

Modelling Method Design: An ADOxx Realisation

Nesat Efendioglu

BOC Asset Management GmbH

Operngasse 20B

1040, Vienna, Austria

nesat.efendioglu@boc-eu.com

Robert Woitsch

BOC Asset Management GmbH

Operngasse 20B

1040, Vienna, Austria

robert.woitsch@boc-eu.com

Abstract—The importance of Modelling Method Engineering is equally rising with the importance of Domain Specific Languages and individual modelling approaches. 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. Based on practical experience in business, more than twenty EU-projects and other research initiatives, this paper presents improved and extended version of a model-driven approach to support the design of a modelling method. The approach is evaluated by two projects in the context of e-Learning, and Business and IT-Cloud Alignment. The paper discusses the evaluation results and derived outlooks.

Keywords— *Meta-modelling; Modelling Method Design; Modelling Method Engineering; Knowledge acquisition; Evolutionary prototyping*

I. INTRODUCTION

Conceptual models are commodities in IT and computer science, hence a growing number of groups around the world design their individual modeling-methods - in addition to existing standards- for a variety of application domains. Therefore, models are not only used for schema or software generation but enable information value-creation. Hence, modeling-methods need to provide the necessary concepts and functionality to perform value creation within a particular application domain. The construction of such applicable modeling methods – also titled as conceptualization process - is complex, especially when considering the mapping of the entire spectrum from language artifacts and corresponding functionality to concrete implementable and deployable software-platform capabilities. Today, different approaches, guidelines and practices for the development of modeling tools are available that do not consider the full spectrum of a modelling method design, which unavoidably leads to limitations in the conceptualization. There are branching knowledge domains into more refined and specialized subdomains, where each domain needs to be characterised by its own abstraction and refinement of concepts from reality [19].

ADOxx is a meta modelling platform that has been used (a) in business from the consulting company BOC [1] in realizing their products ADONIS®, ADOScore®, ADOlog®, ADOit® as well as PROfit®, PROMOTE® and ADVISOR®, (b) in over thirty University Research and Research

Institute collaboration through the Open Models community at OMiLAB as well as (c) in over twenty European Research projects. Although the ADOxx.org community [1] is focusing on the technical realization of modelling tools, it is often also involved in the conceptual specifications of model-based approaches in the context of European Research projects. Based on the experience that has been collected during the realization of model-based approaches in more than twenty European Projects since 2000; a modelling method design environment is proposed that enables the specification, implementation-support and documentation of modelling methods. In order to specify the relevant concepts that are satisfying domain-specific needs, it is necessary to involve both method engineers as well as domain experts in the process of modelling method design [13]. Hence, in order to establish communication, collaboration among actors and supporting knowledge-externalization with an appropriate modelling method design approach is crucial.

The modelling method design approach that is introduced in this paper is based on the overall Agile Modelling Method Engineering approach [12].

This paper is based on the full version published in [8], where the modelling method design approach has been discussed in detail and the first version of its corresponding software prototype introduced. The goals of this paper is (1) to revisit and update the content with introducing second version of the prototype, discuss new evaluation results, and (2) introducing early implementation results of an experimental prototype, which shall complement the approach and be extension to the previous works.

Being a revised and extended version of [8], following the introduction, in Section 2 we revisit our approach with introducing second version of its corresponding software prototype, so-called “Modelling Method Design Environment 2.0”, which is improved based on previous evaluation results. Additionally, in this section, a new extension to the approach is introduced In Section 3, the cases from two different EU projects are briefly presented as evaluation cases for the Modelling Method Design Environment. In Section 4 the evaluation results are discussed. Finally, in Section 5 this work is concluded an outlook on future work is provided.

II. THE MODELLING METHOD DESIGN ENVIRONMENT

A. Modelling Method Conceptualization Process

Generic Modelling Method Specification Framework (GMMF) by Karagiannis and Kühn[14][10] is our reference framework for conceptualization of modelling methods. This section introduces the Modelling Method Conceptualization Process (as depicted in Figure 1) based on the OMILAB Lifecycle [16], which is, again, based on GMMF. The Conceptualization Process has been extended during a series of meta-modelling workshops within the EU-project Learn PAD, as a process to conceptualize modelling methods. Originally, the OMILAB Lifecycle consists of five phases; (1) Create Phase, where the system under study, application scenarios and their requirements are investigated, (2) Design Phase, where the modelling language with its syntax, semantic and notation (concrete syntax) is specified, mechanisms and algorithms are specified, (3) Formalization Phase, where the specification of modelling language is well-formalized in order to ensure a possible implementation into a software. (4) Development Phase, where the specified modelling method is implemented on a meta-modelling environment, (5) Deployment phase, where the implementation is packaged in form of standalone application and offered for use to end-users.

