Generic EA Analysis Framework for the Definition and Automatic Execution of Analyses

Melanie Langermeier and Bernhard Bauer
Software Methodologies for Distributed Systems, University of Augsburg, Universitätstrasse 6a, 86135 Augsburg, Germany

Keywords: Analysis, Enterprise Architecture, Enterprise Architecture Analyses, Domain Specific Language.

Abstract: Analysis is an essential part in the Enterprise Architecture Management lifecycle. An in-depth consideration of the architecture obtains its strengths and weaknesses. This provides a sound foundation for the future evolution of the architecture as well as for decision-making regarding new projects. Current literature provides a large number of different analysis approaches, targeting different goals and utilizing different techniques. To provide a common interface to analysis activities we studied the corresponding literature in previous research. Based on these results we develop a language for the definition of EA analyses as well as an execution environment for their evaluation. To cope with the high variety of meta models in the EA domain, the framework provides a uniform and tool independent access to analysis activities. Additionally it can be used to provide an EA analysis library, where the architect is able to select predefined analyses according to his specific requirements.

1 INTRODUCTION

Today's IT landscapes in organization are typically the product of evolved structures extended with additional parts through merger and acquisitions. Since IT was not always seen as critical factor their management and alignment was partially unattended. This faces organization with the problem of a high complexity and heterogeneity when regarding their IT support. Enterprise Architecture Management (EAM) provides means for organization to capture the essential business and IT elements as well as the dependencies between them. This provides a clear understanding of the structure and enables an organization-wide optimization of the architecture (Lankhorst, 2013).

Therefore Enterprise Architecture (EA) models are used within a plethora of different application scenarios. Among those are IT business alignment, project portfolio planning, business process optimization, sourcing decision and IT service management. To utilize the EA models in these contexts the contained data has to be analyzed, summarized and interpreted with suitable analysis techniques (Bucher et al., 2006). For example during project portfolio planning, analyses support the decision about approval or rejection of a project proposal. They are used to calculate effects and evaluate the quality of the future architecture. Another application scenario for analysis techniques is the provisioning of a dashboard with performance indicators to support management decisions (Frank et al., 2009).

Summarized, EA analyses are means to utilize the established EA models. They increase the understanding of the architecture and provide aggregated information to the management, e.g. through a dashboard. Through an evaluation of the current and target architecture as well as potential change scenarios, they support decision-making and architecture evolution (Sasa and Krisper, 2011).

To cover the various application fields a plethora of different EA analysis approaches can be found in literature. The approaches fulfill various different goals and use a wide selection of techniques to reach them (Rauscher et al., 2017). Each approach typically serves a specific use case and incorporates a specific implementation. Their adaption to other use cases or their use within different EA models is a complex task and made reuse difficult. To take advantage of the existing EA analysis approaches a unified approach to EA analysis is important (Naranjo et al., 2014; Buckl et al., 2009; Johnson et al., 2007). Such an approach could also ease the development of a common analysis catalog as proposed by (Lantow et al., 2016).

The broad application field of EA analyses as well as the specifics of EA models lead to several challenges. First, EA models are large and complex models that are created by using different, often disconnected modeling languages (Naranjo et al., 2015).

Langermeier, M. and Bauer, B.
Generic EA Analysis Framework for the Definition and Automatic Execution of Analyses.
DOI: 10.5220/0006379003160327
In *Proceedings of the 19th International Conference on Enterprise Information Systems (ICEIS 2017) - Volume 3*, pages 316-327
ISBN: 978-989-758-249-3

There exist several frameworks each providing a meta model proposal but no common standard for EA models. Typically the organization chooses a specific framework and additionally makes further adaptions. Second, the EA models of an organization are often not fully specified, i.e. an analysis technique has to deal with incomplete models.

In previous research we analyzed current EA literature and identified different analysis types and categories (functional and technical) with their characteristics (Rauscher et al., 2017). Additional in (Langermeier et al., 2014b) we presented an approach for the execution of EA analyses based on a generic EA meta model. In the following we combine both results to define a framework for the generic EA analysis definition and execution. Therefore we developed an EA analysis definition language (Arla) that allows the architect to define and customize analyses like performance indicators or views, while abstracting from the technical details. For evaluation the Arla analysis definition is interpreted as executable rules. In order to support a broad spectrum of EA analysis we use a combination of SPARQL, a query language for triple stores, and a data-flow based approach for their execution.

In the next section we present the foundations about EA analysis as well as the results of our previous work. In section 3 the developed language is presented followed by the execution mechanism in section 4. The evaluation is presented in section 5. Therefore we determined the coverage degree of Arla by analyzing the EA literature with respect to their feasibility in Arla. Additionally examples of Arla definitions are presented in section 3, which were executed on different EA models for evaluation purposes.

2 EA ANALYSIS

Enterprise architecture models support the architecture process only in a visual and qualitative way (Franke et al., 2009). Analysis techniques provide means to quantify models, predict future behavior and compare different alternatives. They make use of the contained information in order to support the planning process. The EA cycle encompasses the phases *document*, *analyze*, *plan*, *act* and *control* (Niemann, 2006). Thus they are an essential part in the overall EAM process. Examples for analysis procedures are coverage analysis, analysis of the interfaces, heterogeneity analysis, analysis of complexity, conformity, costs and benefits (Niemann, 2006). To receive more expressive analyses the combination of several techniques and analysis methods is proposed (Naranjo et al., 2015).

The analysis activities during EAM are unforeseeable, since they are not known in detail at the beginning (Naranjo et al., 2015). Changing business or IT requirements as well as strategic changes trigger changes in the analysis necessities. For example a new strategy will lead to new goals for business and IT. Goals are often monitored using KPIs, thus, changing the goals requires an adaption of the KPIs. Additionally generated analysis results provide new information, which influences the future proceeding. For example if a severe impact is calculated for a technology change, the architect wants to analyze the impact in detail.

