A Meta-Language for EA Information Modeling – State-of-the-art and Requirements Elicitation

Sabine Buckl, Florian Matthes and Christian M. Schweda

Chair for Software Engineering of Business Information Systems (sebis),
Technische Universität München,
Boltzmannstr. 3, 85748 Garching, Germany
{buckls,matthes,schweda}@in.tum.de
http://wwwmatthes.in.tum.de

Abstract. Enterprise architecture (EA) management has not only recently gained importance as means to support enterprises in adapting to a changing market environment and in seizing new business opportunities. This twofold role of EA management in transforming enterprises is connected to describing the current state as well as future states of the EA. Although different information models for the description of these states have yet been proposed in literature, no 'standard' information model exists, and the plurality advocates for the idea that such models are enterprise-specific design artifacts.

In this paper, we explore the fundamentals of EA information modeling, namely the meta-languages underlying today's models, and analyze their diversity. Based on the analysis, we elicit requirements for a "unifying" meta-language. By showing that multi-purpose modeling facilities, as the OMG's UML, fail to fully satisfy these requirements, we establish a future field of research – a meta-language for EA information modeling.

Key words: Enterprise architecture, Modeling language

1 Motivation

Adapting to changes of the environment is a critical success factor for today's enterprises. An instrument, which is commonly regarded to be supportive in this context, is the management of the enterprise architecture (EA). Architecture, in this context, is understood as the "fundamental organization of a system, embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution" [1]. Therefore, EA provides a holistic perspective on the enterprise and its constituents as well as on the environment, and considers business as well as IT-related aspects. The goal of EA management is to provide support for organizational change and enhance the alignment of business and IT [2, 3]. Major tasks of EA management are the description and analysis of the current EA state as well as providing support for the planned evolution of the architecture via comparing future scenarios [4, 5], and selecting the project portfolio guiding the transformation [6].

Although the topic of EA management has been under research by a large community of practitioners [7], academic researchers [8, 9, 10, 11], public authorities [12], and tool vendors [13], since the late 70ies, no commonly accepted standard has yet emerged. The absence of such a standard becomes apparent, if architectural descriptions are considered. Various modeling techniques and complementing information models[1], providing the terms and concepts necessary for describing an EA, have been proposed, which differ widely in respect to the concepts that they employ as well as the coverage of the EA that they aim at [8]. The plurality of available models advocates for the idea of such models being organization-specific design artifacts. This fact is backed by techniques, which describe how an enterprise-specific information model can be developed [8, 11].

Albeit the non-existence of a standard information model, which can be ascribed to the enterprise-specificity of such artifact, the meta-language for describing information models seems to be a candidate for more detailed discussions. In particular, one can expect the requirements for such a meta-language not to be organization-specific. We will see some indications supporting the former hypothesis in Section 2, when we revisit the state-of-the-art in EA information modeling. Nevertheless, the models presented in the EA management approaches listed above, differ widely in respect to the underlying meta-languages. Two possible reasons might cause this plurality:

– The information models differ strongly in respect to the grounding abstractions of the modeling domain or
– the information models were developed independently on arbitrarily chosen conceptualizations for describing the modeling domain.

The truth may most likely lay somewhere between these extremes and poses an interesting subject for in-depth research. This is especially true, as the lack of a single dedicated meta-language for EA information modeling hampers the advance in this field of research. On the one hand, different information models are hardly comparable, if they utilize different meta-languages. On the other hand, models grounded on different meta-languages cannot easily be combined into a comprehensive model, if a using enterprise would like to leverage the advantages of the individual models. Finally, some of the aforementioned EA management approaches bring along information models that are grounded in meta-languages that have originally been developed for other purposes. A prominent example for such a meta-language is the UML [14]. This utilization of general purpose meta-languages tends to result in misusing concepts and in developing isolated meta-language extensions (cf. [15]). Furthermore, most of these meta-languages provide specialized concepts for the originally intended usage scenario, which are not needed for a meta-language for EA information modeling. This may lead to the creation of unintended models, i.e. models that use the language concepts in a way not intended for the modeling domain.

On this background, we regard the topic of a meta-language for EA information modeling to be an open issue. This article approaches the experienced gap in a twofold way. In Section 2, we revisit the state-of-the-art in EA information

[1] In line with the terminology of Buckl et al. (cf. [8]), we use the term "information model" when referring to the meta-model used for EA modeling.

modeling with special emphasis on the meta-languages used to build the corresponding information models. Taking the general perspective of a multi-purpose ontology [16], we propose a set of requirements for a meta-language for EA information modeling in Section 3 and link the requirements back to the underlying statements from EA-related literature. Section 4 sketches how currently used modeling facilities fail to completely fulfill the requirements, and concludes with an outlook on future research ideas in the context.

