Towards An Analysis Driven Approach for Adapting Enterprise Architecture Languages

Sybren de Kinderen[1,3], Qin Ma[2,3]

[1] University of Luxembourg, Luxembourg
[2] Public Research Centre Henri Tudor, Luxembourg, Luxembourg
[3] EE-Team, Luxembourg, Luxembourg*
sybren.dekinderen@uni.lu, qin.ma@tudor.lu

Abstract: Enterprise Architecture (EA) modeling languages are increasingly used for various enterprise wide analyses.

In most cases one needs to adapt EA languages to an appropriate level of detail. However such an adaptation is not straightforward. Language engineers currently deal with analysis driven language adaptation in an ad-hoc manner, adapting languages from scratch. This introduces various problems, such as a tendency to add uninteresting and/or unnecessary details to languages, while important enterprise details are not documented. Moreover, adding detail increases the complexity of languages, which in turn inhibits a language's communication capabilities. Yet experience from practice shows that architects often are communicators, next to analysts. As a result, one needs to find a balance between a model's communication and analysis capabilities.

In this position paper we argue for an approach for assisting language engineers in adapting, in a controlled manner, EA languages for model-driven enterprise analyses. Furthermore, we present the key ingredients of such an approach, and use these as a starting point for a research outlook.

1 Introduction

Enterprise Architecture (EA) is increasingly recognized as a steering instrument that covers the complete business-to-IT stack of an enterprise [AW09, OPW+08, Lea13], interrelating an enterprise's products and services, business processes, IT applications and physical IT infrastructure. By emphasizing such a holistic perspective on an enterprise [Lea13], EA can act as an instrument for various *enterprise-wide analyses* (briefly enterprise analyses) to support decision making, such as the enterprise wide impact of access control concerns [FDP+12], the modifiability of an enterprise-wide IT system [LJH10], cost management [Lea13], and more. Two recent surveys among practitioners [MLM+13, LJJ+06] also show a need from industry for such analyses.

*The Enterprise Engineering Team (EE-Team) is a collaboration between Public Research Centre Henri Tudor, The University of Luxembourg, Radboud University Nijmegen and HAN University of Applied Sciences (www.ee-team.eu)

EA modeling languages, prominently the Open Group standard ArchiMate [IJLP12, Lea13 provide a model driven approach to capture enterprise architectures. Because of the holistic nature, these languages are often on purpose designed for expressing an enterprise at a high level of abstraction [IJLP12]. As a consequence, resulting EA models of such languages are more used to facilitate communication among stakeholders than to support enterprise analyses [IJLP12, MLM+13].

Yet to do a proper enterprise analysis (be it for cost management, security concerns, or otherwise), we also need *domain-specific details* that provide us with a detailed expression of analysis concerns. Thus, to perform an enterprise-wide analysis we essentially need to perform both (1) inter-layer analysis, whereby models with a high level of abstraction allow us to connect different enterprise layers, e.g., business, application and infrastructure layers as defined in ArchiMate, and (2) intra-layer analysis, whereby detailed domain specific models can express analysis concerns to *a level of detail* that is sufficiently amenable for analysis purposes.

A case supporting this argument is the ArchiMate language. As illustrated by the authors of [Lea13, p. 189 - 219], ArchiMate is enriched with domain specific details to enable enterprise analyses, namely:

- either extra attributes assigned to concepts and relations of ArchiMate to capture measures relevant for analysis, e.g., response time of a service and utilization of a resource for performance analysis [JI09], and e.g., importance of a business process and effectiveness of an information system for portfolio analysis [QSL12] (more syntactical details);
- or more details about the meaning of these attributes and the inter-relation between their values [JI09, QSL12] (more semantical details);
- or a higher degree of formality by translating ArchiMate models into mathematical formalisms to support static impact-of-change analysis [dBBJ+04, BBJ+05] (more formal).

As hinted above, it is impractical, if not impossible, to have a universal language that caters to all types of analyses [MLM+13, LPJ10, KGP12], because the diversity of types of analysis and their distinct requirement on the level of detail. Therefore, a tailored solution pertaining to the level of detail for each type of analysis is a better option. Note here that Sect. 3 elaborates further on what we mean by "analysis" and "level of detail".

The right level of detail is not straightforward to achieve. Currently, it is mainly done in an ad-hoc manner, where for each analysis task, a language engineer basically has to start from scratch. This gap is also reflected in literature. The authors of [MLM+13, p.18] state that, for analysis purposes, architects call for extending current languages with extra properties to enhance their expressiveness for a particular domain. Yet this language extension is not considered a trivial task as one may tend to add excessive detail to a language [MLM+13, p.13], describing aspects of a system that are trivial or uninteresting, while the most interesting discussions are not documented.

