Model bundling: Towards a value-based componential approach for language engineering

Sybren de Kinderen[3,4], Qin Ma[1,4], and Henderik A. Proper[1,2,4]

[1] Public Research Centre Henri Tudor, Luxembourg, Luxembourg
[2] Radboud University Nijmegen, Nijmegen, the Netherlands
[3] University of Luxembourg, Luxembourg
[4] EE-Team, Luxembourg, Luxembourg**
erik.proper@tudor.lu,qin.ma@tudor.lu,sybren.dekinderen@uni.lu

Abstract. This paper takes stock of an ongoing language merging experiment on value-based componential language engineering. In line with earlier work, we discuss that conceptual modelling languages can be perceived from a value perspective, and that doing so opens up the possibility of "model bundling". This means that we unbundle languages into valuable constituent parts, and subsequently mix-and-match these parts to fit with situation-specific modelling purposes.
The contribution of this particular paper is to discuss an ongoing language engineering experiment on the merge of e^3value and ArchiMate. Particularly, we show how different modelling purposes lead to different configurations (or bundles) of e^3value and ArchiMate, and how economic language design necessitates the removal of obsolete concepts. Finally, we discuss directions for further work.

Keywords: Value modelling, model bundling, language engineering, e^3value

1 Introduction

Enterprise modelling languages allow for an analysis of the as-is/to-be state of an enterprise [1, 2], ensuring acceptance of business decisions [2], analyzing cost structures [1] and workflows, and more. Examples of such languages include ArchiMate for enterprise architecture modelling, e^3value/REA for modelling a value perspective on an enterprise, and BPMN/UML activity diagrams/petri nets/Event Process Chains for process modelling.

We observe that languages are increasingly combined to address the context-specific needs of a modelling exercise, as also pointed out by [3]. This is evident from recent efforts to (1) extend the ArchiMate language with security concerns [4] and (essential) business model concerns [5], so as to address the

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

enterprise-wide impact of security aspects, respectively business model proposals.(2) combine e^3value with BPMN, strategic and IT modelling, to provide conceptual modelling support for business-IT alignment across aforementioned perspectives [6], (3) define a transformation of e^3value to ArchiMate [7, 8], to provide lightweight modelling support for analyzing the implementation of a proposed value constellation. Obviously, this list is non-exhaustive, as each enterprise modelling effort calls for context-specific language needs.

Creating a "one-size-fits-all" language is one way of addressing the plethoria of language needs. An example of such a language is ArchiMate, which recently has incorporated motivation and migration extensions, and which may be further extended to capture suggested security and business model concerns. However, when one merely keeps extending one language with new concerns, one is inevitably going to run into *economic* issues in language design [9, 10]. We posit that creating a one-size-fits-all language leads to "modelling spaghetti": cluttered models, that are difficult to design and interpret.

As a response, earlier work [11] introduced first ideas for a component-based approach towards language engineering that emphasizes an *economic* design of situation-specific modelling languages. Here, a key idea is to treat languages as "building blocks" that can be decomposed and recomposed to provide the value required for purposes of the specific modelling exercise at hand. Thus we essentially treat language composition as "model bundling" (cf. [11]): we unbundle languages into more elementary constituent parts, and subsequently rebundle these to fit situation specific modelling needs. We assume here that existing languages provide many of the ingredients required for situation-specific modelling purposes. What is mostly needed, then, is to intelligently mix-and-match existing language (parts) so that they are fit for the situation at hand.

For model bundling we reuse a method originally developed for needs-driven service bundling. This service bundling method takes the *value* provided by services as a point of departure for a match to customer needs. For example: "Skype" may be conceptualized in terms of the value "voice communication", which satisfies the need "communicate with someone abroad". As we will show in Section 3, this value based service conception can also be used for conceptualizing model (parts), whereby we want to move towards combinations of models to fit with the purposes of a modelling exercise.

As a contribution of this paper, we report on an ongoing experiment that builds on the notion of model bundling. This experiment merges two languages: e^3value and ArchiMate. Particularly we discuss our experimental aims, and our focus on language engineering as opposed to (the more generic) method engineering (the focus of [11]). Furthermore, we reflect on lessons learned so far, discussing (1) how different modelling purposes lead to different configurations of e^3value and ArchiMate, (2) how bundling may result in removal of concepts due to redundancies. For example, both ArchiMate and e^3value contain the "value" notion. Such removal is also considered important in the light of *economic* language design.

