Application of a Domain-Specific Language to Support the User-Oriented Definition of Visualizations in the Context of Collaborative Product Development

Thomas Reschenhofer — Ivan Monahov — Florian Matthes

Chair for Informatics 19 (sebis)
Technische Universität München
Boltzmannstr. 3, 85748 Garching bei München, Germany

reschenh@in.tum.de
ivan.monahov@in.tum.de
matthes@in.tum.de

ABSTRACT: *In the domain of Enterprise Architecture Management (EAM), multiple stakeholders with different responsibilities and backgrounds have to collaborate to achieve different predefined enterprise-related goals. To enable the definition of stakeholder-specific views onto the overall Enterprise Architecture (EA), we developed and prototypically implemented a domain-specific language (DSL) for defining model-based metrics and visualizations as management decision support in a commercial EAM tool. Since the domain of Collaborative Product Development (CPD) seemed to be similar to EAM with respect to the involvement of a diversity of stakeholders and the need for collaboration, we applied our prototype in the CPD domain by implementing an automated project status analysis and visualization called SmartNet Navigator as part of the EU research project SmartNets. However, since we faced several limitations and shortcomings (e.g., missing type checking and insufficient DSL syntax) during the evaluation of our prototype in the field of EAM, we redesigned the DSL and improved the prototype to solve the problems. This paper firstly provides an overview of the problems and improvements of our solution. Afterwards we present the application of our improved prototype in the domain of CPD by reimplementing the SmartNet Navigator and compare it to its initial prototype. Finally, we discuss possible benefits by the application of the improved DSL in the field of CPD and outline the need of further evaluation with experts from the SmartNet project according to the design science research method.*

KEY WORDS: *Domain-Specific Language, Enterprise Architecture Management, Collaborative Product Development, Metric, End-User Development*

1. Introduction

To cope with the continuous evolution of an enterprise's environment, it has to align its IT perpetually to the ever-changing business requirements to ensure effective and efficient IT support for the business. In this context, the Enterprise Architecture (EA) is defined as "the fundamental and holistic organization of an enterprise embodied in its components, relations and its environment" (ISO/IEC 42010:2007). Hence, to facilitate this alignment, enterprise architects have to manage the flexibility, efficiency and transparency of the EA. The corresponding management discipline is named Enterprise Architecture Management (EAM) and includes the development, implementation and control of the EA's evolution, i.e., the evolution of its components, attributes, and relations (Ahlemann et al. 2012).

However, due to the increasing complexity and the dynamics of today's enterprise architectures, qualitative models (e.g., visualizations) are not sufficient for efficient decision support (Kaisler et al. 2005). Therefore, quantitative models (e.g., metrics and performance indicators) are used to provide a meaningful and reliable assessment of an EA. Furthermore, the increasing size and complexity of EAs also implies an increasing need for adequate tool-support (Handler, Wilson 2012; Buckl et al. 2008), which also includes support for quantitative models by the EAM tool. Moreover, to cope with the need for a flexible EA model due to the changing environment and the need for collaboration support due to a plethora of involved stakeholders (Lucke et al. 2010), the *Wiki4EAM* approach (Matthes, Neubert 2011) makes use of the model-based and collaborative EAM tool *Tricia*. In order to enable the definition of metrics and thus to equip this EAM tool with quantitative modelling capabilities, we developed a domain-specific language (DSL) named model-based expression language 1.0 (MxL 1.0) for the user-oriented definition of metrics at runtime and integrated it in Tricia (Monahov et al. 2013).