2 Meta-languages used for EA information modeling

In this section, we analyze the meta-languages underlying prominent EA information modeling approaches. The selection of approaches is based on a literature survey of Aier et al. [17], and the analysis of Schelp and Winter [18].

The Archimate approach to enterprise modeling is among others presented by Jonkers et al. [19] and was further refined in [20]. In these publications, the relevant concepts for describing an EA are introduced using a somewhat intuitive notation. Put in other words, the Archimate information models are presented in a proprietary diagrammatic notation without explicit reference to an underlying meta-language. Only a side-note grants a glimpse on the meta-language, more precisely states, that two distinct types `Thing` and (binary) `Relation` are contained therein. Whereas, other concepts, as e.g. `Properties`, are not directly introduced, there is evidence that an identifying `name` property is associated with every `Thing`. Complementing the information model, Jonkers et al. provide a dictionary of terms, i.e. textually describe the meanings of the concepts introduced in the model.

From an information model point of view, a central publication of the St. Gallen approach to EA management is the "core business metamodel" as introduced by Österle et al. [21]. The publication names the UML [14] as the underlying meta-language of the information model, more precisely, only a "pragmatic" subset of thereof [22]. Mainly, the concepts `Class` and (binary) `Association` are used, while in occasional cases the associations are refined to `Aggregations` or `Association Classes`. Aggregations are thereby often used to describe hierarchies in the information model. Finally, `Generalizations` and `Specializations`, respectively, are used to build the information model of the St. Gallen approach. No further concepts from the UML are employed, such that the meta-model most evidently lacks the capabilities to specify `Properties` owned by the classes as well as `Multiplicities` for constraining the valid instantiations. This fact is nevertheless partially mediated as in instantiations of the core business metamodel or parts thereof the corresponding objects are evidently `named`.

The EAM approach developed at the Royal Institute of Technology in Stockholm, more precisely details of the underlying information modeling technique, are described in the group's book on EA analysis [5]. It has to be emphasized that the approach employs two different modeling techniques – one underlying the relevant "EA viewpoints", as they call the information models, and a different one backing their analysis models. The former technique, more precisely the underlying meta-model is not explicitly alluded to, but its concepts strongly resemble UML concepts as `Class` and (binary) `Association`, where associations are further constrained via `Multiplicities`. The models also employ the concepts of `Generalization` and `Specialization`, respectively, while prominently the concept of `Property` is omitted. Again, evidence exists that at least a `name` property is supplied with every information model class. Properties play also an important role in the meta-model backing the analysis models, denoted as "influence diagrams" in the approach. Influence diagrams are used to operationalize EA-relevant goals to "abstract qualities" that are further detailed towards observable qualities or properties related to architectural elements. In this sense, the meta-model of the influence diagrams introduces the concept of `Goal` and `Quality/Property` as well as two types of relationships: `Definitional Relationships` and `Causal Relationships`, of which the later describe that a change in one property is most likely to cause the change of a related property.

The Open Group Architecture Framework (TOGAF) in its most recent version 9 [7] provides the "architecture content framework", which is grounded in a "content metamodel" introducing the relevant concepts. The meta-language underlying the TOGAF's information model is not explicitly alluded to, but the employed terminology points towards the utilization of an object-oriented meta-language as the UML [14]. More precisely, a subset of the UML concepts is used, namely `Class`, `Property`, and (binary) `Association`. Further, the mechanisms of `Generalization` and `Specialization` are used. Other concepts from the UML are not employed, such that evidently `Multiplicities` for the associations and `typing` for the attributes are missing. Special to TOGAF's "content metamodel" is some sort of packaging mechanism that partitions the information model into six distinct units "Core", "Process", "Governance", "Motivation", "Data Modeling", and "Infrastructure Consolidation". While no precise semantics of the packaging mechanism are given in TOGAF [7], the provided examples exert strong similarities with the UML `Package Merge`, i.e. a mechanism that allows to provide additional specification for one class in a package different from the package, where the class was initially defined. Complementing the information model, TOGAF provides a comprehensive dictionary textually defining both the classes and the properties used in the "content metamodel".

