

A unified framework for enterprise architecture analysis

Melanie Langermeier, Christian Saad, Bernhard Bauer

Angaben zur Veröffentlichung / Publication details:

Langermeier, Melanie, Christian Saad, and Bernhard Bauer. 2014. "A unified framework for enterprise architecture analysis." In *Proceedings of the 2014 IEEE 18th International Enterprise Distributed Object Computing Conference Workshops and Demonstrations, 1-2 September 2014, Ulm, Germany*, edited by Georg Grossmann, Sylvain Hallé, Dimka Karastoyanova, Manfred Reichert, and Stefanie Rinderle-Ma, 227–36. Los Alamitos, CA: IEEE. <https://doi.org/10.1109/edocw.2014.42>.



A unified framework for Enterprise Architecture analysis

Melanie Langermeier
Software Methodologies for
Distributed Systems
University Augsburg
Email: langermeier@ds-lab.org

Christian Saad
Software Methodologies for
Distributed Systems
University Augsburg
Email: saad@ds-lab.org

Bernhard Bauer
Software Methodologies for
Distributed Systems
University Augsburg
Email: bauer@ds-lab.org

Abstract—Enterprise Architecture (EA) analysis is an important tool for leveraging EA models. However, due to the diverse nature of the EA field, analysis techniques must be able to deal with variability caused by different modeling standards as well as their adaption to specific organizational needs. Additionally, the definitions of the relevant measures often vary between different stakeholders and organizations. To address these challenges we propose a (meta) model independent framework for the analysis of architecture models. Based on a generic representation of architectural data, we employ a data-flow based analysis approach to enable a context-sensitive evaluation of organization specific measures. We demonstrate the generic applicability of this framework through a re-implementation of three different analyses from literature.

I. INTRODUCTION

Today's IT landscapes in organizations are getting more complex and affect more and more business domains. The dependencies between the IT layer and the business layer are difficult to grasp. Enterprise Architecture (EA) supports an organization in getting a clear understanding of the essential business and IT elements [1]. It is a holistic approach covering all layers in an organization, thereby enabling not only local optimization within one domain.

Several EA frameworks (i.e. Zachman [2], TOGAF [3], DoDAF in the military domain [4], RM-ODP for open distributed systems [5], the FEEF for US federal agencies and other governmental agencies [6] or the 4+1 View model of Architecture [7]) have been established, defining elements and layers of an enterprise architecture. Such EA meta models consist merely of concepts and relationships between these concepts (i.e. nodes and edges). In practice, the organizational business and IT landscapes are often very diverse. As a consequence, existing (EA) frameworks have to be adapted to each organization's specific needs. It is obvious, that this can easily lead to problems when applying techniques that aim to extract information from enterprise models. On the one hand, there exists a wide variety of different modeling standards for the description of EA systems, that in turn may be modified for different application scenarios. Analyses of EA models that rely on a specific meta model, therefore have to be adapted for each application. On the other hand, many analysis techniques only provide a limited set of features, that may be suitable only for a certain set of use cases.

Nevertheless, EA analysis is an important issue, when trying to get benefit from the established enterprise models.

To support decision making, the as-is architecture, as well as planned future architectures can be assessed [8]. One technique is the quantification of the models according to defined measures like cost or benefit. The values of such measures can be used to indicate weaknesses in the current model, as well as compare different planned future architectures to the current one as well as to each other. Such a global assessment on the overall architecture provides a valuable input for making decision, especially in an early design stage [9]. Apart from cost and benefit further measures can be properties like business IT-alignment and security [10].

Despite the benefits that analyses bring to enterprise models, this concept is not very common in current architecture practice [10]. In contrast, a lot has been done for how to represent an architecture model best, but less for using these models for analysis purposes [11]. One drawback for a widespread use of analysis techniques is certainly the lack of general and easy adaptive methods. Existing approaches often hardly rely on a specific analysis meta model [10] or require a model normalization before execution [15]. Both cases require a transformation of the original EA model respectively a meta model mapping to be able to perform the analysis. Since the techniques used for execution rely on the meta model, routines for handling missing data are difficult to integrate. Additionally regarding current analysis approaches, a quantitative one, considering all architecture layers, is hard to find [9]. On a technical level, the Object Constraint Language (OCL) enables the specification of constraints on modeling languages. However, it suffers from several shortcomings which render it unsuitable for the intended purposes. More specifically, the presented methodology relies on the principles of information propagation and fixed-point computation, both of which are not supported by OCL.

As a solution to these problems, we propose the usage of a combination of two techniques: We employ a very generic meta model, that is able to act as a universal interchange format for EA data, specified in an arbitrary modeling language (section III-A). Based on this unified representation, we are then able to implement different kinds of static analyses (section III-C). For this purpose, we employ the Data-flow Analysis formalism, a powerful method originating from the area of compiler construction, that allows the computation of context-sensitive information based on declarative specifications (section III-B). Since flow analysis relies on the principle of information propagation rather than fixed navigation state-

ments, it is possible to anticipate a wide array of changes and adaptations to the underlying modeling language. In a sense, this method can be interpreted as an extension of the capabilities of traditional model-based techniques such as OCL.

Our approach enables us to combine semantic-less graph analysis like reachability or shortest paths with a semantic-full determination of quantitative measures like cost or performance. The declarative specification of information propagation enables the execution of quantitative analyses, without adapting the EA meta model with analysis specific concepts. Nevertheless the analysis specifications have to be adapted, if semantic knowledge i.e. names of specific concepts, is used.

We demonstrate the viability of this framework, as well as its generic applicability in the context of three existing analyses described in the literature which have been re-implemented based on the presented techniques (section IV). As a result, all of these analyses are not only built on the same underlying technology, but they also prove to be very robust towards changes to the employed EA language.

II. QUANTITATIVE EA ANALYSIS

Enterprise architecture analysis can be defined as the "application of property assessment criteria on enterprise architecture models" [12]. Whereas architectural models "provide only visual and qualitative support" [13], the use of analysis techniques enables the quantification of these models. Especially in case of appearing changes, such quantified measures enable the comparison of different alternatives and measurement of the efficiency of investments [14]. But it is still a challenge for an organization to quantify the information in the models and make the value or cost of a change visible [14]. EA analysis does not encompass only one domain, quite the contrary, the greatest benefit will be reached if a method considers all architectural domains. E.g. architectural layers are dependent on each other in a manner that higher layers impose workload on lower layers and vice versa, the performance of the lower layers (i.e. server capacity) has influence on the higher layers (i.e. availability of a service) [9].