Jonkers and Iacob propose a differentiation between design and analysis (Jonkers and Iacob, 2009). The design space of EA is modeled using languages like UML, business process modeling language like BPMN or architectural description languages like ArchiMate. For the analysis space they propose a special-purpose language that enables the later analysis. Our approach focuses on the analysis space. We deal with the analysis of the model data of an enterprise architecture, i.e. how can the data be analyzed and how can the analysis procedure be defined and executed. This includes the decomposition in smaller model parts and the evaluation of those. The result of an evaluation is either a model part, the reduction of the architecture or an architecture part to an attribute or quantitative value or the extensions of an architecture or architecture part with calculated attributes or quantitative values. The visualization of the result is basically covered in our approach in order to make the results verifiable. Further research discusses the topic of visualization and visual analysis in detail e.g. (Naranjo et al., 2015)

In previous work we analyzed the current literature about enterprise architecture analysis. In (Rauscher, 2013) we identified 105 analysis approaches, which are roughly grouped in 40 analysis types. The groups are created based on the scope of the respective analyses. The goals and realization methods summarized in one type can differ. Examples of analysis types are 'Analysis of Service Response Time' (e.g. (Närman et al., 2014)), 'Change Impact Analysis' (e.g. (Sunkle et al., 2013)) and 'Performance and Workload Analysis' (e.g.(Lankhorst, 2013)). Based on this work we propose a two-dimensional categorization of EA analyses in (Rauscher et al., 2017) in order to derive typical characteristics of EA analyses and requirements for their execution. Each analysis approach was assigned to one technical category and at least one functional category. A technical category summarizes ap-

proaches utilizing a similar method with similar steps regardless of the addressed goal and subject. In contrast a functional category summarizes approaches addressing the same objectives and goals. Since an approach can fulfill different goals, there are several approaches with more than one functional category. For example the cost analysis of (Niemann, 2006) is assigned to the technical category *KPI* and the functional category *Financial*. Whereas the quality analysis of (Närman et al., 2008) is assigned to the technical category *Bayesian Networks* and to the functional categories *System, Attribute, Quality* and *Data*.

We identified 10 functional categories: System, Attribute, Dependencies, Quality, Design, Effects, Requirements, Financial, Data and Business Objects. The technical dimension concluded with 17 categories: Bayesian Networks, Business Entities, Probabilistic Relational Models, Social Network, Analytic Hierarchy Process, Time Evaluation, Tree, KPI, Comparison, Views, Lifecycle, Ontology, Extended Influence Diagrams, Weak Points, Matrices, Design and Structural. The assignment of the analyses to the categories is described in detail in (Rauscher et al., 2017) and (Rauscher, 2015). About 30% of the analysis approaches are assigned to a technical category describing probabilistic approaches. Further important techniques are the calculation of measures and KPIs to address the architecture quality and attributes as well as the definition of views on the architecture. Additionally the functional category Dependencies is often used. The technical realization for analyses addressing dependencies varies widely.

Current approaches that try to cover different analysis types are rare. The majority of the approaches are isolated ones that cannot be related to each other. A uniform interface to the different EA analyses is missing. Further challenges that are given only little attention are recursive analysis definitions. Cyclic dependencies in the models as well as incomplete models are sparsely considered. Only a few approaches (e.g. (Kurpjuweit and Aier, 2009)) deal with the indirect relationships in EA models. (Sunkle et al., 2013) addresses EA analysis using ontologies. A lot of approaches utilize probabilistic techniques (e.g. (Närman et al., 2008; Franke et al., 2009)). But none of these approaches is able to cover the various different goals that are addressed by EA analyses. (Naranjo et al., 2014) propose the PRIMROSe framework for visual analysis of EA models. This is a graph based, modular approach to compose predefined analysis function and create sound visualizations by utilizing selectors and decorators. The architect can define the analysis chains, although the scope of the analysis functions is restricted to those that are prespecified. Customization is possible through the defining of a composition chain.

Current EA modeling tools provide extensive analysis support. Limitations are caused by to the modeling approach, the supported meta models and the technical analysis capabilities (Naranjo et al., 2015). Such capabilities can for example only include conformity checks or the generation of predefined views. Some EA tools provide also a possibility to query the model either by providing a DSL or by integrating for example a SQL interface. EA tools provide currently two major approaches to enterprise architecture analysis: Either they are shipped with a predefined and static meta model (e.g. iteraplan, leanIX). Upon their meta model these tools typically provide several analysis techniques out of the box. In the other case the meta model of the tool can be customized to the organization needs (e.g. planningIT). In this case the analysis functions of a tool typically have also be adapted, which is associated with a high effort.

3 ARCHITECTURE ANALYSIS LANGUAGE

The Architecture Analysis Language (Arla) provides a universal interface for the definition and execution of analyses. Thereby Arla abstracts from technical details. The architect defines only ”what” he is interested in, how this information is retrieved is generated from the Arla analysis definition. This execution process is described in section 4. The language supports the specification of analyses for a concrete EA model, but also the definition of templates using placeholder variables for EA stereotypes. In order to apply a template on a concrete EA model, the declared variables have to be mapped to existing stereotypes. A re-definition of the analysis is not necessary.

Another concept to ease the re-use of analyses and support template definition is node and edge classes. Those classes are used to abstract from EA specific stereotypes during analysis definition. Based on a review of EA frameworks we classified relationship types according to their semantics. We identified five classes of relationship types: *Provide*, *ConsumedBy*, *LocalizedAt*, *StructuralDependentOf* and *BehavioralDependentOf*. These edge classes are successfully used in previous work for change impact analysis (Langermeier et al., 2014a) in order to abstract form the concrete EA meta model. Additionally we adapted this concept to the element types. Thereby we identified three types, that are used in all common EA meta models: *Process*, *Application* and *In-*

frastructureElement. The mapping between the EA specific stereotypes and the respective classes has to be defined once. The concluding type declarations are generated automatically during model import while the original stereotype name is also kept. For example the relationship types *realizes* and *access* from ArchiMate can be mapped to the edge class *provide*. In DoDaF the relationship types *provide* and *performs* can be mapped to this class. If the classes are insufficient for an analysis definition, it is always possible to use also stereotypes expressions.