As part of the Systemic Enterprise Architecture Methodology (SEAM), Lê and Wegmann provide in [23] an "object-oriented modeling language for EA". According to the text, the information model's underlying meta-language is the UML [14], of whose concepts conversely only a subset is used. Putting it more precisely, the information model builds on the concepts of `Class` and (binary) `Association`, respectively, of which the latter can further be refined to `Compositions` or via `Association Classes`. Compositions are used as means to impose hierarchies in the architecture model. Also the mechanisms of `Generalization` and `Specialization` are used. Finally, the associations are constrained by `Multiplicities` and by additional formal `Constraints` supplemented in a set-theoretic language. Abstaining from utilizing other concepts of the UML, the meta-language lacks the concept of `Property`. Complementing the information model, Lê and Wegmann (cf. [23]) textually describe the semantics of the information model classes.

The Multi-Perspective Enterprise Modeling (MEMO) approach introduced by Frank in [24] and refined over the years in multiple publications (see among others [25, 26]) puts critical emphasis on the topic of modeling. In this light, it is not surprising that the approach not only brings along a comprehensive set of information models underlying the different special-purpose modeling languages, which constitute the approach. MEMO also explicitly alludes to the meta-language (MML), on which the different MEMO languages are built. This meta-language is described by Frank in [9] and presents itself as an object-oriented language comprised of the conceptions of `MetaEntity`, `MetaAttribute`, and `MetaAssociationLink`[2]. Both properties and associations supply the concept of `Multiplicity`, whereas properties are further *strongly-typed* in a domain-agnostic type-system consisting of basic `Datatypes`. Classes in the MML can be related via the mechanisms of `Generalization` and `Specialization`, and can further be flagged as `abstract`. Beside the rather basic concept of the `Constraint` expressed in an OCL-like syntax [27], the MML supports the sophisticated notion of the `intrinsic` concept. This concept plays a crucial role, when multiple levels of *metaization* are considered. For example an intrinsic property specified on meta-level $n+2$ is linguistically instantiated into a "normal" property on meta-level $n+1$ and can finally be assigned to values on meta-level n. Intrinsic concepts resemble a *potency* of limited depth as introduced by Atkinson and Kühne in [28]. To round up the analysis of the MEMO approach, we should have a closer look on the ScoreML, a special-purpose modeling language outlined by Frank et al. in [26]. The conceptual model of ScoreML introduces the notion of "goal" and decomposes the model towards operationalized "performance indicators". These indicators are conversely associated to classes and concepts from an arbitrary special purpose language for the corresponding relevant part of the overall EA. In the ScoreML, further concepts for defining and relating "performance indicators" are supplied.

The pattern-based approach to EA management as presented in the EAM pattern catalog of Technische Universität München [29] presents a set of information model fragments, called *I-pattern*, for modeling EAs. These fragments use the UML as meta-language, more precisely concepts for describing static aspects as also reflected in the Meta-Object Facility (MOF) [30]. The approach uses abstract and non-abstract `Classes`, *typed* `Properties`, and (binary) `Associations`, which can further be refined to `Compositions` and via `Association Classes`. For typing the properties, domain-specific `Datatypes`, as e.g. `Money` are used where necessary, but lack a comprehensive definition. The mechanisms of `Generalization` and `Specialization`, `Multiplicities` for both properties and associations, and `Constraints` in OCL syntax [27] are used throughout the information model fragments. In addition, selected infor-

[2] Translated to the terminology of the UML, the three concepts denote `Classes`, `Properties`, and `AssociationEnds`, respectively.

mation model fragments represent relevant architecture performance indicators via `derived` properties that are also supplied with *derivation rules* expressed in OCL. Annotating the information model fragments, the EAM pattern catalog [29] further supplies textual descriptions for the classes' meanings. In a joint publication of KTH Stockholm, TU Berlin, and Technische Universität München [31] Buckl et al. discuss along a practical example how the MOF can be extended to support goal-specific dependency modeling in the EA. In this context the "definitional dependencies" are introduced to the meta-language provided by MOF (cf. also Buckl et al. [15]).

Complementing the forestanding analysis of the state-of-the-art in information models for describing EAs as found in literature, we further analyze the models and underlying meta-languages as used in today's prominent EA management tools. Abstaining from enumerating the very details of the meta-language tool-by-tool, we summarize common characteristics here and come back to "exotic" characteristics, when eliciting the meta-language requirements in Section 3. The most prominent tools as analyzed by Matthes et al. in [13] build on an object-oriented meta-language comprised of the concepts `Class`, `Property`, and (binary) `Association`, while the latter concept is sometimes substituted by *mutual* properties. Further, the tools only support *strongly-typed* properties, often providing a rich set of domain-appropriate `Datatypes` as *money* or *date*. Rounding up this short exposition of common characteristics of the tool's meta-languages, we can say that the mechanisms of `Generalization` and `Specialization` are widely supported.

