Classification and Definition of an Enterprise Architecture Analyses Language

Julia Rauscher[1], Melanie Langermeier[1], and Bernhard Bauer[1]

University Augsburg, Augsburg, Germany,
rauscher@ds-lab.org

Abstract. Enterprise Architecture Management (EAM) deals with the assessment and development of business processes and IT components. Through the analysis of as-is and to-be states the information flow in organizations is optimized. Thus EAM analyses are an essential part in the EAM cycle. To cover the needs of an architect the analyses pursue different goals and utilize different techniques. In this work we examine the different EA analysis approaches according to their characteristics and requirements. For that purpose we design a generic analysis language which can be used for their description. In order to manage the numerous approaches from literature we develop a categorization. The categories are created based on the goals, constructs and kind of results. We propose a two-dimensional classification into functional and technical categories. The goal is to provide a common description for EA analyses for an easy access to their goals and execution requirements.

Key words: Enterprise Architecture Management, Analysis, EA Analysis, Categorization, Characteristics, Requirements, Domain Specific Language, Meta Language

1 INTRODUCTION

Analyses are one of the most important artifacts integrated in Enterprise Architecture Management (EAM) and are indispensable in the EAM cycle. The EA process contains five phases [Niemann, 2006]: Document, Analyze, Plan, Act and Check. Thus, analysis is an essential part in order to create and implement future plans. It supports decision making through an evaluation of the current architecture as well as potential future scenarios [Sasa and Krisper, 2011]. The result of analysis and planning actions is finally the creation of a target architecture. Those actions enable planning, acting, controlling and documenting through all layers.

The creation of an Enterprise Architecture (EA) model is time and cost consuming. Therefore, support for decision making and planning generates value and increases the acceptance of the EA initiative in an organization [Lankhorst, 2013]. Thus, analysis support is essential in order to generate value from an EA model. The execution of an analysis decomposes the analyzed object in its components. Those single elements are examined and evaluated as well as the relationships and interactions between them. Applying existing analyses on established models is expensive, since the corresponding meta models typically require some adaptions [Langermeier et al., 2014]. This makes reuse of existing solutions and research findings hard.

In current literature a great variety of analysis possibilities are described. In previous work we provided an overview of the different analysis approaches [Rauscher, 2013]. These approaches are mainly isolated, integrated ones are rare. The well-known EA frameworks deal only secondary with EA analysis, a common understanding of the term is not established yet [Buckl et al., 2009b, Närman et al., 2012]. Nevertheless due to the importance of EA analyses, methods are required to specify them consistently [Buckl et al., 2009b, Johnson et al., 2007b].

The analyses rely on different technologies, like ontologies [Sunkle et al., 2013], probability networks [Närman et al., 2008] or expert interviews [Kazienko et al., 2011], have different preconditions and provide different kinds of results. E.g. preconditions can be required properties for model elements or specific data structures. Typical results are quantitative ones like an overall metric for the architecture, measures for specific architecture elements, but also a determined set of elements. Several analysis approaches focus on the dependencies between the elements of an architecture. For example, they are used to determine the impact of changes (e.g. [de Boer et al., 2005]). Other analyses focus on specific attributes of elements. For instance an availability analysis predicts the availability value for an element, dependent on several other factors modeled in the EA [Närman et al., 2012]. Accordingly metrics based on attribute values can be calculated. They can be used as key performance indicator for the evaluation of the architecture and for decision making [Matthes et al., 2011]. In those calculations also the relationships between the elements as well as the attributes of dependent elements can be considered. The analysis of timing aspects is very important for EAM. Therewith the evolution of the architecture can be monitored [Matthes et al., 2011]. Every analysis supports a different goal and thus, for a sound evaluation of the architecture different kinds of analyses are required.

In this paper we want to analyze existing EA analyses from literature to determine their characteristics and their requirements for execution. Additionally we want to provide a uniform description technique to specify the different requirements, goals and outputs of the analyses. Due to the high number of analyses in literature we first categorize them and determine the requirements per category. The main goal of the categorization is to create a possibility to conduct analyses organized and controlled. Additionally the categorization enables the creation of new analyses and the selection of the best suited analysis depending on the goal and requested technique. Therefore we study the single analysis approaches regarding their requirements for execution and their provided results. This provides a sound overview of analysis approaches in the context of EA and of the issues they address. Based on the results we categorize them once according to their functional dimension, and once according to their technical dimension. The characteristics of the resulting categories are formalized while establishing a Domain Specific Language (DSL). Such a language allows us to make the requirements of an analysis visible in a structural way. The calculated outcome as well as the preconditions in order to execute the analysis is easily accessible.

The remaining paper is structured as followed. First we introduce foundations of EA analysis (section 2). Following we present in section 3 our approach for determining the analysis categories. The categories themselves are also presented shortly with their main characteristics. The DSL to describe the analyses is introduced in section 4. Its application is shown exemplary for one category. Finally we discuss the categorization and the DSL (section 5).

2 ENTERPRISE ARCHITECTURE ANALYSIS

Architectures are used to describe the elements of a system and the relationships between them. The term also comprises the process of creation and maintenance, the architecture work [Lankhorst, 2013]. EA focuses on elements like business processes, applications and the technical infrastructure of an organization. Often used layers are the strategic layer to represent the organization's strategy with its goals, the business layer describing the business processes and products, the information layer with the information objects, and the application layer as well as the infrastructure layer describing the soft- and hardware components [Lankhorst, 2013, Winter and Fischer, 2006]. Despite the examination of different layers the focus of an EA are the dependencies between layers, i.e. how business and IT relate to each other. Layers are dependent according to the Align-Enable-Principle. The lower layers are the foundation for the upper ones, and the upper ones adjust the lower ones [Winter and Fischer, 2006, Krcmar, 2015].

The main reasons for analyses are receiving an overview of the architecture, its components and connections, and examining the as-is state [Langermeier et al., 2014]. Furthermore weak points can be revealed, new advantages be discovered and various design alternatives be evaluated [Zia et al., 2011]. Results of analysis activities are the development of a to-be architecture as well as an improved decision making. The focus of every analysis depends on the analysis type. Additional the questions of what is feasible and what is desirable are crucial [Johnson et al., 2007a]. The process of analysis can be segmented in different phases and activities [Wan and Carlsson, 2012]. We used the parts "system thinking", "modeling", "measuring", "satisfying", "comparing with requirements" and "comparing alternatives" in this work to identify the characteristics of analysis categories.

