Enterprise Architecture Modelling[α]

Purpose, requirements and language

Hend*erik* A. Proper*†, Marija Bjeković*, Bas van Gils‡, Sybren de Kinderen§
*Luxembourg Institute of Science and Technology, Luxembourg
†University of Luxembourg, Luxembourg
‡Strategy Alliance, the Netherlands
§University of Duisburg-Essen, Germany
E-mail: E.Proper@acm.org, M.Bjekovic@list.lu,
Bas.vanGils@strategy-alliance.com, Sybren.deKinderen@uni-due.de

***Abstract*—This paper is concerned with the engineering of modelling languages, and the engineering of modelling languages for enterprise architecture in particular. In line with this, this paper aims to provide two key contributions. Firstly, it aims to contribute to the field of (enterprise) modelling language engineering by refining an existing approach for the engineering of requirements on modelling languages. It will do so, by suggesting specific classes of purpose and quality criteria for models and their creation. The second contribution involves an initial application of the latter approach, in terms of a critical reflection on the language requirements as formulated for the ArchiMate standard.**

***Keywords*—enterprise modelling, enterprise architecture, modelling language engineering**

I. Introduction

In line with [72, 11, 62] we consider models to be purposefully created artefacts. Even more, we regard the modelling languages used to express these models as purposely created, or selected, artefacts as well. In doing so, we certainly do not want to exclude the possibility of modelling languages emerging, or evolving, naturally when creating actual models [80]. However, we do take the perspective that in our engineering field(s), be it information systems engineering, business process engineering, or enterprise engineering, it is important to reflect on the purposeful design / selection of the modelling languages we use. Ultimately, the purposes of models also define the requirements on modelling languages [48, 21].

We have been involved in several language engineering efforts in the context of e.g. Object-Role Modelling [58, 27, 12], Business Process Modelling [6, 5] and Enterprise Architecture Modelling [45, 14, 25, 20, 22, 54]. This language engineering work, has also triggered reflections on different aspects of the engineering of modelling languages, including experiences in formalising languages [71], the need to explicitly identify the added value of different constructs in modelling languages [59, 16], and the practical usefulness of the languages [24, 61], eliciting and capturing the requirements on modelling languages [31, 15], the quality of models and modelling processes [75] and a fundamental perspective on models, modelling and languages [8, 7].

In this paper, we focus on (modelling) language engineering in the context of enterprise architecture management. In doing so, this paper aims to provide two contributions. First, it aims to contribute to the field of (enterprise) modelling language engineering by refining an existing approach for requirements engineering used in this context [21, 15]. It will do so, by suggesting specific classes of purpose and quality criteria for models and their creation [40, 31, 75]. Secondly, the paper aims to contribute towards the field of enterprise architecture in terms of a critical reflection on the requirements of the ArchiMate [47, 32] language, based on the latter approach.

The remainder of this paper is structured as follows. Section II focuses on the suggested refinements for the requirements engineering approach for language engineering. Section III, then focuses on a reflection of the requirements used in the creation of ArchiMate, while also putting this in the context of recent extensions made to the language, and the general evolution of the field of enterprise architecture. The conclusion of the paper will briefly revisit the contributions of the paper, while also identifying its limitations, and our planned future research, in particular in terms of further development of the requirements engineering approach for the design, selection and tuning of modelling languages.

II. Requirements engineering for enterprise modelling languages

A. Eliciting and capturing requirements

As also argued in [39, 21], not much work has been done with regards to the elicitation of requirements on the design of (conceptual and / or enterprise) modelling languages. Existing work in this regard tends to focus more on technical language engineering considerations, see e.g. the Eclipse Modelling Framework [70].

Furthermore, the requirements of existing modelling languages are not always captured explicitly. This might be understandable for an ad-hoc domain / purpose specific language. However, for the design of standards such as BPMN [51], UML [50], and ArchiMate [32] this is quite surprising, as their designs involve a large amount of labour, and the use of these standards represents an even large amount of labour. One can, for instance find statements such as "*This specification represents the amalgamation of best practices within the business modelling community to define the notation and semantics ...*" [51], but what remains unclear are the underlying requirements and design considerations that have been

[α]This work has been partially sponsored by the *Fonds National de la Recherche Luxembourg* (www.fnr.lu), via the ValCoLa project.

taken into account. In the case of ArchiMate, the document describing the standard [32], does include a discussion on the design approach used for the language, thus providing insight into its design considerations. However, to find its (for version 1.0) requirements requires one to look elsewhere [10, 45].

The (domain specific) modelling language engineering proces as suggested by [21] is, to our knowledge, one of the few explicit descriptions on how to do language engineering. As shown in Figure 1, it involves three phases pertaining to requirements engineering for language design:
1: *Clarification of scope and purpose*, with the aim to produce a description of essential design objectives and (where relevant) a definition of the needed budget.
2: *Analysis of generic requirements*, which aims to specify generic requirements on the modelling language. These requirements are generic in the sense that they should essentially apply to all (domain specific) modelling languages. Examples would include generic requirements with regards to the notation to be used, based on the work by e.g. Moody [49].
3: *Analysis of specific requirements*, aiming to gather the requirements that should apply to the specific language that is to be engineered.

As also suggested in Figure 1, these phases are designed to be highly cyclic in nature.