Närman et al. [10] define a process for EA analysis in the context of decision making. The five steps towards a decision are: (1) Define scenarios, (2) Determine properties of interest, (3) Modeling scenarios using a meta model, (4) Analyze the scenario properties, and (5) Make a decision. In this work we focus on step four, how to analyze a given enterprise model with a given measure for the property.

Buckl et al. [8] compared current analysis approaches according to the categorization schema they developed. Every characteristic of the classification is covered by at least one approach (except multi-level self-referentiality). They identified further work to establish approaches that cover multiple characteristics, for example through an integration of existing ones. As example they propose a combination of rule based analysis with indicator based analysis. A rule based analysis refers to certain architectural constellations, that should be present or absent. An indicator based analysis is used to quantify the model using architectural properties [8].

When regarding the assessment of specific quantitative measurement, i.e. analyses for one specific use case a lot of

work has to be done. Jonkers and Iacob describe a performance and cost analysis through a top-down calculation and propagation of the workloads from the top layer, followed by a bottom-up calculation and propagation of the performance measures [15]. Their analysis is defined with and following limited to the use of ArchiMate. The authors also do not propose an implementation to integrate the analysis in a tool. In [16] a meta model for enterprise systems modifiability analysis is presented, i.e. assessing the cost of making changes to enterprise-wide systems. In their work probabilistic relational models (PRM) are used for the formalization. Furthermore, Närmann et al. present a framework with four viewpoints for information system analysis [10]. These are application usage, data accuracy, service availability and service response time. For each viewpoint they established a meta model and p-OCL code to execute the analysis. Each approach describes merely the calculation of one specific measurement. Additionally they are restricted to the specific underlying meta model. Using them with another meta model in another context requires adaption. None of these approaches proposes a technique how to define a specific analysis.

For the definition of analyses, techniques like XML [17], SPARQL [18], extended influence diagrams [19], probabilistic relational models [20], p-OCL [21] or architecture theory diagrams [22] have been proposed in current literature. These techniques are dependent on the underlying meta model or schema, an adaption of defined analyses to another context or due to changes requires much effort.

To cope with the variability of meta models, Jonkers and Iacob propose a differentiation between design space and analysis space. The design space is expressed by using languages like UML, business process modeling languages or architectural description languages. The analysis space is expressed by using special-purpose languages, which enable the later analysis. Following their method to perform analysis includes a model transformation, then the execution of the analysis with calculation of the properties, and finally a reverse transformation back to the design space (with the results of the analysis) [15].

III. UNIFIED FRAMEWORK FOR EA ANALYSIS

Analyses are typically dependent on the meta model, since the techniques, commonly used for implementation of them, are closely related to the meta model (cf. section II). If an organization wants to execute a specific analysis, they must use the respective analysis meta model (or define a meta model mapping). In the following we describe techniques and how they work together to enable meta model independent analyses. The application of the developed techniques will be demonstrated in the context of the *MIDWagen* example (figure 1), a model that is shipped with MID's *Innovator for Enterprise Architects* tool [23]. The *MIDWagen* model specifies the organizational structure of a car rental company, consisting of business processes, applications and the underlying infrastructure along with their respective services.

The analysis framework relies on the combination of two techniques: (1) A generic meta model which provides a unified representation for enterprise architecture data and (2) a method that allows the extraction of context-sensitive

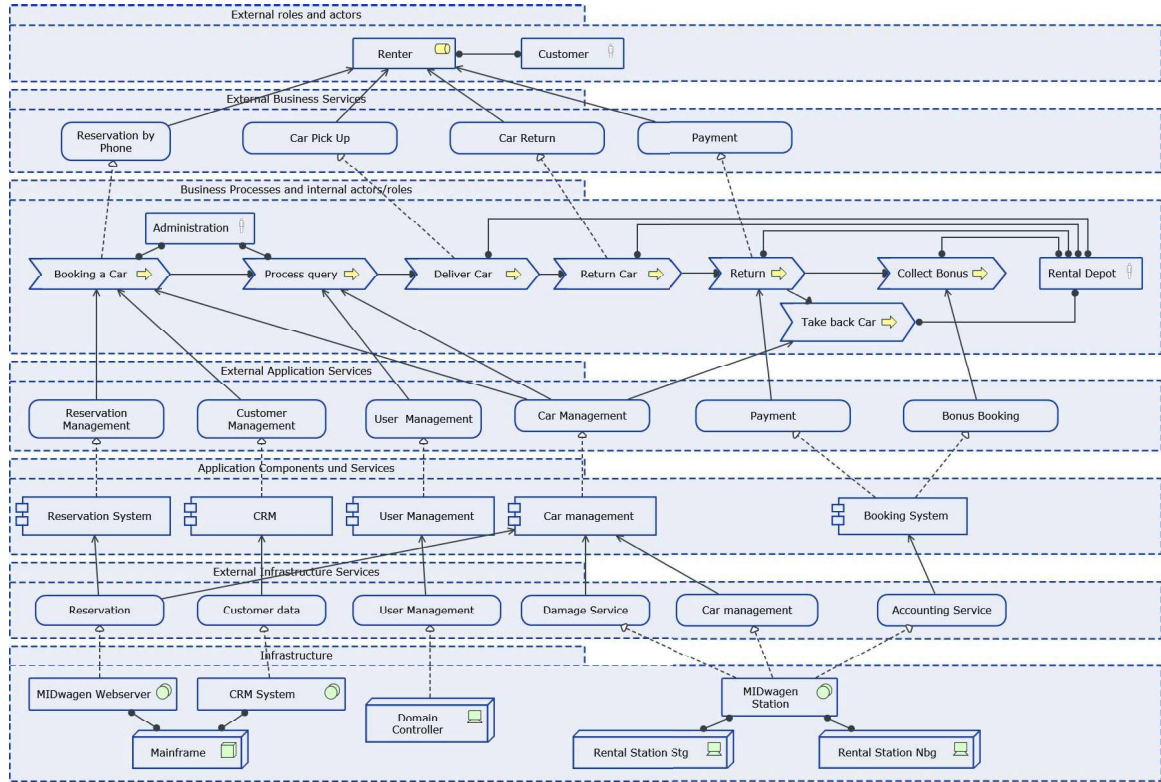


Fig. 1. Archimate model of the MIDWagen Use Case [23]

information from models. The former implements a generic data structure on which the analyses can be defined. To fulfill this requirement, the generic meta model - described in detail in section III-A - must be abstract enough to accommodate mappings to arbitrary architectural modeling formats while, at the same time preserving the relevant meta information. The chosen method for model analyses is inspired by the Data-flow Analysis (DFA) approach commonly employed in the field of compiler construction to derive semantic information from a program's control-flow graph. In section III-B, we summarize the approach proposed in [24] which adapts the DFA technique for use in the modeling domain. Finally, in section III-C, we show how these two methods can be combined to realize a universal framework for enterprise architecture analysis.