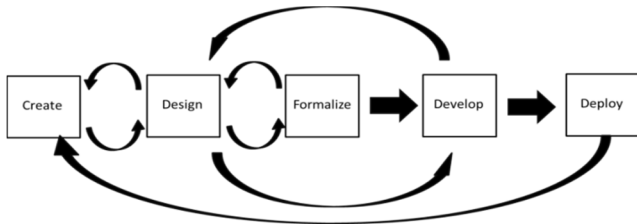


Figure 1. Modelling Method Conceptualization Process (based on OMILAB Life Cycle [16])

On one hand, having stakeholders with varying knowledge backgrounds and perspectives involved in the conceptualization of a modelling method, and on the other hand, having end-users with different objectives, the feedback channels proposed by OMILAB Lifecycle would be not enough to react quickly to required changes and enable continuous improvement. Therefore, the OMILAB Lifecycle is adapted (as depicted in Figure 1) by adding feedback channels between creation and design in order to prove, if the designed modelling language covers identified application scenarios and considers the identified requirements; between design and formalization to ensure formal approval of modelling language; as well as between design and development phases in order to improve modelling language in earlier stages before it is released and deployed.

The abstraction of domain context-specific reality and design of the “system” in an appropriate way have the most significant impact on the rest of the Conceptualization Process. Hence we concentrate in this work on

development of an appropriate modelling method for modelling method design supporting these two phases.

”Model Driven Design for e-Applications: The Meta Model Approach” by Karagiannis and et al. [13] identifies the actions to investigate application scenarios and requirements of a domain as well as to design that domain specific modelling method (for detailed descriptions of those actions please refer to [13]).

B. Modelling Method Design Environment

Following the model-driven approach, we propose a modelling method, the so-called “Modelling Method Design Environment (MMDE)”, to support the design of other modelling methods. During the development of MMDE, a subset of UML has been taken and extended with new concepts based on lessons learned from state-of the art analysis.

Having in mind that the MMDE itself is a modelling method which aims to cover requirements of Create and Design Phases during the Conceptualization Process, it also has a Modelling Language, Mechanism & Algorithms and a Modelling Procedure.

In the following sections, (1) we specify, first of all, the concepts in the Modelling Language of MMDE categorizing them as support for Create or Design Phases and then (2) we specify functionalities of MMDE, which are –with another words- its mechanisms and algorithms.

1) Modelling Language of Modelling Method Design Environment

As aforementioned, the Modelling Method Design Environment (MMDE) aims to support the Create and Design Phase (Based on the lessons learned in state-of the art analysis and based on the expertise of authors of this work, who have been involved in several (one of them 5, the other more than 30) modelling method development projects for varying domains, (1) UML has been identified as the best possible starting point. We have taken a subset of UML and extended the modelling language with required concepts and model types in order to meet shortcomings, (2) developed the modelling method “Modelling Method Design Environment” and (3) implemented a corresponding fully-fledged modelling editor, the so-called “Modelling Method Design Environment Modelling Toolkit” is available to download free of charge with sample models in [2]. In the following sections we specify the MMDE with introducing its modelling constructs as solutions for the Create and Design Phases with giving snippets from sample models modelled on the Modelling Method Design Environment Modelling Toolkit version 2.0..

a) Solutions to Support Create Phase.

First of all, to be able to describe and trace the requirements a model type called “Requirements Model Type” has been specified.

The Requirements Model Type (as depicted in Figure 2.) allows describing the 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 Meta-Model Classes models, (b) graphical notation (concrete syntax) definitions modelled in Graphical Notation models, (c) model types/modules modelled in Modelling Stack models and (d) to the functionalities modelled in Mechanisms & Algorithms models.