Note that this paper focuses on valuable aspects of language engineering, less so on the more technical aspects such as concerns related to model merging.

Note also that, in Situational Method Engineering, some early work exists on goal-oriented selection of method fragments [12–15]. However more research remains to be done, as also pointed out by [16, p.465] who in a recent state-of-the-art report on Situational Method Engineering argue that to move from stakeholder requirements to an optimal set of method fragments in a semi-automated way is an important research challenge. Related are also feature diagrams from software engineering (e.g. [17]), which allow for managing requirements in software product lines. However, such diagrams consider everything as a feature, do not address stakeholder motivations, and are not geared to engineering (visual) enteprise modelling languages specifically.

In Section 2, we introduce the notion of model bundling. Section 3 introduces our language merging experiment, followed in Section 4 by a discussion of lessons learned. Section 5 concludes with directions for further research.

2 Conceptual Models as collections of valuable components

To keep this paper self-contained, we now introduce the notion of a model bundle. For a more elaborate description, see [11].

Figure 1 shows an example model bundle: the ArchiMate language. Though ArchiMate is one language, it can in fact be considered as a good example of a model bundle. This is because the main added value of the language lies in relating individual perspectives on an enterprise, which are usually considered in separate languages. Arguably, bundling even is the driver for the ArchiMate language, as individual languages exist that are better tailored to modelling its individual perspectives (as in: these languages possess more expressivity, and are less ambiguous).

In line with [11], we show bundling of languages with a **value interface** which groups **value ports**. Value ports provide **value objects**, which denote elementary model outcomes (in line with [18]). **Consequences** result from both value objects and value interfaces. They denote *how* bundling, respectively a value object, are valuable. For example: from Figure 1 we observe that ArchiMate as a bundle is valuable because it provides "Link business process/IT perspectives".

Furthermore, value objects are produced by **model fragments**, which can consist of more elementary model fragments. At this point it is important to remark that in e^3value such decomposition would be impossible, as in fact the mirror concept of a "value activity" does not allow for this. However, for model bundling we deliberately opted for the option of nesting method fragments as it allows us to represent *nesting* of model fragments without further redundancy. For example, by capitalizing on the nesting of models we can use the model fragment "ArchiMate 1.0 Modelling" as input for a bundle together with e^3value,

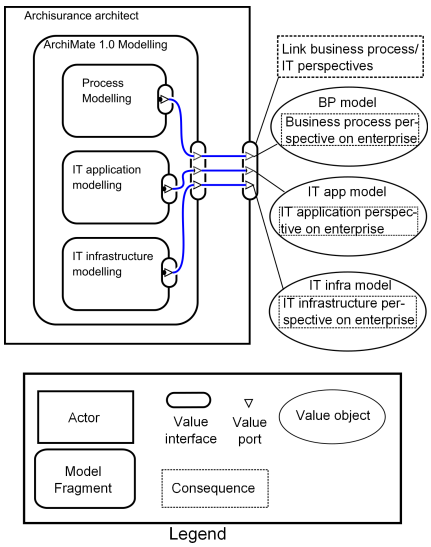

Fig. 1. ArchiMate as a model bundle

by treating "ArchiMate 1.0 Modelling" as a black box (i.e.: abstracting from its constituent parts).

3 The ArchiValue experiment

3.1 Merging e^3*value* into ArchiMate

As input for our component-based language engineering approach we conduct an experiment to merge two languages: e^3*value* and ArchiMate. The aim of this experiment is twofold: (1) to see how a componential language engineering approach could *technically* work. Here an example concern is how to actually merge two languages. The hierarchical approach towards language engineering detailed in [10] is used as a first candidate approach for this. (2) to see how a componential language engineering approach could work from a *value* point of view.

Obviously, the focus of the current paper is on the value point of view. However, it is worthwhile to tackle these two perspectives in tandem as the technical perspective may inform pragmatic (value) language engineering concerns and vice versa. For one, in terms of required effort, tackling language merging by means of the ArchiMate hierarchical approach is "expensive" compared to more traditional language transformation approaches.

