A Toolbox Supporting Agile Modelling Method Engineering: ADOxx.org Modelling Method Conceptualization Environment

Nesat Efendioglu(✉), Robert Woitsch, and Wilfrid Utz

BOC Asset Management GmbH, Operngasse 20B, 1040 Vienna, Austria
{nesat.efendioglu, robert.woitsch,
wilfrid.utz}@boc-eu.com

Abstract. The importance of Modelling Method Engineering is equally rising with the importance of domain specific modelling methods and individual modelling approaches. In order to capture the most relevant semantic primitives that address domain specifics needs, it is necessary to involve both, method engineers as well as domain experts. Due to complexity of conceptualization of a modelling method and development of regarding modelling tool, there is a need of a guideline and corresponding tools supporting actors with different background along this complex process. Based on practical experience in business, more than twenty EU projects and other research initiatives, this paper introduces a toolbox to support the conceptualization of a modelling method. The realized toolbox is introduced and evaluated by two EU-funded research projects in the domain of e-learning and cloud computing as well as additionally by an in-house development project in the area of decision modelling extensions. The paper discusses the evaluation results and derived outlooks.

Keywords: Meta-modelling · Modelling method design · Agile modelling method engineering · Conceptualization

1 Introduction

The importance of Modelling Method Engineering is equally rising with the importance of Domain Specific Conceptual Modelling Methods and individual modelling approaches. In addition to existing standards (e.g. BPMN, DMN, CMMN), a growing number of groups around the world design their individual modelling-methods (in accordance with the definition of such a method by [1, 2]) for a variety of application domains. The engineering of such applicable modelling tools as a result of the conceptualization process of modelling methods, is complex, especially when considering the mapping of the entire spectrum from language artefacts and corresponding functionality to concrete implementable and deployable modelling tool capabilities. Besides that, there are branching knowledge domains into more refined and specialized subdomains, where each domain needs to be characterized by its own abstraction and refinement of concepts from reality. Hence, in order to capture the most relevant semantic primitives that address domain specific needs, it is necessary to involve both the method engineers as well as domain experts. Today, there are different approaches,

J. Horkoff et al. (Eds.): PoEM 2016, LNBIP 267, pp. 317–325, 2016.
DOI: 10.1007/978-3-319-48393-1_23

guidelines and practices for the development of modelling tools available that do not consider the full spectrum of the design and collaborative development of a modelling method, which unavoidably leads to limitations in the conceptualization of it [3]. There is a need of a guideline and corresponding tools supporting method engineers along the complex conceptualization process taking all phases into consideration and ensuring collaboration among stakeholders involved in the process. Karagiannis proposes in [2] the Agile Modelling Method Engineering (AMME) framework. Authors of [4] propose the Modelling Method Conceptualization Process that based on AMME, guides the method engineers during conceptualization. The work at hand proposes a toolbox that supports this process, evaluates it in two European Research projects, and one additional in the context of an in-house research project, and discusses evaluation results.

The remainder of the paper is structured as follows: Sect. 2 briefly revisits AMME, the Modelling Method Conceptualization Process and outlines each tool in the toolbox. Section 3 presents evaluation cases and discusses the evaluation results, while Sect. 4 concludes the paper and gives an outlook on future work.

2 Modelling Method Conceptualization Environment

AMME is proposed in [2] to support modelling method engineering during propagation and evolution of modelling requirements. The OMiLab Lifecycle [5] instantiates AMME and defines the internal cycle of a modelling method conceptualization with five phases; (1) "Create" as a mix of goal definition, knowledge acquisition and requirements elicitation activities that capture and represent the modelling requirements; (2) "Design" specifies the meta-model, language grammar, notation and functionality as model processing mechanisms and algorithms; (3) "Formalize" aims to describe the outcome of the previous phase in non-ambiguous, formal representations with the purpose of sharing results within a scientific community; (4) "Develop" produces concrete modelling prototypes and finally (5) "Deploy/Validate" involves the stakeholders in hands-on experience and the evaluation process as input for upcoming iterations.