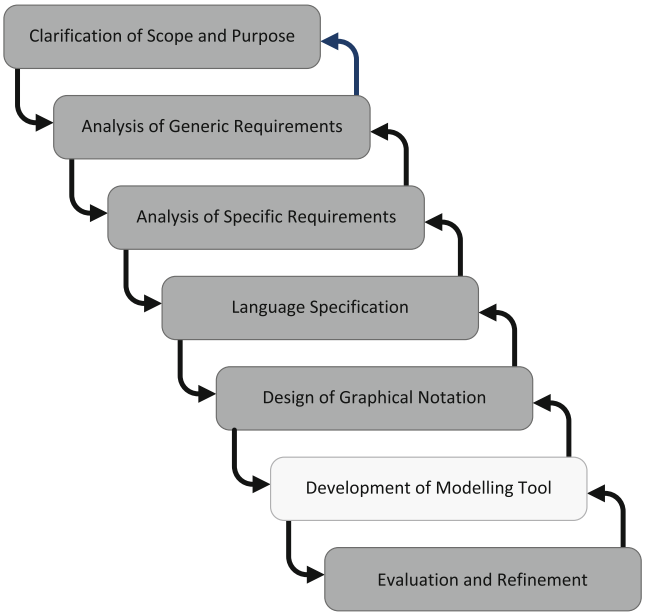

Fig. 1. Conceptual Modelling Language Development (source: [21])

In [15], a possible refinement of this process is provided that enables the explicit capturing of the goals of stakeholders as well as reasoning about the added value of the language constructs towards the achievement of these latter goals. This has resulted in the vGREL (*v*alue- and *G*oal-oriented *R*equirements *E*ngineering for conceptual modelling *L*anguages) approach [15], which builds on the earlier work as reported in [59, 17, 16] and also includes a value profile for GRL (Goal-oriented Requirements Language) [36] which can be used to capture the desired / offered added value of modelling languages and their constructs.

In terms of Figure 1, vGRL enables a value and goal oriented elicitation and capturing of the *clarification of scope and purpose*, while towards the *analysis of generic requirements* as well as the *analysis of specific requirements* it offers the ability to express the *qualities* required from a modelling language, and map these (in terms of added value) to *stakeholder goals*.

B. Targeted level of genericity

In the context of enterprise architecture management, and the ArchiMate case study in particular, it is important to explicitly identify the desired level of genericity of the modelling language. As illustrated in a design science context [43, 2], it is quite important to identify the situations / scopes of applicability of designed artefacts (such as modelling languages). In this regard, it is also interesting to note that (practitioners created) methods such as TOGAF [74] explicitly refer to the need to tune the methods (and modelling frameworks) to the needs of a specific context. This is, for example, an important task in TOGAF's *preliminary phase*.

In the context of enterprise modelling languages, we suggest to identify three main levels of genericity:

1) *Engagement specific* This is the level at which a language would be tuned towards, or defined, for a specific project or programme in a specific organisational context.
2) *Enterprise specific* At this level, a language would be tuned towards, or designed, for a specific enterprise (as undertaken by one or more organisations), but be generic with regards to the engagements in which it will be used. Of course, within larger organisations, there might be a need to differentiate this based on different levels of organisational aggregation (units specific, departments specific, corporation wide).
3) *Generic* This is the most generic level. The language would be applicable (and tuneable) to different enterprise contexts, and different engagements. This is where we find general purpose languages such as ArchiMate [32], BPMN [51] and UML [50].

Due to their genericity, and the need to be applied in specific enterprise and engagement contexts, general purpose modelling languages typically have a built-in specialisation and / or extension mechanism to allow for situation specific tuning. In defining the requirements on a modelling language, it is therefore important to identify the desired level of genericity, and the need / desire to be able to extend/tune the language in specific situations.

C. Purposes for the creation of models

Inspired by a.o. [67, 62, 19, 72], we [7] define a model as *an artefact acknowledged by the observer(s) as representing some domain for (a) particular purpose(s)*. To define the requirements on a modelling language, it is important to first understand the intended purposes of the models that should be expressible in the language.

We argue that there are fundamentally two top-level classes of purposes for the creation of a model. The most familiar class of purposes concern some *use* of the model as an artefact. Nevertheless, the collaborative elements of the model's creation process can be crucial in arriving at a shared understanding, and / or commitment, among different stakeholders. In such

a situation, the purpose would more strongly involve the collaborative elements of the creation process, than the quality of the resulting model. So, the second top-level class of purpose pertains to its *creation* process.

In the context of our engineering fields, purposes relating to the *use* of the model as an artefact, can of course be specialised / concretised further. For example, in [69, 46], three types of usage contexts have been identified for architecture viewpoints / models:

1) *Designing:* Design viewpoints support architects and designers in the design process from initial sketch to detailed design.
2) *Deciding:* Decision support views assist managers in the process of decision making by offering an insight into cross-domain architecture relations, typically through projections and intersections of underlying models, but also by means of analytical techniques.
3) *Informing:* These viewpoints help to inform any stakeholder about the enterprise architecture, in order to achieve understanding, obtain: commitment, and convince adversaries.

Similarly, purposes pertaining to the *creation* process can be specialised further as well. For instance, in [31, 57] the creation and use of models is approached from the perspective of *knowledge sharing* (and creation) among the actors involved in a development effort. More specifically, in terms of three main classes of goals for modelling processes. The first class are those goals where the aim is to make actors *aware* of knowledge by the use and / or creation of the model. These goals also require an acceptance by the involved actors that the model is indeed a proper representation of the given domain for the given purpose. The second class of goals are the ones that aim to have actors *agree* to the consequences / meaning of the model and its purpose. For example, the model might give insight into a problem that needs to be solved. In that case, an agreement must be reached that this indeed is a problem that needs solving. The third class of goals pertains to goals where the aim is to get actors to *commit* to the consequences / meaning of the model and its purpose. For instance, a model representing a future design of the enterprise may require deep commitment from the stakeholders. It is important to separate agreement and commitment. It is possible to reach an agreement on matters, but is one really committed to bare the consequences?

These three classes of knowledge sharing goals, lead to three classes of purposes for the (collaborative) creation of a model: create *awareness*, achieve *agreement* and obtain *commitment*.

D. Required qualities of models

Based on the *purposes* of a model, and their specific *creation and usage contexts*, we can consider the qualities that would make a model fit for purpose in these specific situations. A modelling language should of course support the creation of models that offer the qualities that will make it fit for purpose.

In [7] it is suggested that modelling involves two main aspects:

1) *Representation:* The extent to which model indeed represents the desired domain / aspect with the desired level of formality / precision / explicitness.
2) *Manifestation:* The manifestation of the model as a physical / digital artefact.

