

ADOxx Modelling Method Conceptualization Environment

Nesat Efendioglu*, Robert Woitsch, Wilfrid Utz, Damiano Falcioni

BOC Asset Management GmbH, Innovation Group, AT-1040, Austria

ARTICLE INFO

Article history:

Received: 03 March, 2017

Accepted: 07 April, 2017

Online: 18 April, 2017

Keywords :

Meta-modelling, Modelling
Method Design, Agile Modelling
Method Engineering
Conceptualization

ABSTRACT

The importance of Modelling Methods Engineering is equally rising with the importance of domain specific languages (DSL) and individual modelling approaches. In order to capture the relevant semantic primitives for a particular domain, it is necessary to involve both, (a) domain experts, who identify relevant concepts as well as (b) method engineers who compose a valid and applicable modelling approach. This process consists of a conceptual design of formal or semi-formal of modelling method as well as a reliable, migratable, maintainable and user friendly software development of the resulting modelling tool. Modelling Method Engineering cycle is often under-estimated as both the conceptual architecture requires formal verification and the tool implementation requires practical usability, hence we propose a guideline and corresponding tools to support actors with different background along this complex engineering process. Based on practical experience in business, more than twenty research projects within the EU frame programmes and a number of bilateral research initiatives, this paper introduces the phases, corresponding a toolbox and lessons learned with the aim to support the engineering of a modelling method. "The proposed approach is illustrated and validated within use cases from three different EU-funded research projects in the fields of (1) Industry 4.0, (2) e-learning and (3) cloud computing. The paper discusses the approach, the evaluation results and derived outlooks.

1. Introduction

This paper is an extension of work originally published in 2016 IEEE 20th International Enterprise Distributed Object Computing Workshop (EDOCW)[1]

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 (de-facto-) standards (e.g. Business Process Model and Notation (BPMN)[2], Decision Model and Notation (DMN)[3], Case Management Model and Notation (CMMN)[4]), a growing number of groups around the world design their individual modelling-methods (in accordance with the definition of such a method by [5], [6] for a variety of application domains.).

This is often seen as necessary, when model-based approaches are transferred in new application domains and hence require adaptations for modelling methods. A simple sample can

demonstrated using the well-known standard for business process BPMN. Although BPMN can be used to design a administrative process, such as sending an invoice, it cannot be used to design a simple production process like producing a chair. The successor relation that indicates that one activity follows the other does not have properties like distance to the station, which is not necessary when sending an invoice, but is of utmost importance, when producing a chair. When analysing more complex scenarios like a car manufacturer shop floor (we faced in projects DISRUPT [60], GOODMAN [65]), the adaptation requirements for a modelling language like BPMN becomes quite complex. Hence, providing well-known model-based approaches requires the adaptation by e.g. introducing the concept "distance" between two activities.

Challenging question is, how to support the generation of modelling tools that can range from a minor adaptation like the one introduced above, till the complete realisation of totally new modelling approach like a cyber threat modelling for cloud computing .

The authors of [8] believe that supporting the automatic generation of modelling tools can open a new quality in

*Corresponding Author: Nesat Efendioglu BOC Asset Management GmbH, Innovation Group, AT-1040, Austria
Email: nesat.efendioglu@boc-eu.com

information systems development for engineers and customized design as well as encourage the use of modelling languages that are fitting to the custom needs. Often customer needs are defined by desired features like visualisation, querying, simulation or configuration and transformation, which are applied onto the model. Individual solutions enable the generation of light-weighted solutions with targeted provision of features that reduce the developers' aversion against overloaded modelling languages and inflexible or expensive modelling tools.

The engineering of such modelling tools is a result of the so called "conceptualization process of modelling methods", which is disseminated by the world-wide community of OMiLAB[10]. The complexity lies in the mapping of language artefacts and their corresponding functionality to concrete implementable and deployable modelling tools. In addition, knowledge domains are divided into more refined and specialized subdomains, where each domain needs to be characterized by its own abstraction and refinement of concepts from the so-called "real world" objects from the "subject under study". Hence, in order to capture the 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, guidelines and practices for the development of modelling tools available that do not consider the full lifecycle from the design and collaborative development of a modelling tool, which unavoidably leads to limitations and inconsistencies in the conceptualization [7]. We, hence, propose a guideline and corresponding tools supporting method engineers along the conceptualization process supporting all phases and ensuring collaboration among involved stakeholders.

Karagiannis proposes in [6] the Agile Modelling Method Engineering (AMME) framework. Authors of [9] propose the Modelling Method Conceptualization Process that based on AMME and guides the method engineers during conceptualization. The same authors in [35] propose a toolbox that supports this process. The work at the hand introduces an extended version of this toolbox so-called "Modelling Method Conceptualization Environment" as well as introduces corresponding services. Hence the paper introduces a product-service-system proposal for modelling method conceptualisation. Moreover the paper evaluates this product-service system in three European Research projects, and one additional in the context of an in-house research project, and discusses evaluation results.

In this respect, the remainder of the paper is structured as follows: Section 2 briefly presents results of our state of the art analysis from [9], revisits AMME, the Modelling Method Conceptualization Process. Section 3 outlines the toolbox Modelling Method Conceptualization Environment with an emphasis on new extension to ADOxx Library Development Environment, so-called ADOxx-JAVA-MM-DSL. Section 4 presents Modelling Method Conceptualization Services, Section 5 introduces evaluation cases and discusses the evaluation results, while Section 6 concludes the paper and gives an outlook on future work.

2. State of the Art in Modelling Method Definition and Development Approaches

We published the initial version of this state of the art analysis in [9]. In this paper, we revisit shortly it for completeness reasons.

We analyse modelling standards from five well-known standardization organizations, which provide intensively used industry modelling standards in last decade; (1) OASIS (www.oasis-open.org), (2) OMG (www.omg.org), (3) Open Group (www.opengroup.org), (4) W3C(www.w3.org) and (5) WfMC(www.wfmc.org). It is highly possible that each organization follows same or similar approach and technologies to specify standards within its organization. Therefore, we assume that, it would be enough selecting control samples from each standardization organization in order to understand, which approach they are following, which tools they are using to define modelling standards. As the control samples we select UBL [39], ebXML BPSS [40] and WS-BPEL [53] from OASIS; BPMN [2] CMMN [4] and DMN [3] from OMG; ArchiMate® [42] from Open Group; OWL [44], SPARQL [45] and WSDL [46] from W3C, finally XPDL [47] from WfMC.

We investigate, how the modelling language of modelling standards has been specified. We focus on four aspects specification of (1) abstract syntax-, (2) semantic- and (3) notation (concrete syntax) of the modelling languages as well as (4) samples provided to ease to understand modelling language specification and its usage.