A. Generic meta model

To derive meaningful information from a model, an analysis must incorporate knowledge about the semantics of the language constructs. In general, the structure of any modeling language therefore has a significant impact on the way analyses are implemented and executed. This can present a problem in areas such as enterprise architecture modeling, where a large number of competing standards and practices exist. To circumvent this complication, we introduce a generic meta model (GMM) which acts as a universal interchange format for EA models, and provides a common ground for analysis specifications.

As mentioned before, analyses heavily rely on semantic

information since the evaluation of a model element usually depends on its meta model type. The GMM therefore not only has to encode the actual model data but also the related meta information, i.e. the classes and associations of the respective meta model, and establish proper *«instanceof»* relationships between both artifacts. As a consequence, each instance of the GMM conforms to a representation of the target EA model as well as the EA language itself.

This approach has to benefits: On the one hand, the established EA meta models includes only those concepts, that are actually used in the EA model. On the other hand, it enables to define any analysis over the EA, without adapting the EA meta model. All required adaptations can be covered through an adequate analysis specification (c.f. section III-C)

The depiction of the GMM shown in figure 2 aligns the concepts of the meta model according to their responsibilities. The root element *EAModelContainer* contains an *EAMetaModel*, an *EAModel* and a *Configuration* element, each of which acts as a container for the elements of the respective artifact type. The GMM defines several additional types such as *NamedElement* which have been omitted in this illustration for reasons of clarity.

The layout of the proposed format conforms to a directed graph in which each node may possess an arbitrary amount of data fields (named properties). For this purpose, each node, edge and property type defined in the *EAMetaModel* specializes the abstract *MetaModelStereotype* class. On the other hand, each instance of a node, edge or property in the *EAModel*

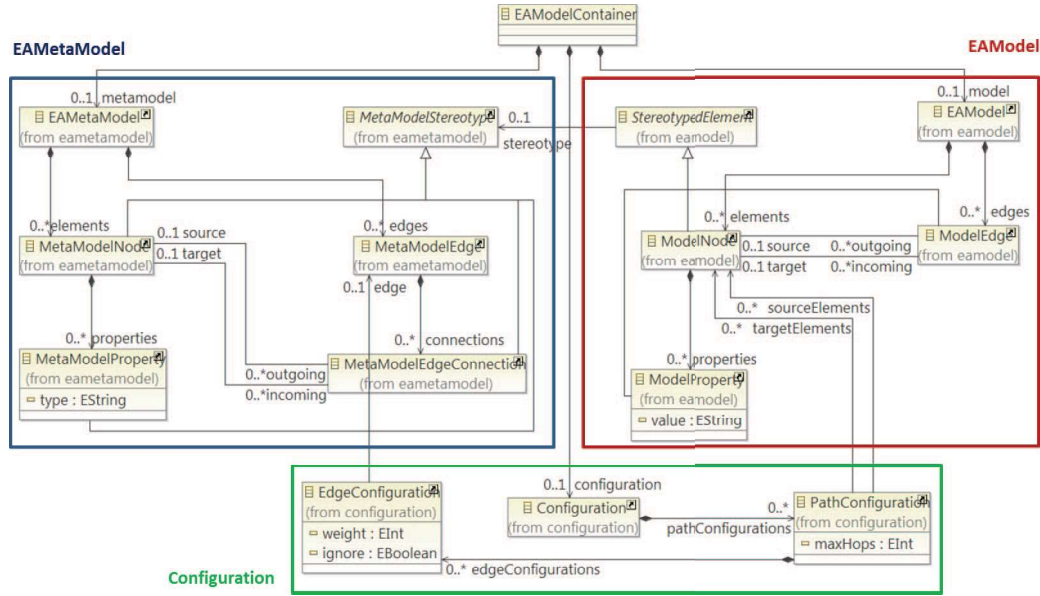


Fig. 2. Meta model for generic measure calculation

is a *StereotypedElement* which references a stereotype from *EAMetaModel*.

In order to map the structure of an existing EA modeling language to the GMM format, the defined classes have to be translated to *MetaModelNodes* while their associated class attributes become *MetaModelProperties*. Edge types, on the other hand, require a mapping to two concepts: *MetaModelEdge* and *MetaModelEdgeConnection*. This distinction is motivated by the fact that the specifications of various EA standards may conflict with restrictions imposed by the current state of modeling technology. In the Unified Modeling Language (UML) for example, it is not possible for a class to declare two incoming or outgoing associations with the same name. This usually requires to implement workarounds such as assigning a unique numeric index to each association. The downside to this approach is the loss of semantic information as associations of the same type but with different indices are treated as separate entities by modeling tools. The definition of a *MetaModelEdge* type that can be shared by multiple *MetaModelEdgeConnections* solves this problem.

As mentioned above, the *EAModel* as part of the GMM, specifies the structure for instances of the type declarations in the *EAMetaModel*. More specifically, it defines *ModelNodes* along with *ModelProperties* (which possess a *value* field that accommodates the property's data) and *ModelEdges* which establish connections between *ModelNodes*.

The last section of the GMM consists of the *Configuration* which provides the possibility to encode analysis-specific data in the model. The contained elements can be adapted to suite the needs of a particular use case. Here, the *Configuration* is used to restrict the computation of paths between model elements to paths of a specific length (*PathConfiguration*→*maxHops*) and to ignore transitions of certain types (*EdgeConfiguration*→*ignore*). An advantage of this approach is that the analysis configuration is directly tied to the spec-

ific elements of *EAMetaModel* and *EAModel* and can therefore be easily processed by the analyzer.

Figure 3 shows an excerpt of the *MIDWagen* use case to illustrate the use of the GMM.

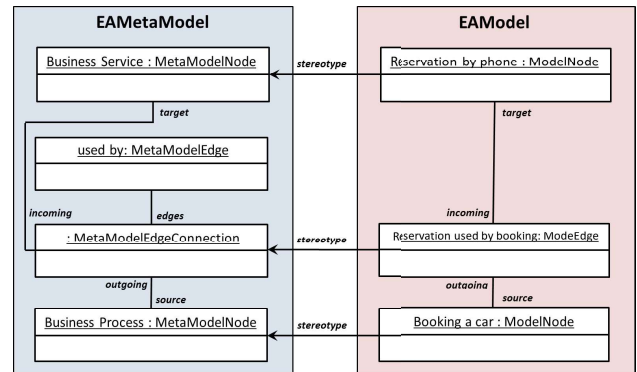


Fig. 3. Example for the instantiation of the generic meta model.

