unique identifiers so that a relation will reference an element in the other files.

2. **Data preprocessing:** This step includes transforming raw data into a relevant format and cleaning the data set. Since EA models represent components and relationship between the components, this can be easily depicted as a graph-like structure where components represent nodes, and the relationship between the components represents edges in the graph. Thus, raw data obtained from elements and relations files will be parsed to obtain graph data object using igraph library [46].

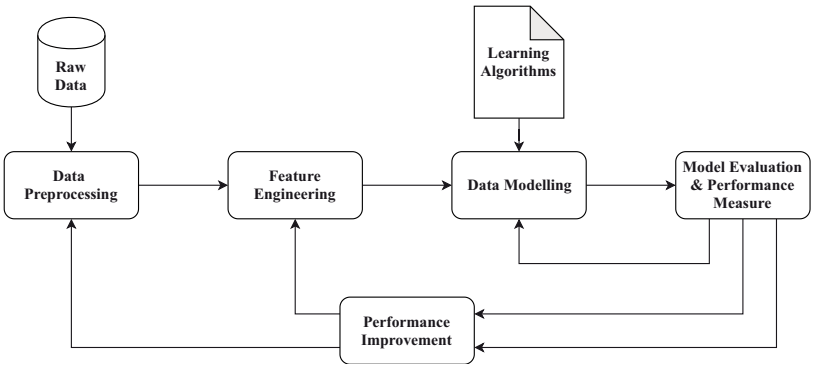


Figure A.4.: Work-flow Framework to compare different ML algorithms.

Data normalization is necessary to standardize the range of independent variables or features of data. Also, isolated components may tend to increase the size of the repository. This should be inspected and removed.

3. **Feature engineering:** The next step is to extract or compute relevant graph feature statistics. The features extracted from data will directly influence the predictive models and the results which we can obtain. Extracting a right feature set may boost the performance of a model.

4. **Data modeling:** After extracting the relevant data set and feature set, the next step is to perform the core ML task. Making use of supervised/unsupervised learning algorithms, we can evaluate similarity and investigate clusters/communities formed in order to analyse EA models in different context.

5. **Model evaluation & Performance measure:** Model evaluation mainly helps to find the best performing model based on suitable performance metrics that represent the data and how well the chosen model will work in the future.

6. **Performance improvement:** Iterating over different graph analytic or ML algorithms and feature set can result in obtaining optimized results which improve the overall performance of the algorithms under inspection.

Appendix B.

Abbreviations

ADM Architecture Development Method

AHP Analytic Hierarchy Process

ALM Application Lifecycle Management

ATD Architecture Theory Diagram

AWP Anwendungssystemportfolio

BI Business Intelligence

BIM Business-IT-Management

BITAM Business-IT-Alignment

BMM Business Motivation Model

BPMN Business Process Model and Notation

BSI Bundesamt für Sicherheit in der Informationstechnik

CAN Controller Area Network

CD Continuous Delivery

CEO Chief Executive Officer

CIO Chief Information Officer

CMDB Configuration Management Database

CSV Comma Seperated Values

DoDAF Department of Defense Architecture Framework

DSL Domain Specific Language

DSR Design Science Research

EA Enterprise Architecture

EAF Enterprise Architecture Framework

EAQF EA Quality Framework

ECU Electronic Control Unit

EISA Enterprise Information Security Architecture

ENISA European Network and Information Security Agency

ESB Enterprise Service Bus

FEAF Federal Enterprise Architecture Framework

GPS Global Positioning System

HRM Human Resource Management

IDE Integrated Development Environment

IDPS Intrusion Detection and Prevention System

IoT Internet of Things

IS Information System

ISM Information Security Management

ISMS Information Security Management System

IT Information Technology

ITIL IT Infrastructure Library

ILM Infrastructure Lifecycle Management

JSON JavaScript Object Notation

KPI Key Performance Indicator

LAN Local Area Network

LDAP Lightweight Directory Access Protocol

LIN Local Interconnect Network

LIP Linear Integer Program

LSA Latent Semantic Analysis

MAL Meta Attack Language

ML Machine Learning

OCL Object Constraint Language

OWL-DL Web Ontology Language - Description Logic

P^{2}AMF Predictive, Probabilistic Architecture Modeling Framework

PBX Private Branch Exchange

PM Project Management

ROI Return on Investment

SCM Software Configuration Management

SD Standard Deviation

SE Software Engineering

SLR Systematic Literature Review

SOA Service Oriented Architecture

SVD Singular Value Decomposition

TF-IDF Term Frequency - Inverse Document Frequency

TOGAF The Open Group Architecture Framework

TSP Traveling Salesperson Problem

UAT User Acceptance Test

UML Unified Modeling Language

XMI XML Metadata Interchange

XML Extensible Markup Language

Related Interesting Work from the SE Group, RWTH Aachen

Agile Model Based Software Engineering

Agility and modeling in the same project? This question was raised in [Rum04]: "Using an executable, yet abstract and multi-view modeling language for modeling, designing and programming still allows to use an agile development process." Modeling will be used in development projects much more, if the benefits become evident early, e.g with executable UML [Rum02] and tests [Rum03]. In [GKRS06], for example, we concentrate on the integration of models and ordinary programming code. In [Rum12] and [Rum16], the UML/P, a variant of the UML especially designed for programming, refactoring and evolution, is defined. The language workbench MontiCore [GKR+06, GKR+08] is used to realize the UML/P [Sch12]. Links to further research, e.g., include a general discussion of how to manage and evolve models [LRSS10], a precise definition for model composition as well as model languages [HKR+09] and refactoring in various modeling and programming languages [PR03]. In [FHR08] we describe a set of general requirements for model quality. Finally [KRV06] discusses the additional roles and activities necessary in a DSL-based software development project. In [CEG+14] we discuss how to improve reliability of adaprivity through models at runtime, which will allow developers to delay design decisions to runtime adaptation.