The results of the analysis are presented in Table 1. In respect of abstract syntax specification organization used (1) graphical approaches and/or, (2) formal textual approaches and/or textual informal approaches. All of organizations utilize UML-Class Diagram [41] to specify abstract syntax of most of their modelling languages. In addition to UML-Class Diagram all standardization institute use natural language to specify the abstract syntax. Additionally, in order to ensure interoperability between systems using the given modelling standard and collaboration among parties utilizing the standard, most of the organizations attach importance to formal textual specification of abstract syntax either using with BNF [48] or XSD [49].

It seems that, regarding to specification of semantics there is no standard/approach used commonly by all organization. W3C uses formal languages such as RDF [50], Z Notation [54] [51] and standard mathematical notation. However, most of the organizations utilize semi-formal keywords defined in RFC-2119 [52] in order to define requirement level and constraints. Additions to these, all organizations use natural language to specify semantics of all modelling standards.

Regarding to specification of notation (concrete syntax), obviously the modelling standards, which have been developed preferentially in the first place for human interpretation, precise has concrete syntax that is illustrated with images and described with natural language. On the other side, the standards, which have been developed for machine interpretation, have concrete syntax specified with formal textual approaches. Mostly, XSD is utilized for specification of concrete syntax of these standards. Finally, all the organizations choose to introduce features of their modelling standards with samples. The samples can be found in either in form of graphical or textual models.

According to analysis results, for the specification of selected modelling standards, the UML (UML Class Diagram) has been mostly utilized to specify the meta-model of the modelling language.

Table 1 Figure 1 Approach, standards and artifacts used to specify modelling languages in control samples

		Abstract Syntax			Semantics			Notation			Sample
		Graphical	Textual (formal)	Textual informal)	Formal Languages	Semi-formal Languages	textual (informal)	Graphical	textual (formal)	textual (informal)	
OASIS	Universal Business Language (UBL)	UML-Class Diagram	BNF	Natural Language			Natural Language		XSD	Natural Language	Sample Graphical Models
	The eBusiness eXtensible Markup Language (ebXML) Business Process Specification Schema (BPSS)		XSD	Natural Language		RFC-2119			XSD	Natural Language	Sample Textual Models
	Web Services Business Process Execution Language (WS-BPEL)	UML-Class Diagram	XSD	Natural Language		RFC-2119			XSD	Natural Language	Sample Graphical Models
OMG	Business Process Model and Notation (BPMN)	UML-Class Diagram	XSD	Natural Language		RFC-2119; WS-BPEL	Natural Language	Image	XSD	Natural Language	Sample Graphical Models
	Case Management Model and Notation (CMMN)	UML-Class Diagram	XSD	Natural Language		RFC-2119	Natural Language	Image		Natural Language	Sample Graphical Models
	Decision Model and Notation (DMN)	UML-Class Diagram	XSD; BNF	Natural Language			Natural Language	Image		Natural Language	Sample Graphical Models
Open Group	ArchiMate	UML-Class Diagram		Natural Language			Natural Language	Image		Natural Language	Sample Graphical Models
W3C	Web Ontology Language (OWL)	UML-Class Diagram	BNF	Natural Language	RDF	RFC-2119			XSD ;BNF; RDF	Natural Language	Sample Textual Models
	SPARQL Protocol and RDF Query Language (SPARQL)		BNF	Natural Language	Standard mathematical notation		Natural Language		BNF;RFC-3986; RFC-3987	Natural Language	Sample Textual Models
	Web Services Description Language (WSDL)		XSD; BNF		Z Notation	RFC-2119	Natural Language		XSD	Natural Language	Sample Textual Models
WFMC	XML Process Definition Language (XPDL)	UML-Class Diagram	XSD	Natural Language			Natural Language		XSD	Natural Language	Sample Textual Models

2.1. UML as a Modelling Method Design Approach

As UML is utilized for specification of most of modelling standards in our set, we extend our literature research to analyze UML as a prominent modelling method, which is used during create and design phases of the conceptualization process. The pragmatic objective of this analysis is to understand if UML - in the context of modelling method design - covers all requirements for creating and designing a modelling method, or if there are any shortcomings that we have to consider during the development of our approach.

Glinz evaluates UML in [56] as a requirement specification language. He states that definition of requirements with UML Use Case Diagrams is possible but Use-Case-Diagrams alone are not sufficient. Definitions of concrete functional and non-functional requirements as well as definition of relation between the concepts in domain specific modelling method and requirements are not possible. The authors of [58] indicate the same problem and propose again a UML-based solution to define non-functional requirements and relation between the components in system by combining constructs from UML Class Diagram and UML Component Diagram. Hence the first shortcoming needs to be considered in our approach is;

1. Definition of functional and non-functional requirements and their relation between the concepts in modelling methods.

According to Selic [59] the domain specific modelling languages specify different viewpoints of a complex system. Hence, the complex system should be represented from different viewpoints and some features will be presented in several viewpoints. Moreover the language, presumably, will support multiple viewpoints for different sub-domains, which means language should allow use of different abstract and concrete syntax. With UML Class Diagrams the whole meta-model (abstract syntax) of language can be fragmented. But depicting all alternatives of abstract syntax together can make the class diagram very complex and hard to read. To the best of our knowledge the UML does not offer definitions of different concrete syntax for same concept of modelling language. Hence the next shortcomings are

2. Fragmenting whole meta-model into individual meta-models composing concepts for different sub-domains and still be able to define link between concepts in different individual meta-models
3. Having another abstraction layer to represent modules and layers of modelling language as well as relation among them without representing complexity of abstract and concrete syntax
4. Assigning different concrete syntax to the concepts in modelling language.

According to authors of [55], nowadays, models are used to elaborate design decisions, sort out different concepts and exchange the ideas in mainstream software development. They state the importance of traceability during the transformation of information, communication of decisions etc. from design into implementation and vice versa. They indicate that in order to

establish the traceability the support of modelling environment is as essential as the approach itself. To the best of our knowledge UML itself does not support such traceability. Then the next shortcoming is:

5. Traceability of information (e.g design decisions, changes, suggestions etc.) during the knowledge exchange among experts within design phase and also from design to implementation vice versa.

