THE ROLE OF PIVOT METAMODEL IN AUTOMATIC GENERATION OF METAMODEL EXTENSIONS

Igor Zečević, Branko Perišić, Petar Bjeljac, Danijel Venus
Faculty of Technical Sciences, University of Novi Sad, Novi Sad, Serbia
{igor.zecevic, perisic, pbjeljac, danijelvenus}@uns.ac.rs

Abstract – **Until recently, the approaches of creating domain specific languages "from scratch" and extending existing languages represented adverse solutions that cannot be integrated in a unique model driven process. The main reason was the lack of a suitable formal theoretical framework that, based on the domain specific languages metamodel, enables the automatic generation of a final number of syntactically valid metamodel extensions. In this paper, the use of the Pivot metamodel concept is described. This metamodel enables the transformation of elements from one metamodel into extensions of another, regardless the compatibility level of their meta-metamodel. The rules and restrictions for transforming random models into the Pivot metamodel are also presented.**

I. INTRODUCTION

One of the main goals in the Model Driven Engineering (MDE) approach to software development [1] is the specification of Domain Specific Languages (DSLs) [2] which enable a semantically rich description of a designed software product at high abstraction level. Among many approaches to DSL development, two are dominant:

- adapting (extending) existing languages, that are commonly used in concrete problem domains [3] or
- creating completely new modeling languages, based on the characteristics of the application domain and the base rules for formal languages specification [4].

Until recently, in practice these two approaches were considered adverse and impossible to synchronize in a unique MDE process [5]. This standpoint was justified by the lack of a methodology that could provide a good enough level of interoperability for different approaches, methods, techniques and tools. It is possible to establish the synergic use of these two approaches through a formal theoretical framework that, based on the DSL metamodel, supports the automatic generation of a final number of syntactically valid metamodel extensions. The implementation of this formal theoretical framework, in the form of a workspace, enables:

- Interoperability of different meta-modeling tools based on the transformations established between their meta-models;
- Automatic model transformations, for models representing the instances of meta-models for which transformations have been defined;
- The integration of artifacts originating from models, which represent instances of meta-models for which transformations have been defined.

The specification of a formal theoretical framework requires the definition of rules that support the transformation of the DSL metamodel elements into extensions of existing modeling languages meta-models. Based on analysis of published research [6, 7, 8, 9, 10, 11, 12, 13], it is impossible to completely automate the transformation process between meta-models that are derived from different meta-meta-models. In general, there exist a large number of different combinations between meta-models derived from different meta-meta-models. This makes the individual transformations an exhausting activity that is hardly economically justifiable.

In this paper, an approach that raises the level of transformation automation based on a mediator (Pivot metamodel) is described. Using this approach, it is only necessary to establish a two-way transformation between a metamodel and its Pivot metamodel. In order to fulfill their role, in the formal theoretical framework context, the Pivot meta-models need to have the capacity to define any arbitrary metamodel that is the subject of transformation.

In this paper, the Pivot metamodel is described in the context of a formal theoretical framework for generating metamodel extensions. Furthermore, the rules and restrictions for transforming meta-models into the Pivot metamodel are presented.

The paper is organized as follows: in Section 2, the definition of models in a three-level model architecture and the concept of Technological spaces are presented. In Section 3, the theoretical framework for metamodel extension generation is described. The definition of the Pivot metamodel and the rules that guide the transformation of arbitrary metamodel into the Pivot metamodel are presented in Section 4. In section 5, based on the mappings established on meta-metamodel level, the conditions for generating the Pivot metamodel are described. In Section 6, an example of transforming a simple metamodel into the Pivot metamodel is presented. Section 7 concludes the paper and describes the directions of future work.

II. TECHNOLOGICAL SPACES AND METELEVEL BASED ARCHITECTURES

From an organization point of view a model can be defined as follows [2]:

Definition 2.1*:* A directed multigraph $G = (N_G, E_G, F_G)$ consists of a finite set of nodes N_G, final set of edges E_G and function $F_G : E_G \rightarrow N_G \times N_G$ which maps edges to their source and target nodes.