The left hand side shows the type definitions of *EAMetaModel* while the right hand side contains the model data.

To execute an analysis on a GMM instance, the (meta) model must first be translated. This requires the implementation of an adapter for each language (or EA tool). The simplest way to accomplish this task consists of a traversal of the source model, creating corresponding GMM nodes, properties and edges on-the-fly. As part of this process, the meta information of the element has to be evaluated as well, and the *EAMetaModel* has to be extended or updated accordingly. This approach ensures that all relevant information is transferred into the GMM representation while unused parts of the original EA language are automatically excluded.

Figure 4 shows a refined version of the meta model that

has been extracted for the *MIDWagen* model depicted in 1. In contrast to the Archimate meta model, it only contains a subset of EA classes and relationships, namely the concepts which are actively used in the example.

In conclusion, we can state that this approach can be applied to all architectural models that are defined in an object oriented manner, i.e. models which distinguish between type definitions and instances.

B. Data-flow based Model Analysis

Data-flow Analysis (DFA, [25]) is a technique that originates from the area of compiler construction. During the process of translating source code into machine code, a variety of validation and optimization steps are applied to the program. DFA is commonly used to statically derive optimizations based on the structural composition of program instructions. By examining each basic block in its overall context, it is possible to derive information that holds true for each possible execution of a program. Canonical examples include the calculation of reaching definitions and variable liveness.

For this purpose, the program is usually represented as a control-flow graph with the nodes conforming to the basic blocks and the edges denoting the flow of control. Subsequently, a set of data-flow equations is computed at each node. The output of this function not only relies on the contents of the respective node itself but also - recursively - on the results computed at its predecessors. As a consequence, results are propagated along the graph's edges and the evaluation of each node incorporates the context of its predecessors. Since the presence of loops leads to an infinite number of execution paths (and consequently cyclic DFA equation systems), fixed-point evaluation semantics are employed to approximate the - otherwise undecidable - runtime behavior of programs.

In [24] we discussed how this analysis technique can be adapted to the modeling domain, resulting in a "generic 'programming language' for context-sensitive model analysis". The approach defines a declarative specification language that allows the annotation of data-flow attributes at meta model classes. These attributes can subsequently be instantiated and evaluated for arbitrary models. This technique provides two significant advantages: Since information can be propagated along model edges, each model element can be evaluated in its overall context thus eliminating the need for static navigational expressions as are common in languages such as OCL. This is an important benefit in the EAM domain where both the structure of meta models and models is highly dynamic. Secondly, the usage of fixed-point semantics allows the implementation of a correct handling of cyclic paths.

The application of flow-based model analysis can be illustrated in the context of a simple example: We assume the existence of a simple meta model for control-flow graphs that specifies the class *node* along with two specializations *startnode* and *endnode* as well as a class *edge*. A Data-flow Analysis can easily determine whether a node is reachable from the *startnode*. For this purpose we assign a data-flow attribute *is_reachable* of type boolean to the *node* class (and thereby implicitly to its sub-classes). We then define two data-flow equations for *is_reachable*. Since, by definition, a *startnode* is always reachable, its equation always returns *true*.

The second equation is used for the other node types and only returns *true* if the value computed for *is_reachable* at one of its direct predecessors is *true*. The instantiation of this analysis for a specific model attaches an instance of the *is_reachable* attribute to each *node*. The DFA solver is then responsible for determining the dependencies between the attribute instances and executing the data-flow equations in a valid order.

A reference implementation of this approach exists in the form of the Model Analysis Framework (MAF, [26]) which has been successfully employed to specify and carry out analyses in different domains, for example the analysis of AUTOSAR models [27].

C. Analysis Execution

In the following, we will demonstrate how the techniques, which have been presented in the previous sections, can be combined to construct a framework for the computation of quantitative measures. For this purpose, the respective measure has to be clearly defined and it must be known, how it can be derived from the model data. The required information for the calculation must be available in the EA model.

At this point, we assume that a transformation is available that converts the EA artifacts into the unified GMM representation (section III-A). In the first step of the analysis specification process, the quantitative measure has to be formalized. We exemplify this process by using the workload calculation developed by [15]. The authors propose a framework for performance analysis, which is defined through a top-down calculation of the workload, followed by a bottom-up propagation of the respective performance. Thereby the workload for a node *a* in an architectural model is defined as

$$\lambda_a = f_a + \sum_{i=1}^{d_a^+} n_{a,k_i} \lambda_{k_i} \quad (1)$$

where f_a denotes the node's arrival frequency, d_a^+ the out-degree, n the edges weight and k_i a child of *a*. This definition requires a property *arrival frequency* for nodes and a property *weight* at the model edges. To calculate the measure it must be ensured, that this information is available.

This definition has to be translated into a data-flow specification. Since we want to calculate the workload, we consequently define a data-flow attribute *workload* in the context of the class *MetaModelNode*. Instance results are computed by the following data-flow equation:

Algorithm 1 Data-flow equation for the attribute *workload*

```

1: public Object node_workload(Object context) {
2:   float result = node.arrivalFrequency;
3:   for (ModelEdge outEdge : context.getOutgoing()) {
4:     if (outEdge.target.stereotype != context.stereotype)
5:       result += outEdge.weight * outEdge.target.getWorkload();
6:   }
7:   return result;
8: }
```

The DFA solver invokes this function with a parameter *context* which signifies the current execution context (comparable to OCL's *self* variable). Line 2 initializes *result* with the value