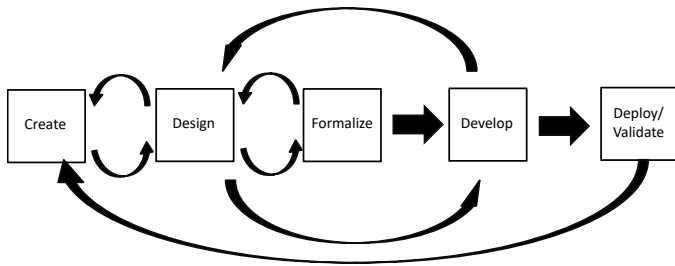
Karagiannis and Kühn in [5][57] argue that modelling methods have three major components (1) Modelling Language, (2) Modelling Procedure and (3) Mechanisms & Algorithms. Hence besides the modelling language of domain specific modelling method, we have to consider also designing modeling procedure and mechanisms & algorithms during the design of a modelling method. The design of modelling procedure and mechanisms & algorithms can be possible with using UML (e.g UML Activity diagram for design of the modelling procedure and if we consider description of mechanism and algorithms as sequence of object interactions and message exchange, the UML Sequence diagram can be used for description of mechanism and algorithms). Hence we would not see any shortcoming in UML for this issue. But in order to emphasize requirement of ability to design modelling procedure and mechanisms & algorithms besides the modelling language of a domain specific modelling method and requirement, and requirement of having corresponding supportive modelling environment, we would list this issue here rather as a general requirement than shortcoming of UML that we have to consider in our approach. Hence last but not least;

6. Possibility to design modelling procedure and mechanisms & algorithms of a domain specific modelling language.

2.2. Agile Modelling Method Engineering and Conceptualization Process

Having roots in software engineering, as it is in agile software development, during the modelling method engineering, involved stakeholders need procedures, structures and supportive tools allows high iterative process with as less as possible bureaucracy, and offers agile value and follows principles in Agile Manifesto [37]

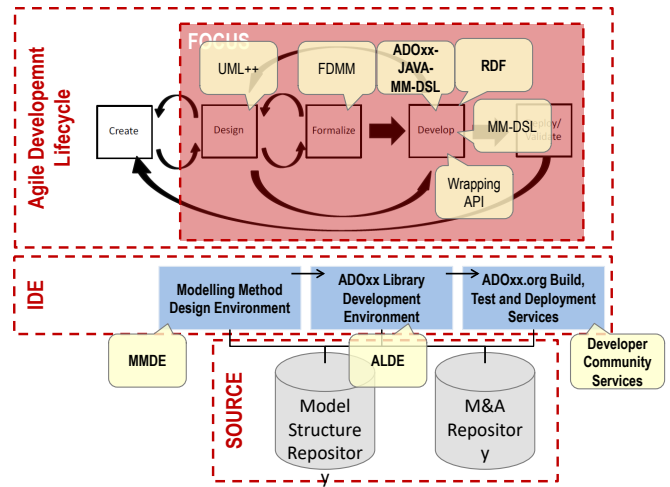
AMME is proposed in [6] to support modelling method engineering during propagation and evolution of modelling requirements. The OMiLab Lifecycle [10] instantiates AMME and defines the internal cycle of a modelling method conceptualization with five phases; (1) "Create" as a mix of goal definition, knowledge 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 [9] propose so-called Modelling Method Conceptualization Process (as depicted in Figure 1) by adding additional feedback channels into the 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.

3. Modelling Method Conceptualization Environment

The work at hand introduces the “Modelling Method Conceptualization Environment from ADOxx.org. A toolbox (as its high-level architecture depicted in Figure 2) that initially has been introduced in [35] and that instantiates the above-mentioned conceptualization process and supports method engineers during each phase. 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) shall be freely selected. Modelling Method Conceptualization Environment proposes a combination of tools in sense of Integrated Development Environment (IDE), such as the Modelling Method Design Environment (MMDE, available to download and install at [11]) for the Design, the ADOxx Library Development Environment (ALDE) for Formalize and Develop, ADOxx.org Build, Test and Deployment Services (available at [22]) for Deploy/Validate Phases.



As depicted in Figure 3, typical application scenario would be; during the create phase domain experts and method engineers come together, define goal of modelling method, acquire and elicit requirements, in design phase method engineers with tight collaboration of domain experts specifies the meta-model, language grammar, notation and processing functions on MMDE, as method engineers formalize design of modelling method collaboratively and commit on ALDE, developer(s) based on that formalization implements concrete modelling toolkit prototype within ALDE and ADOxx Development Toolkit. Developer(s) uploads the prototype into ADOxx.org build server, semi-automatic service behind starts with completeness check, building installation package, testing of installation package and optionally deploy it on selected developer’s space to allow to download the toolkit, to be tested and validated by community members, so to get feedback from them or the build services simply sends a link to corresponding owner to download and/or share the modelling toolkit.

It is worth to mention that one of the objectives is to provide loosely coupled tools, so involved actors 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

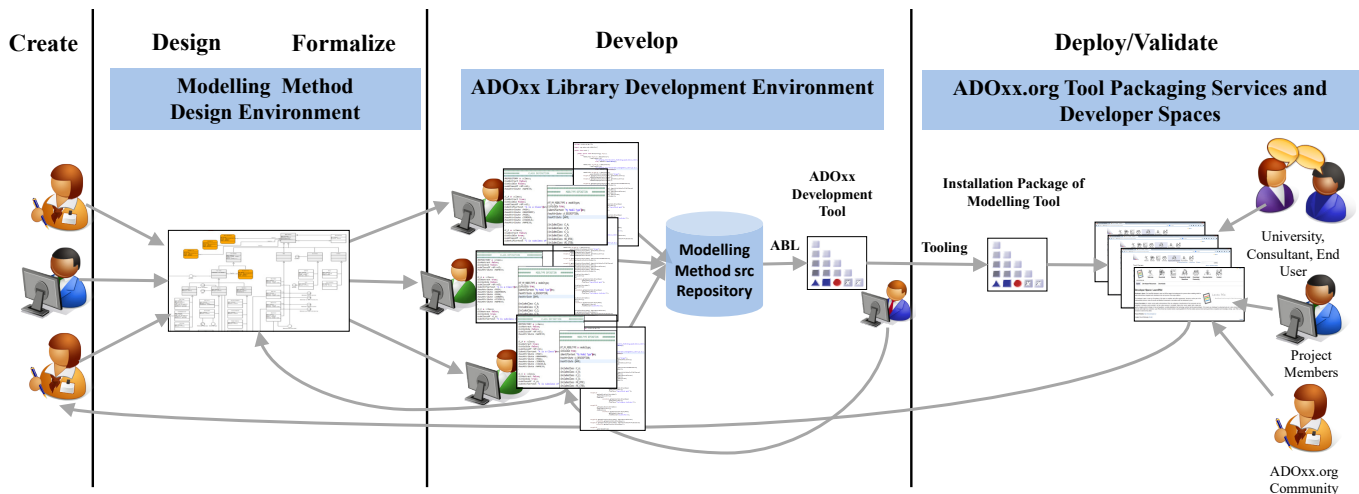


Figure 3 Life-cycle within Modelling Method Conceptualization Environment from Users’ Perspective

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.

3.1. 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 from results of the state of the analysis, which is discussed in previous chapter, and from the experience of the authors, who have been involved in more than 20 modelling method/tool development projects for varying domains. Based on these lessons learned, UML [12] has been identified as a 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 after the state of the art analysis: **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 (as depicted in Figure 4) 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.