	I Creation of ideas	II Concept development	III Prototyping	IV Sampling	V Production and Marketing
Planning	Innovation culture Innovation strategy and objectives Identification of opportunities	Framework def. (Concept) IPR protection planning Project planning (Concept)	Framework def. (Prototype) Project planning (Prototype)	Framework def. (Sample) Planning of sourcing Project planning for sample development	Planning of market introduction and marketing Planning of production and life-cycle handling
Execution	Idea generation Idea formulation	Concept elaboration Functional description Tech. feasibility Market study Business plan Marketing plan Protection of IPR	Prototype elaboration Prototype test (α-test)	Sourcing for sampling Process implementation Production of samples Sample test (β-test)	Market introduction Continuous marketing Continuous production and life-cycle handling
Control	Screening and first evaluation Evaluation of IPR situation Recommendation of project	Assessment of concept Evaluation of studies Financial assessment (Concept) Launch prototyping	Technical evaluation Market-oriented evaluation Financial assessment (Prototype) Launch for sampling	Evaluation of test results Evaluation of process reliability Financial assessment (Sample) Launch for production	Evaluation of market response Financial success control

Figure 1. *A sketch of the SmartNet Navigator.*

In addition to the DSL's application in EAM we also applied MxL 1.0 in the domain of Collaborative Product Development (CPD), since most of the general conditions of EAM (e.g., involvement of a plethora of stakeholders and collaborative tasks) also hold for CPD (Hauder et al. 2013). In this context, MxL 1.0 was employed in the EU project *SmartNets* by implementing the so-called *SmartNet Navigator* – a visualization of the aggregated status of a certain development project (Matheis 2013). The SmartNet Navigator aggregates the status of individual tasks and meetings to the status of corresponding activity types, development phases, and in the end to an overall project status and visualizes it by a certain color-coding.

Figure 1 shows a sketch of the SmartNet Navigator for an exemplary project. The Navigator's columns represent development phases, the rows represent management activity types, and the cells consist of multiple activity types, which in turn are related to the project's tasks and meetings. In the concrete example in Figure 1, the first development phase of the project is finalized (green), the second phase is in progress (orange), and the remaining phases are still open (grey).

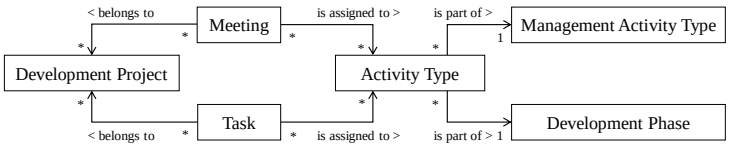

Figure 2. *The SmartNet Navigator's underlying meta-model*

We implemented the SmartNet Navigator in MxL 1.0 based on the meta-model shown in Figure 2. Hence, users are able to adjust the SmartNet Navigator's implementation at runtime, which makes it possible for the users to respond immediately to certain changes of the environment, e.g., a change of the rule for aggregating the status of multiple tasks. Figure 3 shows an excerpt of the SmartNet Navigator's implementation.

However, the implementation of the SmartNet Navigator in MxL 1.0 suffers from several drawbacks, which we describe in the next Section.

Custom MxL Function Development Phase::smartnetNavigatorHeaderCell

Parameters dp

Method Stub

```
"<td class='".concat(this.smartnetStatusOfProcessPhase(dp)).concat("'>")

    /* Display the phase's order as a roman number, followed by
       the phase's name */
    .concat(this["Order"].first().roman()).concat("<br/>").concat(this["Name"].first())

.concat("</td>")
```

Figure 3. *An MxL 1.0 function (Monahov et al. 2013) for the generation of HTML mark-up as part of the SmartNet Navigator's implementation. This function generates an HTML cell whose style is determined by the status of the corresponding process phase. The content of the cell is defined as the number of the process phase (in roman number format) followed by a line break and the name of the process phase.*

2. Redesign of MxL and reimplementation of the SmartNet Navigator

While we have done the implementation of the SmartNet Navigator with MxL 1.0 (Hauder et al. 2013), the evaluation of MxL 1.0 in the domain of EAM revealed some weaknesses of the DSL and its implementation in Tricia. The most relevant to our understanding are:

W1 One of the goals of MxL 1.0 was to keep it minimal regarding its expressiveness and syntax. Consequently, we waived common language constructs (e.g., infix-notation for algebraic operators) and implemented them by function calls. However, this purely functional approach yields to incomprehensible expressions.