From the perspective of Krogstie's well known SEQUAL [40] framework for quality of (conceptual) models, the *representation* aspect can be refined in terms of Semantic Quality, Domain quality, and Knowledge Quality, while the *manifestation* aspect can be refined to Physical Quality, Empirical Quality and Syntactic Quality respectively.

E. Putting it together

In terms of the three initial phases shown in Figure 1, we suggest to go about as follows in combining the above discussed ingredients:

1) *Clarification of scope and purpose:* Identify the desired level of genericity. Also identify a catalogue of the key usage scenario's for models. In the context of enterprise architecture, this could be a list of typical situations in which the models are used / created / manipulated.
2) *Analysis of generic requirements:* Identify general requirements on *representation* and *manifestation*, in particular in terms of the overall needed precision / explicitness and quality of the notation. Also, given the choice for a level of genericity, requirements may be formulated on the desire/need to tune / extend the language towards more specific situations.
3) *Analysis of specific requirements:* Using the catalogue of key usage scenarios of the model, one can classify these based on the question if it involves *designing*, *deciding* and *informing*, and the relative importance of the knowledge sharing goals: *aware*, *agree* and *commit*. This can be captured in a structured in terms of underlying goals and added value in terms of vGREL [15].
   Based on this analysis, more specific requirements towards *representation* and *manifestation* can be formulated, linked to the different usage scenarios.

III. Reflection on the ArchiMate requirements

At the start of the development of the ArchiMate language, a survey was conducted among the key stakeholders to identify the key requirements of the language [10], which was then used as a base to design the language [45].

As discussed in [44], requirements / features of relevant other architecture frameworks / languages (e.g. [35, 33, 34, 63, 73]) were considered as well in the development of ArchiMate. Nevertheless, the focus was on the requirements as stipulated by the architects from the industry partners participating in the development project. These requirements were based on their day-to-day work as architects, and associated modelling activities.

Based on the approach as suggested in the previous section, we now proceed to analyse the original ArchiMate requirements. In doing so, we mainly base ourselves on the documentation provided in [10].

As described in [10]: "*Requirement: In a first phase customer-specific requirements were elicited by interviewing and studying architectural material at the customer site. Customer requirements together with a general description of the customers architectural practice have been put down in customer-specific requirements documents*". The requirements documented in [10] indeed offer an aggregated level, based on the individual requirements brought forward by the participating client organisations. Furthermore, a project steering board (with representatives of the client organisations) classified the requirements based on 'Must Have' and 'Nice to Have' and a apriori estimation of the feasibility ('Yes' and 'Difficult').

It is also interesting to note the statement "*Requirement: During the requirements engineering process we have found that it was not possible to determine a detailed set of requirements concerning concepts and relations between concepts. Therefore, the requirements in this document should be viewed as high level directions and preconditions for the project and not as detailed specifications for the ArchiMate products.*" which pays witness to the fact that one needs to iterate over the phases as shown in Figure 1.

A. Clarification of scope and purpose

ArchiMate's requirements document does not make an explicit choice with regard to the desired level of genericity of the resulting language. However, one can infer from statements such as "*Requirement: ... the conceptual work of ArchiMate also includes mappings of these concepts to company-specific concepts (or to other description languages used within the companies ...*" that the language was clearly intended to be at the generic level. At least, generalising over the participating client organisations: ABN-AMRO, ABP and the Dutch Tax Administation. Arguably, this implies a generalisation over "information intensive service organisations". As such, one cannot, a priori, expect ArchiMate to be applicable in e.g. a Defence context or a manufacturing environment. This is indeed where one may consider 'competing' standards, such as GERAM [34], UPDM [52] and ARIS [63]. At the same time, as mentioned above, it remains import to derive the language requirements from the actual intended users, rather than relying on a generic 'one-size-fits many' reference framework.

The ArchiMate requirements document does provide a reflection on the general role of architecture in organisations. As it is based on interviews with architecture experts from the participating industrial partners, it does reflects the different *Requirement: knowledge sharing purposes* of a model, and their specific *Requirement: usage contexts* for enterprise architecture models in the participating organisations. Regretfully, these individual interviews are not publicly available for further investigation.

Towards future evolution of the ArchiMate standard, it might be worth while to update the understanding of the usage context of ArchiMate. Especially since the understanding of the role of enterprise architecture (management) has evolved considerably since the start of the ArchiMate project. Enterprise architecture is increasingly seen as the key instrument to support organisations in managing the direction in which it evolves [53, 30, 4, 64, 56, 25, 24]. At its maximum, enterprise architecture management can provide insight into the current structures of an organisation and its enterprise(s), the impact of different alternatives towards the future, as well as monitor the compliance of actual changes made in the organisation (e.g. in terms of development projects) in relation to a desired path towards the future. Recent work also provides valuable empirical insights into the different ambition levels at which organisations actually position and use enterprise architecture (management) [42, 41].

B. Analysis of generic requirements

The ArchiMate requirements document not only contains requirements towards the language itself, but also refers to requirements on the research project as such. For example it refers to the need to have a "*Requirement: overview of the current state of the art in modelling languages and frameworks for architects*", which indeed makes sense as a way to provide further input into the specific requirements. Regretfully, it is not clear how this overview resulted in more detailed requirements. The book reporting on the results of the project as a whole, does indeed contain the requested overview [32].

Conceptual integrity of the language, i.e. in terms of a well integrated and consistent meta-model, is also identified as an important generic requirement. As stated in [45]: "*Requirement: Cobbling together a language from many different sources (viz. UML) is not the ideal way to achieve this.*"

Even though ArchiMate provides an extension mechanism, the general tendency of standards to grow in size, and the many extensions being suggested, really makes it essential to enable an effective extension and profiling system. It might possibly be wise to conduct re-factoring of the existing language into a number of extensible building blocks, that may be situationally composed. This is likely to be needed to warrant the desired *Requirement: conceptual integrity* of future versions of the standard, especially with mounting pressure to add more and more extensions.

C. Analysis of specific requirements

What we cannot currently trace back from the documentation is to what extend these requirements were linked to specific enterprise architecture usage scenarios. Given the broad user base of ArchiMate, it would certainly be feasible in future research to trace the use of the existing language (and document desired changes) in terms of actual usage scenarios and classify these in terms of the *Requirement: knowledge sharing purposes* of a model, and their specific *Requirement: usage contexts*, as discussed in the previous section.