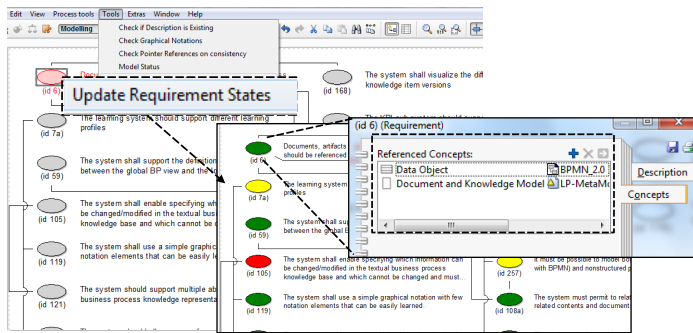


Figure 4. Example: Requirement Model before and after Updating the Status of Requirements

For **Ch2** and **Ch3**, we extend the class diagram from UML (as depicted in Figure 5) 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].

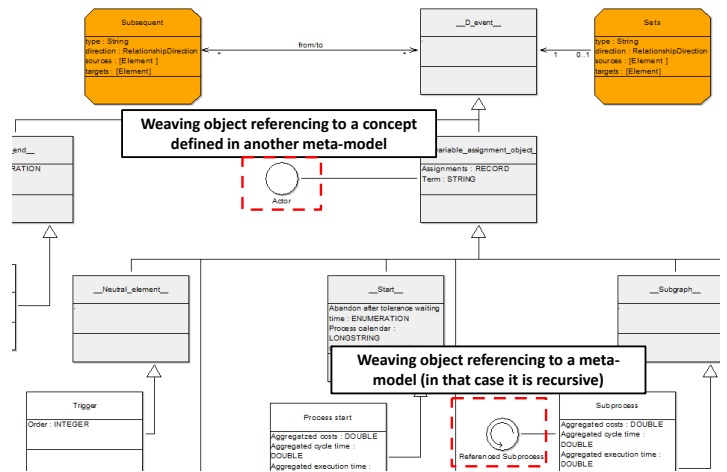


Figure 5 A sample meta-model showing usage of weaving concept

The modularization and layering of modelling language is essential to avoid complexities during the design of domain specific modelling methods [15][16] Hence, we propose representation of the Modelling Stack as the “Meta-models Stack model type (as depicted in Figure 6) allowing method engineers to differentiate meta-models in sense of different model types that target different fragments of the system.

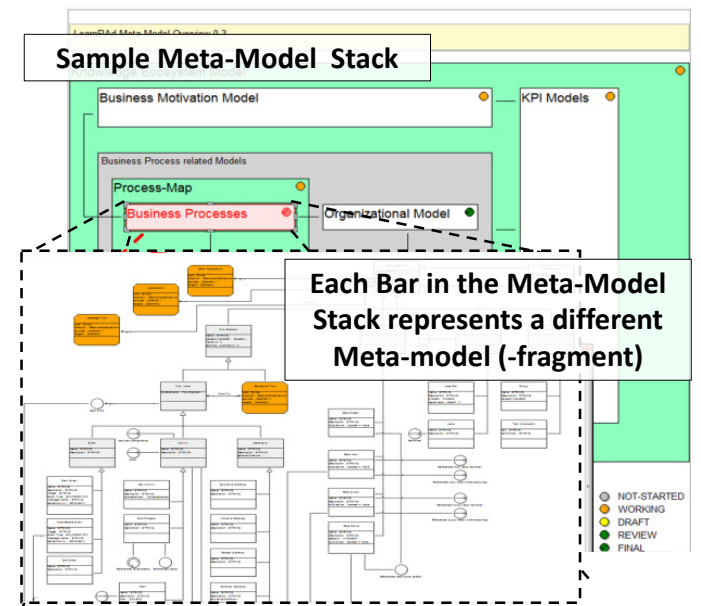


Figure 6 A Sample Meta-model Stack Model

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 (as depicted in

Figure 7) 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 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)




End Event		As the name implies, the End Event indicates where a Process or Choreography will end.
Activity		An Activity is a generic term for work that company performs in a Process. An Activity can be atomic or non-atomic (compound). The types of Activities that are a part of a Process Model are: Sub-Process and Task, which are rounded rectangles. Activities are used in both standard Processes and in Choreographies.
Gateway		A Gateway is used to control the divergence and convergence of Sequence Flows in a Process and in a Choreography. Thus, it will determine branching, forking, merging, and joining of paths. Internal markers will indicate the type of behavior control.

Figure 7 Sample Graphical Notation Model

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 (as depicted in

Figure 8) allows method engineers to define the steps with their required inputs and produced outputs, as well as the sequence of steps based on the input – output relationships, in order to introduce case specific proper usage of their modelling method.

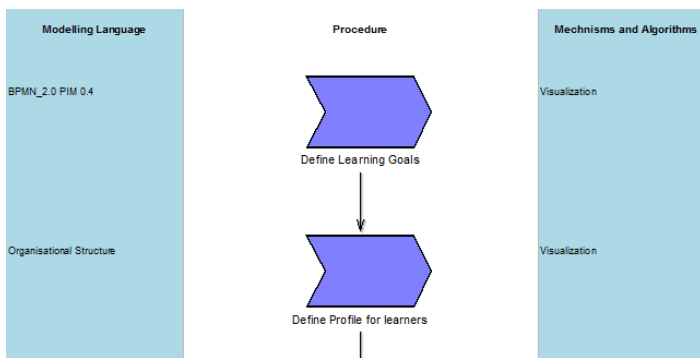


Figure 8 A Sample Modelling Procedure Model

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 [9]

3.2. ADOxx Library Development Environment

The ADOxx Library Development Environment (ALDE) aims to enable 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 is uses the Resource Description Framework (RDF) as a format for data interchange [17]

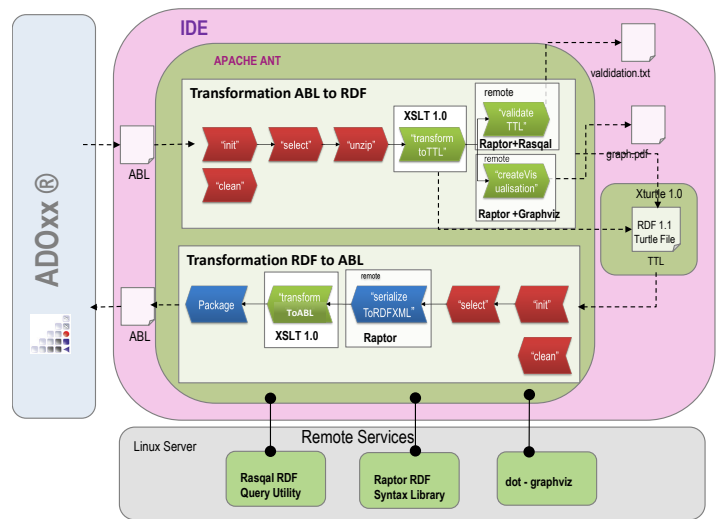


Figure 9 Architecture of ADOxx Library Development Environment (ALDE)

The Figure 9 depicts the architecture of ALDE. It is a development environment based on the Eclipse IDE [18] and includes a meta-meta-model defined in RDFS, the ALDE vocabulary. Having the vocabulary and utilizing Apache Ant® [19] as a build mechanism, 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 [20] has been used (the Figure 10 depicts code snippets produced by the environment) and includes the RDF XTurtle Editor developed by [21]. 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. On the other hand, ALDE includes verification services to ensure that the newly developed, edited or merged libraries are consistent with ADOxx platform requirements.

```

##### 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 #####
:MT_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_CTOD;
    
```

Figure 10 Snippet of Class and Model Type Definitions in ALDE in RDF

The new extension to ALDE is a new DSL based on Java, which has a working title of “ADOxx-JAVA-MM-DSL”. The ADOxx-JAVA-MM-DSL is developed according to feedback on

previous version of toolbox, which are presented and discussed in [35].