Generative Software Engineering

The UML/P language family [Rum12, Rum11, Rum16] is a simplified and semantically sound derivate of the UML designed for product and test code generation. [Sch12] describes a flexible generator for the UML/P based on the MontiCore language workbench [KRV10, GKR+06, GKR+08]. In [KRV06], we discuss additional roles necessary in a model-based software development project. In [GKRS06] we discuss mechanisms to keep generated and handwritten code separated. In [Wei12] demonstrate how to systematically derive a transformation language in concrete syntax. To understand the implications of executability for UML, we discuss needs and advantages of executable modeling with UML in agile projects in [Rum04], how to apply UML for testing in [Rum03] and the advantages and perils of using modeling languages for programming in [Rum02].

Unified Modeling Language (UML)

Starting with an early identification of challenges for the standardization of the UML in [KER99] many of our contributions build on the UML/P variant, which is described in the two books [Rum16] and [Rum12] implemented in [Sch12]. Semantic variation points of the UML are discussed in [GR11]. We discuss formal semantics for UML [BHP+98] and describe UML semantics using the "System Model" [BCGR09a], [BCGR09b], [BCR07b] and [BCR07a]. Semantic variation points have, e.g., been applied to define class diagram semantics [CGR08]. A precisely defined semantics for variations is applied, when checking variants of class diagrams [MRR11c] and objects diagrams [MRR11d] or the consistency of both kinds of diagrams [MRR11e]. We also apply these concepts to activity diagrams [MRR11b] which allows us to check for semantic differences of activity diagrams [MRR11a]. The basic semantics for ADs and their semantic variation points is given in [GRR10]. We also discuss how to ensure and identify model quality [FHR08], how models, views and the system under development correlate to each other [BGH+98] and how to use modeling in agile development projects [Rum04], [Rum02]. The question how to adapt and extend the UML is discussed in [PFR02] describing product line annotations for UML and more general discussions and insights on how to use meta-modeling for defining and adapting the UML are included in [EFLR99], [FELR98] and [SRVK10].

Domain Specific Languages (DSLs)

Computer science is about languages. Domain Specific Languages (DSLs) are better to use, but need appropriate tooling. The MontiCore language workbench [GKR+06, KRV10, Kra10, GKR+08] allows the specification of an integrated abstract and concrete syntax format [KRV07b] for easy development. New languages and tools can be defined in modular forms [KRV08, GKR+07, Völ11] and can, thus, easily be reused. [Wei12] presents a tool that allows to create transformation rules tailored to an underlying DSL. Variability in DSL definitions has been examined in [GR11]. A successful application has been carried out in the Air Traffic Management domain [ZPK+11]. Based on the concepts described above, meta modeling, model analyses and model evolution have been discussed in [LRSS10] and [SRVK10]. DSL quality [FHR08], instructions for defining views [GHK+07], guidelines to define DSLs [KKP+09] and Eclipse-based tooling for DSLs [KRV07a] complete the collection.

Software Language Engineering

For a systematic definition of languages using composition of reusable and adaptable language components, we adopt an engineering viewpoint on these techniques. General ideas on how to engineer a language can be found in the GeMoC initiative [CBCR15, CCF+15]. As said, the MontiCore language workbench provides techniques for an integrated definition of languages [KRV07b, Kra10, KRV10]. In [SRVK10] we discuss the possibilities and the challenges using metamodels for language definition. Modular composition, however, is a core concept to reuse language components like in MontiCore for the frontend [Völ11, KRV08] and the backend [RRRW15]]. Language derivation is to our believe a promising technique to develop new languages for a specific purpose that rely on existing basic languages. How to automatically derive such a transformation language using concrete syntax of the base language is described in [HRW15, Wei12] and successfully applied to various DSLs. We also applied the language derivation technique to tagging languages that decorate a base language [GLRR15] and delta languages [HHK+15a, HHK+13], where a delta language is derived from a base language to be able to constructively describe differences between model variants usable to build feature sets.

Modeling Software Architecture & the MontiArc Tool

Distributed interactive systems communicate via messages on a bus, discrete event signals, streams of telephone or video data, method invocation, or data structures passed between software services. We use streams, statemachines and components [BR07] as well as expressive forms of composition and refinement [PR99] for semantics. Furthermore, we built a concrete tooling infrastructure called MontiArc [HRR12] for architecture design and extensions for states [RRW13b]. MontiArc was extended to describe variability [HRR+11] using deltas [HRRS11, HKR+11] and evolution on deltas [HRRS12]. [GHK+07] and [GHK+08] close the gap between the requirements and the logical architecture and [GKPR08] extends it to model variants. [MRR14] provides a precise technique to verify consistency of architectural views [Rin14, MRR13] against a complete architecture in order to increase reusability. Co-evolution of architecture is discussed in [MMR10] and a modeling technique to describe dynamic architectures is shown in [HRR98].

Compositionality & Modularity of Models

[HKR+09] motivates the basic mechanisms for modularity and compositionality for modeling. The mechanisms for distributed systems are shown in [BR07] and algebraically underpinned in [HKR+07]. Semantic and methodical aspects of model composition [KRV08] led to the language workbench MontiCore [KRV10] that can even be used to develop modeling tools in a compositional form. A set of DSL design

guidelines incorporates reuse through this form of composition [KKP+09]. [Völ11] examines the composition of context conditions respectively the underlying infrastructure of the symbol table. Modular editor generation is discussed in [KRV07a]. [RRRW15] applies compositionality to Robotics control. [CBCR15] (published in [CCF+15]) summarizes our approach to composition and remaining challenges in form of a conceptual model of the "globalized" use of DSLs. As a new form of decomposition of model information we have developed the concept of tagging languages in [GLRR15]. It allows to describe additional information for model elements in separated documents, facilitates reuse, and allows to type tags.