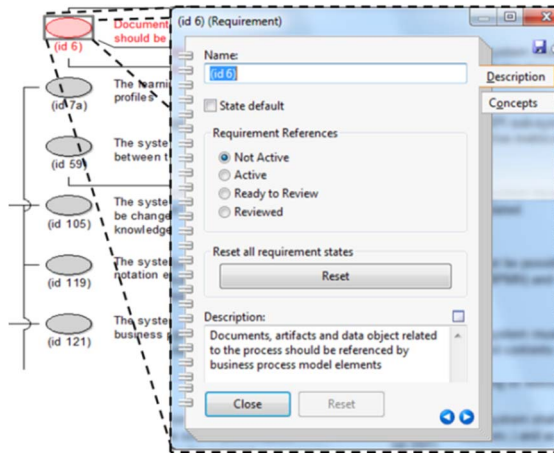


Figure 2. Sample Requirement Model and Description of Requirements

As depicted in Figure 3. , depending on the status of those constructs (if they are considered, reviewed or dismissed), the status of the requirements will be updated accordingly. It is enough, if the user clicks on the menu item “Update Requirements Status”, the design environment will check if the requirement has been considered in any constructs within the designed modelling method - by checking referenced requirements in all modelling objects in models designed for given modelling method. The MMDE will update the status of a requirement accordingly and a list of constructs will be updated and presented in the notebook of the given requirement indicating where the requirement has been considered. The MMDE differentiates the status of requirements with colour codes; Grey, if the requirement is not considered yet, Red if the requirement is not considered and dismissed, Yellow, if the requirement is still in consideration, and Green if the requirement is considered and reviewed.

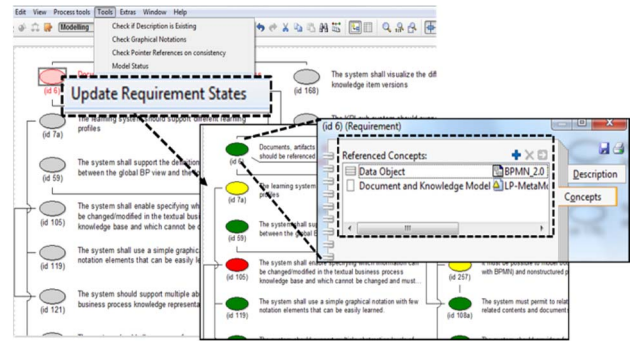


Figure 3. Before and After automatically Updating the Status of Requirements

b) Solutions to Support Design Phase

As it has been discussed, the Design Phase is the most crucial and effort-intensive phase of modelling method engineering. Moreover, since the Design involves the specification of concrete modelling language, procedure and mechanisms & algorithms are being made and when the more intensive collaboration and knowledge exchange is needed. The following solutions are offered by the MMDE for the Design Phase.

First of all, to ease the readability and navigation we specified a new model type “Modelling Stack”

Modelling Stack Model Type (as depicted in Figure 4) Domain specific modelling methods usually specify different fragments of complex systems. The different fragments of the system are represented by different model types with merging-relevant information[13]. The modularization and layering of the modelling language is essential to avoid complexities during the design of domain-specific modelling methods. Hence, we propose a model type called “Modelling Stack Model Type” that allows method engineers to differentiate between meta-models considering them as different model type that targets different fragments of the system. This model type allows the description of hierarchy and dependencies among meta-models on a higher-level, without giving details about each meta-model neither in the form of classes, attributes nor cardinalities etc. This model type consists of navigational modelling concepts which enable aggregation of meta-models, definition of their scope and relations among them. On the other hand this model type allows to trace the development status of each meta-model, in other words the status of each model type in terms of a release workflow. All fragments (in form of model types) defined in this type of models can be linked through a pointer concept to the abstract syntax defined in individual *Meta-Model Classes Models* and concrete syntax defined in individual *Graphical Notation Models*.