3.1 Analysis Definition

An analysis or template definition is composed of a general part and an analysis specific part. The general part comprises attributes like the name, the analysis type, the result type and a description. The analysis specific part enables the definition of configuration information, necessary for the analysis execution. The language is designed in a modular way, i.e. complex analyses are defined through the composition of simpler ones. Supported analyses in the language are:

- **Metric Definition**: Calculates a metric for each element or a metric for the whole architecture
- **Scope Definition**: Defines a part of the model
- **Path Analysis**: Calculates available paths between two elements according to defined requirements
- **Change Impact Analysis**: Calculates the impact of a change in one element
- **Composed Analysis**: Enables the composition of analyses

```
Template ChangeImpact {
    "Change impact analysis"
    Result Attributes changestatus
    as Element Attribute
    defined with change impact configuration myChangeImpact {
        defined with change impact configuration   egation",edgeType:"assignment")
        defined with composition rule              ed by" )
        defined with scope configuration           alization")
            WeakEffect Out (edgeType:"aggregation")
    }
}
```

Figure 1: Arla template definition.

The specification of an analysis respectively template is exemplary shown for the change impact template (see figure 1). Each template definition starts with the signal word *Template*. A specific analysis starts with the signal word *Analysis* respectively. This is followed by the name of the analysis and a description. Afterwards the name of the result attributes (at least one) are specified, here it is one attribute named *changestatus*. Followed by the signal word *as* the analysis type (here *Element*) and the result type (here *Attribute*) are defined. Afterwards the analysis configuration has to be chosen. The configuration propositions are filtered according to the chosen analysis and result type. We choose the change impact configuration, which is followed by an identifier for later reference purposes. Then the analysis or template specific part is followed.

In this case stereotypes are mapped to the respective effect types to define the change semantic. E.g. the last line defines that an outgoing aggregation edge has the change semantic of a weak effect. Thereby `edgeType: aggregation` is only a placeholder variable. For understand-ability reasons it is recommended to give meaningful names to those variables, although every term is possible. In order to execute the analysis on a specific EA model, the variable has to be mapped to a concrete stereotype of the model in a *AdaptedAnalysis*.

3.2 Language Overview

Figure 2 gives a simplified overview of the analysis language and its elements. Due to readability reasons not all concepts of the language are visualized. Arla consists of three packages. The Base Package (in the figure at the bottom and blue) defines those parts that are used by both analysis types, the specific and the generic ones. The Specific Package (in the left and yellow) defines constructs for a concrete analysis definition whereas the Generic Package (on the right and green) defines the constructs for template definition.

The common attributes of an analysis are summarized in the concept *Analysis Header*. This includes the name, the description and the declaration of the result attributes. The Base Package also defines enumeration types like *AnalysisType*, *ResultType*, *NodeClass* and *EdgeClass* and a generic construct for referencing EA stereotypes. The *AnalysisType* defines whether the analysis should be calculated once for the whole architecture (Aggregate) or for each element (Element). The *ResultType* defines the kind of result. Possible results are *Metric, Attribute, Boolean, Modelelement, Modelelementset* and *Pathset*. The result type and the analysis type are attributes of all analyses. Since they can have predefined values for some analysis types, they are specified in the concrete analysis definition. Additionally the configuration for each analysis type is defined in the base package. This includes calculation rules for metrics, conditions for a node set, rules for analysis composition, the definition of a scope, the configuration of paths, the configuration of a change impact and the definition of a performance analysis.

The Specific Package and the Generic Package

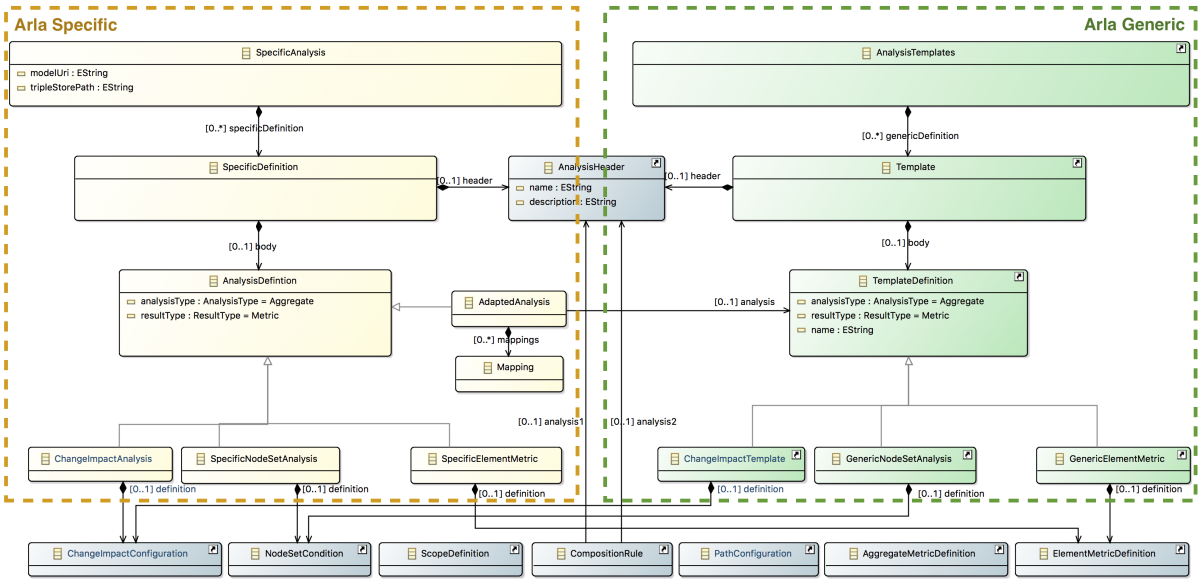

Figure 2: Simplified Arla Overview.

have a very similar structure. Each package has a rule for the main analysis specification. This rule includes a header and a body with the analysis respective template definition. The analysis or template definition rule is specific for every analysis type.

```
ChangeImpactAnalysis:
        analysisTyp=Element
        resultTyp=Attribute
        'defined with change impact configuration' '{'
            definition = ChangeImpactConfiguration
        '}'
;
```

Figure 3: Rule for change impact analysis.