Semantics of Modeling Languages

The meaning of semantics and its principles like underspecification, language precision and detailedness is discussed in [HR04]. We defined a semantic domain called "System Model" by using mathematical theory in [RKB95, BHP+98] and [GKR96, KRB96]. An extended version especially suited for the UML is given in [BCGR09b] and in [BCGR09a] its rationale is discussed. [BCR07a, BCR07b] contain detailed versions that are applied to class diagrams in [CGR08]. To better understand the effect of an evolved design, detection of semantic differencing as opposed to pure syntactical differences is needed [MRR10]. [MRR11a, MRR11b] encode a part of the semantics to handle semantic differences of activity diagrams and [MRR11e] compares class and object diagrams with regard to their semantics. In [BR07], a simplified mathematical model for distributed systems based on black-box behaviors of components is defined. Meta-modeling semantics is discussed in [EFLR99]. [BGH+97] discusses potential modeling languages for the description of an exemplary object interaction, today called sequence diagram. [BGH+98] discusses the relationships between a system, a view and a complete model in the context of the UML. [GR11] and [CGR09] discuss general requirements for a framework to describe semantic and syntactic variations of a modeling language. We apply these on class and object diagrams in [MRR11e] as well as activity diagrams in [GRR10]. [Rum12] defines the semantics in a variety of code and test case generation, refactoring and evolution techniques. [LRSS10] discusses evolution and related issues in greater detail.

Evolution & Transformation of Models

Models are the central artifact in model driven development, but as code they are not initially correct and need to be changed, evolved and maintained over time. Model transformation is therefore essential to effectively deal with models. Many concrete model transformation problems are discussed: evolution [LRSS10, MMR10, Rum04], refinement [PR99, KPR97, PR94], refactoring [Rum12, PR03], translating models from one language into another [MRR11c, Rum12] and systematic model transformation language development [Wei12]. [Rum04] describes how comprehensible sets of such transformations support software development and maintenance [LRSS10], technologies for evolving models within a language and across languages, and mapping architecture descriptions to their implementation [MMR10]. Automaton refinement is discussed in [PR94, KPR97], refining pipe-and-filter architectures is explained in [PR99]. Refactorings of models are important for model driven engineering as discussed in [PR01, PR03, Rum12]. Translation between languages, e.g., from class diagrams into Alloy [MRR11c] allows for comparing class diagrams on a semantic level.

Variability & Software Product Lines (SPL)

Products often exist in various variants, for example cars or mobile phones, where one manufacturer develops several products with many similarities but also many variations. Variants are managed in a Software Product Line (SPL) that captures product commonalities as well as differences. Feature diagrams describe

variability in a top down fashion, e.g., in the automotive domain [GHK+08] using 150% models. Reducing overhead and associated costs is discussed in [GRJA12]. Delta modeling is a bottom up technique starting with a small, but complete base variant. Features are additive, but also can modify the core. A set of commonly applicable deltas configures a system variant. We discuss the application of this technique to Delta-MontiArc [HRR+11, HRR+11] and to Delta-Simulink [HKM+13]. Deltas can not only describe spacial variability but also temporal variability which allows for using them for software product line evolution [HRRS12]. [HHK+13] and [HRW15] describe an approach to systematically derive delta languages. We also apply variability to modeling languages in order to describe syntactic and semantic variation points, e.g., in UML for frameworks [PFR02]. Furthermore, we specified a systematic way to define variants of modeling languages [CGR09] and applied this as a semantic language refinement on Statecharts in [GR11].

Cyber-Physical Systems (CPS)

Cyber-Physical Systems (CPS) [KRS12] are complex, distributed systems which control physical entities. Contributions for individual aspects range from requirements [GRJA12], complete product lines [HRRW12], the improvement of engineering for distributed automotive systems [HRR12] and autonomous driving [BR12a] to processes and tools to improve the development as well as the product itself [BBR07]. In the aviation domain, a modeling language for uncertainty and safety events was developed, which is of interest for the European airspace [ZPK+11]. A component and connector architecture description language suitable for the specific challenges in robotics is discussed in [RRW13b, RRW14]. Monitoring for smart and energy efficient buildings is developed as Energy Navigator toolset [KPR12, FPPR12, KLPR12].

State Based Modeling (Automata)

Today, many computer science theories are based on statemachines in various forms including Petri nets or temporal logics. Software engineering is particularly interested in using statemachines for modeling systems. Our contributions to state based modeling can currently be split into three parts: (1) understanding how to model object-oriented and distributed software using statemachines resp. Statecharts [GKR96, BCR07b, BCGR09b, BCGR09a], (2) understanding the refinement [PR94, RK96, Rum96] and composition [GR95] of statemachines, and (3) applying statemachines for modeling systems. In [Rum96] constructive transformation rules for refining automata behavior are given and proven correct. This theory is applied to features in [KPR97]. Statemachines are embedded in the composition and behavioral specification concepts of Focus [BR07]. We apply these techniques, e.g., in MontiArcAutomaton [RRW13a, RRW14] as well as in building management systems [FLP+11].

Robotics

Robotics can be considered a special field within Cyber-Physical Systems which is defined by an inherent heterogeneity of involved domains, relevant platforms, and challenges. The engineering of robotics applications requires composition and interaction of diverse distributed software modules. This usually leads to complex monolithic software solutions hardly reusable, maintainable, and comprehensible, which hampers broad propagation of robotics applications. The MontiArcAutomaton language [RRW13a] extends ADL MontiArc and integrates various implemented behavior modeling languages using MontiCore [RRW13b, RRW14, RRRW15] that perfectly fit Robotic architectural modelling. The LightRocks [THR+13] framework allows robotics experts and laymen to model robotic assembly tasks.

Automotive, Autonomic Driving & Intelligent Driver Assistance

Introducing and connecting sophisticated driver assistance, infotainment and communication systems as well as advanced active and passive safety-systems result in complex embedded systems. As these feature-driven subsystems may be arbitrarily combined by the customer, a huge amount of distinct variants needs to be managed, developed and tested. A consistent requirements management that connects requirements with features in all phases of the development for the automotive domain is described in [GRJA12]. The conceptual gap between requirements and the logical architecture of a car is closed in [GHK+07, GHK+08]. [HKM+13] describes a tool for delta modeling for Simulink [HKM+13]. [HRRW12] discusses means to extract a well-defined Software Product Line from a set of copy and paste variants. [RSW+15] describes an approach to use model checking techniques to identify behavioral differences of Simulink models. Quality assurance, especially of safety-related functions, is a highly important task. In the Carolo project [BR12a, BR12b], we developed a rigorous test infrastructure for intelligent, sensor-based functions through fully-automatic simulation [BBR07]. This technique allows a dramatic speedup in development and evolution of autonomous car functionality, and thus enables us to develop software in an agile way [BR12a]. [MMR10] gives an overview of the current state-of-the-art in development and evolution on a more general level by considering any kind of critical system that relies on architectural descriptions. As tooling infrastructure, the SSElab storage, versioning and management services [HKR12] are essential for many projects.