We perform the experiment with the well-documented ArchiSurance case study, a fictitious yet realistic case of an insurance company that considers to sell its insurances via an intermediary as opposed to directly to the customer.

ArchiSurance is the principal case study used by the Open Group [5] to illustrate the ArchiMate language.

3.2 ArchiValue: a first version

Until now, we created one version of the merged ArchiValue language. The rationale behind this particular ArchiValue language is twofold. On the one hand, ArchiValue allows for a lightweight analysis of the business processes required for a value constellation. On the other hand, ArchiValue allows one to express the economic rationales for an operational Enterprise Architecture model.

Figure 2 shows an example ArchiValue model for the Archisurance intermediary case. Here, we can observe how the merged language allows for a link of value to business process concerns: via a business service. A business service is originally an ArchiMate concept that allows for an externalization of functionality and as such, it allows for showing the required functionality for implementing a value constellation. Subsequently, Archivalue can show how a business process can realize the functionality implied by a business service. For example, in Figure 2 we see that "Customer insurance management service" is required to support the interaction with the customer, and that "Customer insurance management service" in turn is supported by the business process "Provide insurance".

Observe from Figure 2 as well that there are further technical integration issues at play, such as the centrality of the role concept (inherent to ArchiMate) as opposed to the centrality of a physical actor (inherent to e^3value). However, as this paper concentrates on the value aspects we do not further discuss these here.

4 Lessons learned, so far

Lesson 1: Different configurations for different EA + value modelling needs

So far, we have created one version of ArchiValue, to allow for a lightweight qualitative analysis of linking value and business process concerns. However to allow for a quantitative assessments we would require a more elaborate version of ArchiValue that incorporates both a quantitative appreciation of resources in ArchiMate, and possible links to model fragments from e^3value that enable profitability analysis, such as scenario paths.

Figure 3 provides a goal model reflecting these different motivations, leading to different language needs. Here we see that to achieve goal (1) "see added value of intermediary for ArchiSurance", we can simply rely on the current ArchiMate language, which already has a "value" concept. However, if we want to achieve goal (2) "Analyze ,qualitatively, costs and benefits of ArchiSurance" we require

[5] The Open Group is the standardization body behind the ArchiMate language specification.

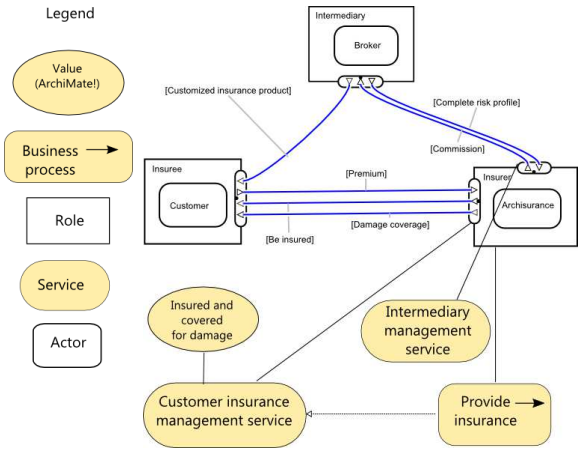

Fig. 2. An ArchiValue model for the ArchIsurance intermediary sales model

the current version of the ArchiValue language, as - through its "bundled" perspectives - ArchiValue allows for an appreciation of both costs and benefits, and for an appreciation of actor dependencies. The ArchiValue bundle that we would then use is depicted in Figure 4. Finally, if we want to achieve goal (4) "Profitability of intermediary business model" we need incorporate further language elements, as mentioned.

Lesson 2: Allow for removal of overlapping concepts. In our experiment we find that, by introducing e^3value concepts into ArchiMate, we essentially make the current "value" concept from ArchiMate obsolete. This is because currently "value" is a catch all concept in ArchiMate for all matters to do with value. As the ArchiMate "value" concept can be readily specified by e^3value notions such as "value object", "value interface", and "value transfer", it can be removed from the new ArchiValue language. This is especially important in the light of our intention to provide for *economic* language design.