Furthermore, [MLM+13] observes a tension between the architect's dual roles as an analyst (the "introvert" architect) and as a communicator (the "extrovert" architect). On the one hand, as stated, architects as analysts call for extending current languages with extra (e.g. domain specific) properties. On the other hand, for communication purposes architects prefer a language that is "simple enough to communicate the right message to stakeholders"[MLM+13, p.18]. To this end, [MLM+13] states that architectural languages should be generic and semi-formal, rather than domain-specific, and detailed.

The above two challenges further emphasize that, for model-driven enterprise analysis, dealing with the model's level detail is a non-trivial issue. In this position paper, we argue for an approach that tackles the challenges from a model-driven, language-based perspective. We envision a generic analysis-driven EA language adaptation approach to assist language engineers in evaluating and adapting, in a controlled manner, EA languages for model-driven enterprise analyses.

As elaborated in Sect. 4, such an approach will address the problem from the following two aspects: firstly, a framework will be established to evaluate the fitness of a candidate EA modeling language for the targeted analysis purpose; secondly, language customization and model integration techniques will be studied to realize the suggested adaptation.

As a starting point for the evaluation part, we can take inspiration from existing work such as model quality [KSJ06, Moo05] to assess the level of detail. For the adaptation part, we can exploit model integration techniques [ZKK07, KM10] to make languages fit for analysis purposes. Each of these elements is a valuable component for creating our language adaptation approach. However, these elements by themselves are not sufficient for our research purposes. First, model quality work is generic, thus lacking a capability to specifically assess the capacity of models for dealing with a particular analysis. Second, how to specialize and combine these individual works in an effective approach to indeed support model-driven enterprise analyses has not received much research attention yet.

Meanwhile, in developing the approach, we will give special care to achieve a balance between using EA models for communication and analysis purposes. We require such attention since the addition of details adds complexity that may inhibit communication. As a starting point we can use literature on (1) the design principles behind the ArchiMate language [LPJ10], which are explicitly aimed at model complexity reduction through, e.g., conceptual integrity principles, (2) model (de-composition), which subdivides models into smaller relevant models to aid communication [MKG13], and (3) model complexity management [Moo09], which provides techniques to construct a visual notation that does not overload the human mind.

As such, the main contribution of this paper is twofold (1) to argue for an approach that can adapt, in a systematic way, EA languages to cope with various enterprise analyses, (2) a first impression of what we consider to be the key elements for such an approach.

This paper is structured as follows. Sect. 2 discusses how the state of the art forms useful input for analysis-driven language adaptation, and where it falls short. Thereafter Sect. 3 argues for systematic analysis-driven language adaptation, whereas Sect. 4 provides a first impression of what we consider to be important elements for such language adapation. Sect. 5 concludes, and provides a research outlook.

2 Background

2.1 Enterprise Architecture modeling

Various enterprise architecture frameworks provide ingredients for enterprise analysis. To name a few: ARIS [SN00, STA05], CIMOSA [KVZ99], DoDAF/MoDAF, ArchiMate [Lea13], MEMO [Fra02], and UPDM [OMG13].

UPDM (Unified Profile for DoDAF/MoDAF) is an OMG standard language that unifies concepts and viewpoints from the enterprise architecture frameworks DoDAF and MoDAF [1]. It provides a standard UML profile for expressing DoDAF/MoDAF concepts [OMG13, p.17]. Furthermore, UPDM provides a mapping to SysML, which provides a starting point for model driven analysis of EA models created with UPDM.

CIMOSA and ARIS are frameworks for creating and managing enterprise architectures, which both provide a process-oriented modeling language for expressing enterprise architectures. Particularly, ARIS provides the well known Event Process Chain (EPC) language [STA05].

While the above mentioned frameworks and languages may provide a starting point for different analyses, they provide few guidelines for adaptation to a particular EA analysis. Dealing with different levels of detail, or how to instantiate SysML models for analyses, is not further specified. This is also true for EPC. While various formalizations of EPC exist in academic literature, they typically remain on a detailed workflow level. Thus, how to deal with the different enterprise perspectives required to do an enterprise wide analysis (e.g. strategic goals or computational resources) , or how to deal with different levels of detail, remains - to the best of our knowledge - underresearched for EPC models.

In summary: while various EA modeling languages provide a good starting point for analysis they remain just that: *a starting point*.

2.2 Model quality

Work on model quality provides us with hints on how to assess the fitness of a language for a particular purpose. Here frameworks evaluate the general quality of a model [KSJ06], for example if the syntax of a model is appropriate for the modeling task at hand. Furthermore, some model quality frameworks focus on evaluating a particular type of model, such as UML activity diagrams [GFSN$^+$11].