Energy Management

In the past years, it became more and more evident that saving energy and reducing CO2 emissions is an important challenge. Thus, energy management in buildings as well as in neighbourhoods becomes equally important to efficiently use the generated energy. Within several research projects, we developed methodologies and solutions for integrating heterogeneous systems at different scales. During the design phase, the Energy Navigators Active Functional Specification (AFS) [FPPR12, KPR12] is used for technical specification of building services already. We adapted the well-known concept of statemachines to be able to describe different states of a facility and to validate it against the monitored values [FLP+11]. We show how our data model, the constraint rules and the evaluation approach to compare sensor data can be applied [KLPR12].

Cloud Computing & Enterprise Information Systems

The paradigm of Cloud Computing is arising out of a convergence of existing technologies for web-based application and service architectures with high complexity, criticality and new application domains. It promises to enable new business models, to lower the barrier for web-based innovations and to increase the efficiency and cost-effectiveness of web development [KRR14]. Application classes like Cyber-Physical Systems and their privacy [HHK+14, HHK+15b], Big Data, App and Service Ecosystems bring attention to aspects like responsiveness, privacy and open platforms. Regardless of the application domain, developers of such systems are in need for robust methods and efficient, easy-to-use languages and tools [KRS12]. We tackle these challenges by perusing a model-based, generative approach [NPR13]. The core of this approach are different modeling languages that describe different aspects of a cloud-based system in a concise and technology-agnostic way. Software architecture and infrastructure models describe the system and its physical distribution on a large scale. We apply cloud technology for the services we develop, e.g., the SSELab [HKR12] and the Energy Navigator [FPPR12, KPR12] but also for our tool demonstrators and our own development platforms. New services, e.g., collecting data from temperature, cars etc. can now easily be developed.

References

[BBR07] Christian Basarke, Christian Berger, and Bernhard Rumpe. Software & Systems Engineering Process and Tools for the Development of Autonomous Driving Intelligence. *Journal of Aerospace Computing, Information, and Communication (JACIC)*, 4(12):1158–1174, 2007.

[BCGR09a] Manfred Broy, María Victoria Cengarle, Hans Grönniger, and Bernhard Rumpe. Considerations and Rationale for a UML System Model. In K. Lano, editor, *UML 2 Semantics and Applications*, pages 43–61. John Wiley & Sons, November 2009.

[BCGR09b] Manfred Broy, María Victoria Cengarle, Hans Grönniger, and Bernhard Rumpe. Definition of the UML System Model. In K. Lano, editor, *UML 2 Semantics and Applications*, pages 63–93. John Wiley & Sons, November 2009.

[BCR07a] Manfred Broy, María Victoria Cengarle, and Bernhard Rumpe. Towards a System Model for UML. Part 2: The Control Model. Technical Report TUM-I0710, TU Munich, Germany, February 2007.

[BCR07b] Manfred Broy, María Victoria Cengarle, and Bernhard Rumpe. Towards a System Model for UML. Part 3: The State Machine Model. Technical Report TUM-I0711, TU Munich, Germany, February 2007.

[BGH+97] Ruth Breu, Radu Grosu, Christoph Hofmann, Franz Huber, Ingolf Krüger, Bernhard Rumpe, Monika Schmidt, and Wolfgang Schwerin. Exemplary and Complete Object Interaction Descriptions. In *Object-oriented Behavioral Semantics Workshop (OOPSLA'97)*, Technical Report TUM-I9737, Germany, 1997. TU Munich.

[BGH+98] Ruth Breu, Radu Grosu, Franz Huber, Bernhard Rumpe, and Wolfgang Schwerin. Systems, Views and Models of UML. In *Proceedings of the Unified Modeling Language, Technical Aspects and Applications*, pages 93–109. Physica Verlag, Heidelberg, Germany, 1998.

[BHP+98] Manfred Broy, Franz Huber, Barbara Paech, Bernhard Rumpe, and Katharina Spies. Software and System Modeling Based on a Unified Formal Semantics. In *Workshop on Requirements Targeting Software and Systems Engineering (RTSE'97)*, LNCS 1526, pages 43–68. Springer, 1998.

[BR07] Manfred Broy and Bernhard Rumpe. Modulare hierarchische Modellierung als Grundlage der Software- und Systementwicklung. *Informatik-Spektrum*, 30(1):3–18, Februar 2007.

[BR12a] Christian Berger and Bernhard Rumpe. Autonomous Driving - 5 Years after the Urban Challenge: The Anticipatory Vehicle as a Cyber-Physical System. In *Automotive Software Engineering Workshop (ASE'12)*, pages 789–798, 2012.

[BR12b] Christian Berger and Bernhard Rumpe. Engineering Autonomous Driving Software. In C. Rouff and M. Hinchey, editors, *Experience from the DARPA Urban Challenge*, pages 243–271. Springer, Germany, 2012.

[CBCR15] Tony Clark, Mark van den Brand, Benoit Combemale, and Bernhard Rumpe. Conceptual Model of the Globalization for Domain-Specific Languages. In *Globalizing Domain-Specific Languages*, LNCS 9400, pages 7–20. Springer, 2015.

[CCF+15] Betty H. C. Cheng, Benoit Combemale, Robert B. France, Jean-Marc Jézéquel, and Bernhard Rumpe, editors. *Globalizing Domain-Specific Languages*, LNCS 9400. Springer, 2015.

[CEG+14] Betty Cheng, Kerstin Eder, Martin Gogolla, Lars Grunske, Marin Litoiu, Hausi Müller, Patrizio Pelliccione, Anna Perini, Nauman Qureshi, Bernhard Rumpe, Daniel Schneider, Frank Trollmann, and Norha Villegas. Using Models at Runtime to Address Assurance for Self-Adaptive Systems. In *Models@run.time*, LNCS 8378, pages 101–136. Springer, Germany, 2014.