As basis for our work we conducted a detailed literature research [Rauscher, 2013, Rauscher, 2015]. We exclusively chose analyses, which purely analyze EA and are not transferred from other topics. Hereof a pure EA analysis has the focus on collecting data and discovering the current state of an enterprise architecture in a quantitative or functional way to create a summary, alter the state or control different aspects. The goal of this selection was to create an overview of current EAM analyses and to receive approaches utilizable for a categorization (e.g. Della Bordella et al., 2011; Johnson et al., 2007a; Razavi et al., 2011). We identified 105 EAM analyses which are roughly grouped into 40 EA analysis types in previous work [Rauscher, 2013]. An analysis type describes analyses, which have the same rough scope and are built independently from the realization method. The goals of the contained analyses can differ significantly. Examples of types are 'Quality Analysis' (e.g. [Närman et al., 2008]), 'Requirements Analysis' (e.g. [Aier et al., 2009]) and 'Analysis of Costs' (e.g.[Niemann, 2006]). The different types of analyses, which have been discovered in the literature research, can be treated as a first categorization. However this categorization only makes raw statements about the rough purpose of the contained analyses. Although analyses of the

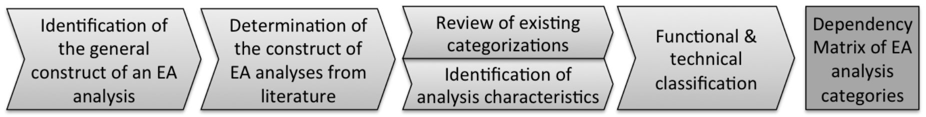

Identification of the general construct of an EA analysis
Determination of the construct of EA analyses from literature
Review of existing categorizations
Identification of analysis characteristics
Functional & technical classification
Dependency Matrix of EA analysis categories

Fig. 1: Categorization approach

same analysis type have the same field of interest their individual goals and implementations can differ. Thus the classification in those types is not detailed enough to derive characteristics. Quality analyses, for example, can be conducted in various ways and can target different goals. For instance, this can be the quality of a whole system or maturity quality of a single artifact.

3 CATEGORIZATION

The huge amount of different approaches clarifies importance of EA analyses and coherence of a successful architecture. To ease analysis activities and to get an understanding about current work we categorized them according to their characteristics. We define characteristics of an EA analyses as all necessary steps and components of an analysis to accomplish its goal. The characteristics are a main part of the categorization because they are guaranteeing the accuracy of the conducted analysis and the achievement of the goal. Figure 1 gives an overview of the categorization approach which is described in detail in section 3.1. The resulting categories are presented in sections 3.3 and 3.4.

3.1 Categorization Approach

Preliminary work for the categorization includes the definition of a general understanding of an analysis to determine a general purpose construct. We could identify three main constructs of an EA analysis, which can be used as foundation for the categorization and determination of characteristics. These are data intake, processing and outcome. Other parts vary per analyses. Based on these parts we determine the meaning and boundaries of a category: Analyses can be merged to a category if they coincide at least in one of the three parts. As optimal condition all parts are equal, but this is usually not given. Therefore we classify different analyses to the same category if they have at least the same target or same processing technique.

After defining our basis we conduct literature research of current EA analysis (procedure see section 2). We determined the construct for each of the identified EA analysis approaches. Thereby we ensured that only papers with a high elaboration level are used. Because of missing details it is not possible to analyze rough approaches and to identify their construct and characteristics. For elaborated analyses with less detailed parts, we made necessary assumptions. In the case that an EA analysis approach is realized using another non-EA related analysis approach, this non-EA analysis is included too. This proceeding ensures the construction of a data basis with categorize-able analyses according to the general construct of intake, processing and outcome.

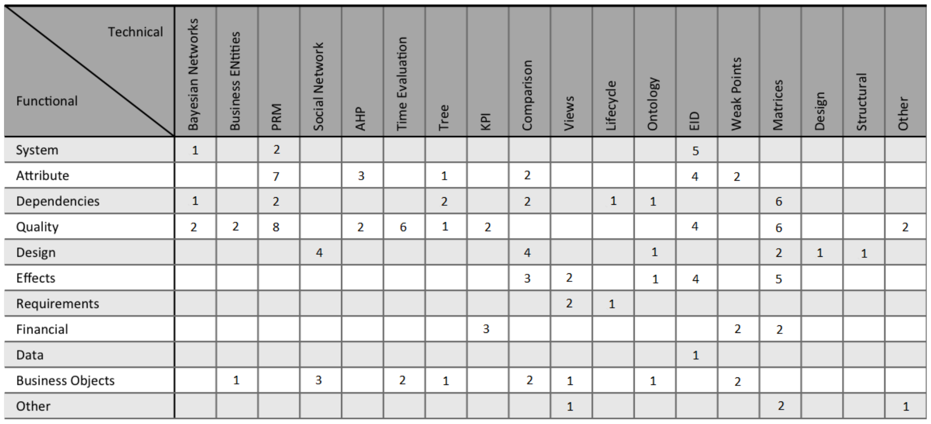
Based on the experiences made while identifying the construct of the EA analysis we were able to refine our categorization approach. Considering different existing kinds of categorization [Lankhorst, 2013, Buckl et al., 2009b] we conducted our approach with two main fields of categories: functional and technical. This decision brought the most advantages in comparison with other approaches because of the division in "How" (technical aspects) and "Why" (functional purpose). The additional distinction in architecture levels is not included in our approach because a plain allocation wouldn't be possible. Most analysis can be conducted in many levels or can only be performed by involving several levels. Through the new and detailed knowledge from the first evaluation of EA analysis constructs we introduce characteristics to ensure accuracy. We used the analysis' properties and steps as characteristics in order to retrieve detailed information about the category of each analysis approach.

After this step the final categorization of the analyses was received based on our main idea of distinction between functional and technical. The business functions of every analysis are determined based on the concepts *purpose dependent division* ("Fundamental", "Main" and "Decision-oriented") and *activity dependent division* ("System thinking", "Modeling", "Measuring", "Satisfying", "Comparing with requirements" and "Comparing alternatives") [Wan and Carlsson, 2012]. We used these concepts to analyze the identified analysis approaches according to their goals and activities. Thereby the functional categories have been determined by using a prepared template of aspects. This template consists of the analysis activities, the intermediate objectives and the main goal. After analyzing all approaches we identified 10 categories from classifying the various analyses goals. Attention should be paid to the fact of multiple classifications. For example, a security analysis is able to analyze dependencies and requirements and therefore can be assigned to both functional categories.

After we completed the functional classification, we conducted the technical categories. This procedure was more detailed and complex because of the large variety of existing methods in EA analysis. Only analyses with the same method and same steps of goal attainment can form a technical category. This constraint is necessary to enable discovery of shared characteristics. The already mentioned template was altered for creating technical categories. The new focus lies on the constructs, methods, techniques (including single steps) and artifacts. First, rough technical categories have been determined based on the dimensions "quantitative", "analytic", "simulation" and "functional" [Lankhorst, 2013]. After this preliminary stage detailed categories were created. Each identified analysis approach passed through this procedure. In contrast to functional categories every analysis was assigned to one specific technical category. As final result we concluded with 17 technical categories.