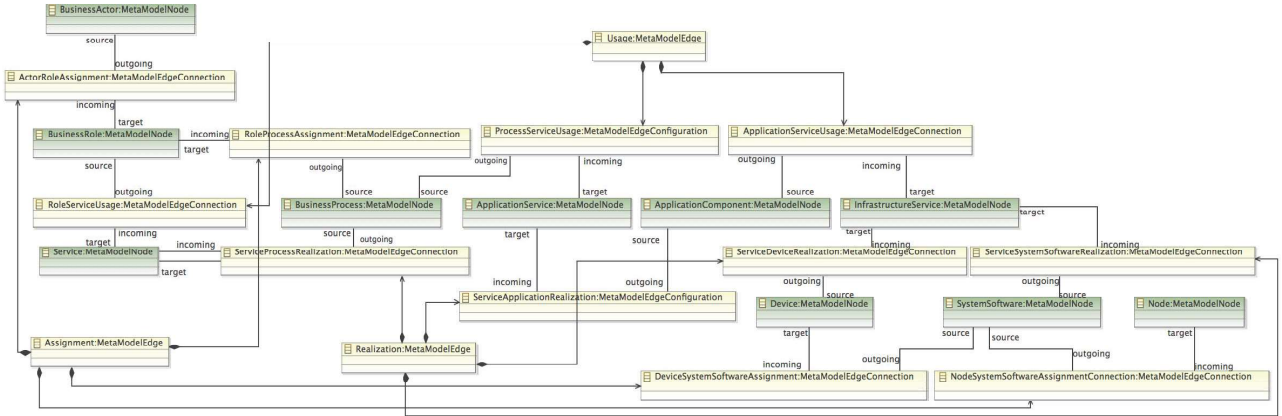


Fig. 4. Specific meta model for the MIDWagen Use Case

of the property *arrivalFrequency* of the currently processed node (if available). The loop in lines 3-6 then iterates over all outgoing edges. If the stereotype of the referenced object differs from the type of the context object (line 4), the result is updated accordingly. This check is required, to get only the children with a different type; connections between elements of the same type, for example between two business processes, are ignored.

Line 5 then multiplies the value of the edge's *weight* attribute with the *workload* result computed at the edge's target node. Since *workload* is a data-flow attribute, the invocation of *getWorkload()* informs the solver, that a recursive fixed-point evaluation is required at this point. It should also be noted that the data-flow functions are also able to access the contents of the GMM's *Configuration*, allowing to parametrize their behavior.

The workload calculation is done for every node in the EA model, depending only on the information of child elements. Thereby we interpret a child, as any connected element, that can be reached via an outgoing edge and must not be of the same type. Of course, this characteristic can also be implemented using different interpretations. The benefit of our interpretation is, that it does not required more semantic information about the EA model, i.e. the specification does not use EA meta model specific concepts. If an analysis required the use of EA specific concepts, this will be implemented using constant, which map to one or more concepts of the EA meta model. This has to be done manually.

IV. EVALUATION

To evaluate the generic applicability of the analysis framework, we implemented three different measures from literature (c.f. subsections IV-A, IV-B, IV-C) and applied them to the MIDWagen example (figure 1).

Obviously the use case is not eligible for all of those analyses, since every analysis requires different information. Independently from the analysis technique, a measure can only be determined if the required information is modeled. For this purpose, we modified the use case (model and meta model)

to incorporate the required information for each analysis. Although we used the notation provided by Archimate to model the use cases, the way the concepts are used differs. Following each adaption results in a different EA meta model, which reflects the variability encountered in real-world scenarios.

However, changes to the EA meta model are not limited to the modification of class attributes. In fact, the structural composition of the elements itself - both in the meta model and in the model - can be highly dynamic. Consider the meta model shown in figure 4: In this definition, *business roles* and *business processes* may be connected either directly or indirectly via intermediary *services*. Depending on organization specific adaptations, only one of these paths may be available or additional alternatives may be introduced. In addition, the interpretation of these alternatives also depends on the respective measure, that is to be computed. This presents a problem in traditional methods for analysis specification that rely on fixed navigational statements. In the previous section, we described how the GMM can be used to handle the technical issues arising from this situation. The propagation of data-flow values along model edges allows the abstraction from the concrete structure of model elements by making information available along transitive paths. As the computation of some measures requires the examination of neighboring elements, we implemented a flow attribute *allpaths* that computes the set of paths starting at the local element to arbitrary target objects and made this information available to subsequent analyses. Alternatively, a computation of the shortest path to each target element can be executed.

In the following, we will describe the implementation of three different application scenarios:

A. KPI Calculation

Matthes et al. defined 52 KPI's to measure EA management goals, based on a literature study [29]. From this set, we chose the calculation of "the coverage of IT continuity plans in respect to business-critical processes", as defined in ([29], page 66) to exemplify the computation of EA KPIs.

In their catalog the authors provide the following specification for this measure: "Number of business-critical processes

relying on business applications not covered by IT continuity plan divided by total number of business-critical processes". We therefore extend the MIDWagen use case with

- the boolean attribute *IT Continuity Plan* and corresponding values for application components and
- the boolean attribute *business critical* and corresponding values for business processes.

Only when this information is present, the coverage measure can be calculated. The KPI is computed by two data flow attributes: *relevant application components* and *continuity covered*. While the values of these attributes are not themselves propagated throughout the model, they nevertheless rely on context-sensitive information, namely the *allpaths* result for each element. The first attribute represents the set of application components supporting a business process and can be calculated by evaluating the following data-flow rule in the context of business processes:

Algorithm 2 Data-flow rule for the attribute *relevant application components*

```

1: public Object node_relevantapplicationcomponents(Node
   currentNode) {
2:   for (Path path : currentNode.getAllPaths() {
3:     if (path.getTarget().type !=
   Constants.APPLICATIONCOMPONENT)
4:       continue;
5:     ignorePath = false
6:     for (PathEntry pathEntry : path.getEntries() {
7:       if (pathEntry instanceof ModelNode){
8:         predecessor = pathEntry;
9:         continue;
10:      }
11:      if (pathEntry.type != (Constants.REALIZE ||
   Constants.USEDBY){
12:        ignorePath = true;
13:        break;
14:      }
15:      if (pathEntry.getSource() != predecessor){
16:        ignorePath = true;
17:        break;
18:      }
19:    }
20:    if (ignorePath)
21:      continue;
22:    relevantApplicationComponents.add(path.getTarget())
23:  }
24:  return relevantApplicationComponents;
25: }

```

Note that this definition does not presume a specific path structure between business processes and application components. Instead, we rely on the value of *allpaths* to identify the relevant connection(s). Lines 3 - 4 ensure that only paths targeting an application component are regarded. For each alternative path, leading to an application component, we then check if the local business process actually relies on the target component. This is implemented by examining the edge stereotypes which must be either match the constant *USEDDBY* or *REALIZE* (line 11 - 14). The constants provide a mapping to the actual used stereotypes in the EA meta model. Additionally, the connection has to be an incoming edge, i.e. opposed to the path direction (line 15 - 18). The path will be discarded if either of these two conditions is not met (line 20). Otherwise,

it is determined that the application component supports the business process and it is therefore added to the set of relevant components (line 22).

The second attribute, *continuity covered*, evaluates to true for a business process, if all of its supporting application components are covered by an IT continuity plan. The computation of the final KPI only requires to divide the amount of business critical processes where *continuity covered* is true by the number of all business critical processes in the model. The screenshot shown in figure 5 shows the result for the MIDWagen example in the analysis plugin developed for the MID Innovator.