Due to the involvement of several stakeholders with varying knowledge backgrounds, perspectives and different objectives, in the conceptualization of a modelling method, the authors of [4] propose so-called Modelling Method Conceptualization Process (as depicted in Fig. 1) by adding additional feedback channels into the

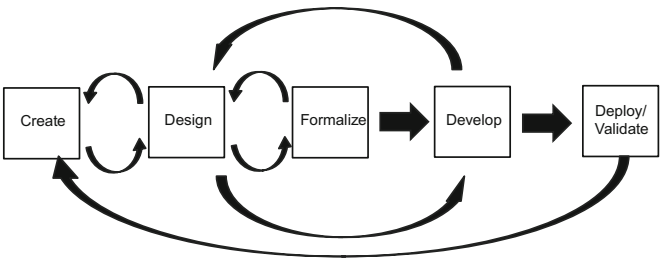

Fig. 1. Modelling method conceptualization process

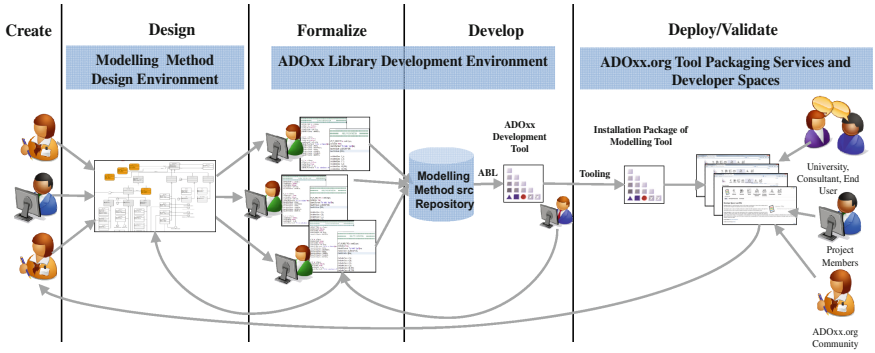

Fig. 2. The toolbox: modelling method conceptualization environment

OMiLab Lifecycle between: (1) Create and Design, to prove, if the designed modelling language covers the identified application scenarios and considers the identified requirements; (2) Design and Formalize to ensure formal approval of modelling language, as well as (3) Design and Develop - to improve modelling language in earlier stages before it is released and deployed.

The work at hand introduces a toolbox called "Modelling Method Conceptualization Environment" (as depicted in Fig. 2) that instantiates the above process and supports method engineers during each phases. The only exception is that of the "Create" phase, as this part is regarded as the most creative phase and standard tools and methods (also in some cases pen and paper can be the most appropriate tools) to can be freely selected. Modelling Method Conceptualization Environment proposes a combination of tools, such as the Modelling Method Design Environment (MMDE, available at [6]) for the Design, the ADOxx Library Development Environment (ALDE) and ADOxx, for Formalize and Develop, ADOxx.org Tool Packing Services and Developer Spaces for Deploy/Validate Phases.

It is worth to mention that one of the objectives is to provide loosely coupled tools, so method engineers have the flexibility to decide to use one, a combination of tools from the toolbox or even use other appropriate tools of their choice, (e.g. method engineer uses MMDE during the Design Phase, but formalize the modelling method design with mathematical models or use another development tool during the Develop Phase and deploys them at the Developer Spaces and enable validation).

In the following sub-sections current abilities of the tools from the environment are shortly presented.

2.1 Modelling Method Design Environment

The Modelling Method Design Environment (MMDE) is itself a modelling tool to design other modelling methods. MMDE has been implemented based on lessons learned and the experience of the authors, who have been involved in modelling method/tool development activities in more than 20 EU research projects for varying domains. Based on these lessons learned, UML [7] has been identified as a fitting

starting point. Hence, the MMDE takes a subset of UML and extends it with required concepts and functionalities in order to overcome the following challenges **(Ch)**, which are introduced by [4] after a state of the art analysis about specification of conceptual modelling methods: **Ch1-**Definition of functional and non-functional requirements and their relation between the concepts in modelling methods; **Ch2-**Fragmenting the whole meta-model into individual meta-models composing concepts for different sub-domains and still be able to define links between concepts in different individual meta-models; **Ch3-**Having another abstraction layer to represent modules and layers of modelling language as well as relation among them without representing the complexity of abstract and concrete syntax; **Ch4-**Assigning different concrete syntax to the concepts in modelling language; **Ch5-**Possibility to design modelling procedure and mechanisms & algorithms of a domain specific modelling language.