We decided to choose a two-dimensional classification and not a more detailed fragmentation, because of the high amount of differences within the approaches. Every analysis has special characteristics when sharing the same technique. Therefore it was not possible to identify categories or classifications on a lower level of abstraction. However a high abstraction level involves the danger of missing necessary details. To involve all aspects of every analysis we observed every analysis in its functional and technical view.

Altogether 105 analysis approaches fulfilled our criteria and have been incorporated. Only nine of them couldn't be classified. These ones were too specific and individual for the mapping to a category. For each analysis we identified exactly one technical category and at least one functional category. To create an overview of all possible combinations of categories three matrices were created. Two matrices represent the combination of the analysis approaches and the categories. The functional matrix has more possible combinations because analyses can achieve more targets simultaneously. However the technical matrix has only one combination per analysis. For example, Närman et al. (2008) is assigned to the functional categories **System Analyses**, **Attribute Analyses** and **Quality Analyses** and to the technical one **Bayesian Networks**. To provide an overview of the functional and technical combinations both matrices have been joint, which resulted in a shared matrix (see figure 2). Thus, we get an overview of the realization techniques of a functional category and also the other way round for which analysis goals a technique is used. The numerical values in the table represent the amount of analyses in current literature, which match to both categories, the functional and the technical one. However the sum of the values is more then 96 because of the fact that some analyses have multiple functional categories.



Technical / Functional	Bayesian Networks	Business ENtities	PRM	Social Network	AHP	Time Evaluation	Tree	KPI	Comparison	Views	Lifecycle	Ontology	EID	Weak Points	Matrices	Design	Structural	Other
System	1		2										5					
Attribute			7		3		1		2				4	2				
Dependencies	1		2				2		2		1	1			6			
Quality	2	2	8		2	6	1	2					4		6			2
Design				4					4			1			2	1	1	
Effects									3	2		1	4		5			
Requirements										2	1							
Financial								3						2	2			
Data													1					
Business Objects		1		3		2	1		2	1		1		2				
Other										1					2			1

Fig. 2: Dependency matrix of the functional and technical categories

3.2 Identification of characteristics

As only properties and steps can show the components responsible for classification, we introduce characteristics to ensure accuracy. We define a characteristic as requirement, since an analysis can only be conducted target-aimed with all indispensable artifacts. Requirements support the achievement of goals and are used to identify hidden characteristics [Van Lamsweerde, 2001]. Whereas properties can differ significantly, on some spots we had to choose the most elaborated or create a higher abstraction level. There are two types of characteristics: category specific ones and general characteristics. The second type includes a meta model and scenarios, determined at the beginning of an analysis. Another universal characteristic is the main goal. These three characteristics have to be conducted for all analyses. Together they provide a high level of abstraction.

For the specific characteristics we distinguished five different kinds. The conducted kinds of characteristics are important for the identification of properties from technical analyses. Whereas functional categories have rough properties, technical categories have similar structure. We identified the following kind of characteristics: *Input*, *Conditions*, *Construct*, *Measurement*, and *Output*. The basis of an analysis is always represented in terms of *Input* data. In every case an architecture or scenarios, in form of an model, are needed to conduct the following steps and final measurements. Before the main part of an analysis can be performed, sometimes *Conditions* are needed. For example the possibility of succeeding must be given. Most of the *Conditions* are analysis independent and therefore can be seen as generally valid. The main part and procedure of an analysis is the *Construct*, containing all details of the procedure. It's required to conduct all details successfully to be able to finish the analysis. Examples are detailed steps, mathematical algorithms, relationship types and weighting of artifacts. To prove and measure the results and its calculation, every analysis needs some kind of *Measurement*. This characteristic is responsible to witness the achievement of goals. Mainly a *Measurement* is conducted using scales, KPIs and metrics to control functional and non-functional goals [Davis, 1989]. This characteristic can vary dependent on the analysis and its goals. As last characteristic kind the *Output* was identified. It includes the way of presentation and type of outcome such as percentage, graphics or matrices. This category is crucial because analyses within the same category should have the same *Output*. We used these characteristic kinds to analyze the approaches again to receive detailed information. Through the new and detailed knowledge we had to rearrange the analysis categories on necessary points. New identified characteristics have been verified on correctness and necessity. After this step the final categorization of the analyses was received. Every analysis has multiple and complex requirements. Therefore a table including all characteristics of a category was not feasible, because of the amount of details. To provide an insight of this amount of different characteristics we present short segments of categories and their requirements.

3.3 Functional categorization

In the following we present the categories of the functional classification. Therefore we will list them combined with an example of an assigned analysis approach. Additional a short description of the characteristics is be given. The complete assignment of all identified analyses to the categories can be found in [Rauscher, 2015]. However only a few characteristics are given and it does not present the whole amount of identified requirements in detail.

System Analyses (e.g. [Närman et al., 2008]) check partial or holistic systems and encompass the analysis types 'Quality Analysis' and 'EID'. Mostly time quality aspects and their optimization are in the main focus. Analyses that are contained in this category are often also part of other functional categories because of possible sub-goals. Examples are an analysis of single quality attributes without considering other parts of a system or an analysis determining a possible impact. Analyses in this functional category have very different realization approaches, thus various different techniques are utilized. Possible techniques are PRM or EID (see section 3.4 for further details).

For instance, specific attributes and their values are analyzed by **Attribute Analyses** (e.g. [Razavi et al., 2011]). Ten analysis types are joint in this category. The observation and management of attributes is the focus such approaches. For instance the different states of attributes with changing input can be analyzed or the availability of attributes can be observed. This category contains many approaches, because of the high demand of attributes in EAM. Following there exist numerous different field of applications as wells different realization techniques. But the focus lies always on attributes. Through the various fields of application, most of contained analyses are also a part of another functional category.

Analyses which prove the relations between the elements are classified as **Dependencies Analyses** (e.g. [Saat, 2010]). The main goal of these approaches is the identification of dependencies in EAs and relations of single components to receive an understanding of the whole architecture. Therewith critical relations are identified and observed. Additionally a risk analysis addresses also financial aspects beside the relations. The methods range from comparison of scenarios to a weak point analysis of the relations.

Quality Analyses have the main focus on various quality questions regarding attributes, systems, architectures and other components and target subjective and measurable goals (e.g. [Närman et al., 2008]). Altogether 13 different analysis approaches target quality issues. This category is based on ISO 9126 standard of software quality metrics and analyzes maintainability, maturity, usability, accuracy, security, efficiency and interoperability. Based on the high variety of contained analyses types, also the possible techniques differ. For instance, PRM and EID can be used to analyze service quality. In most cases this category tries to observe subjective quality through comparisons of alternatives or the usage of metrics.