The ADOxx-JAVA-MM-DSL is a framework that creates several abstraction layers over the ADOxx Library Language (ALL) format, the ADOxx internal language that describes a meta-model [64]. Each layer simplifies and adds features to the bottom one. The framework gives, the possibility to operate and easily perform modification on a meta-model without dealing with its complexity. In order to assure that, an internal structure is managed that represents the ALL structure. This internal structure can be imported from an existing ALL meta-model.

All the constraints and rules present in the ALL syntax are managed, so the framework can guarantee that only syntactically valid ALL conform meta-models can be loaded and generated. The whole internal structure is an instance of java classes, so operations and definitions on the meta-model's concepts can be done using the java features. Parallel to that, additional utilities are integrated in order to bridge the gap between the development phases:

- The compilation to the ALL meta-model to its binary format ABL: The ALL is a text based language used to describe a meta-model, in order to be imported and be usable in ADOxx it need to be compiled in its binary format ABL. A conversion engine is integrated into the framework. In such a way it is possible to automatize a previously manual step, required in order to create an automatized flow between formalization and development phase.
- The importing and deployment of the compiled meta-model in the form of ABL file, into a ADOxx Database: In order to create the respective modelling environment based on the created meta-model, is required to import the ABL compiled meta-model into the Database used by ADOxx. After this step the modelling environment relative to that meta-model can be executed and opened simply specifying at the launch time, the newly created database. The framework contains a feature that automatizes such a process, generating the Database and launching the modelling environment relative to the created meta-model. This step bridges the gap between formalization and development phase.

As depicted in Figure 11, the framework can be divided into three abstraction layers and two transversal layers of primitives and general features:

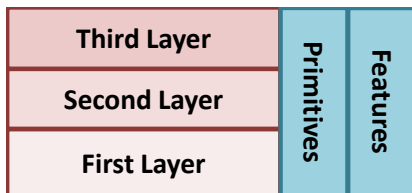


Figure 11 Layers of ADOxx-JAVA-MM-DSL

The first abstraction layer is a package of java classes that exactly reflect the structure of the ALL syntax. This layer is responsible for converting each class in its ALL representation and applies syntax rules. Due to its closeness to the ALL syntax, working directly with this layer is difficult. The advanced of creating a meta-model using this layer instead using directly the ALL syntax is that all possible errors derived from miss-spelling

and unallowed sequences will be avoided, but the complexity in the definition of the meta-model remain the same as the ALL.

The second abstraction layer abstracts all the concepts of the First layer to a more design friendly way, providing the possibility to work directly with the concepts of Libraries, Classes, Relations and Attributes. These concepts are more familiar to method engineers that are familiar with the interactive development approach directly in the ADOxx Development Toolkit. This abstraction layer defines also some semantic rules over the syntactic rules provided by the first layer. The entire concepts created at this level will be mapped to the relative concepts at the first layer in order to reflect the ALL syntax. At this level, certain helpful features are also implemented, such as the possibility to merge two or more libraries or to import one or several classes from a library to another.

The third abstraction layer is a factory layer and contains predefined methods that generate empty libraries as a starting point for the definition of specific meta-models with classes and relations. The main entry point of the framework is also at this layer with the possibility to explore all the features of the development environment.

The transversal primitives layer gives a formal representation of all the primitives allowed in the ALL syntax. The primitives are used at every level in order to define data such as Identifiers or Attribute Values;

The transversal features layer contains functions used at every level and the ALL parser. The parser is the component that allows an ALL to be read from a file and being instantiated and loaded into the framework in order to be extended or modified.

The ADOxx-JAVA-MM-DSL supports following three scenarios:

Definition of a meta-model from scratch: Using the third abstraction layer, it is possible to generate an empty ADOxx library that is the best starting point to create own specific meta-model. Starting from that object the method engineer can add Classes and Relations and their respective Attributes to the meta-model by writing pure java code. Once the java code is executed, it generates the ALL file and/or compile it in the ABL file and/or directly create the ADOxx Database and execute the prototype of newly developed modelling toolkit based on defined meta-model.

Extension or modification of an existing meta-model: Using the parsing module of the transversal layer, it is possible to load an ALL file into an instance of the java classes present in the second and First layer of the developer environment. After the ALL have been loaded is possible to extend, modify or manage it in the same way as creating from scratch. Certain methods to find specific Libraries, Classes, Relations and Attributes that the method engineer wants to work with are available. As in the previous scenario, after the java code is executed generates the ALL file and/or compile it in the ABL file and/or directly create the ADOxx Database and execute the prototype of newly developed modelling toolkit based on defined meta-model

Merging of two or more meta-model in one: This scenario is a particular case of the first two. In particular, with using the framework, it is possible to merge two or more meta-model. As it is in a sample snippet depicted in

Figure 12, thanks to the features present in the second layer is possible to import the whole concepts from a meta-model to

another or do a fine-grained import of specific concepts from each meta-model to another one. It is possible to import a class from a meta-model to another managing automatically all its dependencies like the presence of Super Classes. The merging scenario is supported in order to minimize the conflict that may rise in the merging of two incompatible meta-models, providing useful information to method engineer regarding how to correct occurred conflicts.

```

package caxman_merge_lib;
import org.adoxx.ado.ADOUtils;

public class Main {
    public static void main(String[] args) {
        try {
            ADOAllFile mainLib = ADOLibFactory
                .LoadFromAbLFile(
                    "D:\\merging_test\\Business_Modelling_Application_Li
                    true, ADOUtils.adonisAbLKey);

            ADOAllFile secondLib = ADOLibFactory
                .LoadFromAbLFile(
                    "D:\\merging_test\\Performance_Management_Library_0.
                    true, ADOUtils.adoxxAbLKey);

            ADOAllFile mergedLib = ADOLibFactory
                .generateADOxxTotalEmptyLibrary("Merged");
            mergedLib.getApplicationLibraryNew().getDynamicLibrary()
                .addADOxxEmptyDefaultClasses();
            mergedLib.getApplicationLibraryNew().getStaticLibrary()
                .addADOxxEmptyDefaultClasses();

            mergedLib.getApplicationLibraryNew().importClassesAndRelations(
                mainLib.getApplicationLibraryNew());

            mergedLib.getApplicationLibraryNew().importAttributeProfileClasses(
                mainLib.getApplicationLibraryNew());
            mergedLib.getApplicationLibraryNew().importRecordClasses(
                mainLib.getApplicationLibraryNew());
            mergedLib.getApplicationLibraryNew().importFiles(
                mainLib.getApplicationLibraryNew());
            mergedLib
                .getApplicationLibraryNew()
                .getDynamicLibrary()
                .addClass(
                    secondLib.getApplicationLibraryNew()
                        .getDynamicLibrary()
                        .findClass("Operational goal"));

            mergedLib
                .getApplicationLibraryNew()
                .getDynamicLibrary()
                .addClass(
                    secondLib.getApplicationLibraryNew()
                        .getDynamicLibrary()
                        .findClass("Performance indicator"));

            mergedLib
                .getApplicationLibraryNew()
                .getDynamicLibrary()
                .addRelation(
                    secondLib.getApplicationLibraryNew()
                        .getDynamicLibrary()
                        .findRelation("quantifies"));

            mergedLib.getApplicationLibraryNew().importRecordClasses(
                secondLib.getApplicationLibraryNew());
            mergedLib.getApplicationLibraryNew().importAttributeProfileClasses(
                secondLib.getApplicationLibraryNew());

            mergedLib
                .generateABLFile(