Definition 2.2: A model $M = (G, M_A, \mu)$ is an ordered triplet, where $G = (N_G, E_G, F_G)$ is a directed multigraph, M_A is a model (called the referential model), which is defined using a directed multigraph $G_A = (N_A, E_A, F_A)$, $\mu : N_G \cup E_G \rightarrow N_A$ is a function, which transforms elements (nodes and edges) of G multigraph of the model M into nodes of the G_A multigraph of the model M_A.

The relation between model M and referential model M_A is called conformance (conformsTo). The elements of metamodel M_A (nodes N_A and edges E_A) are called metaelements.

This definition of a model allows an unlimited number of referential model levels. By restricting the number of levels to three, meta-metamodel and metamodel may be defined as follows:

Definition 2.3: A meta-metamodel is a model that represents its own reference model (it conforms to itself).

Definition 2.4: A metamodel is a model such that its reference model is a metametamodel (it conforms to the meta-metamodel).

In a three-level model architecture, levels are usually marked as M1, M2 and M3. M1 level contains models which confirm to concepts of M2 level and are not metamodels. M2 level contains models which are not meta-metamodels and confirm to M3 concepts, while the M3 level is self-defined and contains meta-metamodels.

Technological spaces (TS) is a concept defined by Kurtev et al. in a discussion on problems during the integration of different technologies [14]. The technological spaces concept was initially defined as "a working context with a set of associated concepts, body of knowledge, tools, required skills, and possibilities". Certain technological spaces can be intuitively recognized. It is possible to isolate technological spaces such as executable programming languages TS (JAVA, C#), database management system TS or modeling framework TS (UML, MDA/OMG). In Fig. 2 the three-level model organization for two different TS is presented.

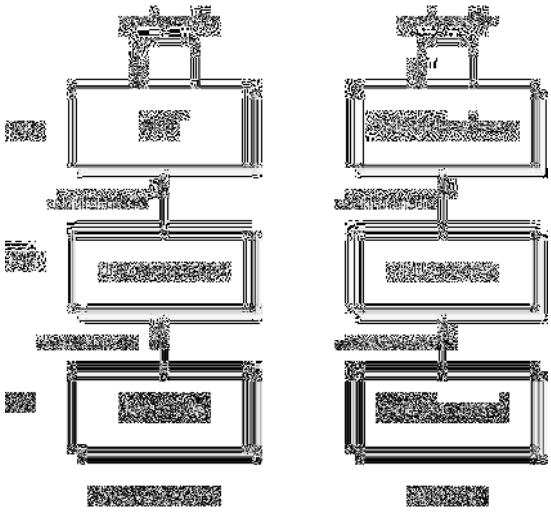

Figure 2. MDA and XML technological spaces [14]

III. FORMAL THEORETICAL FRAMEWORK FOR GENERATING METAMODEL EXTENSIONS

The aim of creating a formal theoretical framework for generating metamodel extensions is to enable a final number of syntactically valid metamodel extensions based on another metamodel. The metamodel from which these extensions are derived represents the source metamodel, and the metamodel subject to the extension represents the destination metamodel. It is assumed that the source and destination metamodel are both part of technological spaces that have a three-level meta-architecture (Fig. 1).

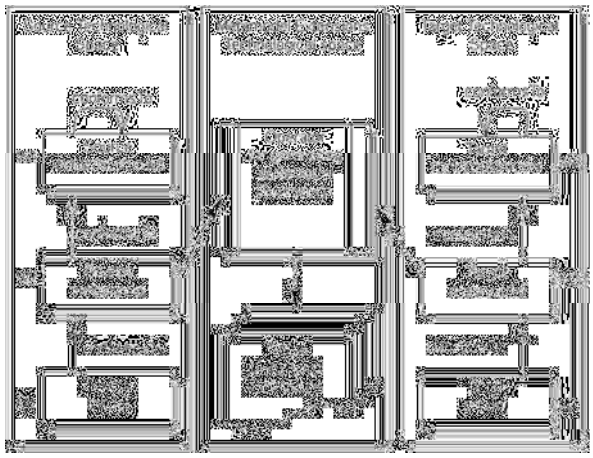

Figure 1. Formal theoretical framework for generating metamodel extensions

The formal theoretical framework should support the use of different TS-s, allow different TS-s for the source and destination metamodel, as well as to allow the source and destination metamodel to conform to different meta-metamodels.