However, as emphasized by [Moo05], the field of model quality is still immature. In particular, a multitude of academic propositions for model quality exist, but few have been extensively validated in practice [Moo05]. Furthermore, there is no agreement on what different model quality characteristics mean. The latter is more recently reflected in [HFL12], who try to build a consensus around the model quality attribute “understandability”. Fur-

[1] DoDAF and MoDAF are frameworks that provide a standard way for planning and managing enterprise architectures, but are not languages.

thermore, to the best of our knowledge, no model quality work exists pertaining to assessing the fitness of a model for analysis purposes. Yet we expect at least some characteristics to be specific for model based analysis, such as the level of formality of a model.

2.3 Enterprise analysis

As mentioned before there exist several instances of model driven enterprise wide analysis, e.g. [JIV+14, JI09, dBBJ+04, QSL12]. These analyses often require models to express the enterprise at different levels of detail. Particularly, this is illustrated by the profitability analysis exposed in [JIV+14].

[JIV+14] introduces an approach for reasoning under uncertainty about profitability analysis of to-be business networks, by extending the net present value calculations from the e^{3}value modeling language with probabilities on the occurrence of future scenarios.

In line with the e^{3}value language, [JIV+14] performs a profitability analysis based on the value exchanges of the actors participating in the business network. Thus they remain at a *high level of abstraction*. However, [JIV+14, p.25] admits that many of the details to do a in-depth, substantial profitability calculation require them to "zoom in" on various modeled elements of the business network. They then go on to argue that such details could typically be obtained by relating their business network profitability approach to approaches for enterprise architecture cost analysis and prediction, prominently [JI09].

Yet, in contrast to [JIV+14], [JI09] remains at a *very detailed level of abstraction*. For example, their insurance case analyzes business processes such as "store damage report", as supported by a "report scanning application" (see [JI09, p.66]). Thus here there is an apparent gap in level of detail between the two types of analyses, which needs to be addressed by language engineers if one wants to perform an in-depth profitability analysis.

Dealing with such differing levels of detail in a controlled manner is not a straightforward task to achieve, as mentioned in the introduction. We actually observed the problem of dealing with different levels of abstraction in our own work, particularly regarding an experiment on bridging the value modeling technique e^{3}value with ArchiMate via the transaction modeling technique DEMO [KGP12]. Here, a key idea behind this experiment is to use DEMO transaction patterns to analyze what business process steps are required to realize economic transactions stemming from e^{3}value, and to subsequently use these business processes as a starting point for ArchiMate modeling. Yet, applying the DEMO transaction patterns yielded detailed process models focused on communication acts, such as "send an acknowledgment receipt". As a result the produced process models were not fully suitable for ArchiMate, which typically expresses process models at a high level of abstraction. As a result, as part of future work, [KGP12] suggests to assess the fitness of connecting DEMO and ArchiMate due to their differing level of detail, and how to deal with this connection. This is actually one instance of the more general problem statement described in this paper.

2.4 Model driven language engineering

Model Driven Engineering (MDE) is an engineering discipline where *models* are systematically used as the primary artifacts throughout the engineering lifecycle. A model is a sound abstraction of an original, being a software system or an enterprise for example, allowing predictions or inferences to be made [Küh06]. Models are expressed in modeling languages. The definition of a modeling language consists of the specification of the following components: abstract syntax, concrete syntax and semantics, as well as mappings between them. Model Driven Language Engineering (MDLE) applies MDE to language engineering [Kle09]. More specifically, models are exploited to capture all the components of a language specification. The mapping between these artifacts are established by model transformations. Models used to define languages are referred to as metamodels, namely, models of models [Küh06, OMG03].

The main contributions of the proposed approach, namely (1) the definition of the notion of level of detail and the evaluation of a candidate EA modeling language for a targeted analysis purpose, (2) the techniques to adapt languages towards the right level of detail, and (3) the support to balance EA model communication capability and the presence of extra complexity, will be largely following the mindset from MDLE. Various types of (meta-)models and model transformation techniques will be identified, defined, and exploited for achieving our goals.

2.5 Language adaptation

Adapting a candidate EA modeling language towards the right level of detail to serve a given analysis purpose involves two directions of manipulations: to remove unnecessary details, and to introduce missing details.

For the former, existing works on metamodel pruning provide inspiration. The idea is to eliminate unnecessary details of a modeling language and obtain a minimal set of modeling elements containing a required subset of elements of interest. Metamodel pruning techniques have been investigated for various purposes such as the construction of model transformations [SMBJ09]. In our own previous work, we developed a generic (meta-)model decomposition technique and applied it to the Eclipse Modeling Framework to improve language comprehension [MKG13]. However, analysis oriented purposes that we will address in adapting languages have not yet been considered by pruning techniques.