Figure 3 shows exemplary the analysis definition rule for the change impact analysis. First, the result type and the analysis type are defined. For the change impact analysis these values are fix, since a change impact analysis calculates the change value for each element. The analysis type is *Element* and the result type is *Attribute*. The respective analysis configuration is defined as reference to the rule in the Base Package. This *ChangeImpactConfiguration* rule is presented in figure 4. There are three different possibilities for a change impact configuration. One of them, the configuration using stereotypes is presented in detail. Thereby stereotypes are mapped to the respective effect types of change. A functional description of the change impact analysis can be found in (Langermeier et al., 2014a).

One exceptionality of the Specific Package is the *AdaptedAnalysis*. This analysis type is used to execute predefined templates on a specific EA model and has therefore no pendant in the Generic Package. Like other analyses, the definition starts with the selection of the result type and the analysis type. Afterwards the analysis configuration from the utilized template is referenced. A mapping of the stereotype variables to concrete stereotypes of the EA model follows this.

3.3 Example Application

We already presented the definition of a change impact analysis above. To further illustrate the usage of Arla, we present the realization of two additional analyses in the following. As representative of the technical category *KPI* we choose a KPI from the EAM KPI catalog (Matthes et al., 2011). The definition of the *Business application technology standards compliance* in Arla is shown in figure 5. The KPI calculates the compliance degree of business applications to technology standards. Therefore the business applications that comply to a technology standard are divided by the total number of business applications. This KPI can be used to reduce operating costs and security breaches as well as increase the homogeneity. The template in the figure includes the description, the name of the result attribute as well as the calculation rule named *complianceDegree*. Executing the analysis results in a single number indicating the compliance degree.

Despite metrics the concept of *Views* is another important analysis approach. Views provide the architect with an excerpt of the architecture according to his current needs. The analysis approaches in this category often do not provide enough details for a concrete analysis specification and execution. There-

```
ChangeImpactConfiguration:
    ChangeImpactConfigurationByStereotpe | ChangeImpactConfigurationByEdgeClasses | StaticChangeImpactConfiguration
;

ChangeImpactConfigurationByStereotpe : {ChangeImpactConfigurationByStereotpe}
        ('WeakEffect In' '(' incomingWeakStereotypes +=EdgeTypeReference ( ',' incomingWeakStereotypes+=EdgeTypeReference)* ')')?
        ('MediumEffect In' '(' incomingMediumStereotypes +=EdgeTypeReference ( ',' incomingMediumStereotypes+=EdgeTypeReference)* ')')?
        ('StrongEffect In' '(' incomingStrongStereotypes +=EdgeTypeReference ( ',' incomingStrongStereotypes+=EdgeTypeReference)* ')')?
        ('WeakEffect Out' '(' outgoingWeakStereotypes +=EdgeTypeReference ( ',' outgoingWeakStereotypes+=EdgeTypeReference)* ')')?
        ('MediumEffect Out' '(' outgoingMediumStereotypes +=EdgeTypeReference ( ',' outgoingMediumStereotypes+=EdgeTypeReference)* ')')?
        ('StrongEffect Out' '(' outgoingStrongStereotypes +=EdgeTypeReference ( ',' outgoingStrongStereotypes+=EdgeTypeReference)* ')')?
;
```

Figure 4: Change impact configuration rule.

```
Template BusinessApplicationTechnologyStandardsCompliance {
    "Measurement of the compliance degree of business
    applications to technology standards."
    Result Attributes ratio
    as Aggregate Metric
    defined with calculation rule complianceDegree :=
        (COUNT( nodeType:"Business Application" AND
                having relation to (nodeType:"Technology Standard")))
            /
        (COUNT( nodeType:"Business Application"));
}
```

Figure 5: Example KPI definition.

fore we employ the viewpoint definitions included in the ArchiMate specification although they are not categorized as analysis. Therein we chose the Application Structure Viewpoint to illustrate the definition of views in Arla (The Open Group, 2012). This view contains elements with the type *Application interface*, *Application component*, *Application collaboration* and *Data object* as well as dependencies between them with the types *aggregation*, *composition*, *used by*, *access* and *specialization* (see figure 6).

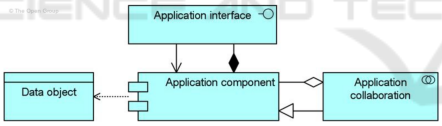

Figure 6: ArchiMate Application Structure Viewpoint (The Open Group, 2012).

Views are defined in Arla as Scope Analysis. There are two different possibilities: The first one is the definition of a node set using conditions and restrictions of node types. This possibility is shown in figure 7. The rule restricts the architecture to elements having one of the four given types.

```
Template ApplicationStructureViewpoint {
    "The Application Structure viewpoint shows the structure of
    one or more applications or components. This viewpoint is
    useful in designing or understanding the main structure of
    applications or components and the associated data"
    Result Attributes viewset
    as Aggregate Modelelementset
    defined with set definition viewpointDefinition :=
        nodeType:"Application component"  OR
        nodeType:"Application interface" OR
        nodeType: "Data object" OR
        nodeType: "Application collaboration"
}
```

Figure 7: Viewpoint definition using node conditions and restrictions.

In the second possibility conditions about the edges are specified (see figure 8). Thereby for each edge class it is specified, whether the edges should be considered one time (single), in a transitive way or not at all. This analysis is executed for a specific architecture element, for example a concrete application component. The elements for the view are then calculated according to the given scope configuration. I.e. in this case all direct and indirect nested components are added to result model part according to the transitive definition of the structural dependency class. Additionally all provided interfaces and data objects of these components as well as the used components are added.

```
Template ApplicationStructureViewpoint2{
    "Application structure viewpoint for one application"
    Result Attributes viewset2
    as Aggregate Modelelementset
    defined with scope configuration applicationStructure {
        ApplicationStructureViewpointEdges  {
        ModelEdge                in: None        out: None
        BehavioralDependentOf    in: None        out: None
        ConsumedBy               in: Single      out: None
        LocalizedAt              in: None        out: None
        Provide                  in: None        out: Single
        StructuralDependentOf    in: Transitive  out: Transitive
        }
    }
}
```

Figure 8: Viewpoint definition with conditions for edge classes.