A concrete implementation of the theoretical framework, containing these characteristics, supports interoperability between different MDE implementation strategies [4, 15, 16] (in the case of different source and destination TS), as well as generating metamodel extensions based on other metamodels (in the case of different source and destination meta-metamodels).

In this formal theoretical framework, the source and destination metamodels are presented using the KM3 metamodel definition language [17]. The purpose of KM3 is to give a relatively simple solution to define the Domain Definition MetaModel of a DSL. It is based on the MOF meta-metamodel [18] and the ECORE meta-metamodel [19]. The source metamodel in the KM3 format represents the source Pivot metamodel, whereas the destination metamodel in the KM3 format represents the destination Pivot metamodel. The Pivot metamodel can be any source or destination pivot metamodel. Each Pivot metamodel conforms to the KM3 meta-metamodel (Fig. 3).

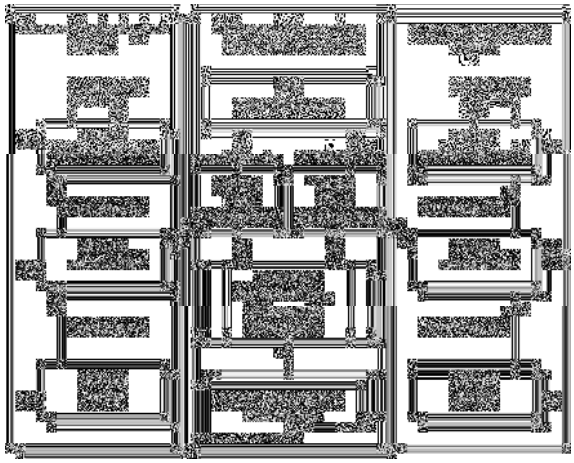

Figure 3. Pivot metamodels in the formal theoretical framework for generating metamodel extensions

IV. PIVOT METAMODELS

According to the definition, every KM3 metamodel is a metamodel. However, not every KM3 model is a Pivot metamodel. Only metamodels from the M2 level can be Pivot metamodels, and KM3 metamodels need to be additionally restricted in order to represent valid pivot metamodels.

Definition 4.1: Pivot metamodel is a KM3 model which represents a metamodel from the three-level metalevel hierarchy.

Based on the definitions of models in different meta levels (Def. 2.2), a metamodel in the three level metalevel hierarchy can be represented as an ordered triplet:

$$M_{MM} = (G_{MM}, M_{MMM}, \mu_{MM}) \quad (1)$$

where $G_{MM} = (N_{MM}, E_{MM}, F_{MM})$ is a directional multigraph containing elements from the M_{MM} metamodel (nodes N_{MM} and edges E_{MM}), $M_{MMM} = (G_{MMM}, M_{MMM}, \mu_{MMM})$ is the referential model for the M_{MM} metamodel (meta-metamodel), and μ_{MM} is a function which transforms elements (nodes and edges) from $\mathrm{G_{MM}}$ multigraph into nodes of $\mathrm{G_{MMM}}$ multigraph ("conformsTo" relation).

Pivot metamodel:

$$piv(M_{MM}) = (G_{PIV}, M_{KM3}, \mu_{KM3}, \omega_{PIV}) \quad (2)$$

is an ordered quadruplet, where $G_{PIV} = (N_{PIV}, E_{PIV}, F_{PIV})$ is a directional multigraph, which contains elements (nodes N_{PIV} and edges E_{PIV}), M_{KM3} is a KM3 metamodel, $\mu_{\mathrm{KM3}}: N_{PIV} \cup E_{PIV} \rightarrow N_{KM3}$ is a function which transforms elements of the $\mathrm{G_{PIV}}$ multigraph into nodes of $\mathrm{G_{KM3}}$ multigraph, and $\omega_{PIV} : G_{MM} \rightarrow G_{PIV}$ is a bijective function, which transforms elements from G_{MM} multigraph into elements of G_{PIV}, multigraph (Fig. 4).