For the latter, language integration techniques provide methods to *actually enrich* current EA modeling languages with analysis capabilities. Such techniques allow for capitalizing on the complementary strengths of languages by (1) merging two languages, or subsets thereof, into a new language. [KBJK03, ZKK07] propose example techniques for this; (2) keeping a federated set of languages, thus establishing links between metamodels of individual languages, but leaving the original metamodels untouched. An exemplar set of federated enterprise models is MEMO [Fra02], which consists of a set of models that each express a relevant perspective on the enterprise; (3) having an intermediate enterprise

modeling language (a "hub"), through which different enterprise modeling languages (the "spokes") are linked. The Unified Enterprise Modeling Language (UEML) [Ver02] is a prominent example of this strategy.

Furthermore, given the model-driven nature of the proposed approach, we can also exploit existing model composition techniques and aspect-oriented modeling techniques when missing details need to be introduced. Examples of such techniques include [BBDF+06, WS08, ODPRK08, SSK+07, KM10].

Yet, current language integration techniques do not sufficiently deal with tensions between languages existing at differing levels of detail, neither do they take care not to sacrifice the communication capability of a language while integrating it with others.

3 Research Objectives

Our objective is to develop an approach for adapting the level of detail of EA modeling languages to cope with different enterprise analyses in a controlled manner.

This section rationalizes such as an approach. We do so by breaking down our main objective into four sub objectives, that we subsequently discuss in further detail.

1. Define the level of detail of a modeling language.

   Different persons may have a different interpretation of abstract vs detailed granularity. Consider two example languages: the Business Process Modeling Notation (BPMN) and ArchiMate. Here, one may argue that ArchiMate is more detailed than BPMN because, *syntactically*, it allows one to express the IT applications and physical IT infrastructure, in addition to business processes. However, one may also argue that BPMN is more detailed than ArchiMate because, *semantically*, the processes captured in BPMN express more specific temporal dependencies (e.g. task A finishes before task B starts).

   The above example illustrates that there is a need to clarify the term "level of detail" so that we can assess languages accordingly. Moreover, we consider that at least syntax and semantics are two important dimensions.

2. Provide a diagnosis of the level of detail of EA languages along different dimensions with respect to an analysis task.

   A language might not be detailed enough for one type of analysis, but sufficient for another. Consider cost management for two stakeholders: an enterprise architect, and a process manager. On the one hand, the enterprise architect concerns himself with having a global overview of costs. To arrive at the global costs overview, he requires syntactical details to capture an enterprise holistically. ArchiMate is an example language having the required level of detail. On the other hand, the process manager is concerned with finding out the costs of each step in a process, and for computing the cost of a process within a time frame. To achieve these, he requires both syntactical details to capture the structure of the business process and

semantical details to capture the dynamic behavior of the process. Languages such as BPMN would be more appropriate candidates to consider than ArchiMate.

The above example shows a need to assess the fitness of a language for performing a particular analysis in terms of level of detail. Furthermore, in case of mismatches, one should also pinpoint discrepancies.

3. Adapt the level of detail of EA languages pertaining to a particular type of analysis.

   Following up on the diagnosis of fitness of a language for an analysis, we need a systematic approach to language adaptation. As we observed from the existing work, languages adapted in an ad-hoc manner have the following shortcomings [MLM+13]: (1) unnecessary details might be introduced which makes the resulting models too complex; (2) necessary details might be overlooked which prevents the provisioning of analysis results relevant to end users; (3) inconsistency might emerge in the adapted language as a result of introducing concepts that overlap and/or conflict with existing ones. Hence, we need techniques to adapt EA languages in a controlled manner towards the right level of detail and meanwhile following guidelines to avoid unnecessary complexity.

4. Balance extra level of detail required for analysis with communication.

   In line with [MLM+13] facilitating communication is deemed important for industry uptake and use of architecture modeling languages. Yet, [MLM+13] also shows that the focus on analysis for languages - as predominant in academia - has at least partly hindered their communicability. Hence, while designing our approach for currently used EA modeling languages we should take care not to sacrifice communicability of the models for the sake of analysis.

4 How to adapt

Now that we have discussed the objectives for and rationale behind an analysis driven approach for adapting enterprise architecture languages, we discuss a first version of the envisioned approach. Particularly, we envision that our approach will consist of the following three parts: (1) a granularity scale framework, (2) an analysis-driven language adaptation method, (3) techniques and tools for balancing between a model's level of detail and its communication capability.