The intersection of both view definitions provides the ArchiMate Application Structure Viewpoint. The resulting architecture part contains only the given element and edge stereotypes as well as is restricted to the perspective of one element.

4 ANALYSIS EXECUTION PLATFORM

The execution environment consists of three main components. Figure 9 gives an overview of the structure. The analysis definition language Arla was already described in the previous section. The Executor is the essential component of the execution platform. It provides the interfaces to the analysis language and the model storage as well as contains the logic to con-

vert Arla into evaluable constructs. Therefore it has interfaces to the evaluation technologies. We choose SPARQL for structural requests and Data-flow Analysis (DFA) for behavioral requests and recursive definitions. SPARQL is a graph-based query language for RDF. The Data-flow Analysis formalism is a powerful method originating from the area of compiler construction that allows computing context-sensitive information based on declarative specifications. Since flow analysis relies on the principle of information propagation rather than fixed navigation statements, it is possible to anticipate a wide array of changes and adaptions to the underlying modeling language (Saad and Bauer, 2013; Saad and Bauer, 2011).

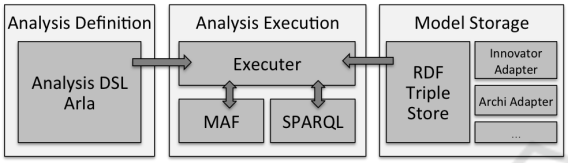

Figure 9: Execution architecture.

The model data is stored in a RDF Triple Store. For the data import we currently developed two adapters: An adapter for the modeling tool Innovator and an adapter for CSV files, which is exemplary used for loading models of the open source EA tool Archi. Further adapters for other tools can easily be added. The model storage and analysis execution is described in detail in the following sections.

4.1 Model Storage

The analysis platform utilizes a generic data schema for the storage of EA models, the Generic Meta Model (GMM). This meta model represents an EA model as a stereotyped graph with nodes and edges. Each element in the EA model is either a *ModelNode* or a *ModelEdge*. *ModelEdges* represent the relationships between *ModelNodes*. Both, nodes and edges, can have one or more *ModelProperty* representing attributes of them like a strategic impact. All three types have a relation to the respective stereotype. Possible stereotypes for *ModelNodes* are *Process* or *Application*. Example for *ModelEdges* are *used by* or *provides*. A stereotype is represented as *MetaModelNode* respectively *MetaModelEdgeConnection* and *MetaModelProperty*. A *ModelEdge* can be further specialized as *LocalizedAt*, *Realize*, *UsedBy*, *BehavioralDependentOf* or *StructuralDependentOf*. These subclasses of *ModelEdge* are used to categorize the individual stereotypes in an EA model and ease the reuse of analysis definitions. They are equivalent to the concept of *EdgeClasses* in Arla. For *ModelNodes* currently supported stereotype categories are *Process*, *Application* and *Infrastructure Component* (accordingly to the concept of *NodeClasses* in Arla). Further details about the GMM can be found in previous work (Langermeier et al., 2014b).

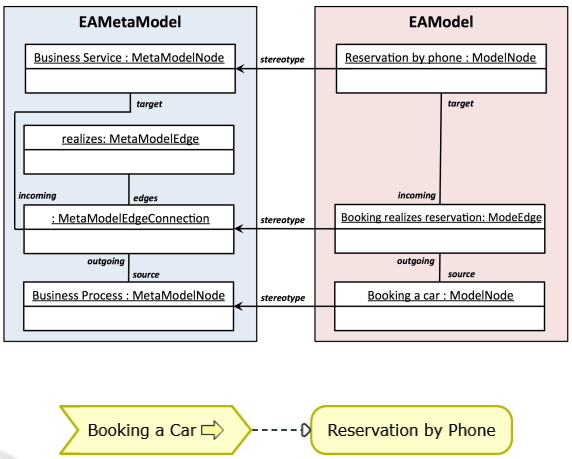

Figure 10: Example for the instantiation of the generic meta model (Langermeier et al., 2014b).

In order to run analyses on an EA Model, it has to be transformed into a GMM model. Due to the universality of GMM, this is possible for most of the EA models. Figure 10 shows the instantiation of the generic meta model. The example shows the provisioning of the business service *Reservation by phone* by the business process *Booking a car*. For an import we capture the syntax of the serialized EA model using an ANTLR[1] grammar. From this grammar ANTLR generates a parser that enables the creation and walk through of the parse tree. We utilize this parser to create an EMF model using the GMM scheme. This EMF model is then persisted in RDF using the Framework EMFTriple[2]. The whole model loading process is illustrated in figure 11.

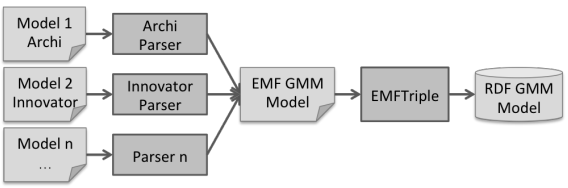

Figure 11: Model loading.

EMFTriple is a framework that provides a RDF binding for EMF. Therewith we were able to persist our EMF models in RDF instead of XMI. The framework provides also interfaces to RDF data stores. We

[1] http://www.antlr.org
[2] https://github.com/ghillairet/emftriple

choose the Triple Store from Apache Jena, the TDB[3], as data store. This component allows the storage and querying of RDF data via the Jena API.

4.2 Analysis Execution Process

In order to execute an analysis from an Arla file, the first step is to parse the analysis definition. We used the Xtext language infrastructure[4] for this task. Despite a model parser Xtext also provides a text editor with syntax highlighting and auto completion. Parsing an Arla file finally provides us an EMF model of the analysis definition. In a second the step the utilized template definition are processed. Each template is transformed into a specific Arla definition using Text2Text Transformation. Thereby only those analyses are transformed that are referenced in an *Adapted-Analysis*. Afterwards all required analysis definitions are available and processable. The following evaluation process is illustrated in figure 12.