3 Requirements

Based on our findings from Section 2, we present requirements for a meta-language for EA information modeling. The subsequent list is thereby not meant to be exhaustive, but delineates requirements that can be grounded well in existing literature on EA management. In this respect, we aim at presenting the most relevant of these requirements as basis for a subsequent analysis of the suitability of multi-purpose modeling facilities. To illustrate our requirements, we give, where possible, illustrative object-oriented models (using the UML[3] [30]), reflecting a typical situation, in which the corresponding requirement applies.

(R1) ***Modeling primitives*** The different EA information models are built on a small set of modeling primitives that conversely must be supported by the corresponding meta-language. Most prominently, these primitives are `Classes`, typed `Properties`, and binary `Associations`. On both properties and associations prominently multiplicity constraints apply, i.e. a *lower bound* and an *upper bound* can be specified. In particular, one must have the chance to express

[3] The utilization of the UML should not be misinterpreted as statement to use UML as meta-language for EA information modeling. We nevertheless found it a both commonly used and convenient language for describing object-oriented models of any kind.

that a property is *mandatory*. Reflecting the forestanding primitives against the background of Guizzardi's ontology [16], we could rise the question, whether a `name`-property should be specifically accounted for. In the sense of Guizzardi, any "thing", i.e. instance of class in our terms, has an *identifying* property. In the context of EA information modeling, we could sensibly assume that the name of a thing would be such property. Delving deeper into ontological subtleties, we could ask on the exact understanding of "class". More precisely, we could generalize the class concept towards the concept of the "universal", as discussed by Guizzardi in [16]. For reasons of brevity, we abstain from detailing such considerations here, which would nevertheless be beneficial to refine the modeling primitives. The same applies for the subtleties of `Generalization` mechanisms, whereas basic inheritance must be supported by the meta-language.

As a lightweight counterpart for classes, the meta-language must also supply a `Datatype` concept complemented with a set of basic data types reflecting typical EA related concepts as *money*, *time*, or *probabilities*. In extension to this, an `Enumeration` concept is needed in the meta-language to specify domain limitations for certain properties, see e.g. [8, 32, 33]. In respect to associations, the meta-language must support a concept to *reify* associations, i.e. understand instantiated associations as "things" that themselves can have properties and associations again. The need to reify associations is clearly supported by the models of Österle [21] and Kurpjuweit, as well as the ones found in the EA management pattern catalog of Technische Universität München [29], which all use the UML concept of the `Association Class`. In line with the argumentation of Guizzardi in [16], a `Relator`-concept should be used to reify an assocation providing a clear distinction between relationship and thing-nature of an element.

(R2) ***Hierarchy modeling*** Hierarchies are prominently used throughout modeling EAs. Thereby, the models reflect hierarchic, i.e. tree-like, structures in the real-world enterprise, e.g. organizational structures, business process hierarchies, or the architecture of component-based business applications. In structures, like the aforementioned ones, the outgoing relationships of *supernodes*, i.e. elements on higher hierarchy-level, are derived from their corresponding *subnodes*. A typical model fragment, illustrating such hierarchy modeling is shown in Figure 1, although pure UML is not sufficient to clearly constrain the model to a hierarchy. To achieve this, further constraints, e.g. using the OCL [27] would be necessary to demand that the parent-child-relationships and its transitive closure, respectively, are acyclic. Examples of hierarchy modeling can be found in different EA management approaches, e.g. the approach presented by Fischer and Winter [34]. According to Matthes et al. [13], many of the currently available EA management tools support hierarchy modeling.

Resorting to the ontological foundations presented by Guizzardi in [16], modeling hierarchies can be regarded a special case of *whole-part-relationships*. The corresponding ontological discipline of *mereology* presents a broad field of possible properties that whole-part-relationships may have. Especially the question, if such a relationship is transitive, would deserve special attention. We abstain from in-depth considerations on this topic here, which may in accordance to Kurpjuweit [35] be also relevant in the context of EA information modeling.

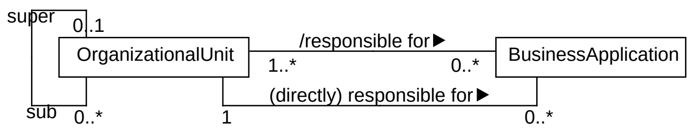

Fig. 1. Object-oriented model of a hierarchy

(R3) ***Constraint specification*** The meta-language must support language concepts for specifying quantified mathematical and logical expressions over the information model, acting as constraints in model instantiation. While as far as possible, the concepts of the meta-language itself should supply mechanisms that restrict their instantiation in order to prevent *unintended models*, there might exist multiple domain-specific constraints that cannot be incorporated in terms of e.g. multiplicities. A simplistic example for such domain-inherent constraint is described in the EA management pattern catalog [29], where a constraint is used to demand that a project starts before it ends (`startDate < endDate`).