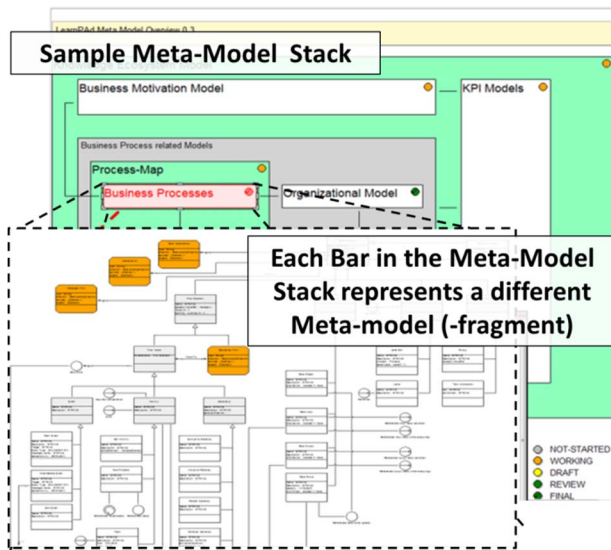


Figure 4. Meta-Model Stack Model Type Sample

Meta-Model Classes Model Type: Model Types in domain specific languages are composed of abstract syntax, concrete syntax and semantics. This model type enables the specification of abstract syntax of individual model types in form of meta-models as with their semantic primitives; classes, attributes and associations (as depicted in Figure 5). In this model type, the MMDE allows –semantic mapping (as depicted in Figure 6) using existing ontologies defined in RDF or OWL format with utilizing one of the semantic lifting scenarios proposed in [12]. Moreover this model type provides a specific relation called “Pointer” in order enable weaving or specify relation among constructs in different model types (representing different fragments of a system).

Having notation of Class Diagrams from UML suggested by OMG as the initial notation in mind for this model type, we evaluate the graphical notation via multi-user tests in several iterations. We have suggested some adaptations such as differentiating the representation of abstract classes, classes and relation classes from each other, as well as dynamic representation of Pointer construct -depending on which weaving type it is representing, in order to make this model type more intuitive and easy to read.

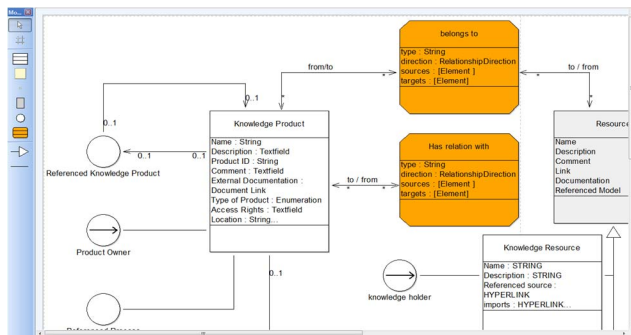


Figure 5. Meta-model Classes Model Type Sample

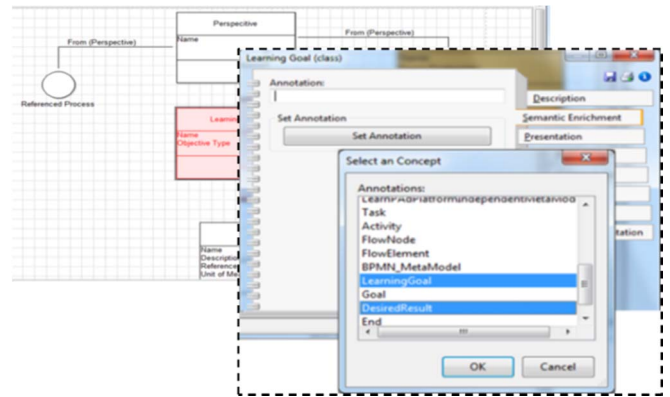


Figure 6. Definition of Semantics via Semantic Annotation Functionality

In order to specify a proper graphical representation (concrete syntax) of each concept in a meta-model, we specify another model type called “Graphical Notation Model Type”

Graphical Notation Model Type (as depicted in Figure 7) allows the definition of a concrete syntax of model types with specifying graphical representations for each constructs in meta-models considering non-functional requirements identified during the Create Phase. This model type allows the description of graphical representations with the assignment of concrete images.

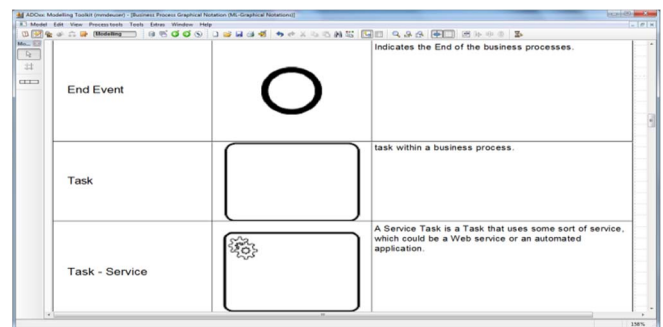


Figure 7. Graphical Notation Model Type Sample

In order to define the applied modelling technique by defining the steps and the results we specified a model type called “Modelling Procedure Modelling Type”

Modelling Procedure Model Type (as depicted in Figure 8) allows method engineers define the steps with their required inputs and produced outputs and the sequence of steps based input – output relationship in order to introduce case specific proper usage of their modelling method.