To overcome **Ch1,** "Requirements" model type is implemented that allows the elicitation of requirements, specifying their status as well as dependencies among them. The described requirements in this model type can be referenced to (a) all the modelling classes modelled in the related model type "Meta-Model" classes, (b) graphical notation (concrete syntax) definitions modelled in the "Graphical Notation" model type, (c) the "Modelling Stack" definition and (d) to the functionalities modelled in "Mechanisms & Algorithms" models. For **Ch2** and **Ch3,** we extend the class diagram from UML with concepts, so method engineers can differentiate between class and relation class as well as relate different meta-models (-fragments) with each other using "Weaving" techniques as they are introduced in [8, 9]. The modularization and layering of modelling language is essential to avoid complexities during the design of domain specific modelling methods [10, 11]. Hence, we propose representation of the modelling stack as the "Meta-models Stack model type allowing method engineers to differentiate meta-models in sense of different model types that target different fragments of the system. In order to target **Ch4** and specify a proper graphical representation (concrete syntax) of each concept in a meta-model, we introduce another model type called "Graphical Notation" model type (allows definition of concrete syntax of model types with specifying graphical representations for each constructs in meta-models. This model type allows the description of graphical representations either assignment of vector graphics code written in GraphRep Language [31] or with the assignment of concrete images in PNG, JPG or Bitmap format including a description of the functionalities in the notation (e.g. attribute-value dependent visualization, context related views) In order to target **Ch5** to define the applicable modelling technique as steps and corresponding results we propose a model type called "Modelling Procedure" model type". The Modelling Procedure Model Type allows method engineers to define the steps with their required inputs and produced outputs, as well as the sequence of steps based on input – output relationships, in order to introduce case specific proper usage of their modelling method. Based on this procedural view, concrete Mechanisms and Algorithms can be derived and depicted as Sequence and Component Diagrams from UML (therefore these diagram types has been implemented as model types in MMDE). Further details about MMDE can be found in [4].

2.2 ADOxx Library Development Environment

The ADOxx Library Development Environment (ALDE) aims to enable formalization and parallel development of modelling tools libraries based on the designs deriving from Design Phase, merging different libraries and ensuring maintainability. As an experimental prototype ALDE uses the Resource Description Framework (RDF) as a format for data interchange [12].

ALDE is a development environment based on the Eclipse IDE [13] and includes a meta^{2}model defined in RDFS, the ALDE vocabulary. Having the vocabulary and utilizing Apache Ant® as a build mechanism [14], the environment enables the definition of the transformation processes from ADOxx Library Languages to RDF and vice versa. Moreover ALDE serializes libraries in an arbitrary RDF format; for the prototypical realization RDF Turtle [15] has been used and includes the RDF XTurtle Editor developed by [16]. Having libraries in RDF Turtle format and a RDF Turtle Editor available, method engineers can adapt declaratively and script libraries collaboratively using standard functionalities of source-code management systems. Merging several libraries or integration of parts of libraries in each other becomes possible.

2.3 ADOxx.org Tool Packaging Services and Developer Spaces

The ADOxx.org Tool Packaging Service [17] is a web-based service that allows method engineers of the ADOxx community to build verified, professional and installable distribution packages that can be distribute to interested stakeholders. The service combines and validates all available inputs, integrates all elements, compiles the necessary artifacts and signs the outcomes and creates the actual installer as a download archive.

As a collaboration space for the development and deployment phases, the concept of "Developer Spaces" has been introduced in ADOxx.org [18]. These spaces enable sharing of intermediate/release results, discussing development resources from all pre/past phases in the form of source code, snippet, examples and distribution packages with the community.

3 Evaluation

The toolbox introduced above has been applied in three different cases for evaluation: two EU-funded research projects in the domain of eLearning and cloud computing and additionally in an in-house development project, in the area of decision modelling extensions. These cases have been selected since the involved partners have varying profiles and expertise in development and in modelling tool engineering. In the following subsection we introduce the cases and their requirements in method engineering manner. Afterwards the evaluation results are discussed.

Case 1: Conceptualization of a Modelling Method for E-Learning: The FP7 project Learn PAd [19] proposes a process-driven-knowledge management approach based on conceptual and semantic models for transformation of public administration

organizations into learning organizations. Learn PAd proposes a model-driven collaborative learning environment. In this case, 4 domain experts and method engineers have been involved. In addition, two developer teams, each consisting of 4 developers worked on the implementation of the tool. The results of the conceptualization process of this modelling method can be found at Learn PAd Developer Space [20], as well as the developed prototypes [21] can be downloaded and feedback can be given.

Case 2: Conceptualization of Modelling Method for Cloud Computing: The H2020 project CloudSocket [22] introduces the idea of Business Processes as a Service (BPaaS), where conceptual models and semantics are applied to align business processes with Cloud-deployed workflows [23]. In this case, 6 domain experts and method engineers have been involved, as well as two developer teams, one with 5 developers, the other one with 2 members The results of the conceptualization process of this modelling method can be found at CloudSocket Developer Space [24], as well as developed prototypes [25] can be downloaded and feedback can be given.