```

Figure 12 Snippet of a Selective Merge of two Meta-models

3.3. Adoxx.org Build, Test and Deployment Services.

Adoxx.org Build, Test and Deployment Services [38] are web-based services that allow 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 artefacts and signs the outcomes and creates the actual installer as a download archive.

www.astesj.com

As a collaboration space for the development and deployment phases, the concept of “Developer Spaces” has been introduced in ADOxx.org [23]. 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.

4. Modelling Method Conceptualization Services

In addition to the development tools described in the previous chapter, an appropriate service support is foreseen to support the modelling method engineers. The services are provided on the ADOxx.org portal, supporting a community of more than 1.300 modelling method engineers world-wide.

- 1. Download:** For the download, ADOxx.org provides the Meta Modelling Platform ADOxx in combination, Installation Instructions, Frequently Asked Questions, Startup-Package as well as a set of more than 30 available application libraries, which can be used to start with.
- 2. Get Started:** For getting started, ADOxx.org provides important readings, provides a Forum that is structured according active communities, lists tutorial and training events that are offered free of charge, provides tutorial material for both the students – in form of guide samples and slides – as well as for the trainer – in form of a trainer handbook and offers tutorial videos and webinars.
- 3. Development and Support:** For the development, ADOxx.org provides aforementioned tools and additional developer utilities, 3rd parties add on like but not limited to simulation, documentation, dashboards or Web-APIs. A collection of 200 graphical representations that introduce the major elements conclude the development support.
- 4. Community:** For collaborative development within the ADOxx.org community a map is provided indicating the ten laboratories – nine in Europe, one in Asia, indicating the hot spots of developers, the participating research institutes, a set of 24 modelling tools as a result of [66], and development spaces that enable a collaborative development and enable the use from solutions and tools from other projects.
- 5. Documentation:** A complete specification and documentation is offered, where each relevant element of the modelling method is (a) explained based on the corresponding theory, (b) introduced with hand-on samples, (c) demonstrated with real-world scenarios, (d) mapped to forum entries of the community and finally (e) supported with tools where possible.

The operation of this service centre is provided via the portal, social media like Twitter, Facebook and LinkedIn, or via email. In justified cases an onsite support is possible, where either the method engineer is trained, supported or critical implementation steps are performed by the ADOxx.org service centre.

5. Evaluation

Given that usually each modelling method engineering case differs from each other in sense of complexity of domain, variety of aspects to be targeted, number of involved actors, to calculate quantified evaluation means is difficult, and – to best of our knowledge - there is no similar conceptualization environment, hence, it is difficult to bench-mark our proposal and quantify the

evaluation and provide statistically objective results. On the other hand, the most important tangible and objective evaluation result would be deployed and ready to use modelling toolkits, specification of modelling methods and communication of community members as proofs of concept. Those proofs of concepts for each are online and freely accessible (with exception of in-house project case). The links to access those proofs of concepts for each case are provided under regarding sub-section below.

The conceptualization environment introduced above has been applied in four different cases for evaluation: three EU-funded research projects in the domain of multi-stage manufacturing, 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 given domains, in development and in modelling method engineering. In the following sub-section we introduce the cases and their requirements in method engineering manner.

Case 1: Conceptualization of a Modelling Method for E-Learning: The FP7 project Learn PAd [24] 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 [25], as well as the developed prototypes [26] can be downloaded and feedback can be given.

Case 2: Conceptualization of Modelling Method for Cloud Computing: The H2020 project CloudSocket [27] 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 [28]. 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 [29], as well as developed prototypes [30] can be downloaded and feedback can be given.

Case 3: Conceptualization of Modelling Method for holistic Manufacturing System Management:

The H2020 project DISRUPT [60] deals with the integration of innovative technologies into a holistic manufacturing system and optimization of production flow. The DISRUPT projects needs a modelling method to describe manufacturing system from supply-chain level down to shop-floor level. In this case 2 domain experts, one requirement engineer and one method engineer have been involved. Preliminary results can be found on DISRUPT Developers Space [61].

Case 4: Integration of existing BPMN and DMN Modelling Methods: The in-house project requires integration of an already implemented DMN Modelling Method into existing BPMN 2.0 realization as part of a commercial product. In this case, 3 domain experts and method engineers, and a team of two developers have been involved.

www.astesj.com

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.

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). Hence, double effort in the design and in the formalisation and or development is currently necessary.

Outlook: The MMDE is currently updated, to offer an XML export, which then can be transformed into different formats like the one that is used for the ADOxx-Java-MM-DSL.

In addition, several improvements on the modelling language are implemented to (a) enable the design of user scenarios, (b) better describe the features of the modelling method and their corresponding components as well as (c) enable a more detailed representation of the method procedure enabling the mapping from components and the corresponding elements of the modelling method.

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.; it re-quires different transformation scripts for different meta-modelling technologies (such as ADOxx, EMF).

Outlook: The semantic-based verification of meta model is seen as a useful extension of the ADOxx-Java-MM-DSL, hence an integration will be experimented. However, we see the necessary skill level for the meta model developer currently as inappropriate and tend not to follow this path.

Feedback on ADOxx-Java-MM-DSL:

Pro: It is possible to merge libraries and start libraries from scratch. Furthermore, the code base can be stored and versioned in a versioning system enabling several developers in parallel to work on one library. Built scripts enable the automatic generation and deployment of the tool.

Contra: The current code maturity needs improvement and documentation, enabling also non specialists to handle the tool.

Outlook: This tool will be further improved and tested in two EU-H2020 research projects and will consequently be taught at the

ADOxx.org Training Days and Webinars to achieve the required maturity.

Feedback on ADOxx.org Tool Packing Services and Developer Spaces.

Pro: It is possible to have an installation package to distribute to interested stake-holders, building your own community around the modelling method, and get feed-back 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.