[CGR08] María Victoria Cengarle, Hans Grönniger, and Bernhard Rumpe. System Model Semantics of Class Diagrams. Informatik-Bericht 2008-05, TU Braunschweig, Germany, 2008.

[CGR09] María Victoria Cengarle, Hans Grönniger, and Bernhard Rumpe. Variability within Modeling Language Definitions. In *Conference on Model Driven Engineering Languages and Systems (MODELS'09)*, LNCS 5795, pages 670–684. Springer, 2009.

[EFLR99] Andy Evans, Robert France, Kevin Lano, and Bernhard Rumpe. Meta-Modelling Semantics of UML. In H. Kilov, B. Rumpe, and I. Simmonds, editors, *Behavioral Specifications of Businesses and Systems*, pages 45–60. Kluver Academic Publisher, 1999.

[FELR98] Robert France, Andy Evans, Kevin Lano, and Bernhard Rumpe. The UML as a formal modeling notation. *Computer Standards & Interfaces*, 19(7):325–334, November 1998.

[FHR08] Florian Fieber, Michaela Huhn, and Bernhard Rumpe. Modellqualität als Indikator für Softwarequalität: eine Taxonomie. *Informatik-Spektrum*, 31(5):408–424, Oktober 2008.

[FLP+11] M. Norbert Fisch, Markus Look, Claas Pinkernell, Stefan Plesser, and Bernhard Rumpe. State-based Modeling of Buildings and Facilities. In *Enhanced Building Operations Conference (ICEBO'11)*, 2011.

[FPPR12] M. Norbert Fisch, Claas Pinkernell, Stefan Plesser, and Bernhard Rumpe. The Energy Navigator - A Web-Platform for Performance Design and Management. In *Energy Efficiency in Commercial Buildings Conference(IEECB'12)*, 2012.

[GHK+07] Hans Grönniger, Jochen Hartmann, Holger Krahn, Stefan Kriebel, and Bernhard Rumpe. View-based Modeling of Function Nets. In *Object-oriented Modelling of Embedded Real-Time Systems Workshop (OMER4'07)*, 2007.

[GHK+08] Hans Grönniger, Jochen Hartmann, Holger Krahn, Stefan Kriebel, Lutz Rothhardt, and Bernhard Rumpe. Modelling Automotive Function Nets with Views for Features, Variants, and Modes. In *Proceedings of 4th European Congress ERTS - Embedded Real Time Software*, 2008.

[GKPR08] Hans Grönniger, Holger Krahn, Claas Pinkernell, and Bernhard Rumpe. Modeling Variants of Automotive Systems using Views. In *Modellbasierte Entwicklung von eingebetteten Fahrzeugfunktionen*, Informatik Bericht 2008-01, pages 76–89. TU Braunschweig, 2008.

[GKR96] Radu Grosu, Cornel Klein, and Bernhard Rumpe. Enhancing the SysLab System Model with State. Technical Report TUM-I9631, TU Munich, Germany, July 1996.

[GKR+06] Hans Grönniger, Holger Krahn, Bernhard Rumpe, Martin Schindler, and Steven Völkel. MontiCore 1.0 - Ein Framework zur Erstellung und Verarbeitung domänspezifischer Sprachen. Informatik-Bericht 2006-04, CFG-Fakultät, TU Braunschweig, August 2006.

[GKR+07] Hans Grönniger, Holger Krahn, Bernhard Rumpe, Martin Schindler, and Steven Völkel. Textbased Modeling. In *4th International Workshop on Software Language Engineering, Nashville*, Informatik-Bericht 4/2007. Johannes-Gutenberg-Universität Mainz, 2007.

[GKR+08] Hans Grönniger, Holger Krahn, Bernhard Rumpe, Martin Schindler, and Steven Völkel. MontiCore: A Framework for the Development of Textual Domain Specific Languages. In *30th International Conference on Software Engineering (ICSE 2008), Leipzig, Germany, May 10-18, 2008, Companion Volume*, pages 925–926, 2008.

[GKRS06] Hans Grönniger, Holger Krahn, Bernhard Rumpe, and Martin Schindler. Integration von Modellen in einen codebasierten Softwareentwicklungsprozess. In *Modellierung 2006 Conference*, LNI 82, Seiten 67–81, 2006.

[GLRR15] Timo Greifenberg, Markus Look, Sebastian Roidl, and Bernhard Rumpe. Engineering Tagging Languages for DSLs. In *Conference on Model Driven Engineering Languages and Systems (MODELS'15)*, pages 34–43. ACM/IEEE, 2015.

[GR95] Radu Grosu and Bernhard Rumpe. Concurrent Timed Port Automata. Technical Report TUM-I9533, TU Munich, Germany, October 1995.

[GR11] Hans Grönniger and Bernhard Rumpe. Modeling Language Variability. In *Workshop on Modeling, Development and Verification of Adaptive Systems*, LNCS 6662, pages 17–32. Springer, 2011.

[GRJA12] Tim Gülke, Bernhard Rumpe, Martin Jansen, and Joachim Axmann. High-Level Requirements Management and Complexity Costs in Automotive Development Projects: A Problem Statement. In *Requirements Engineering: Foundation for Software Quality (REFSQ'12)*, 2012.

[GRR10] Hans Grönniger, Dirk Reiß, and Bernhard Rumpe. Towards a Semantics of Activity Diagrams with Semantic Variation Points. In *Conference on Model Driven Engineering Languages and Systems (MODELS'10)*, LNCS 6394, pages 331–345. Springer, 2010.

[HHK+13] Arne Haber, Katrin Hölldobler, Carsten Kolassa, Markus Look, Klaus Müller, Bernhard Rumpe, and Ina Schaefer. Engineering Delta Modeling Languages. In *Software Product Line Conference (SPLC'13)*, pages 22–31. ACM, 2013.

[HHK+14] Martin Henze, Lars Hermerschmidt, Daniel Kerpen, Roger Häußling, Bernhard Rumpe, and Klaus Wehrle. User-driven Privacy Enforcement for Cloud-based Services in the Internet of Things. In *Conference on Future Internet of Things and Cloud (FiCloud'14)*. IEEE, 2014.