Case 3: Integration of Exiting BPMN and DMN Modelling Methods: The in-house project requires integration of an already implemented DMN [26] Modelling Method into existing BPMN 2.0 [27] realization as part of the a commercial product. In this case, 3 domain experts and method engineers, and a team of two developers have been involved.

3.1 Evaluation Results

The evaluation process was enacted in the following steps: (1) Provisioning: the tools -of the toolbox have been provided to the stakeholders in the involved cases. (2) Team Formation: representatives, -of the stakeholders in the project created development teams consisting of domain experts and method engineers following the conceptualization process and developing tools individually. (3) Feedback Phase: individual results have been consolidated periodically through video conferences and workshops, constituting the evaluation results. In all cases each tool from the toolbox except ALDE has been utilized, ALDE has been utilized just in Case 3.

Feedback on MMDE

Pro: It is possible to specify requirements and dependencies among them as well as tracing them; (2) to define modelling language fragments and modules, (3) layering the modelling language with navigational constructs; (4) definition of syntax, semantic and assignment of notation (concrete syntax); (5) definition of weaving among construct in different meta-models; (6) assignment of (multiple-) graphical notation (concrete syntax); (7) explicit definition of modelling procedure;

Contra: It is not possible to define application scenarios and use cases, and design results can be exchanged solely using ADOxx specific formats or as static content (image, PDF or HTML).

Feedback on ALDE

Pro: It is possible to transform libraries in a machine as well as human interpretable format, ability to use reasoning algorithms, due to standard semantic formats; reduces complexity to edit, merge and maintain libraries.

Contra: To take over results from Design Phase require manual steps. Without knowledge of RDF and Turtle syntax, it is difficult for software engineers that using well-known programing languages (e.g. Java, C++), to get used familiar with; it requires different transformation scripts for different meta-modelling technologies (such as ADOxx, EMF).

Feedback on ADOxx.org Tool Packing Services and Developer Spaces

Pro: It is possible to have an installation package to distribute to interested stakeholders, building your own community around the modelling method, and get feedback from them.

Contra: Setting up and handling issues of a certain Developer Space involves certain manual steps, such, as the interested stakeholder has to send an e-mail to the administrator with a request of an own Developer Space.

4 Conclusion and Outlook

In this paper we introduce a toolbox instantiating the Modelling Method Conceptualization Process, which supports agile modelling method engineering. The toolbox has been evaluated through an analysis of three different cases: two EU research projects and one in-house project. The evaluation results put forward that having an approach and a corresponding toolbox following the idea of model-driven engineering approach is effective in terms of transferring knowledge from the analysis of requirements up to the development of solutions. Being two main tools, MMDE and ALDE, experimental prototypes that are at very early stage of development, lack of full integration or automatic data exchange ability, and the need of manual steps building Developer Spaces came out as major limitations of the toolbox. As an outlook the following items derived from the evaluation: (1) currently we are evaluating development alternatives of MM-DSLs with using Java, Xtend [28] or RDF; building on existing work [29] in the field, (2) enabling graphical modelling method design to transform into machine understandable format, (3) formalization of modelling method design using mathematical models such as FDMM [30], (4) automatization of tooling services and deployment onto developer spaces, (5) full integration of tools around new MM-DSL.

Acknowledgment. This work has been partly supported by the European Commission co-funded projects Learn PAd (www.learnpad.eu) under contract FP7- 619583 and CloudSocket (www.cloudsocket.eu), under contract H2020-ICT 644690.