5 Concluding outlook

Building on earlier model bundling work, this paper reports on first results from an experiment that merges two languages: e^3value and ArchiMate. Key lessons learned so far are: (1) that different modelling purposes require different bundles of ArchiMate and e^3value and (2) that, for economic language design, we should strongly consider removal of redundant concepts as well.

We are currently continuing our model merging experiment, addressing amongst others the following questions:

- *What is an "indivisible" model fragment?* Currently, we consider a part of a language as divisable as long as it provides value in its own right. However, this raises questions such as: if I find a "value interface" useful for my language, do I consider it as a meaningful model fragment? And if so, what happens to

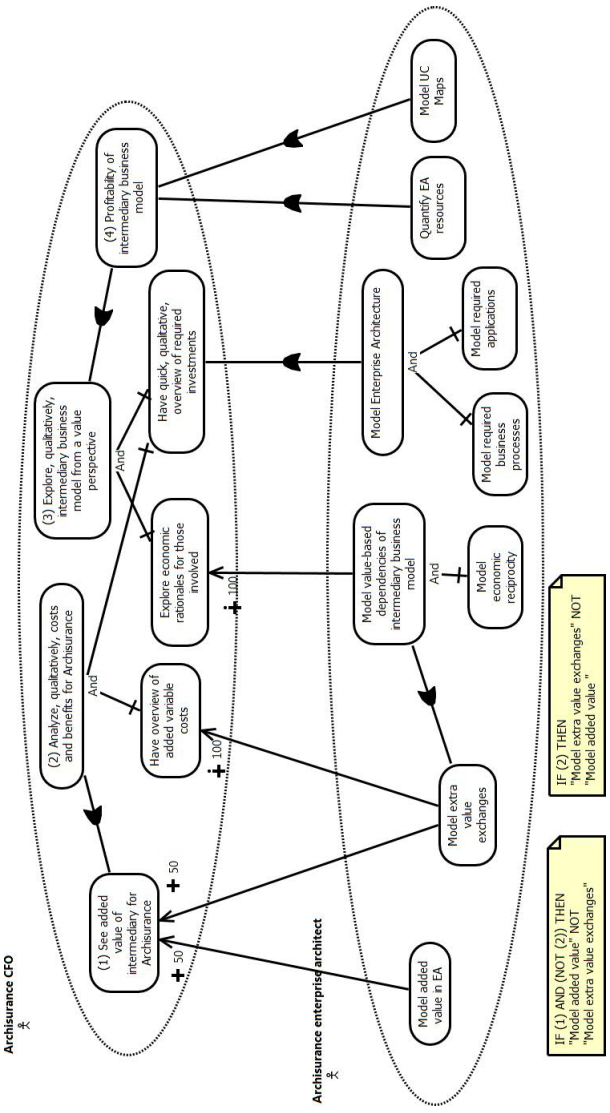

Fig. 3. A first goal model for different language needs

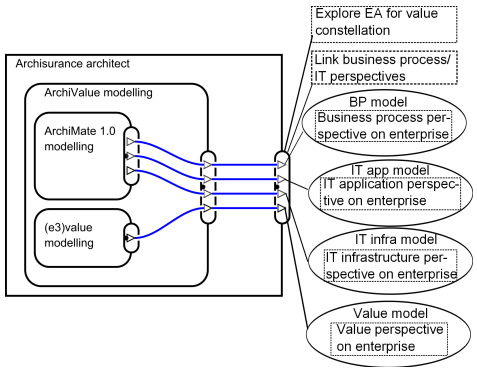

Fig. 4. The ArchiValue model bundle

"technical" dependencies, such as the value ports grouped by an interface? Also, if a value interface is cut out of an original language and placed in a new one, does this not give rise to a possible different interpretation, as opposed to a "value interface" as intended by e^3value?

-*To what extent do we deal with customization, next to bundling?* Currently our approach is purely componential, yet one can imagine that a layer of customization is required on top of the merged models. Such customization can for example entail the introduction of new concepts, in addition to the bundle. An interesting example of such customization, for e^3value, can be found in [19]. Here, the notion of shared value activities is introduced so as to express that joint perfomance of an activity in a partnership.