(R4) ***Dependency explication*** In the context of EA management, many authors, e.g. Niemann [36], express that architectural descriptions are mostly about the relationships between the architectural elements. Dependency explication extends the simple understanding of the relationships towards a more dynamic notion of relation, i.e. dependency, where the EA model can express that an architectural property of one concept is dependent on architectural properties of related concepts. Such dependency modeling can take the simple form of *rules for derivation* as presented by Lankes and Schweda [33] or Frank et al. [26]. But also more complex *cause-effect relationships* between architectural properties may exist, reflecting the behavioral dynamics of the EA. These dependencies are accounted for by different relevant approaches in the field of EA management, e.g. the ones of Buckl et al. [31], Johnson and Ekstedt [5] or of Yu et al. [37]. Dependency modeling can further be understood as generalization of transitive relationship modeling as presented by van Buuren et al. in [38].

(R5) ***Multi-level modeling*** The demand for multi-level modeling applies to many fields in which – speaking in terms of Guizzardi's ontology, cf. [16] – *things* and their corresponding *sortals* should be modeled simultaneously. A related discussion is undertaken by Engelbert and Heymans in [39]. To exemplify the demand for multi-level modeling in the context of EA management, we present a typical *type-item* pattern found in an information model for EA management in the EAM Pattern Catalog [40], see Figure 2. The information model Technology and Connector Usage is used to model architectural standardization on the one hand on the level of architectural guidelines (e.g. three-tier architecture) and on the other hand on the level of actual technologies and technology stacks. This leads to the typical type-item dichotomy that can be found multiple times in other EA models, e.g. described by Matthes et al. [13] or Frank [24]. The problems of modeling the type-item dichotomy by using object-oriented means, become apparent not only along the duplication of concepts, but also with the demand to add further constraints to ensure modeling consistency.

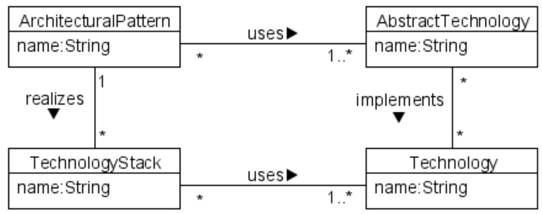

Fig. 2. EA information model exemplifying the type-item pattern

(R6) ***Packaging and package relationship mechanisms*** Relating back to the stakeholder-centric perspective on architectural modeling as advocated by the ISO standard 42010 [1], it seems sensible to decompose the overall architecture into different *areas of interest*, which reflect the stakeholders' architectural concerns. These concerns do nevertheless not exist in isolation, but relate to each other in manifold ways. This can be illustrated along a simple example describing the relationships between the business applications in an enterprise. While a software architect might only be interested to know about the relationships between the application, an infrastructure architect might have a more detailed concern in this respect, needing additional information on the used information exchange protocols, etc. In this example, one might sensibly say that the software architect's concern is totally included in the infrastructure architect's one. The different concerns of the stakeholders thereby correspond to their *base-level* of architectural knowledge, such that in quite some cases inclusion relationships as the one illustrated above, may be derivable. A prominent example for relationships of that type can be found in the stakeholder-oriented approach to EA management presented by Aier in [11]. Similarly, the pattern-based approach to EA management presented by Ernst in [41] establishes relationships between the different information model patterns, reflecting relationships in their represented concerns. In this vein, the types of relationships between patterns as discussed by Noble in [42] may serve as basis for defining the relevant classes of relationships between EA information model fragments. This modeling of relationships between model fragments is further advocated in the discussions of Kurpjuweit and Aier in [22], where they propose to utilize a *composition operator* to consistently aggregate architectural model concepts. The authors argue that the thereby abstracted models are useful for creating EA descriptions, as the modeler is not forced to specify intermediary concepts, if he does not have knowledge about them, i.e. allow for switching the base-level perspective.