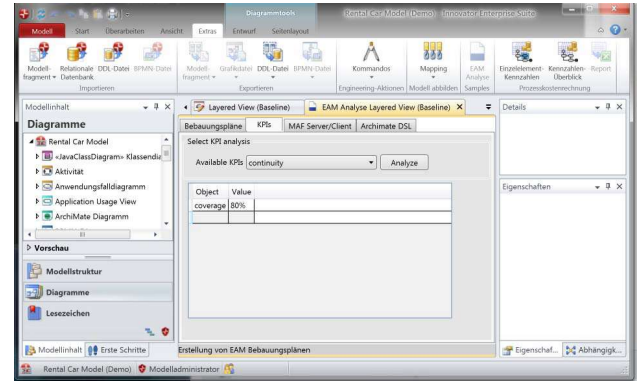


Fig. 5. Screenshot of the calculation of "coverage of IT continuity plans" in the MID Innovator [23].

B. Performance analysis

Jonkers and Iacob propose a performance analysis based on a top-down propagation of workloads and a bottom-up propagation of the utility [15]. The analysis relies on the following set of properties which have to be incorporated into the EA meta model:

- Weight n_e for any relation e
- Service time S_a for service a
- Capacity C_r for any resource r (= actor, application component, device and node)
- Arrival frequency f_a for business services and business processes

In [15] formulas are given that iteratively calculate the respective measures workload I_a , processing time T_a , response time R_a and utilization U_R . The computation of the workload was already presented in section III-C. Algorithm 3 shows the rule for calculating the response time which is defined as $R_a = \frac{T_a}{1 - U_{r_a}}$, where r_a denotes the realizing resource of a .

For the definition of the realizing resource r_a we consider two cases: (1) the realizing resource is directly connected to the service and (2) it is indirectly connected via a behavior element. Therefore the rule first iterates over all incoming realizing edges (line 2 and line 3). If the respective source element has a utilization attribute, this one will be used. If not, all indirectly connected elements will be determined (line 8), and the utilization of the assigned element will be chosen

(line 9) instead. The utilization as well as the processing time (line 14) are also defined as data flow attributes according to the formulas in [15]. Calling their get-Methods instructs the solver to retrieve the respective values, potentially triggering the recursive invocation of other rules.

Algorithm 3 Data-flow rule for the attribute *responsetime*

```

1: public Object node_responsetime(Node currentNode) {
2:   for (ModelEdge inEdge : currentNode.getIncoming()) {
3:     if (inEdge.type != "REALIZE")
4:       continue;
5:     utilization = inEdge.source.getUtilization()
6:     if (utilization >= 0)
7:       continue;
8:     for (Node referencedNode :
9:         inEdge.source.getConnectedNodeSet()) {
10:      if (edge.type != "ASSIGN")
11:        continue;
12:      utilization = referencedNode.getUtilization();
13:    }
14:   return currentNode.getProcessingtime() / (1 -
15:     utilization);
16: }
```

In contrast to Jonkers and Iacob it is not necessary to perform a normalization step before running the analysis and we do not have to make the assumption of a 1-to-1 mapping of a behavior element to a service/resource.

C. Application Usage

In their EA framework for multi-attribute information system analysis Närman et al. describe an application usage analysis [10]. It addresses voluntary application usage, i.e. why users apply a certain application and not another. The application component usage attribute is derived through the following linear regression model:

$$\text{Usage} = \alpha + \beta_1 * \text{TTF}_1 + \dots + \beta_n * \text{TTF}_n + \beta_{n+1} * \text{PU} + \beta_{n+1} * \text{PEoU} \text{ with}$$

- α, β_1, \dots : Constants determined by processing empirical survey of data of application usage
- PU: Perceived Usefulness (comes from evaluation)
- PEoU: Perceived Ease of Use (comes from evaluation)
- TTF: Task Technology Fit (calculated from *Task Fulfillment* of the Business Process and the *Functionality* of the Application Service)

This use case depends on the properties *Task Fulfillment* for Business Process, *Functionality* for Application Service and Application Function, *PU* and *PEoU* for Actor-Component relationships, *Regression Coefficient TTF*, *Regression Coefficient PU* and *Domain Constant* for application components and finally *Regression Coefficient PEoU* for Process-Service relationships. We also introduced a new relationship *used by* between application components and actors.

For the analysis of the usage, we define the following data flow attributes: *weightedTTF* for *used by* edges from application services to business processes and *weightedTAM* for application components. The weighted TTF is calculated

for each *used by* edge connecting an application service to a business process. The values of these attributes are propagated to the corresponding application components via the application functions of those services. The weighted TAM (weighted PU + PEoU) is calculated for each application component by iterating over all outgoing edges to an actor. To get the final usage result, each component has to add both values to the domain constant.

In contrast to the implementation described in [10], our technique does not require an extension of the model with intermediate helper entities to support the aggregation of values.

D. Discussion

The presented framework addresses two major challenges: The lack of a standard EA meta model combined with the fact that analysis specifications are highly dependent on the underlying modeling language as well as overcoming the limitations in current analysis techniques such as OCL. The challenge of diversity in EA languages is addressed by a common representational format and generic analysis definitions. Consequently, it is only necessary to adapt the semantic mappings to be able to execute specified analyses for any EA language (provided that the required information to calculate the measure is contained in the model). With the proposed data-flow based analysis technique, the measure itself and potential interim results are modeled as data-flow attributes. This method enables the specification of an information propagation mechanism on the meta level. The declarative specifications possess a distinct advantage over imperative methods since the propagation is automatically managed by the underlying DFA solver, thus freeing the developer from having to deal with implementation-specific details. Furthermore, the inherent capability to process cyclic dependencies enables the computation of fixed-point results.

A potential weakness of this approach consists of the static mapping technique for semantic concepts as it only allows straightforward one to one or one to many mappings. This can for example present a problem if a single edge concept in the generic analysis specification maps to a chain of multiple relationships in the actual EA model. In future work we will therefore examine the application of more sophisticated mapping methods based, for example, on semantic web technologies.

In general, there are two approaches for the implementation of analysis capabilities into enterprise architecture tools: (1) Integration of the analysis framework in the tool or (2) exporting the model data and executing the analysis outside of the tool (decoupled). While the latter method is easier to accomplish, the former approach is more user friendly.