Specific requirements that are explicitly included in ArchiMate's requirements document are listed below. We have organised them into two categories: *Requirement: representation* and *Requirement: manifestation*. For some, we have added some additional reflections as *Requirement: remarks*.

Representation

Concept coverage: Several domains for grouping concepts have been identified, such as product, process, organisation, information, application, technology (infrastructure, system development, maintenance). The concepts in the language must at least cover the concepts in these domains.

This concept coverage is strongly related to the architectural practices as where in use with the client organisations. See the earlier remark on a focus on 'information intensive service organisations'.

Enterprise level concepts: At an enterprise level, it is important to be able to represent the core elements from the different domains such as product, process, et cetera, as well as the coherence between these aspects. In enterprise architecture models, coherence and overview are more important than specificity and detail. This also implies the need for more coarse grained modelling concepts. As mentioned before, the concepts which can typically be found in modelling language used at the level of specific development projects, such as UML and BPMN, were found to be too fine-grained.

In different usage scenarios to answer e.g. performance, risk analysis, or security concerns, one does see a desire to add more details to the ArchiMate language [20, 22]. Drawing an a priori line between an 'architecture' and a 'project' level of details seems to be difficult if not impossible.

Concept mapping: Organisations and s or individual architects must be able to keep using their own concepts and descriptions in development projects. This requires a mapping from the coarse grained concepts in ArchiMate to the fine-grained concepts used in languages at the project level.

This indeed implies a desire to be able to tune/extend the language towards specific situations.

Unambiguous definitions of concepts: The meaning and definition of the modelling concepts offered by the language must be unambiguous. Every concept must be described taking into account: informal description, specialisation, notation, properties, structuring, rules and restrictions and guidelines for use.

Conformance to international standards: The architecture description language must follow, and whenever possible, influence international standards.

Structuring mechanisms: Composition & decomposition, generalisation & specialisation, and aggregation of concepts must be supported.

Abstraction: It must be possible to model relations at different abstraction levels. For example, relations can be formulated between concepts, groups of concepts or different architectural domains.

Analysis of architectural properties: It must be possible to perform qualitative and quantitative analysis of properties of architectures.

Impact of change analysis: Impact of change analysis must be supported. In general, such an analysis describes or identifies effects that a certain change has on the architecture or on characteristics of the architecture.

Consistency: It must be possible to perform consistency checking of architectures.

Tracing of design decisions: It must be possible to register, trace and visualise the requirements, constraints, design decisions and architectural principles that are used in the construction of the architecture.

This did really not make it to the first version of the language. Has indeed been remedied in version 2.0 release of the standard, and has also been the subject of suggested extensions [54].

Extensibility: The language should be easy to maintain and extend, should the need for new concepts arise.

This requirement re-confirms the desire to be able to tune/extend the language towards specific situations, which could threaten the general requirement of maintaining conceptual integrity.

Practical experiences in the use of the language already indicate a need to extend re-assess some of the above requirements. Based on [13, 24, 61], example considerations include: *Scopes of model use:* There is a need to better deal with shifts in scopes at which the language (and models) are to be used simultaneously. For example, one can produce ArchiMate models at the level of a value chain crossing different organisations. Parts of these models will have more specific interpretations within the scope of a specific participating organisation. This needs refinement relations between the interpretations / models, as well as ways to manage the unavoidable issues with homonyms and synonyms. It would require support for explicit concept management [9].

Stratified implementation design: The need exists to be able to better deal with a distinction between the 'ideal implementation design' versus a 'opportunistic implementation' design. For example, for modularisation reasons one might cluster needed software functionalities into two components. Now imagine that there is a COTS (commercial off the shelve) package / cloud-service available that would cover the needed functionalities of both components. How to capture such trade-offs and associated decisions?

Note: architecture frameworks such as Capgemini's IAF [76] suggest to include a distinction between the ideal *logical* implementation design, and the actual *physical* one based on availability of opportunities in the market.

Explicit capturing of design making status: Even though ArchiMate originally involved viewpoints making a distinction between parts of models there were still sketchy, and part of models that were already more firmly materialised and agreed to, there is a need for more explicit support for such modalities.

Versioning of the models: Though one might argue it is not formally part of a language, there is a growing need to enable different levels of versioning when dealing with architecture models. This may include versions while working on a specific model for a specific purpose, or it might be versions showing the evolution of the enterprise over time [66, 65]. Of course, this links back to past research on model evolution and versioning [38, 55].

Indeed, more recent versions of the ArchiMate standard supports the capturing of different versions over time in terms of a so-called plateaux planning.

Manifestation

Representation of concepts: The visual representation of concepts must be easily adaptable.

As also discussed in [45], this implies a clear separation between modelling concepts and their notation. One could say, a clear separation between abstract syntax and concrete syntax.

Consistency of presentation: The visual presentation needs to be consistent and unambiguous.

Visualisation independence: Visualisation techniques and solutions should be independent of the actual concepts used in a model, i.e., it should be possible to modify / add the concepts in a model without consequences for visualisation techniques.

Visualisation generation: Automatic generation of visualisations from architecture models must be supported.

Viewpoint definition: A viewpoint definition must: state the stakeholder(s) it is created for, define the concerns covered by the viewpoint, and explain how views are created for this viewpoint (in terms of the concepts to be presented and the format of the presentation).

Adaptability of viewpoints: Viewpoints must be adaptable and extensible independent of visualisation techniques.

Viewpoint coverage: ArchiMate has to support often used 'general viewpoints, i.e., viewpoints for frequently occurring stakeholders.

In requirements on the viewpoint we indeed see some traces of making a distinction between the *knowledge sharing purposes* of a model, and their specific *usage contexts*.

Recent research [3, 1, 60] also provides indications that in an enterprise architecture context, models can / should play the role of boundary objects [68] (i.e. bridging between different groups / communities). This of course has ample implications on qualities required from models to play this role, in particular with regards to the *knowledge sharing purposes*.