Another category represents the analysis of architecture design (**Design Analyses**), examples are [Aier et al., 2011, Kazienko et al., 2011]. Through receiving an overview of the architecture construct all design variants can be identified. Beside the analysis of holistic or partial design, business entities, procedures and components can be analyzed and used to optimize the architecture. An example for the concentration on a single part is the analysis of interfaces. Without this analysis a holistic overview of the architecture would not be possible. Therefore it is indispensable for EAM. Analysis in this category utilize specific techniques with rare reuse like social network analysis.

All approaches which control impacts in architectures and actions are joint in **Effect Analyses** (e.g. [de Boer et al., 2005]). This includes 'Gap Analysis' and 'Sensitivity Analysis'. In contrast to dependencies analyses these approaches observe the direct impact and effects of changes in architecture elements. To conduct the effect the change of an artifact is simulated in the model. These artifacts can be data, attributes and quality features. The simulation is done identifying different perspectives and comparisons or using the method of extended influence diagrams.

Requirements Analyses identify the requirements to achieve states or goals (e.g. [Aier et al., 2009]). Therefore the specific conditions have to be determined. Results are either specified values or features and specific business entities, like operations. The main goal of this category is to identify all requirements of an enterprise architecture. Examples are 'Security Analysis' and 'Survival Analysis'. If requirements are not analyzed, processes can not be aligned optimal and goals are not achievable. It's possible to additionally analyze a life cycle and its changing requirements.

To identify costs and benefits **Financial Analyses** are used (e.g. [Niemann, 2006]). On top financial weak points and possible impacts can be discovered. These analyses present a measurement with mathematical calculations. Therefore key figures and metrics determine the outcome. However receiving affected entities is a side effect of the result. Consequently financial analyses observe too high costs or uncertainty and hence are an indicator of necessary architecture and procedure changes. While costs and benefits are only calculated in this analyses, weak points and risks can also be identified for example with dependencies analyses. Financial analyses evaluate the economical success through assessing the costs and benefits architecture and trigger actions to improve them.

However **Data Analyses** cover all kinds of data (e.g. [Närman et al., 2009]). The focus lies on quality and accuracy, because data is a critical factor in enterprises. This category has only one technical category, EID. Thereby the data values are analyzed for evaluation purposes. **Data Analyses** target mainly the data quality because the accuracy of data is fundamental for all EA operations.

Finally the category **Business Object Analyses** was identified. Approaches like [Della Bordella et al., 2011] are included. Business objects of every kind, e.g. operations, artifacts and entities, which are part of the architecture are addressed here. This category analyzes single business artifacts and whole operations. Example for measurement procedures are the evaluation of time and therewith an optimal operation or the creation of views. Next to 'Business Process Support Analysis' and 'Business Entity Analysis' also 'Social Network Analysis' belong to this category.

3.4 Technical categories

In the following we describe the 17 technical categories. Therefore we use the introduced characteristic kinds with their properties and goals. For the description we chose the most important and marked characteristics (see section 3.2). Again only special chosen characteristics will be presented and not all necessary steps for the execution are conducted. Since nearly all technical categories require an architecture model, scenarios and goals as *Input* we won't mention it below.

The first technical category represents analyses conducted with **Bayesian Networks** (e.g. [Närman et al., 2008]). Analyses of this category utilize this technique to analyze the quality of systems and architectures. It is reused in other analyses as part of their procedures, e.g. **PRM** analyses. Requirements of the *Input* are a meta model with entities, attributes and references, different architectural layers and at least two scenarios. These requirements are the most common *Input* prerequisites. The *Condition* are defined attribute states and connection types. They must have discrete areas and are either a causal relation or a definitional relation. In the *Construct*, firstly a model with Bayesian Networks is built, including all nodes and connections of the architecture. A node represents a variable with conditional probability distribution. Therefore in the next step probabilities of attributes and the whole model can be calculated while creating matrices with discrete ranges, connections and weighted attributes. In conclusion this category has probability values as *Output* and can answer questions about the probability of an attribute's status.

Business Entities are a method to receive artifacts on the one hand and on the other hand to analyze quality (e.g. [Della Bordella et al., 2011]). Here it can be distinguished between analyzing single entities, combined entities or their quality. As *Input* and *Condition* BMM and UMD diagrams with all relations and processes, the goal type, strategies and quality features have to be determined. The first step of the *Construct* detects advantages, operations and elements of strategies. As a second step, influencer and strategies are combined to observe the goals. Additionally matching operations and their entities are identified and assigned. The *Measurement* quantifies the goal. Dependent on the chosen goal type, the strategy elements are evaluated. For instance an observation of maturity can be conducted by weighting elements with a scale. Therewith the strategy with the highest efficiency is identified. The *Output* contains valued strategies, quality values and identified operations and entities.

Probabilistic Relational Models (PRM) contains 14 analyses and is therewith the most used technique (e.g. [Buschle et al., 2011]). For instance dependency and quality analyses can be conducted with PRM. Therefore artifacts and effects are the main focus. An EA model, scenarios, problems and goals are the *Input*. *Conditions* require controllable attributes and determinable goals and criteria for the later determination of metrics. As a prestep of the *Construct* connections are defined and uncertainties are formalized. Hereafter a concrete model is built and the PRM is used to calculate the conditional probability of all scenarios and of the dependencies and attributes. PRM can be seen as template of an architecture model. This model has a set of classes. Every class has attributes, values and references. The connections can be one out of five states. This model is conducted with every scenario of the input data. Therefore it is possible to calculate the probability of every scenario. Additional the probability of attributes is determined by using Bayesian Networks. The *Output* contains a probability for attribute values, scenarios and uncertainty values.

Social Network analyses (e.g. [Kazienko et al., 2011]) differ deeply from the other categories. For their conduction questionnaires and all available documents, like bills and connections are required. For the *Input* all available nodes (= entities), connections and needed data sources have to be defined. Entities are persons, groups and companies which can have roles. Only if the methods are accepted, the analysis can be successful, because of the needed support of employees. As *Construct* clusters are built and properties can be checked. Additional new entities and connections are found. One method is combining the socio metric data analysis, questionnaires and other data sources to identify new entities and connections and to evaluate them afterwards. For the *Measurement* a matrix with entities and factors is created for quantitative evaluation with factors or for identification of weak points. An overview of the whole architecture model and its entities and connections on a social basis is found in the *Output*.

Analytic Hierarchy Process (AHP) can be used for analyzing attributes and quality aspects and is one of the most elaborated EA analysis technique (e.g. [Razavi et al., 2011]). Therefore the requirements for execution as well as the procedure of the identified approaches are described detailed. Typical requirements for execution are a model, scenarios and uncertainties values. *Conditions* are expert knowledge, which is used for weighting, as well as quality attributes and their level of success. First in the *Construct* and *Measurement* the quality attributes with their criteria, subcriteria and level of importance are determined. Then the quality attributes are weighted through a pairwise comparison according to the architectural layers by experts. All weightings of the importance level are summarized in a vector and in the next step a prioritized vector of the layers is created. This prioritized list of quality attributes is used for a concrete definition of the scenarios. These scenarios are also compared pairwise to each other, which concludes in a table with the priorities of the scenarios according to the quality attributes. Finally the most suitable scenario is selected and the level of suitability and uncertainty of the selection is calculated. The results are described in the *Output* as prioritized lists of quality attributes and scenarios.