In general, in order for a model to be considered a Pivot metamodel, two conditions need to be met. First, the model needs to conform to the KM3 metamodel. Second, in a three tier metalevel architecture, a metamodel needs to exist such that all elements from this

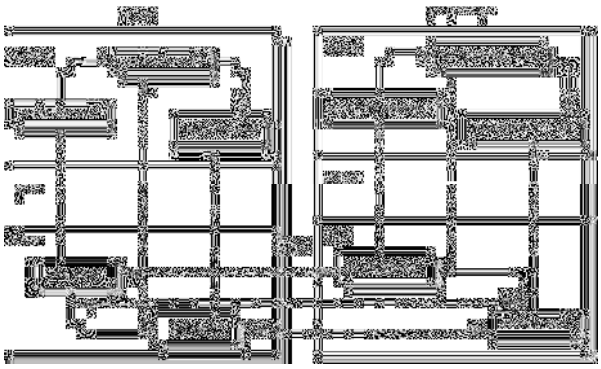

Figure 4. Pivot metamodel

metamodel have exactly one matching element in the Pivot metamodel and vice versa. The KM3 model presented in Fig. 7 represents a metamodel for the faculty organizational structure.

For M_{MM} metamodel, with a total of $|\mathrm{G_{MM}}|$ elements, there are at most

$$|piv(M_{MM})| = |\omega_{PIV}| * |\mu_{\mathrm{KM3}}| \quad (3)$$

pivot metamodels. Function ω_{PIV} transforms each element from the M_{MM} metamodel into exactly one element of the pivot metamodel, and there is a total of $|\omega_{PIV}| = |\mathrm{G_{MM}}|!$ different bijective functions, whereas each of the pivot metamodels contains a total of $|G_{PIV}| = |G_{MM}|$ elements. On the other hand, function μ_{KM3} defines an element of the KM3 meta-metamodel to which each element of the pivot metamodel conforms.

For a pivot metamodel with a total of $|G_{PIV}|$ elements and $|N_{KM3}|$ nodes in the KM3 metamodel, there are at most

$$|\mu_{\mathrm{KM3}}| = |N_{KM3}|^{|G_{PIV}|} \quad (4)$$

different functions μ_{KM3}. Finally for M_{MM} metamodel with $|\mathrm{G_{MM}}|$ elements there are at most:

$$|piv(M_{MM})| = |\mathrm{G_{MM}}|! * |N_{\mathrm{KM3}}|^{|G_{MM}|} \quad (5)$$

different pivot metamodels. Depending on the abstract and concrete M_{MM}, metamodel syntax, the total number of possible pivot metamodels can be smaller. Based on (5), the maximum number of different pivot metamodels for a certain metamodel has an exponential rise in relation to the number of metamodel elements $|G_{MM}|$.

Definition 4.2: The source pivot metamodel is the pivot metamodel of the source metamodel.

Definition 4.3: The destination pivot metamodel is the pivot metamodel of the destination metamodel.

In order to be able to transform a metamodel into a KM3 model, the model needs to have mappings from each element of the metamodel to an element in the KM3 metamodel, which conforms to the KM3 meta-metamodel. Based on these mappings, it is possible to define a transformation of a metamodel into a KM3 model. This means that a new transformation needs to be defined each time a new source or destination metamodel,

which needs to be represented in the form of a pivot metamodel, is introduced.

V. USING M3B APPROACH IN GENERATING PIVOT METAMODELS

Using the M3B approach [20,21] it is possible to reach a certain level of automation in transforming metamodels into KM3 models. Using this approach, it is possible to automatically generate model transformations on lower metalevels, based on established mappings between elements of models on the highest metalevel.