Outlook: This tool packaging service will be stepwise opened, so that the developer can also include own software components, which are then composed into a single tool package.

6. 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 four different cases: three 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 three main tools, MMDE, ALDE and ADOxx-Java-MM-DSL, prototypes that are at about Technology Readiness Level 5, hence 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 as future work: (1) currently we are evaluating development alternatives of DSLs with using Xtend [62] or RDF; building on existing work [63] in the field and integrating it into ADOxx-Java-MM-DSL, (2) enabling graphical modelling method design to transform into machine understandable format, (3) formalization of modelling method design using mathematical models such as FDMM [33] or Proof of Concept prototyping, (4) automatization of tooling services and deployment onto developer spaces, (5) full integration of tools within a holistic development environment.

Conflict of Interest

The authors declare no conflict of interest.

Acknowledgment

This work has been partly supported by European Union's Horizon 2020 research and innovation programme within the projects DISRUPT(www.disrupt-project.eu) under grant agreement No: 723541 and CloudSocket (www.cloudsocket.eu) under grand agreement no: 644690.

References

- [1] N. Efendioglu, R., Woitsch, "Modelling Method Design: An ADOxx Realisation" in 20th IEEE International Enterprise Distributed Object Computing Workshop, EDOC Workshops 2016, Vienna, Austria, 2016
- [2] Object Management Group (OMG), "Business Model and Notation Version 2.0," 2011. [Online]. Available: <http://www.omg.org/spec/BPMN/2.0/>. [Accessed 15 July 2016].

- [3] Object Management Group (OMG), "Decision Model and Notation, Version 1.0," 2015. [Online]. Available: <http://www.omg.org/spec/DMN/1.0/>. [Accessed 15 July 2016].
- [4] Object Management Group (OMG), "Case Management Model and Notation", [Online] Available: <http://www.omg.org/spec/CMMN> [Accessed 22 February 2017]
- [5] D. Karagiannis and H. Kühn, "Metamodelling platforms," in In Proceedings of the 3rd International Conference EC-Web 2002, Dexa 2002, France, Springer-Verlag, 2002, p. 182.
- [6] D. Karagiannis, "Agile Modeling Method Engineering," in Proceedings of the 19th Panhellenic Conference on Informatics, Athens, Greece, ACM, 2015, pp. 5-10.
- [7] V. Hrgovcic, D. Karagiannis and R. Woitsch, "Conceptual Modeling of the Organizational Aspects for Distributed Applications: The Semantic Lifting Approach," in COMPSACW, IEEE, 2013.
- [8] J. Michael, F. Al Machot, M. C. Heinrich "ADOxx Based Tool Support for a Behavior Centered Modeling Approach" in Proceedings of the 8th ACM International Conference on Pervasive Technologies Related, PETRA'15, Corfu, Greece, 2015
- [9] N. Efendioglu, R. Woitsch and D. Karagiannis, "Modelling Method Design: A Model-Drive Approach," in IIWAS '15: Proceedings of the 17th International Conference on Information Integration and Web-based Applications, Brussels, Belgium, ACM, 2015.
- [10] Open Models Laboratory (OMILab), "Idea and Objectives," 2015. [Online]. Available: <http://austria.omilab.org/psm/about>. [Accessed 15 July 2016].
- [11] ADOxx.org, "LearnPAD Developer Space," 2015. [Online]. Available: <https://www.adoxx.org/live/web/learnpad-developer-space/design-environment>. [Accessed 07 July 2016].
- [12] Object Management Group (OMG), "Documents Associated With UML Version 2.0," 2005. [Online]. Available: <http://www.omg.org/spec/UML/2.0/>. [Accessed 12 July 2015].
- [13] H. Kühn, "Methodenintegration im Business Engineering, PhD Thesis (in German)," University of Vienna, 2004.
- [14] R. Woitsch, "Hybrid Modeling: An Instrument for Conceptual Interoperability," in Revolutionizing Enterprise Interoperability through Scientific Foundations, Hershey, 2014, pp. 97-118.
- [15] B. Selic, "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, Springer Berlin Heidelberg, 2011, pp. 290-321.
- [16] D. Karagiannis, V. Hrgovcic and R. Woitsch, "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, 2011, pp. 451-454.
- [17] W3C, "RDF-Resource Description Framework," 2014. [Online]. Available: <https://www.w3.org/RDF/>. [Accessed 14 July 2016].
- [18] Eclipse Foundation, "Eclipse IDE for Java EE Developers," 2016. [Online]. Available: <http://www.eclipse.org/downloads/packages/>. [Accessed 14 July 2016].
- [19] The Apache Software Foundation, "Apache Ant Download," 2016. [Online]. Available: <https://www.apache.org/dist/ant/binaries/>. [Accessed 14 July 2016].
- [20] W3C, "RDF 1.1 Turtle Terse RDF Triple Language," 2014. [Online]. Available: <https://www.w3.org/TR/2014/REC-turtle-20140225/>. [Accessed 14 July 2016].
- [21] The Research Group Agile Knowledge Engineering and Semantic Web (AKSW), University of Leipzig, "Xturtle," 2015. [Online]. Available: <http://aksw.org/Projects/Xturtle.html>. [Accessed 15 July 2016].
- [22] ADOxx.org, "AutoPDP Tool Packaging Service," 2016. [Online]. Available: <https://www.adoxx.org/live/autopdp-packaging-service>. [Accessed 03 March 2017].
- [23] ADOxx.org, "ADOxx.org Developer Spaces," 2016. [Online]. Available: <https://www.adoxx.org/live/development-spaces>. Accessed 03 March 2017].