W2 Although the syntactic correctness of MxL 1.0 expressions (e.g., bracket matching) is checked at compile-time, the validation of an MxL 1.0 expression's static semantics (Voelter et al. 2013) is not performed at compile-time, but at runtime.

W3 Due to the lack of validation of the static semantics of an MxL 1.0 expression, these expressions are not analyzable at compile-time. Hence, the dependencies to MxL functions, attributes, types, etc. are not automatically observable.

W4 Changes of the underlying meta-model (e.g., renaming of attributes) affect the semantic consistency of all MxL 1.0 expressions referring to the changed elements.

Due to these shortcomings identified in the EAM domain, we redesigned the DSL and developed an improved version called MxL 2.0 (Reschenhofer 2013). To assess the added value of MxL 2.0 in the field of CPD, we reimplemented the SmartNet Navigator with MxL 2.0 to compare it to the initial prototype. Figure 4 shows an excerpt of the implementation of the SmartNet Navigator in MxL 2.0, while in the next Section we outline the benefits of the new prototype of the SmartNet Navigator.

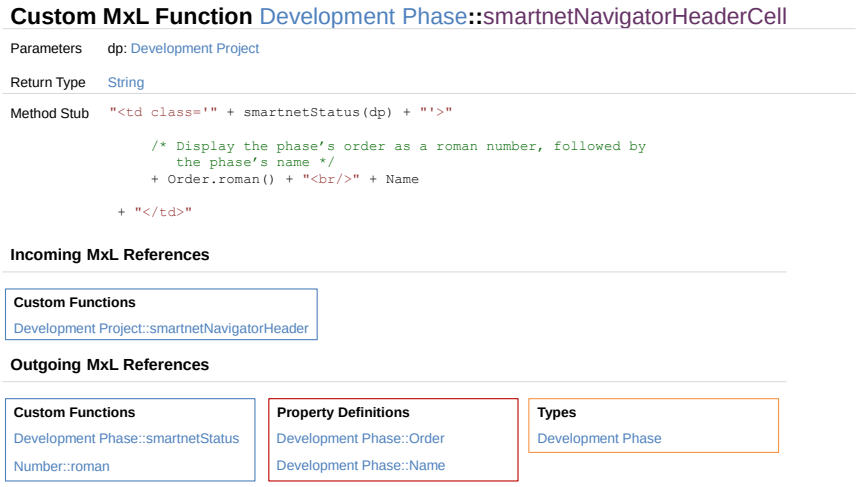

Figure 4. *Reimplementation of the SmartNet Navigator in MxL 2.0 (compare to Figure 3).*

3. Improvements and evaluation of the reimplemented SmartNet Navigator

In this paper, we elucidate the improvements by comparing an excerpt of the SmartNet Navigator's initial prototype (c.f. Figure 3) with a corresponding excerpt of the reimplemented prototype using MxL 2.0 (c.f. Figure 4). The main features of MxL 2.0 are an improved syntax as well as a type checker component as part of the MxL compiler (Reschenhofer 2013), leading to the following improvements, whereas each improvement Ix addresses the corresponding weakness Wx from Section 2:

I1 In MxL 2.0, we introduced infix-operators (e.g., plus operator for the string concatenation) as well as semantic enhancements (e.g., implicit *this*), so that the implementation of the improved prototype is more readable.

I2 The MxL 2.0 type checker validates the static semantics (Voelter et al. 2013) of MxL 2.0 expressions and therefore ensures semantic consistency at compile-time. For example, based on the expression in Figure 4 this means that the type checker ensures the existence of the attributes *Order* and *Name*. Moreover, the type checker determines the return type of the expression (e.g., *String*)

I3 The type checker enables the analysis of expressions, which means the observation of an MxL 2.0 expression's dependencies to MxL functions, attributes, types, etc. We use this expression analysis for generating and maintaining a computation graph. The nodes of the computation graph are MxL 2.0 expressions as well as objects these expressions refer to (e.g., attributes and types). Its edges represent the dependencies between them. For example, Figure 4 shows the *Incoming MxL References* (Expressions, which are referring to the current one) as well as the *Outgoing MxL References* (Objects, the current expression refers to).