(R7) ***Intentional semantics*** EA models, i.e. instantiations based on EA information models, are used as means to support communication among different interest groups in an organization. The employees in these interest groups most likely have differing educational backgrounds and may use a different terminology in respect to the enterprise. The EA information models target to comprehensively describe the "universe of discourse", i.e. the relevant parts of the enterprise, and hence may fall for ambiguities concerning the understanding of the used terms. To prevent communication issues, the meta-language must provide techniques and mechanisms suitable for describing the meaning of the modeled elements. In this sense, two generally different approaches can be distinguished. The meta-language may allow to supply a description for each modeled element. In line with the understanding of Kamlah and Lorenzen [43], such method would shape a *linguistic community* embracing all relevant EA stakeholders. An alternative approach would comprise mechanisms to supply stakeholder-specific descriptions and names for the modeled elements. This approach accounts for the linguistic plurality in an enterprise, and provided a distinct and consistent terminology for every relevant linguistic community in the enterprise.

4 Summary and Outlook

Summarizing, we can state that mostly two meta-languages are used to build EA information models, namely UML (or more precisely subsets of the UML infrastructure) and the special purpose MML of Frank [9]. Table 1 shows how the two meta-languages fulfill the requirements specified in Section 3. The fulfillment of each requirement ranges from nearly complete fulfillment (●) via partial fulfillment (◑) to complete lack of support (○).

	R1[1]	R2[2]	R3	R4[3]	R5	R6	R7[8]
MML + OCL + ScoreML	◑	◑	●	◑	◑[4]	○[6]	○
UML (infrastructure) + OCL	◑	◑	●	◑	○[5]	◑[7]	○

[1] Both UML and MML do not support domain-specific datatypes as money or date.
[2] Hierarchies can be modeled using additional constraints in OCL.
[3] OCL allows to specify and operationalize dependencies but does not support pure specification without derivation rule.
[4] The MML provides the notion of the "intrinsic" feature, that allows two-level instantiation.
[5] The UML follows a strict class-object-dichotomy, i.e. a single-level instantiation.
[6] The MML only supplies a simple packaging mechanism without package composition.
[7] The UML package merge allows model element re-use on class level.
[8] Both UML and MML do not supply a mechanism for specifying the meaning of a concept.

Table 1. Comparison of possible meta-languages for EA information modeling

Forestanding Table 4 indicates that as-of-today none of the used meta-languages fully satisfies the elicited requirements. While one might argue that this may ascribe to the fact that yet no such language was needed, we take a different position. The absence of clear references to the underlying meta-language in many of the EA information modeling approaches outlined in Section 2 seems to us an indication towards the missing engagement in this field.

Our paper does not present a comprehensive meta-language for EA information modeling, nor does it claim to present an embracing set of requirements for such a language. The requirements presented in Section 3 in contrast formulate a "base line" for any meta-language for EA information modeling, retaining potential for future extensions. Especially, the aspect of *uncertainty* (cf. Johnson et al. [44] or Aier et al. [45]) as well as the aspect of *temporality*, as discussed by Buckl et al. in [6], may be of relevance for a meta-language. In the context of temporality also the question of *non-rigid* typing, see e.g. Guizzardi [16], may play an important role and lead to additional requirements.

The findings of the paper may – notwithstanding the aforementioned limitation – provide substantial input for the development of the topic, i.e. for finding or designing a domain appropriate meta-language for EA information modeling. We see such language as very beneficial for the overall advancement of the field, as clear and concise modeling of relevant concepts may allow to compare and relate the different models proposed in the EA management approaches. Finally, a specialized meta-language could lay the basis for a toolset for EA information modeling that should also be valuable for implementing and supporting EA management functions in practical environments.