The method of **Time Evaluation** (e.g. [Lankhorst, 2013]) observes the quality of business entities and operations through the calculation of time values. Analyses which try to optimize the performance utilize this technical category. For example processes and entities are checked for weak points. As additional information to the common requirements, trigger and arrival times are required as *Input*. Rules are necessary to cut the architecture in views and conclude with five perspectives for single time measurements (*Condition*). In this category *Construct* and *Measurement* are combined and require specific time values and calculation metrics. For each view the specific time values are calculated. Examples are “Costumer View” and “Process View” with “Processing Time” and “Response Time”. First, the workload calculation is conducted with a top-down approach. Afterwards the calculation will be applied backwards with a bottom-up approach. Finally all calculated times are summed up to a total time.

Trees are used to analyze and identify dependencies, coherence’s and quality features. The *Output* of those analyses delivers the probability for the occurrence of a failure or specific quality attribute. All necessary operations, procedures, scenarios and time values are included in the *Input* dependent on the goal type. In the beginning of the *Construct* the goals, entities and relations are defined. Afterwards a fault tree is built using Bayesian Networks, containing all steps or events required for the execution. Thereby all given scenarios have to be conducted. For every component of this tree a conditional probability matrix is created to receive the probability of failures or quality attributes [Närman et al., 2011]. For the *Measurement* the time values are summed up or probabilities are calculated.

The technique of **KPI** (Key Performance Indicator) is used in most analyses with quantitative measurement. Because of the high variety of contained analyses, a high level of abstraction was developed. As *Input* an UML meta model with layers is required. The *Condition* is very important for this kind of analysis. A goal has to be defined according to the SMART criteria: It has to be specific, measurable, achievable, realistic and time-bound. The *Construct* starts with the identification of all artifacts that have to be analyzed and and with the determination of the matching KPIs. In the *Measurement* the artifacts are evaluated and the determined values are compared. It is possible to measure single artifacts or to summarize them and evaluate the whole system. Another method for measurement is the usage of matrices, where two different dimensions have to be selected. For example a matrix can present the costs dependent to different organization units. The result in this analysis category represents the goal achievement, unsatisfied quality constraints or the financial situation [Niemann, 2006].

Comparison is a simple but powerful method (e.g. [de Boer et al., 2005]). Next to whole alternatives, also single scenarios, processes, attributes and dependencies can be compared to each other. It is possible to compare different state in times, i.e. the as-is with the to-be, but also different alternatives, i.e. potential future scenarios. Requirements constraint the alternatives and support the achievement of the desirable vision. Additional rules are used to create a consistent model with all requirements and suitable to the end product. First in the *Construct* viewpoints are chosen and a model is created containing all components which should be analyzed. This model can differ dependent on the analysis object. Afterwards the models are compared with a previous state or another alternative. In addition the single elements will be changed and the impacts observed. On this way all possible states can be observed and the best alternative to achieve the goals is identified. The results of the *Output* show what is required to achieve the to-be state and the different impacts dependent on the input.

The technique **Views** is used to analyze aspects in detail or to create different perspectives. It is a powerful tool for EAM, nevertheless only a few analyses use this technique explicitly. However we identified several analyses utilizing the concept of views in their procedure (e.g. [Sasa and Krisper, 2011]). The necessary *Input* and *Conditions* are chosen views, a distinct goal and determinable connections. Criteria and their desirable perspective have to be specified in the *Construct*. Examples are time measures like response time or processing time. After this the views can be built with all required components. A definite *Measurement* is not contained in this category. However, views can be evaluated with criteria to observe whether the view can achieve its goals, for example focus on the processing time.

A less popular methodology is the observation of **Lifecycle** (e.g. [Saat, 2010, Aier et al., 2009]). These analyses ascertain requirements and dependencies through consideration of different lifecycle phases. Therewith changes are identified and it is possible to determine the state of an artifact at a specific point in time. For the conduction the life cycle phases (states) of the artifacts have to be given as attribute assignments and time values in the *Input*. In the analysis *Construct* the lifecycle of the artifacts in the respective architecture part is determined. Afterwards, to check the state of an artifact at a specific point in time the life table method is used. Thus the change cycles are preserved and the validity of dependencies can be determined. In the *Measurement* the probability for an artifact being in a specific state at a specific point in time is calculated. The *Output* is either this probability or the change cycle.

Using **Ontologies** is an uncommon analysis technique in the domain of EAM (e.g. [Sunkle et al., 2013]). The contained analyses, ‘Change Impact Analysis’ and ‘Structural Analysis’ analyze dependencies respectively the architectural construct. A special meta model created with ontological rules is required as *Input* and *Condition*. The meta model defines the entities (i.e. artifacts and dependency types) of the EA model whereas the rules define the dependencies between the elements. The *Construct* analyzes the entities and dependencies in order to determine specific sets of them. Afterwards dependencies, viewpoints and special factors are evaluated for the outcome.

EID (Extended Influence Diagram) is the third most technique in order to conduct EA analyses (e.g. [Johnson et al., 2007a]). Possible results can be statements about maintainability, security and availability. Therefore systems, attributes, quality aspects, impacts and data are analyzed. The steps of the procedure are independent of the analysis type. The scenarios and alternatives have to be conducted and a goal must be defined. As *Condition* it has to be secured that the contained components can be built with EID. Afterwards all scenarios, goals and entities are represented as EID nodes and connections. For *Measurement* Bayesian Networks are used to calculate the probabilities of the attributes. Thus it is possible to analyze dependencies by inferring changes and altered values.

For the identification of **Weak Points** and their costs the following requirements can be determined (e.g. [Xie et al., 2008]). *Input* data are workflows and resources as well as their availability requirements. In the *Construct* a matrix of the workflows and resources is created, which is used for the availability calculation. Whenever the availability is higher as the availability requirement, the condition is fulfilled. If this is not the case a enhancement parameter helps to calculated the current level of availability. Afterwards the expected availability for every workflow is calculated. In addition it is possible to weight resources and receive alternatives with higher availability. The *Output* is the assignment of availabilities to resources.

Another identified technique is the usage of a **Matrices** (e.g. [Szyszka, 2009]). Application fields are for instance 'Coverage Analysis' or 'Maturity Analysis'. Matrices can be used in various ways, mostly for the presentation of results. The *Input* is a common architecture model with classes, types and relations. Additional the goal and application area is required. In order to built and evaluate the matrix, the dimensions and the kind of measurement have to determined. Additionally the elements have to be aligned within the matrices. Dependent on the measurement method a quantitative evaluation or a scale for discrete areas is conducted. The results can vary from quantitative outcomes to weak points, redundant artifacts and functional dependencies.