[HHK+15a] Arne Haber, Katrin Hölldobler, Carsten Kolassa, Markus Look, Klaus Müller, Bernhard Rumpe, Ina Schaefer, and Christoph Schulze. Systematic Synthesis of Delta Modeling Languages. *Journal on Software Tools for Technology Transfer (STTT)*, 17(5):601–626, October 2015.

[HHK+15b] Martin Henze, Lars Hermerschmidt, Daniel Kerpen, Roger Häußling, Bernhard Rumpe, and Klaus Wehrle. A comprehensive approach to privacy in the cloud-based Internet of Things. *Future Generation Computer Systems*, 56:701–718, 2015.

[HKM+13] Arne Haber, Carsten Kolassa, Peter Manhart, Pedram Mir Seyed Nazari, Bernhard Rumpe, and Ina Schaefer. First-Class Variability Modeling in Matlab/Simulink. In *Variability Modelling of Software-intensive Systems Workshop (VaMoS'13)*, pages 11–18. ACM, 2013.

[HKR+07] Christoph Herrmann, Holger Krahn, Bernhard Rumpe, Martin Schindler, and Steven Völkel. An Algebraic View on the Semantics of Model Composition. In *Conference on Model Driven Architecture - Foundations and Applications (ECMDA-FA'07)*, LNCS 4530, pages 99–113. Springer, Germany, 2007.

[HKR+09] Christoph Herrmann, Holger Krahn, Bernhard Rumpe, Martin Schindler, and Steven Völkel. Scaling-Up Model-Based-Development for Large Heterogeneous Systems with Compositional Modeling. In *Conference on Software Engineeering in Research and Practice (SERP'09)*, pages 172–176, July 2009.

[HKR+11] Arne Haber, Thomas Kutz, Holger Rendel, Bernhard Rumpe, and Ina Schaefer. Delta-oriented Architectural Variability Using MontiCore. In *Software Architecture Conference (ECSA'11)*, pages 6:1–6:10. ACM, 2011.

[HKR12] Christoph Herrmann, Thomas Kurpick, and Bernhard Rumpe. SSELab: A Plug-In-Based Framework for Web-Based Project Portals. In *Developing Tools as Plug-Ins Workshop (TOPI'12)*, pages 61–66. IEEE, 2012.

[HR04] David Harel and Bernhard Rumpe. Meaningful Modeling: What's the Semantics of "Semantics"? *IEEE Computer*, 37(10):64–72, October 2004.

[HRR98] Franz Huber, Andreas Rausch, and Bernhard Rumpe. Modeling Dynamic Component Interfaces. In *Technology of Object-Oriented Languages and Systems (TOOLS 26)*, pages 58–70. IEEE, 1998.

[HRR+11] Arne Haber, Holger Rendel, Bernhard Rumpe, Ina Schaefer, and Frank van der Linden. Hierarchical Variability Modeling for Software Architectures. In *Software Product Lines Conference (SPLC'11)*, pages 150–159. IEEE, 2011.

[HRR12] Arne Haber, Jan Oliver Ringert, and Bernhard Rumpe. MontiArc - Architectural Modeling of Interactive Distributed and Cyber-Physical Systems. Technical Report AIB-2012-03, RWTH Aachen University, February 2012.

[HRRS11] Arne Haber, Holger Rendel, Bernhard Rumpe, and Ina Schaefer. Delta Modeling for Software Architectures. In *Tagungsband des Dagstuhl-Workshop MBEES: Modellbasierte Entwicklung eingebetteterSysteme VII*, pages 1 – 10. fortiss GmbH, 2011.

[HRRS12] Arne Haber, Holger Rendel, Bernhard Rumpe, and Ina Schaefer. Evolving Delta-oriented Software Product Line Architectures. In *Large-Scale Complex IT Systems. Development, Operation and Management, 17th Monterey Workshop 2012*, LNCS 7539, pages 183–208. Springer, 2012.

[HRRW12] Christian Hopp, Holger Rendel, Bernhard Rumpe, and Fabian Wolf. Einführung eines Produktlinienansatzes in die automotive Softwareentwicklung am Beispiel von Steuergerätesoftware. In *Software Engineering Conference (SE'12)*, LNI 198, Seiten 181–192, 2012.

[HRW15] Katrin Hölldobler, Bernhard Rumpe, and Ingo Weisemöller. Systematically Deriving Domain-Specific Transformation Languages. In *Conference on Model Driven Engineering Languages and Systems (MODELS'15)*, pages 136–145. ACM/IEEE, 2015.

[KER99] Stuart Kent, Andy Evans, and Bernhard Rumpe. UML Semantics FAQ. In A. Moreira and S. Demeyer, editors, *Object-Oriented Technology, ECOOP'99 Workshop Reader*, LNCS 1743, Berlin, 1999. Springer Verlag.

[KKP+09] Gabor Karsai, Holger Krahn, Claas Pinkernell, Bernhard Rumpe, Martin Schindler, and Steven Völkel. Design Guidelines for Domain Specific Languages. In *Domain-Specific Modeling Workshop (DSM'09)*, Techreport B-108, pages 7–13. Helsinki School of Economics, October 2009.

[KLPR12] Thomas Kurpick, Markus Look, Claas Pinkernell, and Bernhard Rumpe. Modeling Cyber-Physical Systems: Model-Driven Specification of Energy Efficient Buildings. In *Modelling of the Physical World Workshop (MOTPW'12)*, pages 2:1–2:6. ACM, October 2012.

[KPR97] Cornel Klein, Christian Prehofer, and Bernhard Rumpe. Feature Specification and Refinement with State Transition Diagrams. In *Workshop on Feature Interactions in Telecommunications Networks and Distributed Systems*, pages 284–297. IOS-Press, 1997.

[KPR12] Thomas Kurpick, Claas Pinkernell, and Bernhard Rumpe. Der Energie Navigator. In H. Lichter and B. Rumpe, Editoren, *Entwicklung und Evolution von Forschungssoftware. Tagungsband, Rolduc, 10.-11.11.2011*, Aachener Informatik-Berichte, Software Engineering, Band 14. Shaker Verlag, Aachen, Deutschland, 2012.