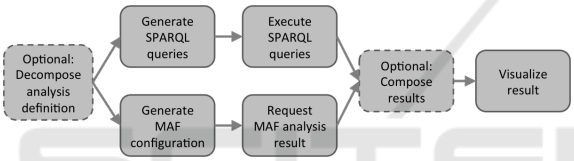

Figure 12: Workflow analysis execution.

In the case of an composite analysis or a metric the analysis definition is composed in its components in a first step. According to the chosen analysis configuration the evaluation is done through the creation and execution of a SPARQL query or through triggering a predefined data-flow analysis. Typically, before executing a DFA several parameters have to be set in a configuration file. The evaluation details of the different analysis types is described in the following:

- **Element or Aggregate Metric**: Decomposition in its components and evaluation of the atomic expressions. Expressions questioning a set of nodes or edges are evaluated through the creation and execution of SPARQL, a result reference triggers the execution of the referenced analysis, a SUM respectively COUNT triggers the execution of the contained expression for each addressed element and merges the results.
- **Node Set Condition**: Creation of a SPARQL query to retrieve the requested node set.
- **Scope**: Generate the configuration file and trigger the DFA.
- **Change Impact** Adapt the configuration parameters, generate configuration file and trigger the DFA.
- **Path Analysis** Adapt configuration and trigger the DFA.
- **Adapted Analysis** Execute the respective generated specific analysis.
- **Composed Analysis** Three different composition possibilities: Successive execution, result combination and within a calculation rule.
- **DFA, e.g. Performance Analysis** Triggers the execution of the respective static DFA (e.g. performance analysis)
- **Custom Query** Executes the explicit specified SPARQL query in the analysis definition

The successive execution is defined as: `Apply` *Analysis2* `on` *Analysis1*. This composition rule is realized while executing analysis 1 (A1) first. The result of A1 is written back into the model. Then analysis 2 (A2) is executed on this new model. The result of analysis 2 is the final result. This execution process is illustrated in figure 13a.

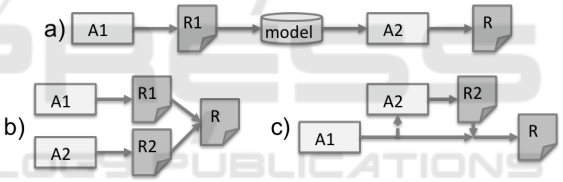

Figure 13: Arla composition rules.

In figure 13b the possibility of result combination is presented. Thereby the two analyses A1 and A2 can be executed independently from each other. The retrieved results, R1 and R2, are combined afterwards to create the final result R. Therefore different combination possibilities exist: In the default mode the results are combined according to the union semantics. It is only possible to combine results of the same type. Additionally converting the path set into a node set allows the combination with a node set. The combination of a element result and a aggregated result is only possible, if the single element results can be summarized to one result, i.e. union for sets. Additionally node set results can be composed using the operators `intersection` and `difference`. The operators are implemented according to set theory. Also in this case a path result is transformed into a node set result and element result are aggregated to one set.

At least analyses can be composed using a calculation rule. Therefore references to result attributes of other analyses can be used within a calculation rule. At execution time the first analysis is evaluated and

[3]http://jena.apache.org/documentation/tdb/index.html
[4]http://www.eclipse.org/Xtext/

within this process the second analysis execution is triggered. The process is illustrated in figure 13c.

The analysis composition is an important mean to deal with incomplete models. Before executing a specific analysis, the model can be restricted to the relevant part of the architecture that is sufficiently described. For example performance parameters are not available for all servers. Before executing a performance analysis, those servers with their dependencies can be excluded.

If it is not possible to implement an analysis in Arla, for example a recursive definition in a metric, the language provides an adapter to DFA. Instead of defining the execution rules for an analysis, the location of the DFA configuration and the executed strategy is defined. Based on this information the predefined DFA is executed and the results are processed. Additionally it is also possible to define a new analysis type in Arla. In this case the user does not have to deal with the concrete DFA configuration path and strategy name. Currently this is realized for the performance analysis of (Jonkers and Iacob, 2009). Due to the recursive definition it was not possible to realize the analysis solely in Arla. Therefore we implemented it as data-flow analysis (see previous work (Langermeier et al., 2014b)) and defined an analysis type *PerformanceAnalysis* in Arla. A further possibility to extend the scope of Arla is the explicit definition of SPARQL queries in an analysis definition. Therefore deep knowledge of the RDF graph is required, but it provides access to the full expressive power of SPARQL.

The final result of an analysis execution is visualized using Graphviz. Depending on the result types different options exist for the visualization: Node sets are visualized as the respective part of the whole model. Analyses concluding in attribute values can be visualized using colors or using text annotations in the model. Numeric results are visualized as annotations in the model. Path results are visualized as several model excerpts, each representing one path.

5 EVALUATION

For evaluation purposes we specified an analysis for each Arla analysis type and executed them within three use cases. These are a small, fictional model describing a car rental station (also used in (Langermeier et al., 2014a)), the EA model of a medium-sized software product house and the EA model of a medium-sized manufacturing company. In section 3 we already presented the template definitions for a change impact analysis, an KPI calculation and a view definition. The definition were executed in the three use case by adapting them to the specific EA models. Figure 14 shows the adapted analysis definition to calculate the application structure viewpoint for the manufacturing use case.

```
Analysis ApplicationStructureViewpoint4CarRental {
    "Adaption of the Application Structure Viewpoint
     to the manufacturing case"
    Result Attributes viewpoint
    as Aggregate Modelelementset
    adapt viewpointDefinition {
        map "Application collaboration"
            to node:"ApplicationFunction" ["ApplicationFunction"]
        map "Application component" to nodeClass:Application
        map "Application interface"
            to node:"ApplicationService" ["ApplicationService"]
        map "Data object"
            to node:"BusinessObject" ["BusinessObject"]
    }
}
```

Figure 14: Adapted analysis definition to calculate the application structure viewpoint.