Analyses joint in the category **Design** are able to observe architecture design in a specific way ([Aier et al., 2011]). The analysis identifies strengths and weaknesses of the architecture. As *Condition* the considered factors and expert knowledge is required. In the main *Construct* items and data are determined, factors are checked with questionnaires and a cluster analysis is conducted. Similarities and clusters are identified through this way. As *Measurement* a matrix of items and factors is built and evaluated. In the *Output* the results of the matrix evaluation represent the potential of a cluster.

The last technical category contains an analysis with a **Structural** procedure ([Buckl et al., 2009a]). This analysis tries to observe design through displaying obstacles of different architecture versions. Therefore a documentation of the EA is required as *Condition* and the main part of the analysis consists of an observation of changes. The *Output* type is unique and represents potential obstacles caused by different versions.

4 Formalization of analysis requirements

The identified requirements for the 10 functional categories and the 17 technical categories are formalized using a domain specific language. Therewith we can elaborate the integrity and correctness of the requirements, i.e. if they are sufficient to describe the analyses in an adequate way. The language provides a uniform description possibility for EA analyses. This supports the decision making about the implementation of an analysis since their requirements and goals are obvious the uniform description makes them comparable. For the language development we used Xtext[1], a framework that comprises a powerful language for the description of textual languages. The framework generates the model as well a parser, linker, type checker and compiler. The DSL was developed according to the meta model development process for abstract syntax development from [Brambilla et al., 2012]. This incremental and iterative process consists of three phases: The Modeling Domain Analysis phase, elaborating the purpose and content, the Modeling Language Design phase, defining the meta model, and the Modeling Language Validation phase, verifying the correctness and integrity. For the last step we select representative EA analyses for each category and formalize them using our modeling language. Difficulties and mistakes during modeling trigger a new iteration of the development process. The concrete syntax is developed simultaneously with the abstract syntax due to the nature of Xtext.

The developed DSL is structured in a general and in a categorization specific part. Figure 3 shows the main rule for the analysis language and the realization of the dimensions. General requirements that occur in all categories are summarized at beginning in the main rule. This is the name of the analysis, the required meta model and potential

```
MetaLanguage:
  'EAM Analysis Language' '{'
  //Domain Definition: General Requirements
    'Performing Analysis' analysis=STRING
    'Metamodel' model+=UMLModel ('{'
       'Scenarios:' scenarioName+=NameIdentifier (scenarioModel+=UMLModel)*
             (";" scenarioName+=NameIdentifier (scenarioModel+=UMLModel)*)*
    '}')?
    'Goal' goal=STRING
    'Goal Type'':' goalType+=GoalType ('&' goalType+=GoalType )*
  '}'
  //Choice of Dimensions
    ('Category functional Dimension' ':' functional+=Functional)?
    ('Category technical Dimension' ':' technical+=Technical)? ;

//----------Functional Categories---------------------------------------//
Functional:
  SystemAnalysis | AttributeAnalysis |    ...    | BusinessObjectAnalysis;

//----------Technical Categories---------------------------------------//
Technical:
  BayesianNetworks | BusinessEntities |  ...  | Structural ;

//Choice of a possible technique matching to the chosen functional Category
SystemAnalysis:
  'System Analysis' (':')?
  ('Technique' analysisTechnique+=SystemAnalysisTechnique)? ;

SystemAnalysisTechnique:
  EID | PRM | BayesianNetworks ;
```

Fig. 3: DSL for EA analyses - main rule

[1] Xtext https://eclipse.org/Xtext/index.html

scenarios to evaluate. For description of the meta model and the scenarios we developed a language construct that allows to specify them similar to a UML model. The goal of an analysis is modeled using a string and its type is defined with an enumeration. Possible goal types are: percentage, matrix, probability, dependency, object, effect, scenario, number or boolean. The choice of the analysis dimensions is realized considering the later usage behavior. The user can choose first either the functional dimension or the technical one. The rule system of the DSL restricts the second dimension to those that are feasible. For example the functional dimension **System Analysis** has realizations with the technical dimensions **EID**, **PRM** and **Bayesian Networks**. The rule *SystemAnalysisTechnique* ensures the integrity of the selection according to the matrix (figure 2). If the achievement of a planned goal is most important, the functional dimension is decided first. Thereby decisions about the function and purpose of the analysis have to be made, how the analysis is conducted is not in the main focus of the user. As second option the user decides first about the utilized technique. This option is used in case only a specific method or technique should be used, e.g. because of a tool restriction or availability issues. After choosing the technical category, it is possible to discover which goals can be achieved with it (i.e. decide about the functional category).

```
BayesianNetworks:{BayesianNetworks}
    'Bayesian Networks'
    ('Function' analysisFunction+=BayesianNetworksAnalysisFunction)?
    'INPUT'          '{' … '}'
    'CONDITIONS'     '{' … '}'
    'CONSTRUCT'      '{' … '}'
    'MEASUREMENT'    '{' … '}'
    'OUTPUT'         '{' … '}'
;

BayesianNetworksAnalysisFunction:
    SystemAnalysis | DependenciesAnalysis | QualityAnalysis
;
```

Fig. 4: Excerpt of the description of Bayesian Network analysis using the DSL

For each technical category a rule is implemented that satisfies the requirements specified in chapter 3. The rules comprise statements for defining the input, the conditions and construct, the measurement and the output (see section 3.2). Figure 4 shows an excerpt of this part of the DSL. Inside the five blocks the specific characteristics of the analysis are defined. According to the complexity of the conducted analysis two blocks can be summarized into one combined block or a block can also be omitted.

To illustrate the structure of a category definition figure 5 shows an example description of the Information Security Analysis from [Johnson et al., 2007a]. This analysis evaluates the architecture by calculating the probability of quality attributes for security. Corresponding to the main rule the description starts with the analysis name followed by a specification of the meta model and two scenarios. The meta model describes the classes, relationships and attributes that are necessary for the analysis. The two scenarios represent different alternatives that should be evaluated. The scenario description is followed by the goal statement and the goal type, in this case *percentage*. Then the functional and technical dimension is defined. The functional dimension is **Attribute**

```
EAM Analysis Language{ Performing Analysis "Information Security Analysis"
    Metamodel Model"Architecture of Information Security"{
        Class "Application"{ ... }
        ...
    }
    {Scenarios:
        "Scenario 1" Model "UML Model Scenario 1"{     ...     };
        "Scenario 2" Model "UML Model Scenario 2"{     ...     }
    }
    Goal"Probability of quality attributes for security"        Goal Type :Percentage
    }
    Category functional Dimension:Attribute Analysis:
    Technique Extended Influence Diagram
        INPUT{ Metamodel "Architecture of Information Security"{
                Scenario"Scenario 1", Scenario"Scenario 2"
            }
        }CONSTRUCT{
            EID MODEL ELEMENTS{ Chosen Scenario "Scenario 1"
                //Value assumptions
                Scenario Node:  type: Decision Node "Scenario Selection" Value: 0."90"
                Goal:           type: Utility Node "Profit" Value: 0."0"
                Attributes:     type: Chance Node"User Training Process" Value: 0."75"
                                ...
                Relations:      "Scenario Selection" as Causal Relation to "User Training Process"
                                ...
            }}
        MEASUREMENT{
            //Example calculation for one node for one scenario
            Chance Node Selection:"User Training Process"
            Scenario Name "Scenario 1"->"Present":
            Calculation of Section: P("User Training Process")= ...
            Result="0.95"
            Goal Calculation: P(A|B)=P(B|A)P(A)/P(B)
            Result: "Usage of Bayesian network analysis tool GeNIe"
        }
        OUTPUT{
            Results: Best Scenario "Scenario 1"
        }
```