IV. Conclusion

In this paper we discussed two contributions: a possible refinement of a requirements engineering approach for the design of modelling languages, and a critical reflection on an ArchiMate language for enterprise architecture modelling, based on this approach. We realise that these contributions are still rather limited. We foresee much broader evaluations of the requirements on e.g. enterprise architecture related modelling languages. More specifically, we will pursue three parallel streams of work.

As a first stream, we aim to derive a better understanding of the role of models in enterprise architecture (management) in general, as well as differentiated to different ambitions at which organisations may apply enterprise architecture. This stream will be based on theoretical views on the role of enterprise architecture towards the steering of change (transformations, gradual change, etc) in organisations(e.g. [53, 30, 78, 56, 25, 60]), while combining this with broad empirical studies on the role of enterprise architecture (e.g. [4, 64, 42, 41]), as well as in-depth reports on specific experiences in the use of enterprise architecturee (e.g. [28, 13, 77, 79, 23, 24]). architecture (e.g. [28, 13, 79, 24]). This should enable us to more clearly identify the *knowledge sharing purposes* and *usage contexts* for models in an enterprise architecture context.

In the second stream, we will gather requirements on the modelling languages to be used in an enterprise architecture context. We specifically aim to do so, by also involving practitioners to collect experiences from their day-to-day use of models in an enterprise architecture context. Based on this we hope to be able to record their desires / wishes in terms of the vGRL [15] based representations.

In the third stream, we aim to further validate and refine a method for requirements engineering for the design and selection of modelling languages. In this stream, we will also be drawing more explicitly on the field of design science [26]. Since we regard a modelling language as a purposely engineered artefact, it would be logical to integrate more design science thinking in its creation.

References

[1] R. Abraham. Enterprise Architecture Artifacts As Boundary Objects - A Framework Of Properties. In ECIS [18]. http://aisel.aisnet.org/ecis2013_cr/120/

[2] R. Abraham and S. Aier. Extending Enterprise Architecture Management into an Enterprise-Wide Coordination Service. In *Proceedings der 74. Wissenschaftliche Jahrestagung des Verbandes der Hochschullehrer für Betriebswirtschaft e. V., Bozen, Italy*, 2012.

[3] R. Abraham, H. Niemietz, S. de Kinderen, and S. Aier. Can boundary objects mitigate communication defects in enterprise transformation? Findings from expert interviews. In Jung and Reichert [37], pages 27–40.

[4] R. Abraham, J. Tribolet, and R. Winter. Transformation of Multi-level Systems – Theoretical Grounding and Consequences for Enterprise Architecture Management. In H. A. Proper, D. Aveiro, and K. Gaaloul, editors, *Advances in Enterprise Engineering VII – Proceedings of the 3rd Enterprise Engineering Working Conference (EEWC 2013), Kirchberg, Luxembourg*, volume 146 of *Lecture Notes in Business Information Processing*, pages 73–87. Springer, Heidelberg, Germany, May 2013.

[5] A. P. Barros, A. H. M. ter Hofstede, and H. A. Proper. Essential Principles for Workflow Modelling Effectiveness. In G. G. Gable and R. A. G. Webber, editors, *Proceedings of the Third Pacific Asia Conference on Information Systems (PACIS'97)*, pages 137–147, April 1997.

[6] A. P. Barros, A. H. M. ter Hofstede, and H. A. Proper. Towards Real–Scale Business Transaction Workflow Modelling. In A. Olivé and J. A. Pastor, editors, *Proceedings of the Ninth International Conference CAiSE'97 on Advanced Information Systems Engineering, Barcelona, Spain*, volume 1250 of *Lecture Notes in Computer Science*, pages 437–450. Springer, Heidelberg, Germany, June 1997.

[7] M. Bjeković, H. A. Proper, and J.-S. Sottet. Embracing Pragmatics. In E. Yu, G. Dobbie, M. Jarke, and S. Purao, editors, *Proceedings of the 33rd International Conference on Conceptual Modeling (ER 2014), Atlanta, Georgia*, volume 8824 of *Lecture Notes in Computer Science*, pages 431–444. Springer, Heidelberg, Germany, 2014.

[8] M. Bjeković, H. A. Proper, and J.-S. Sottet. Enterprise Modelling Languages – Just Enough Standardisation? In B. Shishkov, editor, *Business Modeling and Software Design, Third International Symposium, BMSD 2013, Noordwijkerhout, the Netherlands, July 8-10, 2013, Revised Selected Papers*, volume 173 of *Lecture Notes in Business Information Processing*, pages 1–23. Springer, Heidelberg, Germany, 2014.

[9] A. I. Bleeker, H. A. Proper, and S. J. B. A. Hoppenbrouwers. The Role of Concept Management in System Development – A practical and a theoretical perspective. In J. Grabis, A. Persson, and J. Stirna, editors, *Forum proceedings of the 16th Conference on Advanced Information Systems 2004 (CAiSE 2004), Riga, Latvia*, pages 73–82, Riga, Latvia, June 2004. Faculty of Computer Science and Information Technology.

[10] H. Bosma, H. W. L. ter Doest, and M. Vos. Requirements. Technical Report ArchiMate Deliverabe D4.1, TI/RS/2002/112, Telematica Instituut, December 2002.

[11] Bubenko, Jr. J. A., A. Persson, and J. Stirna. An Intentional Perspective on Enterprise Modeling. In *Intentional Perspectives on Information Systems Engineering*, pages 215–237. Springer, Heidelberg, Germany, 2010.

[12] L. J. Campbell, T. A. Halpin, and H. A. Proper. Conceptual Schemas with Abstractions – Making flat conceptual schemas more comprehensible. *Data & Knowledge Engineering*, 20(1):39–85, 1996.

[13] A. Coenen, B. van Gils, C. Bouvy, R. Kerkhofs, and S. Meijer. Business Modeling Experience for a State Pension Voluntary Insurance Case. In Harmsen et al. [29], pages 46–60.