By establishing the mapping between meta-metamodels to which the source or destination metamodel conform, and the KM3 meta-metamodel, each metamodel which conforms to this metamodel can be transformed into KM3 metamodel (Fig. 6). In order to make mappings between the meta-metamodel and the KM3 meta-metamodel, it is required that each element in the meta-metamodel, has a matching element in the KM3 meta-metamodel. This means that when a new source or destination meta-metamodel is introduced, a new transformation needs to be defined.

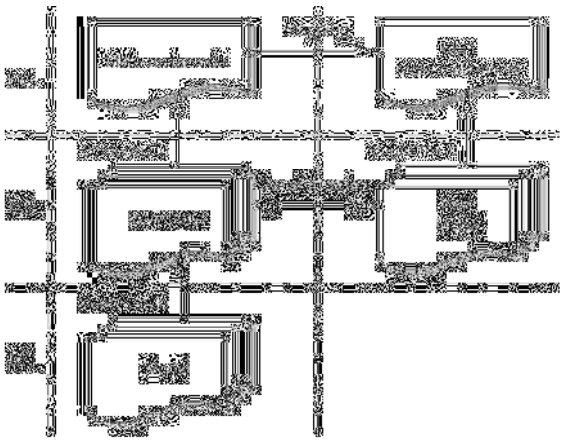

Figure 6. M3B approach in the design of Pivot metamodel

The MB3 approach significantly reduces the number of transformations that need to be implemented in order to represent the source and destination metamodel in KM3 format. Instead of defining new transformations for each new metamodel, only one transformation on meta-metamodel level needs to be defined. If there are two technological spaces with different meta-metamodels, one only needs to define two transformations enabling the automatic transformation of compliant metamodels into KM3 format. As a consequence, meta-metamodels consist of a smaller number of elements than metamodels, thereby simplifying the transformation process.

By establishing correspondence between a meta-metamodel to which a metamodel conforms and the KM3 meta-metamodel it is possible to reduce the number of pivot metamodels. This correspondence can be presented as a function $\omega_{KM3} : G_{MMM} \rightarrow G_{KM3}$, which transforms each element from the G_{MMM} multigraph into exactly one element in the G_{KM3} multigraph.

For a single function ω_{KM3}, there is exactly one pivot metamodel $piv(M_{MM})$ for each metamodel M_{MM} which conforms to the M_{MMM} meta-metamodel. Namely, if there is exactly one matching element in the KM3 meta-metamodel for each element in the meta-metamodel, then each element from a metamodel which conforms to the element in the meta-metamodel has a matching element in the pivot model which conforms to the same element in the KM3 meta-metamodel.

For M_{MMM} metamodel with a total of $|G_{MMM}|$ elements, there are at most

$$|\omega_{KM3}| = |G_{KM3}|^{|G_{MMM}|} \quad (6)$$

pivot metamodels.

Based on this formula, we can conclude that the maximum number of pivot metamodels for a metamodel depends on the number of elements in the meta-metamodel to which this metamodel conforms.

VI. AN EXAMPLE OF PIVOT METAMODEL CREATION

In Fig. 5 a metamodel for a simple faculty organization structure is presented. The metamodel conforms to ECORE meta-metamodel.

The fact that KM3 meta-metamodel is based on ECORE and MOF concepts simplifies transformations of metamodels, which conform to these meta-metamodels,

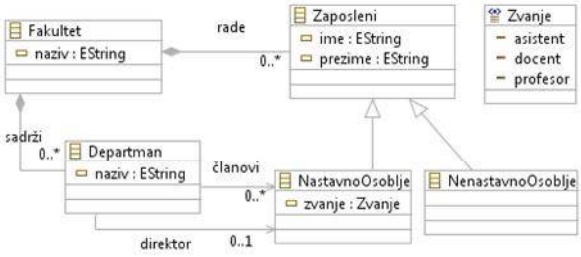

Figure 5. Metamodel for a simple faculty organization structure

into KM3 models. Each class in the ECORE metamodel (Faculty, Department, Employee, Teaching staff, Technical staff) is transformed into an element which conforms to the Class element of the KM3 meta-metamodel. The ECORE metamodel primitive type (EString) is transformed into an element which conforms to the DataType metaelement in the KM3 meta-metamodel. Each ECORE attribute is transformed into an element which conforms to the metaelement Attribute in the KM3 meta-metamodel, and belongs to the Class element, which is an equivalent to the ECORE class, to which the attribute belongs.