Fig. 5: Excerpt of the description of EID analysis using the DSL

Analysis and the technical one is **Extended Influence Diagram**. The remaining structure of the analysis specification is specific for analyses of the category **EID**. The input of the analysis is here straightforward the defined meta model and both scenarios. The construct part defines the requirements in order to create extended influence diagrams. First the chosen scenario is set, and then the nodes, goals and attributes with their types and values are defined. Finally the EID specific relations are declared. In the measurement part for each node in each scenario a matrix with the conditional probability is specified. This is exemplary shown in figure 5. The value of the node ’User Training Process’ is defined with a EID calculation. This calculation determines the probability of the node to be in a specific discrete range, here ’present’, in dependency from further nodes. Finally the result and the goal calculation according to the bayesian theorem are declared. Such a calculation can also be done for a quality attribute to have a specific value in one scenario. The output contains the scenario with the best values according to the measurement.

5 Evaluation and discussion

We evaluated our DSL through formalizing existing analysis approaches from literature. The experiences and limitations are presented below, followed by a discussion about the categorization as well as related work.

All identified categories, functional as well as technical, are integrated in the language and it was possible to formalize a representative from each category. Figure 5 shows an excerpt of the definition of the Information Security Analysis (functional dimension: attribute analysis, technical dimension: extended influence diagram). The language can be reused for the development of new analyses, since it provides a sound foundation of requirements that have to be extended. After further development the language can also be used as an entry point for the specification and execution of analysis. The analysis language itself provides a high abstraction grade that enables also the application for example in service-oriented architectures or business analyses. Additionally the language is easy extendable and without special knowledge understand- and usable.

Most of the requirements for the technical categories are realized in the language. A few requirements are determined as given and not further mentioned, since these requirements are obviously. Examples are the possibility to raise data, i.e. whether data can be used or be accessible. In addition requirements, which are not verifiable couldn't be included. For instance, it is not verifiable whether the meta model can be used to achieve the goals, whether artifacts can be mapped to EID components or whether used nodes are controllable. Additionally the acceptance of a used technique or the availability of expert knowledge is not verifiable and thus not integrated in the language. Requirements that are defined in a graphical way, for example matrices, are difficult to realize in a textual language. Also the definition of patterns is only specified with limitations in the language. The lower number of functional categories in contrast to the technical ones can be explained with the focus on one field of interest. Since we concentrate on pure EA analyses the analysis goals were repetitive. A problem during categorization was the issue that not all aspects from the analyses are described in detail in the available publications. At this point we were only able to identify limited requirements or we had to make assumptions in order to proceed. A interrelated problem is the fact of the low amount of available descriptions of conducted analyses to evaluate our language. Additional some analyses use very specific techniques or modeling approaches. Here, it was not possible to consider all details in order to create a sound categorization. We abstracted from some specifics in order to define the general requirements for a category. We received the general valid requirements by focusing on the approach with the highest level on elaboration and abstracting from it considering the issues of the other approaches. An example is the technical category **KPI** with a high abstraction level. The contained analyses differ deeply in measuring values with different formulas. Therefore the mathematical computation cannot be described in full detail in our language.

Encountered categorization approaches in related work tried to focus on the meta level. However, in contrast to our target they designed an analysis framework independent from the meta model [Langermeier et al., 2014], developed a category independent meta language [Buckl et al., 2011] or had the main focus on characteristics

[Buckl et al., 2010]. Additional EA analyses can be distinguished between the point of execution time. Therefore the analyses are sorted in ex-ante and ex-post to determine whether an analysis will be conducted before or after the adoption of an architecture. It is also possible to separate the analyses according to their execution technique: expert-based, rule-based or indicator-based [Buckl et al., 2009b]. However, both classifications are not detailed enough to identify characteristics and most of the analyses can't be strictly classified within these divisions. Lankhorst (2013) conducted an initial categorization with four dimensions: Quantitative and functional differ at the input and output data. The functional dimension can be further distinguished in static and dynamic. However this division is not detailed enough to identify the explicit requirements of classified analyses and four categories is a rough classification. Therefore an advantage of categorization cannot be accomplished. Regarding the varieties of containing approaches, the existing categorizations are not sufficient.

6 CONCLUSION

In this paper we presented a two-dimensional categorization of EA analyses, based on the characteristics of the approaches found in literature. The first dimension addresses the functional aspect, the second one the technical aspect. Altogether we identified 105 analyses, which are classified in 10 functional categories and 17 technical categories. Using this categorization we can identify 40 different analysis types used in EA. The dependencies between the approaches of the functional and technical dimensions are visualized in a matrix. The dependencies as well as the characteristics of the analysis categories are formalized with a domain specific language. The language provides a structural way to represent the preconditions of an analysis, the technical requirements for execution and also the outcome of it. Additionally the enterprise architecture can use the language to decide whether the outcome of an analysis is from interest for his question, if the analysis is applicable on his EA model and how great the effort of adaption is in order to execute the analysis. The idea of such an EA analysis catalog is the support of reuse of existing work in the domain of EA. Therefore future work has to investigate techniques for context independent execution of those analyses. This could be the development of tool support for the usage of the categories and the DSL. Thus, computations, which need further tools, can be included, new analyses could be created simplified and requirements are checked automatically. Additionally a higher abstraction level of the category characteristics would be conceivable to make the requirements general valid.

References

[Aier et al., 2009] Aier, S., Buckl, S., Franke, U., Gleichauf, B., Johnson, P., Närman, P., Schweda, C. M., and Ullberg, J. (2009). A survival analysis of application life spans based on enterprise architecture models. In *3rd Workshop on EMISA*, pages 141–154.

[Aier et al., 2011] Aier, S., Gleichauf, B., and Winter, R. (2011). Understanding enterprise architecture management design-an empirical analysis. In *Proceedings of 10th Conference on Wirtschaftsinformatik*.

[Brambilla et al., 2012] Brambilla, M., Cabot, J., and Wimmer, M. (2012). *Model-driven software engineering in practice*.