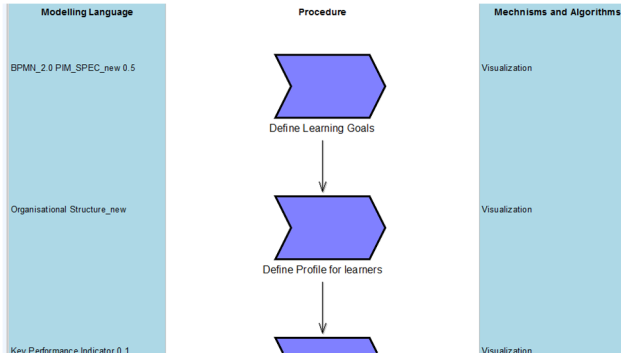


Figure 8. Modelling Procedure Model Type Sample

The version of MMDE is being evaluated in this paper supports design of Mechanisms and Algorithms of other modelling methods with standard UML 2.0 Sequence and Component Diagrams with exact same syntax and semantic.

2) Mechanisms and Algorithms of Modelling Method Design Environment

a) Transformation

Export and Import Models: The actors involved in the Conceptualization Process should be able to exchange their design, their feedback or open questions created in their own MMDE instance with other actor. Therefore MMDE offers model set import and export in formats ADL and XML.

Image Generation: All actors should not necessarily be users of MMDE, but they should still be able to give their feedback on designs. Therefore, we provide the generation of image models, so that actors can also discuss based on images in formats BMP, PCX, JPG, EMF as well as SVG and give their feedback.

HTML Generation: Similarly to the idea behind the generation of images of models, all actors are not necessarily users of MMDE. Instead of distribution of images to individual persons, it should be possible to deploy models on the web, make it accessible over the internet and enable discussion on the central knowledge resource. Therefore, the MMDE enables transformation of specification and images of models in HTML to be deployed on the web.

Modelling Method Specification Report: Beside HTML Generation the MMDE offers a transformation of the specification and the images into reports in PDF format with a pre-defined report template.

b) Analysis

Validation: For implementation of validation logic in the MMDE, two possible technologies are foreseen; (1) using ADOxx specific query-language AQL or (2) using semantic technologies (e.g reasoners). At the moment, MMDE validates meta-models based on the following constraint; **(a)** each model type must have a unique name, **(b)** each model type must contain either at least one class,

(c) each model type must be assigned to one modelling stack **(d)** each class must have a unique name. **(e)** each relation class must have a unique name, **(f)** each relation class must have at least one source endpoint and at least one target endpoint, **(g)** for each relation class endpoint cardinality must be defined, **(h)** each attribute must have a unique name, **(i)** each attribute must have an attribute type, **(j)** each requirement must be assigned to an object in a model within modelling method design. The validation logic based on AQL has already been implemented. The logic based on semantic technologies is under evaluation.

C. New Extension: ADOxx Library Development Environment (ALDE)

The paper [8] is concluded with following future work and research questions;

“(1) How to transform meta-models specified as graphical models into formal models automatically (or semi-automatically) in order to support seamless flow from Design to Development Phase, (2) what is the appropriate formalisms and formal format to transform into”

The investigations and early experiments put forward the following challenges that should be considered for development of a solution to answer the above-mentioned research questions; The solution shall overcome following challenges (C1) transformation of graphical design to machine-interpretable code (modelling editor libraries), (C2) parallel development of modelling editor libraries, (C3) merging different modelling editor libraries, (C4) ensuring maintainability of the libraries.

In this work, we share early results of an experimental prototype implementation that is based on a standard data interchange model, “RDF, so-called “ADOxx Library Development Environment”. The solution is foreseen as an extension to the MMDE.

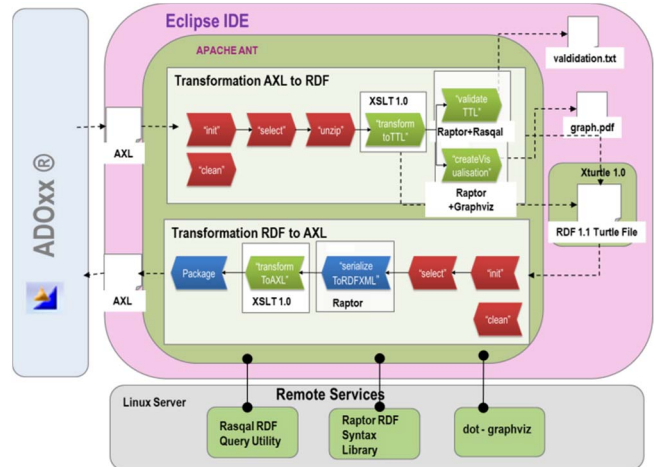


Figure 9. Architecture of ADOxx Library Development Environment (ALDE)