1. Granularity scale framework We define a framework to clarify the notion of level of detail of EA modeling languages and to assess the fitness of an EA modeling language with respect to an analysis task.

Based upon previous work on enriching enterprise modeling languages with analysis capabilities, e.g., [JI09, QSL12, BBJ+05], we initially identify three dimensions along which EA modeling languages can be adjusted, namely the syntax, the semantics and the level of formality. The first two dimensions are aligned with the components of a language

Stakeholder	Concerns	Analysis Description	Required Information	Language Evaluation	
				ArchiMate	BPMN
Enterprise Architect	To find out the costs of all enterprise level components	Examples of enterprise level components include: products and services, business processes, IT applications, and physical infrastructure. The cost of a component can influence the cost of another due to the interdependency between them. For example, the cost of purchasing and maintaining an IT asset should find its way to the costs of business processes supported by the asset, proportionally to their execution.	Products and services, and their costs	1	0
			Business processes, and their costs	1	1
			IT applications, and their costs	1	0
			Physical infrastructure, and their costs	1	0
			Interdependency among enterprise level components	1	0
Process Manager	To find out the cost of a process within a time frame	The cost of a process depends upon the costs of each step in the process, and how often they are executed, which is influenced by the frequency of process execution and probability of branch choice.	Fine-grained business process model	1	2
			Business process execution semantics	0	1
			Cost of carrying out a process activity	0	0

Table 1: Granularity scale for cost management analysis with ArchiMate and BPMN. Scoring: 0) no support, 1) partial support, 2) full support.

specification in model driven language engineering, and the last dimension determines the analysis capability of a language as witnessed by [MLM+13]. These three dimensions together constitute a so called 3D "granularity scale" that will be used by the framework to frame the level of detail of EA modeling languages.

With the granularity scale, the framework will follow a method to assess the fitness of EA languages along different dimensions with respect to an analysis task. For the design of such a method, the procedural methods proposed for assessing model quality [KSJ06, Moo05] provide us with a useful starting point. Furthermore, we seek inspiration from guidelines proposed by ontology mapping literature [CSH06] to identify syntactical and semantical heterogeneity between ontologies[2] for the purpose of pinpointing various types of mismatches along all the dimensions.

The envisioned framework takes as input an EA modeling language and a given analysis task, and produces a qualitative "fitness for purpose" diagnosis elaborated along the three dimensions. Note that, since we aim at focusing our effort on the level of detail, we consider a language's concrete syntax out of scope for our approach.

An early version of such a granularity scale framework is presented in Table 1. Here we see the cost management analysis example for two languages, ArchiMate and BPMN, discussed in Sect. 3, analyzed on: (1) the involved stakeholders, (2) their analysis concerns, (3) the information required for addressing the concerns, and (4) the fitness of a language with respect to the required information. At this point in time, we aggregate the fitness status on our three dimensions into a single score ranging from 0 (no support) to 2 (full support). For example: we see that an enterprise architect requires information on different enterprise components, and that ArchiMate is more suitable in expressing this information than BPMN. This is reflected in the scores: 1 versus 0 (note: ArchiMate natively does not support expression of costs, hence we score it as 1 instead of 2).

[2]Here ontology refers to a formal ontology: "a formal specification of a shared conceptualization" [BAT97]. Similar to a modeling language, ontologies are usually specified in terms of concepts and their interrelations, and are formalized to the point that a computer can process them. Furthermore, discrepancies between ontologies are analyzed in typical modeling language terms, prominently syntax and semantics.

2. Analysis-driven language adaptation In line with the "fitness for purpose" diagnosis, we develop adaptation techniques to integrate EA modeling languages at different levels of detail.

We mainly consider two types of language integration techniques: loosely coupled, i.e., federating a set of languages in coherence by mapping the concepts from different languages; tightly coupled, i.e., decompose existing languages into language fragments then compose the fragments into a Domain Specific Language (DSL) with the right level of detail. An exemplar set of federated enterprise models is MEMO [Fra02], which consists of a set of models that each express a relevant perspective on the enterprise. The Unified Enterprise Modeling Language (UEML) [Ver02] is another prominent example of this strategy. For the DSL based approach, we need techniques in two directions, namely: to remove unnecessary details, and to introduce missing details. For the former, we explore techniques such as metamodel pruning [SMBJ09], model slicing [BLC08, BCBB11] and model decomposition [MKG13] which help in identifying the part of the language relevant for the analysis at hand. For the latter, we explore techniques such as language merging [KBJK03, ZKK07] and model composition [KM10].