[14] S. de Kinderen, K. Gaaloul, and H. A. Proper. Bridging value modelling to ArchiMate via transaction modelling. *Software & Systems Modeling*, pages 1–15, 2012.

[15] S. de Kinderen and Q. Ma. Requirements Engineering for the Design of Conceptual Modeling Languages-a goal-and value-oriented approach. *Applied Ontology*, 2015.

[16] S. de Kinderen, Q. Ma, and H. A. Proper. Model bundling: Towards a value-based componential approach for language engineering. In *Proceedings of the 8th international workshop on Value Modelling and Business Ontology (VMBO 2014), Berlin, Germany*, 2014.

[17] S. de Kinderen and H. A. Proper. e3-RoME: a value-based approach for method bundling. In S. Y. Shin Shin and J. C. Maldonado, editors, *Proceedings of the 28th Annual ACM Symposium on Applied Computing (SAC 2013)*, pages 1469–1471, Coimbra, Portuga, March 2013.

[18] *Proceedings of the 21st European Conference on Information Systems (ECIS 2013), Utrecht, the Netherlands*, 2013. http://aisel.aisnet.org/ecis2013_materials

[19] E. D. Falkenberg, A. A. Verrijn–Stuart, K. Voss, W. Hesse, P. Lindgreen, B. E. Nilsson, J. L. H. Oei, C. Rolland, and R. K. Stamper, editors. *A Framework of Information Systems Concepts*. IFIP WG 8.1 Task Group FRISCO, IFIP, Laxenburg, Austria, 1998.

[20] C. Feltus, E. Dubois, H. A. Proper, I. Band, and M. Petit. Enhancing the ArchiMate standard with a responsibility modeling language for access rights management. In M. Singh Gaur, A. Elçi, O. B. Makarevich, M. A. Orgun, and V. Singh, editors, *Proceedings of the 5th International Conference of Security of Information and Networks (SIN 2012), Jaipur, India*, pages 12–19. ACM Press, New York, New York, 2012.

[21] U. Frank. Domain-Specific Modeling Languages: Requirements Analysis and Design Guidelines. In I. Reinhartz-Berger, A. Sturm, T. Clark, S. Cohen, and J. Bettin, editors, *Domain Engineering – Product Lines, Languages, and Conceptual Models*. Springer, Heidelberg, Germany, 2013.

[22] K. Gaaloul and H. A. Proper. An Access Control Model for Organisational Management in Enterprise Architecture. In California IEEE Explore, Los Alamitos, editor, *The 9th International Conference on Semantics, Knowledge & Grids, Beijing, China*, pages 135–149, 2013.

[23] B. van Gils. Being successful with TOGAF and ArchiMate. Whitepaper, BiZZdesign, Amersfoort, the Netherlands, 2012.

[24] B. van Gils and S. van Dijk. *The Practice of Enterprise Architecture – Experiences, techniques, and best practices.* BiZZdesign Academy, Amersfoort, the Netherlands, 2014.

[25] D. Greefhorst and H. A. Proper. *Architecture Principles – The Cornerstones of Enterprise Architecture*. Enterprise Engineering Series. Springer, Heidelberg, Germany, 2011.

[26] S. Gregor and D. Jones. The Anatomy of a Design Theory. *Journal of the Association for Information Systems*, 8(5):312–335, 2007.

[27] T. A. Halpin and H. A. Proper. Subtyping and Polymorphism in Object–Role Modelling. *Data & Knowledge Engineering*, 15:251–281, 1995.

[28] A. F. Harmsen, H. A. Proper, and N. Kok. Informed Governance of Enterprise Transformations. In H. A. Proper, A. F. Harmsen, and J. L. G. Dietz, editors, *Proceedings of the 1st NAF Academy Working Conference on Practice-Driven Research on Enterprise Transformations (PRET 2009), held at the 21st International Conference on Advanced Information Systems Engineering (CAiSE 2009), Amsterdam, the Netherlands*, volume 28 of *Lecture Notes in Business Information Processing*, pages 155–180, Amsterdam, the Netherlands, June 2009. Springer, Heidelberg, Germany.

[29] A. F. Harmsen, H. A. Proper, F. Schalkwijk, J. Barjis, and S. J. Overbeek, editors. *Proceedings of the 2nd Working Conference on Practice-driven Research on Enterprise Transformation (PRET 2010)*, volume 69 of *Lecture Notes in Business Information Processing*, Delft, the Netherlands, November 2010. Springer, Heidelberg, Germany.

[30] J. A. P. Hoogervorst. *Enterprise Governance and Enterprise Engineering*. Enterprise Engineering Series. Springer, Heidelberg, Germany, Berlin, Germany, 2009.

[31] S. J. B. A. Hoppenbrouwers, H. A. Proper, and Th. P. van der Weide. Understanding the Requirements on Modelling Techniques. In O. Pastor and J. Falcao e Cunha, editors, *17th International Conference on Advanced Information Systems Engineering, CAiSE 2005, Porto, Portugal*, volume 3520 of *Lecture Notes in Computer Science*, pages 262–276. Springer, Heidelberg, Germany, June 2005.

[32] M.-E. Iacob, H. Jonkers, M. M. Lankhorst, H. A. Proper, and D. A. C. Quartel. *ArchiMate 2.0 Specification*. The Open Group, 2012.

[33] IEEE. Recommended Practice for Architectural Description of Software Intensive Systems. Technical Report IEEE P1471:2000, ISO/IEC 42010:2007, IEEE Explore, Los Alamitos, California, Piscataway, New Jersey, September 2000. The Architecture Working Group of the Software Engineering Committee, Standards Department, IEEE.

[34] IFIP-IFAC Task Force on Architectures for Enterprise Integration. GERAM – The Generalised Enterprise Reference Architecture and Methodology. In P. Bernus, L. Nemes, and G. Schmidt, editors, *Handbook on Enterprise Architecture*, International Handbooks on Information Systems, pages 21–63. Springer, Heidelberg, Germany, 2003.

[35] ISO. *Information technology – Open Distributed Processing – Reference model: Architecture*, 1996. ISO/IEC 10746–3:1996(E).