The type of the KM3 attribute is a matching DataType element (if it is a primitive type), or a matching KM3 class. The same transformation rules can be applied to ECORE references. Class inheritance from the ECORE metamodel is represented by an attribute from a matching KM3 class. Finally, ECORE enumeration (Title) is transformed into an element of the KM3 metamodel, which conforms to the Enumeration metaelement of the KM3 meta-metamodel. The Pivot metamodel representing a complete metamodel of the faculty organizational structure is presented in Fig 7.

VII. CONCLUSION

In this paper, the Pivot metamodel concept in the context of a formal theoretical framework for generating

```
package OrganizacionaStruktura{
	datatype String;
	enumeration Zvanje{
		literal asistent;
		literal docent;
		literal profesor;
	}
	class Fakultet{
		attribute naziv [1-1] : String;
		reference sadrzi [*] container: Departman;
		reference rade [*] container: Zaposleni;
	}
	class Departman{
		attribute naziv [1-1] : String;
		reference clanovi [*] : NastavnoOsoblje;
		reference direktor [0-1] : NastavnoOsoblje;
	}
	abstract class Zaposleni{
		attribute ime [1-1]: String;
		attribute prezime [1-1]: String;
	}
	class NastavnoOsoblje extends Zaposleni{
		attribute zvanje [1-1] : Zvanje;

	}
	class NenastavnoOsoblje extends Zaposleni{
	}
}
```

Figure 7. Pivot metamodel for the faculty organizational structure

metamodel extensions is presented. This formal theoretical framework allows the generation of a finite number of syntactically valid extensions of a metamodel (destination metamodel), based on another metamodel (source metamodel).

Pivot metamodels have a mediator role in this formal theoretical framework. These metamodels enable transformation of elements from one metamodel into extensions of another, regardless the meta-metamodels to which they conform. By using strictly defined rules and limitations, Pivot metamodels allow the description of an arbitrary metamodel that needs to be transformed in the formal theoretical framework context. Pivot metamodels are represented in the form of KM3, a textual language for DSL metamodels specification.

Apart from the definition of the Pivot metamodel, rules and limitations for transforming any metamodel into a Pivot metamodel are formulated. Furthermore, a possibility of reaching a certain level of automation in the process of transforming metamodels into KM3 models, based on the M3B approach, is described. Pivot metamodels can be formed directly (by transforming metamodels into KM3 format), or indirectly (using transformations based on correspondences between appropriate meta-metamodels).

In the concrete implementation of the formal theoretical framework, metamodels are formed indirectly, based on mappings between the source meta-metamodel and the KM3 meta-metamodel. The main reason is a potentially unlimited number of different source metamodels which conform to one source meta-metamodel. In practice, only a relatively small number of destination metamodels are subject to extending. Due to this fact, the concrete implementation of the formal theoretical framework contains a finite set of prepared destination pivot metamodels.

The direction of future work is focused the definition of transformations between a certain number of meta-models and KM3 meta-metamodel. This would enable automatic transformation of arbitrary metamodels which conform to selected meta-metamodels into a set of Pivot metamodels.

REFERENCES

[1] Atkinson, C. and Kühne, T., Model-driven development: a metamodeling foundation. Software, IEEE, 20(5), pp.36-41, 2003.

[2] Kurtev, I., Bézivin, J., Jouault, F. and Valduriez, P., 2006, October. Model-based DSL frameworks. In Companion to the 21st ACM SIGPLAN symposium on Object-oriented programming systems, languages, and applications (pp. 602-616). ACM.