We reason that the federation based approach is light-weight in the sense that instead of cutting them off, unnecessary details are simply hidden and the original languages remain untouched. As a consequence, existing tools and models of these languages can be reused. Moreover, in cases where the users are already familiar with the individual languages (which remain untouched), the learning curve of the adapted language might be less steep. However, these advantages come at a price. For example, for automated analysis, extra efforts in terms of model transformations are needed to filter and gather relevant information from original models.

On the contrary, we posit that the DSL based approach calls for more efforts at the language adaptation and EA modeling stage. However, it enjoys all the advantages a DSL brings about compared to a general purpose language [MHS05], being gains in domain expressiveness, ease of use, etc. Moreover, by definition, once created, the DSLs are precisely at the right level of detail for the targeted analyses.

In order to be generic, we will support both approaches and provide guidelines in selecting the appropriate techniques.

3. Balancing communication and level of detail We develop guidelines to control model complexity, and model visualization techniques, implemented into a software tool, to facilitate communication of EA models.

On the one hand, the proposed guidelines aid language engineers during the language adaptation process. Particularly, we aim at controlling model complexity when enriching a language with analysis capabilities. To this end, we can capitalize on literature pertaining to (1) the design principles behind the ArchiMate language [LPJ10], which are explicitly aimed at model complexity reduction through, e.g., conceptual integrity principles, (2) more generally, design principles for engineering complex systems (e.g. [Bro87]) such as "do not introduce what is irrelevant".

On the other hand, the model visualization techniques, and corresponding tools, aid in hid-

ing a model's complexity while communicating. More specifically, we aim to exploit (1) model de-composition, which subdivides models into smaller relevant models to hide unnecessary complexity [MKG13], (2) model complexity management techniques [Moo09], which provides a means to construct a visual notation that does not overload the human mind.

5 Conclusions and outlook

In this paper, we have argued for an approach that assists language engineers in systematically adapting EA languages to make them fit for various enterprise wide analysis. Furthermore we provided an overview of what we consider to be the key elements for such an approach.

We are aware of the ambition level of this research effort. In line with this we foresee the following more concrete research challenges for each of our three key language adaptation elements:

Concerning the *granularity scale framework* we have to clarify further *how* we actually assess the fitness of a language for a particular analysis purpose. One challenge here is to decide on objective versus subjective measuring, the difference being that (1) with objective measuring, one assesses language fitness by analyzing a language specification in the light of the analysis purpose, whereas (2) with subjective measuring, one assesses language fitness by eliciting stakeholder opinions on the fitness of a language for the analysis purpose at hand. Furthermore, we should elaborate on the grading scale compared to the initial version in Table 1, in particular regarding the derivation of scores along each of the three granularity scale dimensions: syntax, semantics, and level of formality.

Concerning the *analysis-driven language adaptation*, we should exhaustively and systematically compare existing language enriching approaches, so as to provide for a road map for selecting a suitable language enrichment approach. To the best of our knowledge such a systematic overview, that covers *multiple fields* (enterprise modeling, model driven engineering, ontology mapping), does not yet exist.

Finally, concerning the *balancing of analysis and communication*, we should further analyze literature on the design of complex systems. In addition, in testing a model's communicability, we should ultimately involve model end users, such as enterprise architects with a modeling background. We plan on involving them in case studies, which will be the primary means for practical validation of our proposed approach.

References

[AW09] Stephan Aier and Robert Winter. Virtual decoupling for IT/business alignment–conceptual foundations, architecture design and implementation example. *Business & Information Systems Engineering*, 1(2):150–163, 2009.

[BAT97] Pim Borst, Hans Akkermans, and Jan Top. Engineering ontologies. *International Journal of Human-Computer Studies*, 46(2):365–406, 1997.

[BBDF+06] Jean Bézivin, Salim Bouzitouna, Marcos Del Fabro, Marie P. Gervais, Frédéric Jouault, Dimitrios Kolovos, Ivan Kurtev, and Richard F. Paige. A Canonical Scheme for Model Composition. In *Proceedings of the 2nd European Conference on Model Driven Architecture - Foundations and Applications (ECMDA-FA 2006)*, volume 4066 of *Lecture Notes in Computer Science*, pages 346–360, 2006.

[BBJ+05] F. S. de Boer, M. M. Bonsangue, J. Jacob, A. Stam, and L. van der Torre. Enterprise Architecture Analysis with XML. In *Proceedings of the 38th Annual Hawaii International Conference on System Sciences - Volume 08*, HICSS '05, pages 222.2–, Washington, DC, USA, 2005. IEEE Computer Society.

[BCBB11] Arnaud Blouin, Benoît Combemale, Benoit Baudry, and Olivier Beaudoux. Modeling Model Slicers. In *Proceedings of the ACM/IEEE 14th International Conference on Model Driven Engineering Languages and Systems (MoDELS 2011)*, pages 62–76, 2011.