Additionally we also executed a path analysis and the performance analysis on these use cases. Figure 15 shows the result of the realizing paths analysis applied to the process *Take back car* in the rental car model. The full model can be found in (Langermeier et al., 2014a). The result visualize two paths that represent two different IT realizations of the process. In this case the difference between the two paths is not on the application layer but in the infrastructure layer.

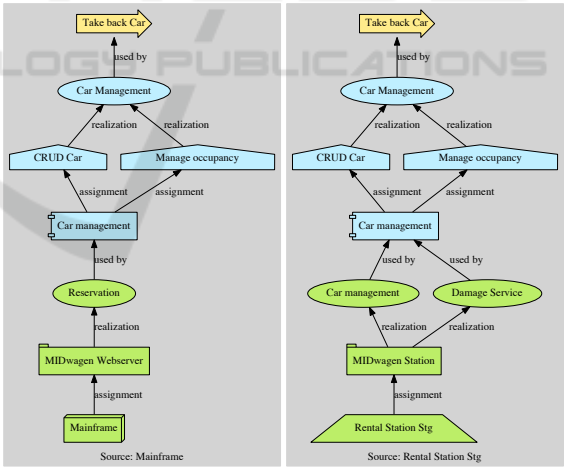

Figure 15: Realizing paths for process *Take back car*.

We were able to execute all specified analysis definitions in the three different use cases. If the EA models contain the required information, for example if applications comply to a technology standard, we retrieve meaningful results. The templates could be executed in all use cases after the definition of the corresponding Adapted Analysis.

Additionally we evaluated the language and its execution possibilities through an examination of

the current work in EA analysis. Thereby analysis approaches relying on expert interviews (e.g. (Della Bordella et al., 2011; Plessius et al., 2012) or approaches addressing the method EAM like maturity analyses (e.g. (Aier et al., 2011)) are not considered in our evaluation. According to these constraints the approaches summarized by the following analysis types are not included in the evaluation: Business Entity Analysis, Design Analysis, Run-time Analysis, Intentional Analysis, Maturity Analysis and Sensitivity Analysis. For the remaining approaches, we decided whether they can be completely described and evaluated in Arla (i.e. they are covered by Arla), partially described and evaluated with Arla or whether Arla does not cover them. For example a partial coverage is given for approaches considering dates. Currently Arla does not comprise special expressions for evaluating time spans or point of times. Indirectly it is possible by utilizing other expression types. Some analyses in the technical category PRM (Närman et al., 2014) are also only partially covered. The probabilistic aspects of these analyses, which are implemented in a Monte Carlo fashion, cannot be reproduced with Arla. By replacing the probability distributions with single values, the analyses can be defined and evaluated. An analysis approach is also declared as partially covered, if it is necessary to specify new DFA implementations or create custom SPARQL queries for its realization.

We aggregated the results of our evaluation using the analysis types as well as the technical and functional categorization. We refer to the types and categories instead of the single approaches, since it is not obvious to decide whether a following publication of an approach is a new one or an extension to an existing one. Hence, referring to the concrete approaches would provide a biased view on the coverage of Arla.

From the 33 analysis types 21 are covered by Arla and additional 7 types are at least partially covered. A total of 6 analysis types are not covered, that is 18%. We say that Arla covers an analysis type, if there exists at least one analysis approach that is assigned to this analysis type and is covered by Arla. If Arla covers none of the approaches but there is at least one approach that is partially covered, then the analysis type is partially covered. Otherwise Arla does not cover the analysis type.

To further evaluate the coverage of Arla we analyzed the results for the technical (figure 16) and functional (figure 17) categorization of the approaches. Most of the technical categories are covered or partially covered by Arla. Only the categories containing probabilistic approaches are not covered. These are *Bayesian Networks*, *Probabilistic Relational Model* and *Extended Influence Diagrams*.

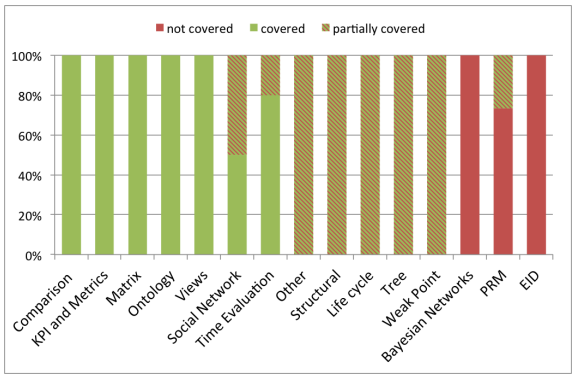

Figure 16: Coverage technical categories.

Concluding, the functional category *System* is not covered by Arla, since the approaches of this category are all assigned to one of these three technical categories. For all other categories there exist approaches that are covered or at least partially covered (*Data*) by Arla.

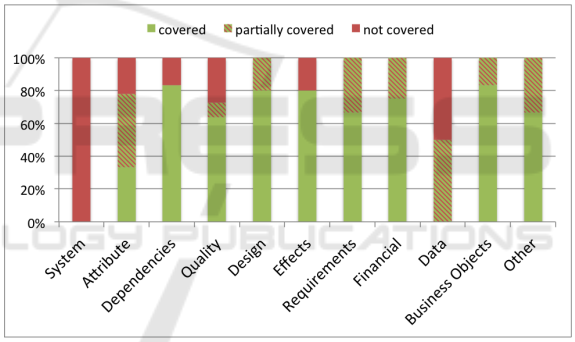

Figure 17: Coverage functional categories.

The overall coverage of Arla regarding existing EA analyses is quite high. The major weakness is the missing support for probabilistic approaches, all other categories are covered by Arla. The implementation within the case studies shows the applicability of the language. The template mechanism in order to enable reuse and predefined analyses supported the analysis execution within the case studies very well. Currently we made no usability tests with architects regarding the ease of use. From our experiences in the case studies we identified several improvements for Arla. If possible, they are already implemented. For example a direct support of dates and date expression is planned for Arla in future work.

For executing the analyses we utilize a combination of SPARQL and data flow based analysis. Both approaches are successfully applied in large application scenarios and thus provide a scalable technical foundation for our analysis execution. The graph

based query language SPARQL provides features to answer structural questions and extract model parts. DFA is perfectly qualified to answer behavioral questions, execute recursive analysis definitions and deal with cyclic dependencies. It enables a forward and also backward traversing of the model. Combining both techniques enables the execution of complex analyses and provides means to deal with an incomplete model.