[Kra10] Holger Krahn. *MontiCore: Agile Entwicklung von domänenspezifischen Sprachen im Software-Engineering*. Aachener Informatik-Berichte, Software Engineering, Band 1. Shaker Verlag, März 2010.

[KRB96] Cornel Klein, Bernhard Rumpe, and Manfred Broy. A stream-based mathematical model for distributed information processing systems - SysLab system model. In *Workshop on Formal Methods for Open Object-based Distributed Systems*, IFIP Advances in Information and Communication Technology, pages 323–338. Chapmann & Hall, 1996.

[KRR14] Helmut Krcmar, Ralf Reussner, and Bernhard Rumpe. *Trusted Cloud Computing*. Springer, Schweiz, December 2014.

[KRS12] Stefan Kowalewski, Bernhard Rumpe, and Andre Stollenwerk. Cyber-Physical Systems - eine Herausforderung für die Automatisierungstechnik? In *Proceedings of Automation 2012, VDI Berichte 2012*, Seiten 113–116. VDI Verlag, 2012.

[KRV06] Holger Krahn, Bernhard Rumpe, and Steven Völkel. Roles in Software Development using Domain Specific Modelling Languages. In *Domain-Specific Modeling Workshop (DSM'06)*, Technical Report TR-37, pages 150–158. Jyväskylä University, Finland, 2006.

[KRV07a] Holger Krahn, Bernhard Rumpe, and Steven Völkel. Efficient Editor Generation for Compositional DSLs in Eclipse. In *Domain-Specific Modeling Workshop (DSM'07)*, Technical Reports TR-38. Jyväskylä University, Finland, 2007.

[KRV07b] Holger Krahn, Bernhard Rumpe, and Steven Völkel. Integrated Definition of Abstract and Concrete Syntax for Textual Languages. In *Conference on Model Driven Engineering Languages and Systems (MODELS'07)*, LNCS 4735, pages 286–300. Springer, 2007.

[KRV08] Holger Krahn, Bernhard Rumpe, and Steven Völkel. Monticore: Modular Development of Textual Domain Specific Languages. In *Conference on Objects, Models, Components, Patterns (TOOLS-Europe'08)*, LNBIP 11, pages 297–315. Springer, 2008.

[KRV10] Holger Krahn, Bernhard Rumpe, and Stefen Völkel. MontiCore: a Framework for Compositional Development of Domain Specific Languages. *International Journal on Software Tools for Technology Transfer (STTT)*, 12(5):353–372, September 2010.

[LRSS10] Tihamer Levendovszky, Bernhard Rumpe, Bernhard Schätz, and Jonathan Sprinkle. Model Evolution and Management. In *Model-Based Engineering of Embedded Real-Time Systems Workshop (MBEERTS'10)*, LNCS 6100, pages 241–270. Springer, 2010.

[MMR10] Tom Mens, Jeff Magee, and Bernhard Rumpe. Evolving Software Architecture Descriptions of Critical Systems. *IEEE Computer*, 43(5):42–48, May 2010.

[MRR10] Shahar Maoz, Jan Oliver Ringert, and Bernhard Rumpe. A Manifesto for Semantic Model Differencing. In *Proceedings Int. Workshop on Models and Evolution (ME'10)*, LNCS 6627, pages 194–203. Springer, 2010.

[MRR11a] Shahar Maoz, Jan Oliver Ringert, and Bernhard Rumpe. ADDiff: Semantic Differencing for Activity Diagrams. In *Conference on Foundations of Software Engineering (ESEC/FSE '11)*, pages 179–189. ACM, 2011.

[MRR11b] Shahar Maoz, Jan Oliver Ringert, and Bernhard Rumpe. An Operational Semantics for Activity Diagrams using SMV. Technical Report AIB-2011-07, RWTH Aachen University, Aachen, Germany, July 2011.

[MRR11c] Shahar Maoz, Jan Oliver Ringert, and Bernhard Rumpe. CD2Alloy: Class Diagrams Analysis Using Alloy Revisited. In *Conference on Model Driven Engineering Languages and Systems (MODELS'11)*, LNCS 6981, pages 592–607. Springer, 2011.

[MRR11d] Shahar Maoz, Jan Oliver Ringert, and Bernhard Rumpe. Modal Object Diagrams. In *Object-Oriented Programming Conference (ECOOP'11)*, LNCS 6813, pages 281–305. Springer, 2011.

[MRR11e] Shahar Maoz, Jan Oliver Ringert, and Bernhard Rumpe. Semantically Configurable Consistency Analysis for Class and Object Diagrams. In *Conference on Model Driven Engineering Languages and Systems (MODELS'11)*, LNCS 6981, pages 153–167. Springer, 2011.

[MRR13] Shahar Maoz, Jan Oliver Ringert, and Bernhard Rumpe. Synthesis of Component and Connector Models from Crosscutting Structural Views. In Meyer, B. and Baresi, L. and Mezini, M., editor, *Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE'13)*, pages 444–454. ACM New York, 2013.

[MRR14] Shahar Maoz, Jan Oliver Ringert, and Bernhard Rumpe. Verifying Component and Connector Models against Crosscutting Structural Views. In *Software Engineering Conference (ICSE'14)*, pages 95–105. ACM, 2014.

[NPR13] Antonio Navarro Pérez and Bernhard Rumpe. Modeling Cloud Architectures as Interactive Systems. In *Model-Driven Engineering for High Performance and Cloud Computing Workshop*, CEUR Workshop Proceedings 1118, pages 15–24, 2013.

[PFR02] Wolfgang Pree, Marcus Fontoura, and Bernhard Rumpe. Product Line Annotations with UML-F. In *Software Product Lines Conference (SPLC'02)*, LNCS 2379, pages 188–197. Springer, 2002.

[PR94] Barbara Paech and Bernhard Rumpe. A new Concept of Refinement used for Behaviour Modelling with Automata. In *Proceedings of the Industrial Benefit of Formal Methods (FME'94)*, LNCS 873, pages 154–174. Springer, 1994.