The Figure 9 depicts architecture of the ADOxx Library Development Environment (ALDE). The ALDE is development environment based on eclipse IDE [7]. The

ALDE includes a meta²model defined in RDFS, the so-called ALDE vocabulary. Having the vocabulary and utilizing Apache Ant® [21], ALDE enables the definition of transformation processes from ADOxx Library Languages to RDF and vice versa. Moreover the ALDE serializes the libraries in RDF Turtle [25] format and includes a RDF Turtle editor. Having libraries in RDF Turtle format and a RDF Turtle editor available, method engineers can adapt, script the libraries collaboratively (SVN is required) (to overcome challenge C2), further as well as merge several libraries or integrate part of libraries in each other (to overcome challenge C3). On the other hand ALDE includes validation services to validated newly developed, edited or merged libraries (to overcome challenge C4). Being in the early stage of the implementation, ALDE has following limitations; it is not integrated to MMDE yet; still it is not possible to transform graphical design (a graph as it is modelled on MMDE) directly into RDF; ALDE still requires ADL or AXL files as input.

```

===== CLASS DEFINITION =====
:REPOSITORY a :class;
:isAbstract false;
:isVisible false;
:subClassOf rdf:nil;
:hasAttribute :NAME33;
.

:C_A a :class;
:isAbstract true;
:isVisible false;
:subClassOf rdf:nil;
:identifiertext "A is a Class"@en;
:hasAttribute :POSY;
:hasAttribute :GRAPHREP;
:hasAttribute :POSX;
:hasAttribute :ZORDER;
:hasAttribute :VISIBLE;
:hasAttribute :NAME33;
.

:C_C a :class;
:isAbstract false;
:isVisible true;
:subClassOf :C_A;
:identifiertext "C is subclass of";
.

===== MODELTYPE DEFINITION =====
:NT_MY_MODELTYPE a :modeltype;
:isVisible true;
:identifiertext "My Model Type"@en;
:hasAttribute :A_DESCRIPTION;
:hasAttribute :NAME;
.

:includesClass :C_A;
:includesClass :C_B;
:includesClass :C_C;
:includesClass :C_D;
:includesClass :RC_DTOC;
:includesClass :RC_CTOO;

```

Figure 10. Sample Class and Model Type Definitions in ALDE

III. EVALUATION CASES

We applied the MMDE in two different cases so far within two different EU-Projects, which are in different domains and have consortiums with partners having highly varying profile. The improved and extended version of MMDE applied again in the same cases within the same EU-Projects during the development of second versions of their modelling methods. In following sub-section, for completeness reasons, we briefly introduce these cases and their requirements again.

A. Design of a Modelling Method for E-Learning in Public Administration

The FP7 project Learn PAd [15] proposes a process-driven-knowledge management approach based on conceptual and semantic models for transformation of public administration organizations into learning organizations. In this scenario the following challenges are identified. The

first challenge is (1) modelling method has to consider the different aspects; (a) business processes, (b) motivation, (c) organizational structure and roles, (d) knowledge resources, (e) learning goals, (f) performance measurements. The second one is, existing technologies that shall realize the overall system need to be considered in order to enable a flexible deployment into the legacy Application. (e.g ADOxx®, XWiki®). The third one is, the modelling language shall be flexible enough to be implemented on different platforms hence the consortium focuses on a new way of defining the meta-model. The fourth and last challenge is, the need of collaboration of multiple experts for development of the modelling method. Given that the multiple experts are from different domains (from software engineering and modelling, business process modelling and analysis, knowledge and learning management, and public administrations shall collaborate to develop the modelling method), the development approach shall be appropriate to varying background of those experts.

B. Design of Modelling Method for Business and IT-Alignment

The H2020 project CloudSocket [6] 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 [22].