Suitable tools for the development of enterprise architecture models include for example PlanningIT¹, Innovator [23] or iteraplan². In many cases, plugin interfaces can be used to integrate analysis capabilities into existing tooling environments. In our case study, we implemented a plugin for Innovator which transmits the model data to MAF's analysis server and

¹<http://www.alfabet.com/en/offering/product//main.aspx>

²<http://www.iteraplan.de/en>

interprets the received results. To (de)serialize the model, we developed a textual notation for the GMM using the Xtext language workbench [28].

V. RELATED WORK

One research area regarding related work is querying languages like SQL [30] for databases and SPARQL [31] for ontologies (proposed for EA in [18]). Both have the weakness, that they are highly dependent on the underlying structure (database schema, respective t-box for ontologies). The constraint language OCL allows the annotation of constraints at meta model elements and their evaluation for models. The expressiveness of OCL is limited due to its static navigational expressions [32], [33]. OCL has also a `closure()` operator³, which was introduced in version 2.3.1 (January 2012) of the specification. But it can only be used with the type `Set` types and for the calculation of the transitive closure of a relationship.

Since the GMM encapsulates both meta and model data, it somewhat resembles use of MOF reflection. However, GMM encapsulates domain-specific information relevant to DFA (which concepts are nodes/edges, where information should be propagated), radically simplifies analysis specification and circumvents the problem that some EA standards are not MOF-compatible.

OCL is applied for EA analysis in P2AMF, a general framework for the prediction of element properties in order to compare design alternatives [34]. Thereby the authors consider the calculation of multi-properties, dependencies between the properties as well as uncertainties regarding the existence of elements and relationships. The value of properties can be described by using probability distributions as well as OCL expressions [34]. Their probabilistic inferencing approach lacks procedures of how to deal with cyclic dependencies as well as they are restricted to the expressiveness of OCL. In earlier publications Johnson et al. propose the use of extended influence diagrams as formal language for EA analysis in [19]. This technique is an extension of bayesian networks, where each node represents a variable with a number of states. CPTs are used to determine the probability that a variable is in state *x*, under the assumption that its "predecessor" is in state *y*. They extend this techniques to be able to consider goals and decision alternatives as well as to consider definitional uncertainty. Johnson et al. propose in [12] an EA analysis tool, where they extend ea meta models with attributes and at least attribute relationships, using conditional probability table (CPT) to express the influence of the attributes to each other. They also consider the probability of correctness of any evidence that is used to create concrete scenario models [12]. Their work is limited to use defined scenarios, how can you ensure that the relevant instances are in the scenario. There is also no procedure mentioned of how to deal with cyclic dependencies. Additionally, a lot of the analysis complexity lies in the meta model and is therefore difficult to adapt to different use cases.

Razavi et al. describe in their paper an AHP based approach for the quantitative assessment of quality attributes in different

scenarios. This is an expert-based EA analysis supporting the selection of the scenario that fits best to prioritized enterprise utilities. In their work the measures for the criteria are determined by experts, but there is no automatic calculation of them [35].

De Boer et al. propose the use of XML for static and dynamic EA analysis. They demonstrate how to represent an enterprise architecture in XML and propose the use of XML parsing tools as well as RML (tools) for executing the analysis [17]. They do not provide detailed implementation details, especially for the static analysis.

The combined representation of meta and model data in the GMM facilitates the specification of generic analyses which are able to dynamically adjust their behavior to the structure of the underlying EA language. In some respects, this approach resembles the MOF reflection mechanism. However, the GMM focuses on the representation of domain-specific information relevant to DFA while employing a fixed graph structure which serves as a foundation for the definition of universal analysis specifications. It furthermore circumvents the problem that some EA standards are not MOF-compatible.

VI. CONCLUSIONS

Although enterprise architecture analysis is a highly relevant topic since it allows one to get additional benefits from EA modeling, it is currently not very widely employed in practice. A major challenge for the application of analysis technology can be found in the high diversity of available and customized meta models in the EA field. Since existing analysis specifications usually presuppose a specific structure of meta models and models, it is very difficult to reuse them with organizational models that do not conform to the respective assumptions. They required a high effort to transform the actual EA model in a manner, that the analysis can be executed, additionally the respective meta model does not make any statements about what concepts are actually used. An additional challenge consists of the need of evaluating model objects in their overall context.

In this paper, we presented a meta model independent analysis framework which addresses these problems. Through the implementation of a generic meta model for the EA domain and the use of an analysis technique based on information propagation, it enables the specification of robust, context-sensitive analyses. The generic meta model enables a representation of EA data which is independent from a concrete modeling language. The method of data-flow analysis for models - inspired by the well-researched practices in the compiler construction domain - enables the replacement of fixed navigational statements with a demand-driven propagation of values along model edges. It also enables the implementation of advanced analyses, which rely on the convergence towards a fixed-point value in the case of circular information flow paths.

To demonstrate the generic applicability of the framework, we implemented three different existing analysis approaches from literature, which can be executed on any EA model that has been translated into the GMM format and comprise the necessary information. If only semantic-less analyses should be executed, like shortest path or the presented workload

³An example use case would be the enforcement of non-cyclic generalization hierarchies for Classifiers:
`self->closure(superClass)->excludes(self)`

calculation, no adaption of the analysis specification is necessary. More semantic-full analyses require the adaptation of the stereotype constants such as "application component" to the respective name of the class in the target modeling language. At the moment the analysis specifications are defined and compiled during design time. Future work has to be done to enable a dynamic definition of analyses. Also the generic meta model approach can be further enhanced to directly include often used concepts like generalization relationships.

ACKNOWLEDGMENT

This work was partially sponsored by "FuE-Programm Informations- und Kommunikationstechnik Bayern". The authors would like to thank MID GmbH for providing their demo use case, licenses for their tool as well as for the support during implementation.