[Buckl et al., 2011] Buckl, S., Buschle, M., Johnson, P., Matthes, F., and Schweda, C. M. (2011). A meta-language for enterprise architecture analysis. In *Enterprise, Business-Process and Information Systems Modeling*, pages 511–525. Springer.

[Buckl et al., 2009a] Buckl, S., Matthes, F., Neubert, C., and Schweda, C. M. (2009a). A wiki-based approach to enterprise architecture documentation and analysis. In *ECIS 2009 Proceedings*, pages 1476–1487.

[Buckl et al., 2009b] Buckl, S., Matthes, F., and Schweda, C. M. (2009b). Classifying enterprise architecture analysis approaches. In *Enterprise Interoperability*, pages 66–79. Springer.

[Buckl et al., 2010] Buckl, S., Matthes, F., and Schweda, C. M. (2010). A Meta-language for EA Information Modeling State-of-the-Art and Requirements Elicitation. In *Enterprise, Business-Process and Information Systems Modeling*, pages 169–181. Springer.

[Buschle et al., 2011] Buschle, M., Ullberg, J., Franke, U., Lagerström, R., and Sommestad, T. (2011). A tool for enterprise architecture analysis using the PRM formalism. In *Information Systems Evolution*, pages 108–121. Springer.

[Davis, 1989] Davis, F. D. (1989). Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS quarterly*, 13(3):319–340.

[de Boer et al., 2005] de Boer, F. S., Bonsangue, M. M., Groenewegen, L., Stam, A., Stevens, S., and Van Der Torre, L. (2005). Change impact analysis of enterprise architectures. In *IEEE International Conf. on Information Reuse and Integration*, pages 177–181.

[Della Bordella et al., 2011] Della Bordella, M., Liu, R., Ravarini, A., Wu, F. Y., and Nigam, A. (2011). Towards a method for realizing sustained competitive advantage through business entity analysis. In *Enterprise, Business-Process and Information Systems Modeling*, pages 216–230. Springer.

[Johnson et al., 2007a] Johnson, P., Lagerström, R., Närman, P., and Simonsson, M. (2007a). Enterprise architecture analysis with extended influence diagrams. *Information Systems Frontiers*, 9(2):163–180.

[Johnson et al., 2007b] Johnson, P., Nordström, L., and Lagerström, R. (2007b). Formalizing analysis of enterprise architecture. In Doumeingts, G., Mller, J., Morel, G., and Vallespir, B., editors, *Enterprise Interoperability*, pages 35–44. Springer London.

[Kazienko et al., 2011] Kazienko, P., Michalski, R., and Palus, S. (2011). Social network analysis as a tool for improving enterprise architecture. In *Agent and Multi-Agent Systems: Technologies and Applications*, pages 651–660. Springer.

[Krcmar, 2015] Krcmar, H. (2015). *Informationsmanagement.* Gabler.

[Langermeier et al., 2014] Langermeier, M., Saad, C., and Bauer, B. (2014). A unified framework for enterprise architecture analysis. In *18th IEEE International EDOC Conference Workshops*, pages 227–236.

[Lankhorst, 2013] Lankhorst, M. (2013). *Enterprise Architecture at Work: Modelling, Communication and Analysis*. Springer.

[Matthes et al., 2011] Matthes, F., Monahov, I., Schneider, A., and Schulz, C. (2011). Eam kpi catalog. Technical Report v 1.0, Technical University Munich.

[Närman et al., 2012] Närman, P., Buschle, M., and Ekstedt, M. (2012). An enterprise architecture framework for multi-attribute information systems analysis. *Software & Systems Modeling*, pages 1–32.

[Närman et al., 2011] Närman, P., Franke, U., König, J., Buschle, M., and Ekstedt, M. (2011). Enterprise architecture availability analysis using fault trees and stakeholder interviews. *Enterprise Information Systems*, 8(1):1–25.

[Närman et al., 2009] Närman, P., Johnson, P., Ekstedt, M., Chenine, M., and König, J. (2009). Enterprise architecture analysis for data accuracy assessments. In *13th IEEE International EDOC Conference*, pages 24–33.

[Närman et al., 2008] Närman, P., Schonherr, M., Johnson, P., Ekstedt, M., and Chenine, M. (2008). Using enterprise architecture models for system quality analysis. In *12th IEEE International EDOC Conference*, pages 14–23.

[Niemann, 2006] Niemann, K. D. (2006). *From enterprise architecture to IT governance*. Springer.

[Rauscher, 2013] Rauscher, J. (2013). Analysen in Unternehmensarchitekturen - Ziele, Techniken, Anwendungsbereiche. *Bachelor Thesis, University Augsburg*.

[Rauscher, 2015] Rauscher, J. (2015). Anforderungen an und Definition von einer Analysesprache für das Enterprise Architecture Management. *Bachelor Thesis, University Augsburg*.

[Razavi et al., 2011] Razavi, M., Aliee, F. S., and Badie, K. (2011). An AHP-based approach toward enterprise architecture analysis based on enterprise architecture quality attributes. *Knowledge and information systems*, 28(2):449–472.

[Saat, 2010] Saat, J. (2010). Zeitbezogene Abhängigkeitsanalysen der Unternehmensarchitektur. In *MKWI*, pages 119–130.

[Sasa and Krisper, 2011] Sasa, A. and Krisper, M. (2011). Enterprise architecture patterns for business process support analysis. *Journal of Systems and Software*, 84(9):1480–1506.

[Sunkle et al., 2013] Sunkle, S., Kulkarni, V., and Roychoudhury, S. (2013). Analyzable enterprise models using ontology. In *CAiSE Forum*, volume 998, pages 33–40.

[Szyszka, 2009] Szyszka, B. (2009). Analysis and classification of maturity models in enterprise architecture management. *Bachelor Thesis, Technical University Munich*.

[Van Lamsweerde, 2001] Van Lamsweerde, A. (2001). Goal-oriented requirements engineering: A guided tour. In *5th IEEE International Symp. on Requirements Engineering*, pages 249–262.

[Wan and Carlsson, 2012] Wan, H. and Carlsson, S. (2012). Towards an understanding of enterprise architecture analysis activities. In *Proceedings of 6th ECIME*.

[Winter and Fischer, 2006] Winter, R. and Fischer, R. (2006). Essential layers, artifacts, and dependencies of enterprise architecture. In *10th IEEE International EDOC Conference Workshops*.

[Xie et al., 2008] Xie, L., Luo, J., Qiu, J., Pershing, J., Li, Y., Chen, Y., et al. (2008). Availability “weak point” analysis over an SOA deployment framework. In *IEEE Network Operations and Management Symposium*, pages 473–480.

[Zia et al., 2011] Zia, M. J., Azam, F., and Allauddin, M. (2011). A survey of enterprise architecture analysis using multi criteria decision making models. In *Intelligent Computing and Information Science*, pages 631–637. Springer.