References

1. International Organization for Standardization: Iso/iec 42010:2007 systems and software engineering – recommended practice for architectural description of software-intensive systems (2007)
2. Henderson, J.C., Venkatraman, N.: Strategic alignment: leveraging information technology for transforming organizations. IBM Systems Journal **32**(1) (1993) 472–484
3. Luftman, J.N.: Competing in the Information Age – Align in the Sand. 2nd edn. Oxford University Press, New York, NY, USA (2003)
4. Holschke, O., Närman, P., Flores, W.R., Eriksson, E., Schönherr, M.: Using enterprise architecture models and bayesian belief networks for failure impact analysis. In Aier, S., Johnson, P., Schelp, J., eds.: Pre-Proceedings of the 3rd Workshop on Trends in Enterprise Architecture Research, Sydney, Australia (2008) 33–46
5. Johnson, P., Ekstedt, M.: Enterprise Architecture – Models and Analyses for Information Systems Decision Making. Studentlitteratur, Pozkal, Poland (2007)
6. Buckl, S., Ernst, A.M., Matthes, F., Schweda, C.: An information model for managed application landscape evolution. Journal of Enterprise Architecture (JEA) **5**(1) (2009) 12–26
7. The Open Group: TOGAF "Enterprise Edition" Version 9. `http://www.togaf.org` (cited 2010-02-25) (2009)
8. Buckl, S., Ernst, A.M., Lankes, J., Schneider, K., Schweda, C.M.: A pattern based approach for constructing enterprise architecture management information models. In: Wirtschaftsinformatik 2007, Karlsruhe, Germany, Universitätsverlag Karlsruhe (2007) 145–162
9. Frank, U.: The memo meta modelling language (mml) and language architecture (icb-research report). Technical report, Institut für Informatik und Wirtschaftsinformatik, Duisburg-Essen, Germany (2009)
10. Lankhorst, M.: Introduction to enterprise architecture. In: Enterprise Architecture at Work, Berlin, Heidelberg, New York, Springer (2005)
11. Aier, S., Kurpjuweit, S., Riege, C., Saat, J.: Stakeholderorientierte dokumentation und analyse der unternehmensarchitektur. In Hegering, H.G., Lehmann, A., Ohlbach, H.J., Scheideler, C., eds.: GI Jahrestagung (2). Volume 134 of LNI., Bonn, Germany, Gesellschaft für Informatik (2008) 559–565
12. NATO: Nato architecture framework version 3. `http://www.nhqc3s.nato.int/ARCHITECTURE/_docs/NAF_v3/ANNEX1.pdf` (cited 2010-02-25) (2007)

13. Matthes, F., Buckl, S., Leitel, J., Schweda, C.M.: Enterprise Architecture Management Tool Survey 2008. Chair for Informatics 19 (sebis), Technische Universität München, Munich, Germany (2008)
14. Object Management Group (OMG): Uml 2.2 superstructure specification (formal/2009-02-02). `http://www.uml.org` (cited 2010-02-25) (2009)
15. Buckl, S., Ernst, A.M., Schweda, C.M.: An extension to the essential meta-object facility (emof) for specifying and indicating dependencies between properties. Technical report, Technische Universität München (2008)
16. Guizzardi, G.: Ontological foundations for structural conceptual models. PhD thesis, CTIT, Centre for Telematics and Information Technology, Enschede, The Netherlands (2005)
17. Aier, S., Riege, C., Winter, R.: Unternehmensarchitektur – literaturüberblick stand der praxis. Wirtschaftsinformatik **50**(4) (2008) 292–304
18. Schelp, J., Winter, R.: Language communities in enterprise architecture research. In: DESRIST '09: Proceedings of the 4th International Conference on Design Science Research in Information Systems and Technology, New York, NY, USA, ACM (2009) 1–10
19. Jonkers, H., van Burren, R., Arbab, F., de Boer, F., Bonsangue, M., Bosma, H., ter Doest, H., Groenewegen, L., Scholten, J., Hoppenbrouwers, S., Iacob, M.E., Janssen, W., Lankhorst, M., van Leeuwen, D., Proper, E., Stam, A., van der Torre, L., van Zanten, G.: Towards a language for coherent enterprise architecture descriptions. In: 7th International Enterprise Distributed Object Computing Conference (EDOC 2003), Brisbane, Australia, IEEE Computer Society (2003)
20. Jonkers, H., Goenewegen, L., Bonsangue, M., van Buuren, R.: A language for enterprise modelling. In Lankhorst, M., ed.: Enterprise Architecture at Work. Springer, Berlin, Heidelberg, New York (2005)
21. Österle, H., Winter, R., Hoening, F., Kurpjuweit, S., Osl, P.: Der St. Galler Ansatz des Business Engineering: Das Core Business Metamodel. Wisu – Das Wirtschaftsstudium **2**(36) (2007) 191–194
22. Kurpjuweit, S., Aier, S.: Ein allgemeiner Ansatz zur Ableitung von Abhängigkeitsanalysen auf Unternehmensarchitekturmodellen. In: 9. Internationale Tagung Wirtschaftsinformatik (WI 2007), Wien, Austria, Österreichische Computer Gesellschaft (2009) 129–138
23. Le, L.S., Wegmann, A.: Definition of an object-oriented modeling language for enterprise architecture. System Sciences, 2005. HICSS '05. Proceedings of the 38th Annual Hawaii International Conference on (2005) 179c
24. Frank, U.: Multi-perspective enterprise modeling (memo) – conceptual framework and modeling languages. In: Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS 2002), Washington, DC, USA (2002) 1258–1267
25. Kirchner, L.: Eine Methode zur Unterstützung des IT-Managements im Rahmen der Unternehmensmodellierung. PhD thesis, Universität Duisburg-Essen, Berlin, Germany (2008)
26. Frank, U., Heise, D., Kattenstroth, H., Schauer, H.: Designing and utilising business indicator systems within enterprise models – outline of a method. In: Modellierung betrieblicher Informationssysteme (MobIS 2008) – Modellierung zwischen SOA und Compliance Management 27.-28. November 2008, Saarbrücken, Germany (2008)
27. OMG: Object constraint language (ocl) available specification, version 2.0 (formal/06-05-01) (2006)