REFERENCES

- [1] Marc Lankhorst, *Enterprise Architecture at Work*. Berlin: Springer-Verlag Berlin and Heidelberg GmbH & Co. KG, 2012.
- [2] J. A. Zachman, "A framework for information architecture," *IBM System Journal*, vol. 26, no. 3, pp. 276–295, 1987.
- [3] The Open Group, *TOGAF Version 9.1*. Van Haren Publishing, 2011.
- [4] U.S. Department of Defense, "The DoDAF Architecture Framework Version 2.02," 2010, in: <http://dodcio.defense.gov/dodaf20.aspx>, accessed 09/04/2013. [Online]. Available: <http://dodcio.defense.gov/dodaf20.aspx>
- [5] ISO/IEC, "ITU-T x.901 | ISO/IEC 10746-1: Information technology - open distributed processing - reference model: Overview," International Standard V1, 1998.
- [6] U.S. Executive Office, "Federal enterprise architecture," 2012, in: <http://www.whitehouse.gov/omb/e-gov/fea/>, accessed 09/04/2013. [Online]. Available: <http://www.whitehouse.gov/omb/e-gov/fea/>
- [7] P. Kruchten, "The 4+1 view model of architecture," *IEEE Software*, vol. 12, no. 6, pp. 42–50, 1995.
- [8] S. Buckl, F. Matthes, and C. M. Schweda, "Classifying enterprise architecture analysis approaches," *Enterprise Interoperability*, pp. 66–79, 2009.
- [9] M. E. Iacob and H. Jonkers, "Quantitative analysis of enterprise architectures," *Interoperability of Enterprise Software and Applications*, pp. 239–252, 2006.
- [10] P. Närman, M. Buschle, and M. Ekstedt, "An enterprise architecture framework for multi-attribute information systems analysis," *Software & Systems Modeling*, pp. 1–32, 2012.
- [11] K. D. Niemann, *From enterprise architecture to IT governance*. Springer, 2006.
- [12] P. Johnson, E. Johansson, T. Sommestad, and J. Ullberg, "A tool for enterprise architecture analysis," in *11th IEEE International Enterprise Distributed Object Computing Conference, EDOC 2007*, 2007.
- [13] U. Franke, W. R. Flores, and P. Johnson, "Enterprise architecture dependency analysis using fault trees and bayesian networks," in *Proceedings of the 2009 Spring Simulation Multiconference*, ser. SpringSim '09. San Diego, CA, USA: Society for Computer Simulation International, 2009, pp. 55:1–55:8.
- [14] M. E. Iacob, L. O. Meertens, H. Jonkers, D. A. C. Quartel, L. J. M. Nieuwenhuis, and M. J. van Sinderen, "From enterprise architecture to business models and back," *Software & Systems Modeling*, pp. 1–25, 2012.
- [15] H. Jonkers and M.-E. Iacob, "Performance and cost analysis of service-oriented enterprise architectures," *Global Implications of Modern Enterprise Information Systems: Technologies and Applications*, IGI Global, 2009.
- [16] R. Lagerström, P. Johnson, and M. Ekstedt, "Architecture analysis of enterprise systems modifiability: a metamodel for software change cost estimation," *Software Quality Control*, vol. 18, no. 4, pp. 437–468, Dec. 2010.
- [17] F. de Boer, M. Bonsangue, J. Jacob, A. Stam, and L. van der Torre, "Enterprise architecture analysis with XML." IEEE Computer Society Press, 2005.
- [18] S. Sunkle, V. Kulkarni, and S. Roychoudhury, "Analyzing enterprise models using enterprise architecture-based ontology," in *Model-Driven Engineering Languages and Systems*. Springer, 2013, pp. 622–638.
- [19] P. Johnson, R. Lagerström, P. Närman, and M. Simonsson, "Enterprise architecture analysis with extended influence diagrams," *Information Systems Frontiers*, vol. 9, no. 2-3, pp. 163–180, 2007.
- [20] M. Buschle, J. Ullberg, U. Franke, R. Lagerström, and T. Sommestad, "A tool for enterprise architecture analysis using the PRM formalism," in *Information Systems Evolution*. Springer, 2011, pp. 108–121.
- [21] P. Johnson, J. Ullberg, M. Buschle, U. Franke, and K. Shahzad, "P2AMF: predictive, probabilistic architecture modeling framework," in *Enterprise Interoperability*, ser. Lecture Notes in Business Information Processing, M. v. Sinderen, P. O. Lutthighuis, E. Folmer, and S. Bosems, Eds. Springer Berlin Heidelberg, Jan. 2013, no. 144, pp. 104–117.
- [22] P. Johnson, L. Nordström, and R. Lagerström, "Formalizing analysis of enterprise architecture," in *Enterprise Interoperability*, G. Doumeingts, J. Mäijller, G. Morel, and B. Vallespir, Eds. Springer London, 2007, pp. 35–44.
- [23] MID GmbH, "MID Innovator for Enterprise Architects," 2014, accessed 28/02/2014. [Online]. Available: <http://www.mid.de/produkte/innovator-enterprise-modeling.html>
- [24] C. Saad and B. Bauer, "Data-flow based Model Analysis and its Applications," in *Proceedings of the 16th International Conference on Model Driven Engineering Languages and Systems (MoDELS'13)*, September 2013.
- [25] G. Kildall, "A unified approach to global program optimization," in *Proceedings of the 1st annual ACM SIGACT-SIGPLAN symposium on Principles of programming languages*. ACM, 1973, pp. 194–206.
- [26] C. Saad and B. Bauer, "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)*, May 2011.
- [27] J. Kienberger, P. Minnerup, S. Kuntz, and B. Bauer, "Analysis and Validation of AUTOSAR Models," in *Proceedings of the 2nd International Conference on Model-Driven Engineering and Software Development (Modelsward'14)*, 2014.
- [28] Eclipse Foundation, "Xtext - Language Development Framework," in: <http://www.eclipse.org/Xtext/>, accessed 28/02/2014. [Online]. Available: <http://www.eclipse.org/Xtext/>
- [29] F. Matthes, I. Monahov, A. Schneider, and C. Schulz, "EAM KPI catalog v 1.0," Technical University Munich, Tech. Rep., 2012.
- [30] ISO/IEC 9075:2011, "Information technology - Database languages - SQL," 2011.
- [31] W3C SPARQL Working Group, "SPARQL 1.1 Overview," 2013, accessed 21/03/2013. [Online]. Available: <http://www.w3.org/TR/sparql11-overview/>
- [32] L. Mandel and M. Cengarle, "On the expressive power of OCL," *Lecture notes in computer science*, pp. 854–874, 1999.
- [33] T. Baar, "The definition of transitive closure with OCL - limitations and applications," in *Perspectives of System Informatics*. Springer, 2003, pp. 979–997.
- [34] P. Johnson, J. Ullberg, M. Buschle, U. Franke, and K. Shahzad, "P2AMF: predictive, probabilistic architecture modeling framework," in *Enterprise Interoperability*, ser. Lecture Notes in Business Information Processing, M. v. Sinderen, P. O. Lutthighuis, E. Folmer, and S. Bosems, Eds. Springer Berlin Heidelberg, 2013, no. 144, pp. 104–117.
- [35] M. Razavi, F. S. Aliee, and K. Badie, "An AHP-based approach toward enterprise architecture analysis based on enterprise architecture quality attributes," *Knowledge and Information Systems*, vol. 28, no. 2, pp. 449–472, 2011.