- [24] Learn PAd Consortium, "The EU Project Learn PAd," 2014. [Online]. Available: <http://www.learnpad.eu/>. [Accessed 03 March 2017].
- [25] LearnPAD Consortium, "LearnPAD Developer Space," 2015. [Online]. Available: <https://www.adoxx.org/live/web/learnpad-developer-space/>. [Accessed 03 March 2017].
- [26] LearnPAD Consortium, "LearnPAD Developer Space - Downloads," 2015. [Online]. Available: <https://www.adoxx.org/live/web/learnpad-developer-space/downloads/>. [Accessed 03 March 2017].
- [27] CloudSocket Consortium, "CloudSocket Project," 2016. [Online]. Available: <https://www.cloudsocket.eu/>. [Accessed 03 March 2017].
- [28] R. Woitsch and W. Utz, "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.
- [29] CloudSocket Consortium, "CloudSocket Developer Space," 2015. [Online]. Available: <https://www.adoxx.org/live/web/cloudsocket-developer-space/>. [Accessed 15 July 2016].
- [30] CloudSocket Consortium, "CloudSocket Developer Space - Downloads," 2015. [Online]. Available: <https://www.adoxx.org/live/web/cloudsocket-developer-space/downloads/>. [Accessed 03 March 2017].
- [31] Eclipse Foundation, "Xtend," 2015. [Online]. Available: <https://eclipse.org/xtend/>. [Accessed 15 July 2016].
- [32] N. Visic and D. Karagiannis, "Developing Conceptual Modeling Tools Using a DSL," in Knowledge Science, Engineering and Management, Sibiu, Romania, Springer, 2014, pp. 162-173.
- [33] H.-G. Fill, T. Redmond and D. Karagiannis, "FDMM: A Formalism for Describing ADOxx Meta Models and Models," in Proceedings of ICEIS 2012, Wroclaw, Poland, Vol. 3, Wroclaw, 2012, pp. 133-144.
- [34] ADOxx.org "GraphRep" 2016 [Online]. Available <https://www.adoxx.org/live/graphrep> [Accessed 31.08.2016]
- [35] N. Efendioglu,, R. Woitsch, W. Utz, "A Toolbox Supporting Agile Modelling Method Engineering: ADOxx.org Modelling Method Conceptualization Environment". In J. Horkoff, M. A. Jeusfeld, & A. Persson, The Practice of Enterprise Modeling (pp. 317-325), 9th IFIP WG 8.1. Working Conference, PoEM 2016, Skövde, Sweden, November 8-10, 2016, Proceedings, Springer
- [36] Principles behind the Agile Manifesto [Online]. Available <http://agilemanifesto.org/iso/en/principles.html> [Accessed 24 January.2017]
- [37] Cees Van Halen; Carlo Vezzoli; Robert Wimmer (2005). Methodology for Product Service System Innovation. Assen: Uitgeverij Van Gorcum. p. 21. ISBN 90-232-4143-6.
- [38] ADOxx.org, "Developer Community," 2017. [Online]. Available: <https://www.adoxx.org/live/community/>. [Accessed 23.January.2017].
- [39] OASIS, "Universal Business Language Version 2.1". 04.November 2013, OASIS Standard [Online] Available at <http://docs.oasis-open.org/ubl/os-UBL-2.1/UBL-2.1.html> [Accessed 13 July 2015]
- [40] OASISI, "ebXML Business Process Specification Schema Technical Specification v2.0.4" 21 December 2006, OASIS Standard Available at <http://docs.oasis-open.org/ebxml-bp/2.0.4/OS/spec/ebxmlbp-v2.0.4-Spec-os-en-html/ebxmlbp-v2.0.4-Spec-os-en.htm> [Accessed 13.July 2015]
- [41] OMG, 2005. Documents Associated With UML Version 2.0. [Online] Available at: <http://www.omg.org/spec/UML/2.0/> [Accessed 12 July 2015]
- [42] The Open Group, 2013. ArchiMate® 2.1 Specification, [Online] Available at <http://pubs.opengroup.org/architecture/archimate2-doc/toc.html> [Accessed 13 July 2015]
- [43] W3C, 2007, Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language [Online] Available at: <http://www.w3.org/TR/wsdl20/wsdl20.pdf> [Accessed 13.July 2015]
- [44] W3C, 2012 OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax (Second Edition) [Online]. Available at <http://www.w3.org/TR/2012/REC-owl2-syntax-20121211/> [Accessed 13 July 2015]
- [45] W3C, 2012, SPARQL 1.1 Query Language [Online] Available at: <http://www.w3.org/TR/owl2-direct-semantics/> [Accessed 13.July 2015]
- [46] Web Services Business Process Execution Language Version 2.0 11 April 2007, OASIS Standard <http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html> [Accessed 13 July 2015]
- [47] WfMC, 2005, Process Definition Interface – XML Process Definition Language [Online] Available at: [http://www.xpdl.org/standards/xpdl-2.2/XPDL%202.2%20\(2012-08-30\).pdf](http://www.xpdl.org/standards/xpdl-2.2/XPDL%202.2%20(2012-08-30).pdf) [Accessed 13. July 2015]
- [48] ISO/IEC 14977:1996, Information technology -- Syntactic metalanguage -- Extended BNF, International Organization for Standardization, 1996 [http://standards.iso.org/ittf/PubliclyAvailableStandards/s026153_ISO_IEC_14977_1996\(E\).zip](http://standards.iso.org/ittf/PubliclyAvailableStandards/s026153_ISO_IEC_14977_1996(E).zip)
- [49] W3C, 2012 W3C XML Schema Definition Language (XSD) 1.1 Part 1: Structures [Online] Available at: <http://www.w3.org/TR/xmlschema11-1/> [Accessed 13.July 2015]
- [50] 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]
- [51] Spivey, J. M., 1992, The Z Notation: A Reference Manual, Second Edition, Prentice Hall.
- [52] Internet Engineering Task IETF, 1997, Key words for use in RFCs to Indicate Requirement Levels, [Online] Available at <http://www.ietf.org/rfc/rfc2119.txt>. [Accessed 20 July 2015]
- [53] OASIS, "Web Services Business Process Execution Language Version 2.0 11" April 2007, OASIS Standard [Online] Available at <http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html> [Accessed 13 July 2015]
- [54] ISO/IEC, 2002 Z formal specification notation –Syntax, type system and semantics First Edition ISO/IEC 13568
- [55] Aagedal, J. Ø., Bézivin, J. & Lington, P. F. 2004. Model-Driven Development (WMDD 2004) in Object-Oriented Technology: (ECOOP) 2004 Workshop Reader, (ECOOP) 2004 Workshops, Oslo, Norway, June 14-18, 2004, Final Reports pp. 148-157
- [56] 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
- [57] H. Kühn, "Methodenintegration im Business Engineering"[in German] PhD Thesis, University of Vienna, April 2004
- [58] 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
- [59] 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.
- [60] DISRUPT Consortium, "Project Overview", 2017 [Online] Available at <http://disrupt-project.eu/about/overview> [Accessed 24 January 2017]
- [61] DISRUPT Consortium, "DISRUPT Developers Space" [Online] Available at <https://www.adoxx.org/live/web/disrupt/> [Accessed 24 January 2017]
- [62] Eclipse Foundation, "Xtend," 2015. [Online]. Available at <https://eclipse.org/xtend/>. [Accessed 15 July 2016].
- [63] N. Visic and D. Karagiannis, "Developing Conceptual Modeling Tools Using a DSL," in Knowledge Science, Engineering and Management, Sibiu, Romania, Springer, 2014, pp. 162-173
- [64] ADOxx.org ADOxx Library Language (ALL), [Online] Available at: <https://www.adoxx.org/live/library-language-all/abl/>, [Accessed 03 March 2017]
- [65] GOOD MAN Consortium,, The project "GOOD MAN: Agent Oriented Zero Defect Multi-Stage Manufacturing" [Online] Available at: <http://goodman-project.eu/> [Accessed 03 March 2017]
- [66] D. Karagiannis, H. C. Mayr, J. Mylopoulos, Domain-Specific Conceptual Modelling, Springer International Publishing, 2016
- [67] D. Karagiannis, H. C. Mayr, J. Mylopoulos, Domain-Specific Conceptual Modelling, Springer International Publishing, 2016