[36] ITU-T. User Requirements Notation (URN)–Language definition, November 2008. Recommendation Z.151 (11/08). URL: http://www.itu.int/rec/T-REC-Z.151/en. Last accessed on 31-07-2014. http://www.itu.int/rec/T-REC-Z.151/en

[37] R. Jung and M. Reichert, editors. *Proceedings of the 5th International Workshop on Enterprise Modelling and Information Systems Architectures (EMISA 2013), St. Gallen, Switzerland*, volume 222 of *Lecture Notes in Informatics*. Gesellschaft für Informatik, Bonn, Germany, 2013.

[38] R. H. Katz and E. Chang. Managing Change in a Computer–Aided Design Database. In *Proceedings of the Thirteenth International Conference on Very Large Data Bases*, pages 455–462, 1987.

[39] L. Kirchner. *Eine Methode zur Unterstützung des IT–Managements im Rahmen der Unternehmensmodellierung*. Logos, Berlin, 2008.

[40] J. Krogstie. A Semiotic Approach to Quality in Requirements Specifications. In L. Kecheng, R. J. Clarke, P. B. Andersen, R. K. Stamper, and E.-S. Abou–Zeid, editors, *Proceedings of the IFIP TC8 / WG8.1 Working Conference on Organizational Semiotics: Evolving a Science of Information Systems*, pages 231–250, Deventer, the Netherlands, 2002. Kluwer.

[41] N. Labusch, S. Aier, and R. Winter. Beyond enterprise architecture modeling – What are the essentials to support enterprise transformations? In Jung and Reichert [37], pages 13–26.

[42] N. Labusch and R. Winter. Towards A Conceptualization Of Architectural Support For Enterprise Transformation. In ECIS [18]. http://aisel.aisnet.org/ecis2013_cr/116

[43] G. Lahrmann, R. Winter, and M. M. Fischer. Design and Engineering for Situational Transformation. In Harmsen et al. [29], pages 1–16.

[44] M. M. Lankhorst, S. J. B. A. Hoppenbrouwers, H. Jonkers, H. A. Proper, L. van der Torre, F. Arbab, de Boer, F. S., M. Bonsangue, M.-.E. Iacob, A.W. Stam, L. Groenewegen, R. van Buuren, R.J. Slagter, J. Campschroer, M.W.A. Steen, S.F. Bekius, H. Bosma, M.J. Cuvelier, H.W.L. ter Doest, P.A.T. van Eck, P. Fennema, J. Jacob, W. P. M. Janssen, H. Jonkers, D. Krukkert, D. van Leeuwen, P. G. M. Penders, G. E. Veldhuijzen van Zanten, and R. J. Wieringa. *Enterprise Architecture at Work: Modelling, Communication and Analysis*. Enterprise Engineering Series. Springer, Heidelberg, Germany, 4th edition, 2017.

[45] 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.

[46] M. M. Lankhorst, L. van der Torre, H. A. Proper, F. Arbab, and M. W. A. Steen. Viewpoints and Visualisation. In *Enterprise Architecture at Work: Modelling, Communication and Analysis* [44], pages 147–190.

[47] M.M Lankhorst, S.J.B.A. Hoppenbrouwers, H. Jonkers, H.A. Proper,

L. van der Torre, F. Arbab, de Boer, F.S., M. Bonsangue, M.-.E. Iacob, A.W. Stam, L. Groenewegen, R. van Buuren, R.J. Slagter, J. Campschroer, M.W.A. Steen, S.F. Bekius, H. Bosma, M.J. Cuvelier, H.W.L. ter Doest, P.A.T. van Eck, P. Fennema, J. Jacob, W.P.M. Janssen, Jonkers, H., D. Krukkert, D. van Leeuwen, P.G.M. Penders, G.E. Veldhuijzen van Zanten, and R.J. Wieringa. *Enterprise Architecture at Work: Modelling, Communication and Analysis*. Enterprise Engineering Series. Springer, Heidelberg, Germany, 3rd edition, 2012.

[48] M. Mernik, J. Heering, and A. M. Sloane. When and how to develop domain-specific languages. *ACM Computing Surveys*, 37(4):316–344, 2005.

[49] D. L. Moody. The "Physics" of Notations: Toward a Scientific Basis for Constructing Visual Notations in Software Engineering. *IEEE Transactions on Software Engineering*, 35(6):756–779, 2009.

[50] OMG. UML 2.0 Superstructure Specification – Final Adopted Specification. Technical Report ptc/03–08–02, Object Management Group, Needham, Massachusetts, August 2003.

[51] OMG. Business Process Modeling Notation, V2.0. Technical Report OMG Document Number: formal/2011-01-03, Object Management Group, Needham, Massachusetts, January 2011.

[52] OMG. Unified Profile for DoDAF and MODAF (UPDM). Technical Report formal/2012-01-03, Object Management Group, Needham, Massachusetts, June 2012.
http://www.omg.org/spec/UPDM/2.0

[53] 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, Heidelberg, Germany, 2008.

[54] G. Plataniotis, S. de Kinderen, and H. A. Proper. EA Anamnesis: An Approach for Decision Making Analysis in Enterprise Architecture. *International Journal of Information System Modeling and Design (IJISMD)*, 5(3):75–95, 2014.

[55] H. A. Proper. Data Schema Design as a Schema Evolution Process. *Data & Knowledge Engineering*, 22(2):159–189, 1997.

[56] H. A. Proper. Enterprise Architecture – Informed steering of enterprises in motion. In *Proceedings of the 15th International Conference (ICEIS 2013), Angers, France – Revised Selected Papers*, volume 190 of *Lecture Notes in Business Information Processing*, pages 16–34. Springer, Heidelberg, Germany, 2014.

[57] H. A. Proper, S. J. B. A. Hoppenbrouwers, and G. E. Veldhuijzen van Zanten. Communication of Enterprise Architectures. In *Enterprise Architecture at Work: Modelling, Communication and Analysis* [44], pages 67–82.