[BLC08] Jung Ho Bae, KwangMin Lee, and Heung Seok Chae. Modularization of the UML Metamodel Using Model Slicing. *Fifth International Conference on Information Technology: New Generations*, 0:1253–1254, 2008.

[Bro87] F.P. Brooks Jr. No silver bullet: essence and accidents of software engineering. *IEEE Computer*, 20(4):10–19, April 1987.

[CSH06] Namyoun Choi, Il-Yeol Song, and Hyoil Han. A Survey on Ontology Mapping. *SIGMOD Rec.*, 35(3):34–41, September 2006.

[dBBJ+04] Frank S de Boer, Marcello M Bonsangue, Joost Jacob, Andries Stam, and L Van der Torre. A logical viewpoint on architectures. In *Enterprise Distributed Object Computing Conference, 2004. EDOC 2004. Proceedings. Eighth IEEE International*, pages 73–83. IEEE, 2004.

[FDP+12] Christophe Feltus, Eric Dubois, Erik Proper, Iver Band, and Michaël Petit. Enhancing the ArchiMate Standard with a Responsibility Modeling Language for Access Rights Management. In *Proceedings of the Fifth International Conference on Security of Information and Networks*, SIN '12, pages 12–19, New York, NY, USA, 2012. ACM.

[Fra02] Ulrich Frank. Multi-perspective enterprise modeling (MEMO) conceptual framework and modeling languages. In *System Sciences, 2002. HICSS. Proceedings of the 35th Annual Hawaii International Conference on*, pages 1258–1267. IEEE, 2002.

[GFSN+11] Marcela Genero, Ana M Fernández-Saez, H James Nelson, Geert Poels, and Mario Piattini. Research review: a systematic literature review on the quality of UML models. *Journal of Database Management (JDM)*, 22(3):46–70, 2011.

[HFL12] Constantin Houy, Peter Fettke, and Peter Loos. Understanding Understandability of Conceptual Models. What Are We Actually Talking about? In Paolo Atzeni, David Cheung, and Sudha Ram, editors, *Conceptual Modeling*, volume 7532 of *Lecture Notes in Computer Science*, pages 64–77. Springer Berlin Heidelberg, 2012.

[IJLP12] M.-E. Iacob, H. Jonkers, Mark M. Lankhorst, and Henderik A. Proper. *ArchiMate 2.0 Specification*. The Open Group, 2012.

[JI09] Henk Jonkers and Maria-Eugenia Iacob. *Performance and cost analysis of service-oriented enterprise architectures*. IGI Global, Hershey, PA, 2009.

[JIV+14] Pontus Johnson, MariaEugenia Iacob, Margus Välja, Marten Sinderen, Christer Magnusson, and Tobias Ladhe. A method for predicting the probability of business network profitability. *Information Systems and e-Business Management*, pages 1–27, 2014.

[KBJK03] Harald Kühn, Franz Bayer, Stefan Junginger, and Dimitris Karagiannis. Enterprise Model Integration. In Kurt Bauknecht, AMin Tjoa, and Gerald Quirchmayr, editors, *E-Commerce and Web Technologies*, volume 2738 of *Lecture Notes in Computer Science*, pages 379–392. Springer Berlin Heidelberg, 2003.

[KGP12] Sybren de Kinderen, Khaled Gaaloul, and Henderik A. Proper. Bridging value modelling to ArchiMate via transaction modelling. *Software & Systems Modeling*, pages 1–15, 2012.

[Kle09] Anneke Kleppe. *Software Language Engineering: Creating Domain-specific Languages Using Metamodels*. Addison-Wesley, 2009.

[KM10] Pierre Kelsen and Qin Ma. A Modular Model Composition Technique. In David S. Rosenblum and Gabriele Taentzer, editors, *FASE*, volume 6013 of *Lecture Notes in Computer Science*, pages 173–187. Springer, 2010.

[KSJ06] John Krogstie, Guttorm Sindre, and Håvard Jørgensen. Process models representing knowledge for action: a revised quality framework. *European Journal of Information Systems*, 15(1):91–102, 2006.

[Küh06] Thomas Kühne. Matters of (Meta-)Modeling. *Software and System Modeling*, 5(4):369–385, 2006.

[KVZ99] Kurt Kosanke, F Vernadat, and Martin Zelm. CIMOSA: enterprise engineering and integration. *Computers in industry*, 40(2):83–97, 1999.

[Lea13] Marc Lankhorst and et al. *Enterprise Architecture at Work: Modelling, Communication and Analysis*. Springer Publishing Company, Incorporated, 3rd edition, 2013.