References

1. Lankhorst, M., et al.: Enterprise Architecture at Work: Modelling, Communication and Analysis. 3rd edn. Springer Publishing Company, Incorporated (2012)
2. Stirna, J., Persson, A.: Purpose driven competency planning for enterprise modeling projects. In Ralyté, J., J., Franch, X., Brinkkemper, S., Wrycza, S., eds.: Advanced Information Systems Engineering. Volume 7328 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2012) 662–677
3. Bjekovic, M., Proper, H.A., Sottet, J.S.: Towards a coherent enterprise modelling landscape. In: Fifth conference on the Practice of Enterprise Modelling (PoEM 2012), ceur-ws (2012)
4. Feltus, C., Dubois, E., Proper, E., Band, I., Petit, M.: Enhancing the archimate standard with a responsibility modeling language for access rights management. In: Proceedings of the Fifth International Conference on Security of Information and Networks. SIN '12, New York, NY, USA, ACM (2012) 12–19
5. Meertens, L.O., Iacob, M.E., Nieuwenhuis, L.J.M., van Sinderen, M.J., Jonkers, H., Quartel, D.: Mapping the business model canvas to archimate. In: Proceedings of the 27th Annual ACM Symposium on Applied Computing. SAC '12, New York, NY, USA, ACM (2012) 1694–1701

6. Pijpers, V., de Leenheer, P., Gordijn, J., Akkermans, H.: Using conceptual models to explore business-ict alignment in networked value constellations. Requirements Engineering **17**(3) (2012) 203–226
7. van Buuren, R., Gordijn, J., Janssen, W.: Business case modelling for e-services. In: 18 th Bled eConference eIntegration in Action, AIS (2005)
8. de Kinderen, S., Gaaloul, K., Proper, H.A.: Bridging value modelling to archimate via transaction modelling. Software & Systems Modeling (2012) 1–15
9. Proper, H.A., Verrijn-Stuart, A.A., Hoppenbrouwers, S.J.B.A.: On utility-based selection of architecture-modelling concepts. In: Proceedings of the 2Nd Asia-Pacific Conference on Conceptual Modelling - Volume 43. APCCM '05, Darlinghurst, Australia, Australia, Australian Computer Society, Inc. (2005) 25–34
10. Lankhorst, M., Proper, H., Jonkers, H.: The anatomy of the archimate language. Int. J. Inf. Syst. Model. Des. **1**(1) (January 2010) 1–32
11. de Kinderen, S., Proper, H.A.: E3rome: A value-based approach for method bundling. In: Proceedings of the 28th Annual ACM Symposium on Applied Computing. SAC '13, New York, NY, USA, ACM (2013) 1469–1471
12. Chiniforooshan Esfahani, H., Yu, E., Cabot, J.: Situational evaluation of method fragments: an evidence-based goal-oriented approach. In: Advanced Information Systems Engineering, Springer (2010) 424–438
13. Chiniforooshan Esfahani, H., Yu, E.: A repository of agile method fragments. New Modeling Concepts for Todays Software Processes (2010) 163–174
14. Ågerfalk, P., Fitzgerald, B.: Exploring the concept of method rationale: A conceptual tool. Advanced topics in database research **5** (2006) 63
15. Rossi, M., Ramesh, B., Lyytinen, K., Tolvanen, J.: Managing evolutionary method engineering by method rationale. Journal of the Association for Information Systems **5**(9) (2004) 356–391
16. Henderson-Sellers, B. and Ralyté, J.: Situational method engineering: state-of-the-art review. Journal of Universal Computer Science **16**(3) (2010) 424–478
17. Schobbens, P., Heymans, P., Trigaux, J.C.: Feature diagrams: A survey and a formal semantics. In: Requirements Engineering, 14th IEEE International Conference. (2006) 139–148
18. Razo-Zapata, I.S., de Leenheer, P., Gordijn, J., Akkermans, H.: Fuzzy verification of service value networks. In Ralyte, J., Franch, X., eds.: 24th International Conference on Advanced Information Systems Engineering (CAiSE'12), Springer (2012)
19. Kort, C., Gordijn, J.: Modeling strategic partnerships using the e3value ontology - a field study in the banking industry. In Rittgen, P., ed.: Handbook of Ontologies for Business Interaction. IGI Global, Hershey, PA (2007)