[58] H. A. Proper and Th. P. van der Weide. EVORM – A Conceptual Modelling Technique for Evolving Application Domains. *Data & Knowledge Engineering*, 12:313–359, 1994.

[59] H. A. Proper, A. A. Verrijn–Stuart, and S. J. B. A. Hoppenbrouwers. Towards Utility–based Selection of Architecture–Modelling Concepts. In S. Hartmann and M. Stumptner, editors, *Proceedings of the Second Asia–Pacific Conference on Conceptual Modelling (APCCM 2005), Newcastle, New South Wales, Australia*, volume 42 of *Conferences in Research and Practice in Information Technology Series*, pages 25–36, Sydney, New South Wales, Australia, January 2005. Australian Computer Society.

[60] H. A. Proper, R. Winter, S. Aier, and S. de Kinderen, editors. *Architectural Coordination of Enterprise Transformation*. Enterprise Engineering Series. Springer, Heidelberg, Germany, 2018.

[61] R. van Rijn, M. Driel, B. van Gils, E. Oord, and A. Santema. *Wegwijzer voor methoden bij enterprise-architectuur - 2de herziene druk*. Van Haren Publishing, 2013. In Dutch.

[62] J. Rothenberg. The Nature of Modeling. In *Artificial intelligence, simulation & modeling*, pages 75–92. John Wiley & Sons, New York, New York, United States of America, 1989.

[63] A.-W. Scheer. *ARIS – Business Process Modeling*. Springer, Heidelberg, Germany, 2000.

[64] D. Simon, K. Fischbach, and D. Schoder. Enterprise architecture management and its role in corporate strategic management. *Information Systems and e-Business Management*, 12(1):5–42, 2014.

[65] P. Sousa, R. Gabriel, G. Tadao, R. Carvalho, P. M. Sousa, and A. Sampaio. Enterprise Transformation: The Serasa Experian Case. In A. F. Harmsen, K. Grahlmann, and H. A. Proper, editors, *Proceedings of the 3rd Working Conference on Practice-driven Research on Enterprise Transformation (PRET 2010)*, volume 89 of *Lecture Notes in Business Information Processing*, pages 134–145, Delft, the Netherlands, September 2011. Springer, Heidelberg, Germany.

[66] P. Sousa, J. Lima, A. Sampaio, and C. Pereira. An Approach for Creating and Managing Enterprise Blueprints: A Case for IT Blueprints. In A. Albani, J. Barjis, and J. L. G. Dietz, editors, *Advances in Enterprise Engineering III*, volume 34 of *Lecture Notes in Business Information Processing*, pages 70–84. Springer, Heidelberg, Germany, 2009.

[67] H. Stachowiak. *Allgemeine Modelltheorie*. Springer, Heidelberg, Germany, 1973.

[68] S. L. Star and J. R. Griesemer. Institutional Ecology, 'Translations' and Boundary Objects: Amateurs and Professionals in Berkeley's Museum of Vertebrate Zoology 1907-39. *Social Studies of Science*, 19(4):387–420, 1989.

[69] M. W. A. Steen, H. W. L. ter Doest, M. M. Lankhorst, and D. H. Akehurst. Supporting Viewpoint-Oriented Enterprise Architecture. In *Proceedings of the 8th IEEE International Enterprise Distributed Object Computing Conference (EDOC 2004), Monterey, California*, pages 20–24, Los Alamitos, California, 2004. IEEE Computer Society.

[70] D. Steinberg, F. Budinsky, M. Paternostro, and E. Merks. *EMF: Eclipse Modeling Framework, 2nd Edition*. Addison-Wesley Professional, 2008.

[71] A. H. M. ter Hofstede and H. A. Proper. How to Formalize It? Formalization Principles for Information Systems Development Methods. *Information and Software Technology*, 40(10):519–540, October 1998.

[72] B. Thalheim. The Theory of Conceptual Models, the Theory of Conceptual Modelling and Foundations of Conceptual Modelling. In *Handbook of Conceptual Modeling*, pages 543–577. Springer, Heidelberg, Germany, 2011.

[73] The Open Group. *TOGAF – The Open Group Architectural Framework – Version 8.1 Enterprise Edition*. The Open Group, Van Haren Publishing, Zaltbommel, the Netherlands, 2005.

[74] The Open Group. *TOGAF Version 9.1*. Van Haren Publishing, Zaltbommel, the Netherlands, 10th edition, 2011.

[75] P. van Bommel, S. J. B. A. Hoppenbrouwers, H. A. Proper, and J. Roelofs. Concepts and Strategies for Quality of Modeling. In T. A. Halpin, J. Krogstie, and H. A. Proper, editors, *Innovations in Information Systems Modeling*, chapter 9. IGI Publishing, Hershey, Pennsylvania, 2008.

[76] J. van't Wout, M. Waage, H. Hartman, M Stahlecker, and A. Hofman. *The Integrated Architecture Framework Explained*. Springer, Heidelberg, Germany, 2010.

[77] R. Wagter, H. A. Proper, and D. Witte. On the Use of GEA at the Dutch Ministry of Social Affairs and Employment. In S. Rinderle-Ma, J. Sanz, and X.-Y. Bai, editors, *Proceedings of the 14th IEEE Conference on Commerce and Enterprise Computing (CEC2012)*, pages 115–119, Hangzhou, China, September 2012. IEEE Computer Society Press, Los Alamitos, California.

[78] R. Wagter, H. A. Proper, and D. Witte. A Theory for Enterprise Coherence Governance. In P. Saha, editor, *A Systematic Perspective to Managing Complexity with EA*. IGI Publishing, Hershey, Pennsylvania, 2013.

[79] R. Wagter, H. A. Proper, and D. Witte. Enterprise Coherence Governance in the Public Sector. In *Proceedings of the 15th IEEE Conference on Business Informatics (CBI 2013), Vienna, Austria*, pages 117–124. IEEE Computer Society Press, Los Alamitos, California, 2013.

[80] Z. Zarwin, M. Bjeković, J.-M. Favre, J.-S. Sottet, and H. A. Proper. Natural Modelling. *Journal Of Object Technology*, 13(3):1–36, July 2014.