The GMM enables a tool and EA meta model independent execution environment for the analyses. The adaption process for a specific EA initiative is kept as simple as possible: The concept of edge classes and node classes provides a mechanism to define generic analyses and execute them with small effort on a specific model. Only during import definition the mapping of stereotype to the classes has to be made once. Additionally, if more detailed semantics about the nodes an edges are required, the analysis templates can use stereotype variables. These variables have to be mapped to concrete EA model stereotypes before execution. Finally the language Arla abstracts from all the technical details. The concrete execution procedures are generated form the Arla definition at runtime.

6 CONCLUSION

In this paper we presented a language for the definition of Enterprise Architecture analyses as well as an execution environment for their evaluation. The language (Arla) provides a universal interface to EA analyses since it abstracts from the technical details of the execution. Thereby Arla overcomes current weaknesses of EA analyses like the isolation of existing approaches or the difficulties in order to adapt them to a specific EA initiative. We integrated different kinds of EA analysis regarding their goals and addressed topics as well a regarding their technical execution. Additionally Arla contains a composition mechanism, which enables the definition of complex analyses by reusing simpler ones. The language with its execution environment provides a foundation for an EA analysis library. Through the possibility of template definition as well as the independence from the meta model, generic analyses can be defined and easily adapted in a specific context. Thereby architects can profit from reuse and experiences from other projects.

The utilization of SPARQL and DFA for analysis execution enables the high coverage of Arla regarding the technical and functional analysis categories. The combination of both techniques in one framework provides means to deal with incomplete models as well as facilitate a high expressiveness and variability for analysis definitions. The presented language Arla abstracts from all those technical details and provides the architect a simple interface to analysis activities.

In future work the scope of Arla will be further extended. Since probabilistic analyses are currently not directly supported, we will elaborate ways to integrate them. Additionally the change impact analysis can be generalized in order to support universal "if-then" analyses based on attributes. Such an analysis type can be used to define failure impact analyses or availability analyses. Another point is the integration of feasibility analyses before the analysis execution. This provides an automatic way to deal with incomplete models and supports the architect in the interpretation of results. Finally we will make use of further RDF features like reasoning in order to deal with indirect relationships.

REFERENCES

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

Bucher, T., Fischer, R., Kurpjuweit, S., and Winter, R. (2006). Analysis and application scenarios of enterprise architecture: An exploratory study. In *10th IEEE International Enterprise Distributed Object Computing Conference Workshops, EDOCW'06*, pages 28–28. IEEE.

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

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.

Frank, U., Heise, D., and Kattenstroth, H. (2009). Use of a domain specific modeling language for realizing versatile dashboards. In *Proceedings of the 9th OOPSLA workshop on domain-specific modeling (DSM)*.

Franke, U., Flores, W. R., and Johnson, P. (2009). Enterprise architecture dependency analysis using fault trees and Bayesian networks. In *Proceedings of the 2009 Spring Simulation Multiconference*, SpringSim '09, pages 55:1–55:8, San Diego, CA, USA.

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

Jonkers, H. and Iacob, M.-E. (2009). Performance and cost analysis of service-oriented enterprise architec-

tures. *Global Implications of Modern Enterprise Information Systems: Technologies and Applications, IGI Global*.

Kurpjuweit, S. and Aier, S. (2009). Ein allgemeiner Ansatz zur Ableitung von Abhängigkeitsanalysen auf Unternehmensarchitekturmodellen. In *Wirtschaftinformatik Proceedings 2009*.

Langermeier, M., Saad, C., and Bauer, B. (2014a). Adaptive approach for impact analysis in enterprise architectures. In *Business Modeling and Software Design*, pages 22–42. Springer.

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

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

Lantow, B., Jugel, D., Wißotzki, M., Lehmann, B., Zimmermann, O., and Sandkuhl, K. (2016). 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. Springer International Publishing.

Matthes, F., Monahov, I., Schneider, A., and Schulz, C. (2011). EAM KPI Catalog. Technical Report v 1.0, Technical University Munich.

Naranjo, D., Sánchez, M., and Villalobos, J. (2014). Towards a unified and modular approach for visual analysis of enterprise models. In *Enterprise Distributed Object Computing Conference Workshops and Demonstrations (EDOCW), 2014 IEEE 18th International*, pages 77–86. IEEE.

Naranjo, D., Sánchez, M., and Villalobos, J. (2015). Primrose: A graph-based approach for enterprise architecture analysis. In *Enterprise Information Systems: 16th International Conference, ICEIS 2014, Lisbon, Portugal, April 27-30, 2014, Revised Selected Papers*, pages 434–452. Springer International Publishing.

Närman, P., Buschle, M., and Ekstedt, M. (2014). An enterprise architecture framework for multi-attribute information systems analysis. *Software & Systems Modeling*, 13(3):1085–1116.

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

Plessius, H., Slot, R., and Pruijt, L. (2012). On the categorization and measurability of enterprise architecture benefits with the enterprise architecture value framework. In *Trends in Enterprise Architecture Research and Practice-Driven Research on Enterprise Transformation*, pages 79–92. Springer.

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

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

Rauscher, J., Langermeier, M., and Bauer, B. (2017). Classification and definition of an enterprise architecture analyses language. In *Business Modeling and Software Design, 6th Int. Symposium*, LNBIP. Springer. Accepted for publication.

Saad, C. and Bauer, B. (2011). The Model Analysis Framework - An IDE for Static Model Analysis. In *Proceedings of the Industry Track of Software Language Engineering (ITSLE) in the context of the 4th International Conference on Software Language Engineering (SLE'11)*.

Saad, C. and Bauer, B. (2013). Data-flow based Model Analysis and its Applications. In *Proceedings of the 16th International Conference on Model Driven Engineering Languages and Systems (MoDELS'13)*.

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

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

The Open Group (2012). ArchiMate 2.1. Technical Report Open Group Standard.