I4 By using the computation graph, a tool implementing MxL 2.0 (e.g., Tricia) is able to propagate changes to those expressions, which are depending on the changing object. For example, if a user renames the attribute *Name* to *Title*, the reference in the expression of Figure 4 will be updated accordingly and therefore keeps consistency regarding its static semantics.

4. Summary and conclusion

Since MxL 2.0 leads to significant improvements in the domain of EAM, we expect also benefits for the field of CPD, because most of the general conditions of EAM also hold for the domain of CPD. In this paper, we outlined these possible benefits by comparing the MxL 1.0 implementation of the SmartNet Navigator with a corresponding MxL 2.0 implementation.

However, to support the claim of achieving significant improvements in CPD, we still have to conduct further evaluation with experts of the field according to the design science research method (Hevner et al. 2004) in our future research activities.

5. References

Ahlemann, Frederik; Stettiner, Eric; Messerschmidt, Marcus; Legner, Christine (2012): *Strategic Enterprise Architecture Management*: Springer-Verlag.

Buckl, Sabine; Ernst, Alexander M.; Lankes, Josef; Matthes, Florian (2008): *Enterprise Architecture Management Pattern Catalog* (Version 1.0, February 2008). Chair for Informatics 19 (sebis), Technische Universität München. Munich, Germany. Available online at http://eampc-wiki.systemcartography.info/.

Handler, Robert A.; Wilson, Chris (2012): *Magic Quadrant for Enterprise Architecture Tools*.

Hauder, Matheus; Roth, Sascha; Matthes, Florian; Lau, Armin; Matheis, Heiko (2013): Supporting collaborative product development through automated interpretation of artifacts. In *3rd International Symposium on Business Modeling and Software Design*.

Hevner, Alan R.; March, Salvatore T.; Park, Jinsoo; Ram, Sudha (2004): Design science in information systems research. In *MIS quarterly* 28 (1), pp. 75–105.

ISO/IEC 42010:2007 Systems and software engineering - Recommended practice for architectural description of software-intensive systems.

Kaisler, Stephen H.; Armour, Frank; Valivullah, Michael (2005): Enterprise Architecting: Critical Problems. In *Proceedings of the 38th Annual Hawaii International Conference on System Sciences*.

Lucke, C.; Krell, S.; Lechner, U. (2010): Critical Issues in Enterprise Architecting - A Literature Review. In *Proceedings of the Sixteenth Americas Conference on Information Systems*.

Matheis, Heiko (2013): SmartNet Navigator and application guidelines. In *Sehenth Framework Programme*.

Matthes, Florian; Neubert, Christian (2011): Wiki4EAM - Using Hybrid Wikis for Enterprise Architecture Management. In *7th International Symposium on Wikis and Open Collaboration (WikiSym)*.

Monahov, Ivan; Reschenhofer, Thomas; Matthes, Florian (2013): Design and prototypical implementation of a language empowering business users to define Key Performance Indicators for Enterprise Architecture Management. In *Trends in Enterprise Architecture Research Workshop*.

Reschenhofer, Thomas (2013): Design and prototypical implementation of a model-based structure for the definition and calculation of Enterprise Architecture Key Performance Indicators. Master's Thesis. Technische Universität München.

Voelter, Markus; Benz, Sebastian; Dietrich, Christian; Engelmann, Birgit; Helander, Mats; Kats, Lennart C. L. et al. (2013): DSL Engineering-Designing, Implementing and Using Domain-Specific Languages: dslbook. org.