The CloudSocket consortium requires defining Business Processes at three levels; (1) Business Process at the Design Level: They should be domain specific business processes that describe the business activities of a worker assigned to that process. They are not executable, neither by a workflow engine within or outside the cloud. (2) Workflows: They should be executable business processes. They are represented as workflows that orchestrate the interaction between software applications. Cloud Deployment Level: They are workflow bundles, which are deployable on cloud. Those bundles packaged include all relevant deployment configurations, so that it can be deployed on demand.

The above mentioned requirements bring the challenge of consideration of meta-models for environments on those three levels. Hence collaboration among stakeholders with varying background, such as domain experts, business process management experts, workflow management system experts, cloud infrastructure (hardware and software) experts as well as brokers between business clients and cloud service providers.

IV. EVALUATION RESULTS

We followed the same evaluation process as in the previous work [8] to evaluate MMDE in the second iteration of development of the above-mentioned modelling methods. The only exception is the ALDE extension, which has been evaluated in an internal project directly by modelling method developers.

The installation package of the new version of MMDE has been prepared and circulated to the corresponding partner

representatives, and they circulated them to their teams. They have continued with further conceptualization as well as with the implementation of corresponding modelling tools. The individual works has been consolidated periodically through video conferences.

The feedback has been given verbally by the participating actors in each workshop and video conferences. We have concentrated on three categories during the evaluation; feedback on (1) the Modelling Language (Constructs) of the MMDE, (2) the Mechanism and Algorithms (Functionalities) of MMDE and finally (3) the new extension ALDE.

Feedback on the Modelling Language

Pro: Ability of definition of requirements and dependencies among them as well as status tracing; Ability of definition modelling language fragments and modules, layering the modelling language with navigational constructs; Release flow; Ability of definition of Syntax, Semantic and assignment of notation (concrete syntax); Definition of Weaving among construct in different meta-models with Pointer Construct; Ability of assignment of (multiple-) graphical notation (concrete syntax); Explicit definition of modelling procedure; Ability of using existing ontologies in well-known formats

Contra: It is not possible to define application scenarios and use cases.

Feedback on the Mechanisms & Algorithms

Pro Ability of exchanging models; Ability of central information as an image in well-known formats; Ability of exporting all information in the model in structured way; Accessing the information via Internet; Ability of all information in the model in structured way; Ability of validation of meta-models within the same environment.

Contra: The existing pre-defined constraints not enough to validate a meta-model; Meta-model validation patterns are nice to have. It is still not possible to export all model content as RDF.

Feedback on the Extension ALDE

Pro: Ability to transform libraries in a machine as well as human interpretable format; Ability to use reasoning algorithms, thanks to RDF; Ability to edit, merge and maintain libraries.

Contra: It requires different transformation scripts for different meta-modelling technologies (e.g ADOxx 1.0, ADOxx 2.0, EMF).

Having a modelling method and its corresponding toolkit, which are following the idea of Model-driven engineering approach, has been proven to be effective in terms of transferring knowledge from the analysis of requirement up to the development of solutions (implemented early domain specific solutions (modelling method and corresponding modelling tool) based on created modelling method designs

on MMDE can be found for case 1 in [4] and for case 2 in [3].

The involved actors agreed on that, the approach presented in this paper has accelerated establishing communication and collaboration, and eased knowledge exchange between actors with different backgrounds, as well as enabled tracing consideration of requirements within the solutions. Evaluation of the MMDE in two concrete cases within complex IT-Projects with multi-discipline and cultural consortiums gave us the occasion to detect further non-functional and functional requirements and required improvements on existing features of the MMDE.

V. CONCLUSION AND OUTLOOK

We introduce our modelling method design approach and corresponding design environment MMDE with revisiting the paper [8].

We evaluated the second version of MMDE and the extension ALDE in two complex IT-Projects and determined that our approach has accelerated establishing communication and collaboration, eased knowledge exchange between actors with different backgrounds as well as tracing consideration of requirements within the solutions. The evaluation results of the preliminary version of ALDE has proven that we are in the right direction to answer research questions posed in previous work [8] based on previous evaluation results.