[LJH10] Robert Lagerström, Pontus Johnson, and David Höök. Architecture analysis of enterprise systems modifiability–models, analysis, and validation. *Journal of Systems and Software*, 83(8):1387–1403, 2010.

[LJJ+06] Åsa Lindström, Pontus Johnson, Erik Johansson, Mathias Ekstedt, and Mårten Simonsson. A survey on CIO concerns - do enterprise architecture frameworks support them? *Information Systems Frontiers*, 8(2):81–90, 2006.

[LPJ10] M.M. Lankhorst, H.A. Proper, and H. Jonkers. The Anatomy of the ArchiMate Language. *International Journal of Information System Modeling and Design (IJISMD)*, 1(1):1–32, 2010.

[MHS05] Marjan Mernik, Jan Heering, and Anthony M. Sloane. When and How to Develop Domain-specific Languages. *ACM Comput. Surv.*, 37(4):316–344, December 2005.

[MKG13] Qin Ma, Pierre Kelsen, and Christian Glodt. A Generic Model Decomposition Technique and its Application to the Eclipse Modeling Framework. *Software & Systems Modeling*, 2013.

[MLM+13] Ivano Malavolta, Patricia Lago, Henry Muccini, Patrizio Pelliccione, and Antony Tang. What industry needs from architectural languages: A survey. *Software Engineering, IEEE Transactions on*, 39(6):869–891, 2013.

[Moo05] Daniel L Moody. Theoretical and practical issues in evaluating the quality of conceptual models: current state and future directions. *Data & Knowledge Engineering*, 55(3):243–276, 2005.

[Moo09] Daniel L Moody. The "physics" of notations: toward a scientific basis for constructing visual notations in software engineering. *Software Engineering, IEEE Transactions on*, 35(6):756–779, 2009.

[ODPRK08] Audrey Occello, Anne-Marie Dery-Pinna, Michel Riveill, and Günter Kniesel. Managing Model Evolution Using the CCBM Approach. In *Proceedings of 15th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems (ECBS-MBD workshop)*, pages 453–462. IEEE Computer Society, 2008.

[OMG03] OMG. MDA Guide v1.0.1, June 2003.

[OMG13] OMG. Unified Profile for DoDAF and MoDAF (UPDM), version 2.1, August 2013.

[OPW+08] M. Op 't Land, H.A. Proper, M. Waage, J. Cloo, and C. Steghuis. *Enterprise Architecture – Creating Value by Informed Governance*. Enterprise Engineering Series. Springer, Berlin, Germany, 2008.

[QSL12] Dick Quartel, Maarten WA Steen, and Marc M Lankhorst. Application and project portfolio valuation using enterprise architecture and business requirements modelling. *Enterprise Information Systems*, 6(2):189–213, 2012.

[SMBJ09] Sagar Sen, Naouel Moha, Benoit Baudry, and Jean-Marc Jézéquel. Meta-model Pruning. In Andy Schürr and Bran Selic, editors, *MoDELS*, volume 5795 of *Lecture Notes in Computer Science*, pages 32–46. Springer, 2009.

[SN00] August-Wilhelm Scheer and Markus Nüttgens. ARIS Architecture and Reference Models for Business Process Management. In Wil Aalst et al., editor, *Business Process Management*, volume 1806 of *Lecture Notes in Computer Science*, pages 376–389. Springer Berlin Heidelberg, 2000.

[SSK+07] Andrea Schauerhuber, Wieland Schwinger, Elisabeth Kapsammer, Werner Retschitzegger, Manuel Wimmer, and Gerti Kappel. A Survey on Aspect-Oriented Modeling Approaches. Technical report, E188 - Institut für Softwaretechnik und Interaktive Systeme; Technische Universität Wien, 2007.

[STA05] August-Wilhelm Scheer, Oliver Thomas, and Otmar Adam. Process modeling using event-driven process chains. *Process-Aware Information Systems*, pages 119–146, 2005.

[Ver02] F. Vernadat. UEML: Towards a unified enterprise modelling language. *International Journal of Production Research*, 40(17):4309–4321, 2002.

[WS08] Ingo Weisemöller and Andy Schürr. Formal Definition of MOF 2.0 Metamodel Components and Composition. In *MoDELS '08: Proceedings of the 11th international conference on Model Driven Engineering Languages and Systems*, pages 386–400, Berlin, Heidelberg, 2008. Springer-Verlag.

[ZKK07] S. Zivkovic, H. Kuhn, and D. Karagiannis. Facilitate modelling using method integration: An approach using mappings and integration rules. In *proceedings of the 15th European Conference on Information Systems (ECIS)*, 2007.