[PR99] Jan Philipps and Bernhard Rumpe. Refinement of Pipe-and-Filter Architectures. In *Congress on Formal Methods in the Development of Computing System (FM'99)*, LNCS 1708, pages 96–115. Springer, 1999.

[PR01] Jan Philipps and Bernhard Rumpe. Roots of Refactoring. In Kilov, H. and Baclavski, K., editor, *Tenth OOPSLA Workshop on Behavioral Semantics. Tampa Bay, Florida, USA, October 15*. Northeastern University, 2001.

[PR03] Jan Philipps and Bernhard Rumpe. Refactoring of Programs and Specifications. In Kilov, H. and Baclavski, K., editor, *Practical Foundations of Business and System Specifications*, pages 281–297. Kluwer Academic Publishers, 2003.

[Rin14] Jan Oliver Ringert. *Analysis and Synthesis of Interactive Component and Connector Systems*. Aachener Informatik-Berichte, Software Engineering, Band 19. Shaker Verlag, 2014.

[RK96] Bernhard Rumpe and Cornel Klein. Automata Describing Object Behavior. In B. Harvey and H. Kilov, editors, *Object-Oriented Behavioral Specifications*, pages 265–286. Kluwer Academic Publishers, 1996.

[RKB95] Bernhard Rumpe, Cornel Klein, and Manfred Broy. Ein strombasiertes mathematisches Modell verteilter informationsverarbeitender Systeme - Syslab-Systemmodell. Technischer Bericht TUM-I9510, TU München, Deutschland, März 1995.

[RRRW15] Jan Oliver Ringert, Alexander Roth, Bernhard Rumpe, and Andreas Wortmann. Language and Code Generator Composition for Model-Driven Engineering of Robotics Component & Connector Systems. *Journal of Software Engineering for Robotics (JOSER)*, 6(1):33–57, 2015.

[RRW13a] Jan Oliver Ringert, Bernhard Rumpe, and Andreas Wortmann. From Software Architecture Structure and Behavior Modeling to Implementations of Cyber-Physical Systems. In *Software Engineering Workshopband (SE'13)*, LNI 215, pages 155–170, 2013.

[RRW13b] Jan Oliver Ringert, Bernhard Rumpe, and Andreas Wortmann. MontiArcAutomaton: Modeling Architecture and Behavior of Robotic Systems. In *Conference on Robotics and Automation (ICRA'13)*, pages 10–12. IEEE, 2013.

[RRW14] Jan Oliver Ringert, Bernhard Rumpe, and Andreas Wortmann. *Architecture and Behavior Modeling of Cyber-Physical Systems with MontiArcAutomaton*. Aachener Informatik-Berichte, Software Engineering, Band 20. Shaker Verlag, December 2014.

[RSW+15] Bernhard Rumpe, Christoph Schulze, Michael von Wenckstern, Jan Oliver Ringert, and Peter Manhart. Behavioral Compatibility of Simulink Models for Product Line Maintenance and Evolution. In *Software Product Line Conference (SPLC'15)*, pages 141–150. ACM, 2015.

[Rum96] Bernhard Rumpe. *Formale Methodik des Entwurfs verteilter objektorientierter Systeme*. Herbert Utz Verlag Wissenschaft, München, Deutschland, 1996.

[Rum02] Bernhard Rumpe. Executable Modeling with UML - A Vision or a Nightmare? In T. Clark and J. Warmer, editors, *Issues & Trends of Information Technology Management in Contemporary Associations, Seattle*, pages 697–701. Idea Group Publishing, London, 2002.

[Rum03] Bernhard Rumpe. Model-Based Testing of Object-Oriented Systems. In *Symposium on Formal Methods for Components and Objects (FMCO'02)*, LNCS 2852, pages 380–402. Springer, November 2003.

[Rum04] Bernhard Rumpe. Agile Modeling with the UML. In *Workshop on Radical Innovations of Software and Systems Engineering in the Future (RISSEF'02)*, LNCS 2941, pages 297–309. Springer, October 2004.

[Rum11] Bernhard Rumpe. *Modellierung mit UML, 2te Auflage*. Springer Berlin, September 2011.

[Rum12] Bernhard Rumpe. *Agile Modellierung mit UML: Codegenerierung, Testfälle, Refactoring, 2te Auflage*. Springer Berlin, Juni 2012.

[Rum16] Bernhard Rumpe. *Modeling with UML: Language, Concepts, Methods*. Springer International, July 2016.

[Sch12] Martin Schindler. *Eine Werkzeuginfrastruktur zur agilen Entwicklung mit der UML/P*. Aachener Informatik-Berichte, Software Engineering, Band 11. Shaker Verlag, 2012.

[SRVK10] Jonathan Sprinkle, Bernhard Rumpe, Hans Vangheluwe, and Gabor Karsai. Metamodelling: State of the Art and Research Challenges. In *Model-Based Engineering of Embedded Real-Time Systems Workshop (MBEERTS'10)*, LNCS 6100, pages 57–76. Springer, 2010.

[THR+13] Ulrike Thomas, Gerd Hirzinger, Bernhard Rumpe, Christoph Schulze, and Andreas Wortmann. A New Skill Based Robot Programming Language Using UML/P Statecharts. In *Conference on Robotics and Automation (ICRA'13)*, pages 461–466. IEEE, 2013.

[Völ11] Steven Völkel. *Kompositionale Entwicklung domänenspezifischer Sprachen*. Aachener Informatik-Berichte, Software Engineering, Band 9. Shaker Verlag, 2011.

[Wei12] Ingo Weisemöller. *Generierung domänenspezifischer Transformationssprachen*. Aachener Informatik-Berichte, Software Engineering, Band 12. Shaker Verlag, 2012.

[ZPK+11] Massimiliano Zanin, David Perez, Dimitrios S Kolovos, Richard F Paige, Kumardev Chatterjee, Andreas Horst, and Bernhard Rumpe. On Demand Data Analysis and Filtering for Inaccurate Flight Trajectories. In *Proceedings of the SESAR Innovation Days*. EUROCONTROL, 2011.