VI. ACKNOWLEDGMENTS

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

VII. REFERENCES

- [1] ADOxx.org Portal, www.adoxx.org, [Accessed]
- [2] ADOxx.org, 2014. *Modelling Method Design Environment*. [Online] Available at: <http://www.adoxx.org/live/web/learnpad-developer-space/design-environment> [Accessed 01 July 2015].
- [3] BOC, "CloudSocket - BPaaS Designer," 2016. [Online]. Available: <https://www.cloudsocket.eu/ADOxxWeb/login.jsp>. [Accessed 27 April 2016]
- [4] BOC, "Learn PAd Developer Space - Learn PAd Modelling Environment," 2016. [Online]. Available: <https://www.adoxx.org/live/web/learnpad-developer-space/prototype-v5.0>. [Accessed 27 April 2016]
- [5] Brown, M. M., & Brudney, J. L. 2003 "Learning organizations in the public sector? A study of police agencies employing information and technology to," *Public administration review*, 63(1), pp. 30-43,
- [6] CloudSocket Project Consortium, 2016. *CloudSocket-Project*. [Online]

- Available at: <https://www.cloudsocket.eu/> [Accessed 27 April 2016]
- [7] Eclipse Foundation, Eclipse IDE for Java EE Developers [Online] Available at <http://www.eclipse.org/downloads/packages/> [Accessed 01 December 2015]
- [8] Efendioglu N., Woitsch R., Karagiannis D., Modelling Method Design: A Model-Driven Approach, in Proceedings of the 17th International Conference on Information Integration and Web-based Applications & Services, iiWAS 2015, 11-13 December 2013, Brussels.
- [9] Glinz, Martin 2000. Problems and Deficiencies of UML as a Requirements Specification Language in *Proceedings of the 10th International Workshop on Software Specification and Design IWSSD'00* IEEE Computer Society
- [10] H. Kühn, "Methodenintegration im Business Engineering" [in German] PhD Thesis, University of Vienna, April 2004
- [11] Karagiannis D., 2015, Agile Modelling Method Engineering, Proceedings of the 19th Panhellenic Conference on Informatics, ACM, New York, 2015
- [12] Hrgovcic, V., Karagiannis D., Woitsch, R. 2013 ., Conceptual Modeling of the Organisa-tional Aspects for Distributed Applications: The Semantic Lifting Approach," in *COMPSACW, IEEE*, 2013.
- [13] Karagiannis, D., Hrgovcic, V. & Woitsch, R., 2011. 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: ACM, pp. 451-454.
- [14] Karagiannis, D.; Kühn, H.: Metamodelling Platforms, DEXA 2002, Aix-en-Provence, France, LNCS 2455, Springer-Verlag, Berlin, Heidelberg, pp. 182.
- [15] Learn PAd Project Consortium, 2014. *The EU Project Learn PAd*. [Online] Available at: <http://www.learnpad.eu/> [Accessed May 2015].
- [16] OMILAB, OMILAB Modelling Methods, <http://www.omilab.org/web/guest/booklet>, 2014, [Accessed 07 September 2015]
- [17] OMILAB, n.d. *OMILAB Life Cycle*. [Online] Available at <http://www.omilab.org/web/guest/about> [Accessed 13 July 2015].
- [18] Saadatmand, M., Cicchetti, A. & Sjödin, M. 2011, UML-Based Modeling of Non-Functional Requirements in Telecommunication Systems in *The Sixth International Conference on Software Engineering Advances*, ICSEA 2011
- [19] Selic, B., 2011. The Theory and Practice of Modeling Language Design for Model-Based Software Engineering—A Personal Perspective. In: *Generative and Transformational Techniques in Software Engineering III*. s.l.:Springer Berlin Heidelberg, pp. 290-321.
- [20] Silingas, D. et al., 2015. Business Architecture For Process-Oriented Learning in Public Administration. In: *Business and Dynamic Change: The Arrival of Business Architecture*. s.l.:Future Strategies Inc., Book Division, pp. 171-185.
- [21] The Apache Software Foundation, Apache Ant®, [Online] Available at <http://ant.apache.org/> [Accessed 01 December 2015]
- [22] Woitsch, R. & Utz, W 2015. "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
- [23] Fill H.G., Karagiannis D., 2013, On the Conceptualisation of Modelling Methods Using the ADOxx Meta Modelling Platform, In: *Enterprise Modelling and Information Systems Architectures, Vol.8, No. 1, March 2013, SIG EMISA 2013*.
- [24] W3C, Resource Description Framework (RDF): Concepts and Abstract Syntax. Graham Klyne and Jeremy J. Carroll, eds. W3C Recommendation, 10 February 2004, [Online] Available at <http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/>. [Accessed 20 July 2015]
- [25] W3C, RDF 1.1 Turtle Terse RDF Triple Language, [Online] Available at <https://www.w3.org/TR/2014/REC-turtle-20140225/> [Accessed 02 Dezember 2015]