28. Atkinson, C., Kühne, T.: Reducing accidental complexity in domain models. Software and Systems Modeling (2007) 345–359
29. Chair for Informatics 19 (sebis), Technische Universität München: Eam pattern catalog wiki. http://eampc-wiki.systemcartography.info (cited 2010-02-25) (2010)
30. OMG: Meta object facility (mof) core specification, version 2.0 (formal/06-01-01) (2006)
31. Buckl, S., Franke, U., Holschke, O., Matthes, F., Schweda, C.M., Sommestad, T., Ullberg, J.: A pattern-based approach to quantitative enterprise architecture analysis. In: 15th Americas Conference on Information Systems (AMCIS), San Francisco, CA, USA (2009)
32. Johnson, P., Johansson, E., Sommestad, T., Ullberg, J.: A tool for enterprise architecture analysis. In: 11th IEEE International Enterprise Distributed Object Computing Conference (EDOC 2007), 15-19 October 2007, Annapolis, Maryland, USA, Annapolis, Maryland, USA, IEEE Computer Society (2007) 142–156
33. Lankes, J., Schweda, C.M.: Using metrics to evaluate failure propagation and failure impacts in application landscapes. In: Multikonferenz Wirtschaftsinformatik, Berlin, Germany, GITO-Verlag (2008)
34. Fischer, R., Winter, R.: Ein hierarchischer, architekturbasierter ansatz zur unterstützung des it/business alignment. In Oberweis, A., Weinhardt, C., Gimpel, H., Koschmider, A., Pankratius, V., Schnizler, eds.: Wirtschaftsinformatik 2007, Karlsruhe, Germany, Universitätsverlag Karlsruhe (2007) 163–180
35. Kurpjuweit, S.: Stakeholder-orientierte Modellierung und Analyse der Unternehmensarchitektur. PhD thesis, Universität St.Gallen (2009)
36. Niemann, K.D.: From Enterprise Architecture to IT Governance – Elements of Effective IT Management. Vieweg+Teubner, Wiesbaden, Germany (2006)
37. Yu, E., Strohmaier, M., Deng, X.: Exploring intentional modeling and analysis for enterprise architecture. In: Proceedings of the EDOC 2006 Conference Workshop on Trends in Enterprise Architecture Research (TEAR 2006), Hong Kong, IEEE Computer Society Press (2006) 32–32
38. van Buuren, R., Jonkers, H., Maria-Eugenia, Strating, P.: Composition of relations in enterprise architecture models. In: The second International Conference on Graph Transformation (ICGT), Roma, Italy (2004) 39–53
39. Englebert, V., Heymans, P.: Towards more extensible metacase tools. In Krogstie, J., Opdahl, A.L., Sindre, G., eds.: CAiSE. Volume 4495 of Lecture Notes in Computer Science., Berlin, Heidelberg, Germany, Springer (2007) 454–468
40. Chair for Informatics 19 (sebis), Technische Universität München: Eam pattern catalog wiki. http://eampc-wiki.systemcartography.info (cited 2010-02-25) (2009)
41. Ernst, A.: Enterprise architecture management patterns. In: PLoP 08: Proceedings of the Pattern Languages of Programs Conference 2008, Nashville, USA (2008)
42. Noble, J.: Classifying relationships between object-oriented design patterns. In: Australian Software Engineering Conference (ASWEC), Los Alamitos, CA, USA, IEEE Computer Society (1998) 98–107
43. Kamlah, W., Lorenzen, P.: Logische Propädeutik: Vorschule des vernünftigen Redens. 3rd edn. Metzler, Stuttgart, Germany (1996)
44. Johnson, P., Nordström, L., Lagerström, R.: Formalizing analysis of enterprise architecture. In: Interoperability for Enterprise Software and Applications Conference, Bordeaux, France, Springer (2006) 10
45. Aier, S., Buckl, S., Franke, U., Gleichauf, B., Johnson, P., Närman, P., Schweda, C.M., Ullberg, J.: A survival analysis of application life spans based on enterprise architecture models. In: 3rd International Workshop on Enterprise Modelling and Information Systems Architectures, Ulm, Germany (2009) 141–154