[3] Fuentes-Fernández, L., Vallecillo, A.: An Introduction to UML Profiles. The European Journal for the Informatics Professional (UPGRADE), vol. 5, pp. 5–13, 2004.

[4] Luoma, J., Kelly, S., Tolvanen, J.-P.: Defining Domain-Specific Modeling Languages: Collected Experiences. 4th OOPSLA Workshop on Domain-Specific Modeling (DSM'04), 2004.

[5] Watson, A.: UML vs. DSLs: A false dichotomy. Whitepaper, 2008.

[6] Abouzahra, A., Bézivin, J., Del Fabro, M.D. and Jouault, F.: October. A practical approach to bridging domain specific languages with UML profiles. In Proceedings of the Best Practices for Model Driven Software Development at OOPSLA (Vol. 5), 2005.

[7] Del Fabro, M.D., Valduriez, P.: Towards the efficient development of model transformations using model weaving and matching transformations. Software & Systems Modeling, 8(3), pp.305-324, 2009

[8] Wimmer, M.:. A semi-automatic approach for bridging DSMLs with UML. International Journal of Web Information Systems, 5(3), pp.372-404, 2009.

[9] Giachetti, G., Marín, B. and Pastor, O.: Integration of domain-specific modelling languages and UML through UML profile extension mechanism. *IJCSA*, *6*(5), pp.145-174, 2009.

[10] Selic, B.: A systematic approach to domain-specific language design using UML. In Object and Component-Oriented Real-Time Distributed Computing, 2007. ISORC'07. 10th IEEE International Symposium on (pp. 2-9). IEEE, 2007.

[11] Bézivin, J., Bruneliere, H., Jouault, F. and Kurtev, I.: Model engineering support for tool interoperability. In Proceedings of the 4th Workshop in Software Model Engineering (WiSME 2005), Montego Bay, Jamaica (Vol. 2), 2005.

[12] Robert, S., Gérard, S., Terrier, F. and Lagarde, F.: August. A lightweight approach for domain-specific modeling languages design. In Software Engineering and Advanced Applications, 2009. SEAA'09. 35th Euromicro Conference on (pp. 155-161). IEEE, 2009.

[13] Lagarde, F., Espinoza, H., Terrier, F., André, C. and Gérard, S.: Leveraging patterns on domain models to improve UML profile definition. In Fundamental Approaches to Software Engineering (pp. 116-130). Springer Berlin Heidelberg, 2008.

[14] Kurtev, I., Bézivin, J. and Akşit, M.: Technological spaces: An initial appraisal, 2002.

[15] Pastor, O., Molina, J.C.: Model-Driven Architecture in Practice: A Software Production Environment Based on Conceptual Modeling. Springer, New York, 2007.

[16] Selic, B.: The Pragmatics of Model-Driven Development. IEEE Software, vol. 20, pp. 19–25, 2003.

[17] Jouault, F. and Bézivin, J.: KM3: a DSL for Metamodel Specification. In Formal Methods for Open Object-Based Distributed Systems (pp. 171-185). Springer Berlin Heidelberg, 2006.

[18] MOF, O., 2002. OMG Meta Object Facility (MOF) Specification v1. 4.

[19] Eclipse Foundation, "Eclipse Modeling Framework.", http://www.eclipse.org/modeling/emf/ Accessed Januar, 2016.

[20] Kern, H. and Kühne, S.: Integration of microsoft visio and eclipse modeling framework using m3-level-based bridges. In Proceedings of Second Workshop on Model-Driven Tool and Process Integration (MDTPI) at ECMFA, CTIT Workshop Proceedings (pp. 13-24), 2009.

[21] Kern, H., Stefan, F., Dimitrieski, V. and Čeliković, M.: Mapping-Based Exchange of Models Between Meta-Modeling Tools. In Proceedings of the 14th Workshop on Domain-Specific Modeling (pp. 29-34). ACM, 2014