References

1. Karagiannis, D., Kühn, H.: Metamodelling platforms. In: Stuckenschmidt, H., Jannach, D. (eds.) EC-Web 2015. LNCS, vol. 239, p. 182. Springer, Heidelberg (2002). doi:10.1007/3-540-45705-4_19
2. Karagiannis, D.: Agile modeling method engineering. In: Proceedings of the 19th Panhellenic Conference on Informatics, Athens, Greece, pp. 5–10. ACM (2015)
3. Hrgovcic, V., Karagiannis, D., Woitsch, R.: Conceptual modeling of the organisational aspects for distributed applications: the semantic lifting approach. In: IEEE COMPSACW (2013)
4. Efendioglu, N., Woitsch, R., Karagiannis, D.: Modelling method design: a model-drive approach. In: IIWAS 2015: Proceedings of the 17th International Conference on Information Integration and Web-based Applications, Brussels, Belgium. ACM (2015)
5. Open Models Laboratory (OMILab): Idea and Objectives (2015). http://austria.omilab.org/psm/about. Accessed 15 July 2016
6. ADOxx.org: LearnPAd Developer Space (2015). https://www.adoxx.org/live/web/learnpad-developer-space/design-environment. Accessed 07 July 2016
7. Object Management Group (OMG): Documents Associated with UML Version 2.0 (2005). http://www.omg.org/spec/UML/2.0/. Accessed 12 July 2015
8. Kühn, H.: Method integration in business engineering, Ph.D. thesis (in German), University of Vienna (2004)
9. Woitsch, R.: Hybrid modeling: an instrument for conceptual interoperability. In: Revolutionizing Enterprise Interoperability Through Scientific Foundations, Hershey, pp. 97–118 (2014)
10. Selic, B.: The theory and practice of modeling language design for model-based software engineering—a personal perspective. In: Fernandes, J.M., Lämmel, R., Visser, J., Saraiva, J. (eds.) Generative and Transformational Techniques in Software Engineering III. LNCS, vol. 6491, pp. 290–321. Springer, Heidelberg (2011)
11. Karagiannis, D., Hrgovcic, V., Woitsch, R.: Model driven design for e-applications: the meta model approach. In: Proceedings of the 13th International Conference on Information Integration and Web-based Applications and Services, iiWAS11, Ho Chi Minh City, Vietnam, pp. 451–454. ACM (2011)
12. W3C: RDF-Resource Description Framework (2014). https://www.w3.org/RDF/. Accessed 14 July 2016
13. Eclipse Foundation: Eclipse IDE for Java EE Developers (2016). http://www.eclipse.org/downloads/packages/. Accessed 14 July 2016
14. The Apache Software Foundation: Apache Ant Download (2016). https://www.apache.org/dist/ant/binaries/. Accessed 14 July 2016
15. W3C: RDF 1.1 Turtle Terse RDF Triple Language (2014). https://www.w3.org/TR/2014/REC-turtle-20140225/. Accessed 14 July 2016
16. The Research Group Agile Knowledge Engineering and Semantic Web (AKSW), University of Leipzig, "Xturtle" (2015). http://aksw.org/Projects/Xturtle.html. Accessed 15 July 2016
17. ADOxx.org: AutoPDP Tool Packaging Service (2016). https://www.adoxx.org/live/autopdp-packaging-service. Accessed 14 July 2016
18. ADOxx.org: ADOxx.org Developer Spaces (2016). https://www.adoxx.org/live/development-spaces. Accessed 14 July 2016
19. Learn PAd Consortium: The EU Project Learn PAd (2014). http://www.learnpad.eu/. Accessed 15 July 2016

20. LearnPAd Consortium: LearnPAd Developer Space (2015). https://www.adoxx.org/live/web/learnpad-developer-space. Accessed 15 July 2016
21. LearnPAd Consortium: LearnPAd Developer Space - Downloads (2015). https://www.adoxx.org/live/web/learnpad-developer-space/downloads. Accessed 15 July 2016
22. CloudSocket Consortium: CloudSocket Project (2016). https://www.cloudsocket.eu/. Accessed 14 July 2016
23. Woitsch, R., Utz, W.: Business process as a service, model based business and IT cloud alignment as a cloud offering. In: ES 2015, Third International Conference on Enterprise Systems, Basel, Switzerland (2015)
24. CloudSocket Consortium: CloudSocket Developer Space (2015). https://www.adoxx.org/live/web/cloudsocket-developer-space/. Accessed 15 July 2016
25. CloudSocket Consortium: CloudSocket Developer Space - Downloads (2015). https://www.adoxx.org/live/web/cloudsocket-developer-space/downloads. Accessed 15 July 2016
26. Object Management Group (OMG): Decision Model and Notation, Version 1.0 (2015). http://www.omg.org/spec/DMN/1.0/. Accessed 15 July 2016
27. Object Management Group (OMG): Business Model and Notation Version 2.0 (2011). http://www.omg.org/spec/BPMN/2.0/. Accessed 15 July 2016
28. Eclipse Foundation: Xtend (2015). https://eclipse.org/xtend. Accessed 15 July 2016
29. Visic, N., Karagiannis, D.: Developing conceptual modeling tools using a DSL. In: Buchmann, R., Kifor, C.V., Yu, J. (eds.) KSEM 2014. LNCS, vol. 8793, pp. 162–173. Springer, Heidelberg (2014)
30. Fill, H.-G., Redmond, T., Karagiannis, D.: FDMM: a formalism for describing ADOxx meta models and models. In: Proceedings of ICEIS 2012, Wroclaw, Poland, vol. 3, pp. 133–144 (2012)
31. ADOxx.org "GraphRep" (2016). https://www.adoxx.org/live/graphrep. Accessed 31 Aug 2016