Domain-Specific Modeling Environment for Developing Domain Specific Modeling Languages as Lightweight General Purpose Modeling Language Extensions

Igor Zečević[1(✉)], Petar Bjeljac[1], Branko Perišić[1], Vladimir Maruna[2], and Danijel Venus[1]

[1] Faculty of Technical Sciences, University of Novi Sad, Trg Dositeja Obradovića 6, 21000 Novi Sad, Serbia
{igor.zecevic,pbjeljac, perisic,danijelvenus}@uns.ac.rs

[2] MD&Profy, Čarlija Čaplina 37, 11000 Belgrade, Serbia
vladimir.maruna@mdprofy.com

Abstract. Compared to General Purpose Modeling Languages (GPMLs), Domain Specific Modeling Languages (DSMLs) have brought increasing expressivity and conciseness through the use of concepts and notations, which are customized by the characteristics of a specific problem domain. However, the creation of completely new languages can significantly increase software development costs. An alternative approach to creating new DSMLs refers to a customization of existing modeling languages. The DSML creation process through a customization of existing languages is a special case of Language Embedding. The embedding provides customization of an inherited infrastructure to a given domain. This paper elaborates a Domain-Specific Modeling environment for embedding an arbitrary DSML into an arbitrary GPML. The environment enables uniform development of DSMLs in the form of a lightweight extension of existing modeling languages.

Keywords: Model-Driven Engineering · Domain-specific modeling languages · Metamodel extension · Language embedding · UML profiles

1 Introduction

Model-Driven Engineering (MDE) [1–3] is a methodological approach to software development, in which models represent formal specifications of the solution automatically transformed into an executable specification for the selected target platform. In specific MDE paradigm implementations, such as Model-Driven Architecture (MDA) [4], Domain-Specific Modeling (DSM) [5, 6], or Model-Driven Software Development (MDSD) [7], appropriate modeling languages [8] take the central role.

Modeling languages enable a direct and formal expression of concepts related to a given domain through the use of base modeling constructs [9]. The use of relevant concepts and relations (domain conceptualization) enables expressing a mental model (domain abstraction) related to the state of certain real world segments. Conceptualization dependency level of a specific domain classifies languages into two categories: General Purpose Modeling Languages (GPML), applicable in any domain, and Domain Specific Modeling Languages (DSML) [10], which are designed for a specific domain, context or industry.

For many years, the Unified Modeling Language (UML) [11] as a GPML has been a de-facto standard in the field of software modeling, and the main building block in MDA approach. Presented as a solution for the issues related to incompatibility of hundreds of various modeling languages, UML resulted in complexity and imprecise interpretation, which were some of the issues [12–14] of its practical use.

Compared to UMLs and other GPMLs, DSMLs have brought increasing expressivity and conciseness through the use of concepts and notations, which are customized, but also limited, by the characteristics of a specific problem domain. DSMLs also ensure higher productivity, increase of final product quality, as well as direct inclusion of end users to the software development process [5, 6]. However, the creation of completely new languages can significantly increase software development costs [15, 16]. Furthermore, without proper tools, the language capabilities become limited [17], resulting in the implementation and maintenance of additional tools which already exists in the context of GPMLs.

An alternative approach to creating new DSMLs refers to extending or customizing existing modeling languages. The use of models, created on the basis of an extension of existing languages, has its advantages and strong arguments in the use of already tested tools, technologies and experiences of users in the implementation of an MDE approach. Extension specification mechanisms have already been embedded in certain modeling languages. The UML Profiles [18], as an extension mechanism of the UML language, belong to the group of "lightweight" extensions [19, 20]. They provide a mechanism for adding constraints and new elements without modifying the base metamodel.

The DSML creation process through a customization of existing modeling languages is a special case of Language Embedding [21, 22]. Embedding, as a variant of language synthesis [23], provides customization of an inherited infrastructure to a given domain. An embedded (*guest*) language uses the existing syntax, libraries, as well as the associated tools of a *host* language. Although a significant number of professional papers address the problem of DSML development using UML [24–26], there is still no possibility for a formal description of embedding a DSML into GPML.

This paper elaborates a DSM environment for embedding an arbitrary DSML into an arbitrary GPML. The environment enables uniform development of DSMLs in the form of an extension of existing modeling languages. To suite the needs of this environment, the KM3eXtension language, a textual DSML intended for formal specification of embedding DSMLs into GPMLs, has been designed. KM3eXtension generalises and customises the concepts of "lightweight" language extensions for general purpose modeling, and uses them for metamodel descriptions derived from the metamodels of existing modeling languages. Unlike the known mechanisms for extension specification, such as UML Profile and EMF Profile [27], KM3eXtension is a generic language, also applicable to languages in which metamodels do not conform to the MOF specification [28].

The rest of the paper is structured as follows. Section 2 gives a brief overview of bases of KM3eXtension language and describes architecture of the environment for embedding DSMLs into GPMLs. In Sect. 3 a KM3eXtension application in the DSM environment for language embedding is presented. The conclusions and future work are presented in Sect. 4.

2 DSM Framework for DSML Development

Development of a DSM environment for DSML development in the form of lightweight GPML extensions is directed by the following requirements:

R1: Each DSML can be represented as a guest language;

R2: Each GPML can be represented as a host language; and

R3: A definition of correspondence between the concepts of the host language and guest language, as well as their integration, has to conform to the principles of "lightweight" GPML extensions. The requirements which managed the UML Profile development [29] have been adopted as referent principles for "lightweight" extensions.

KM3eXtension is a language intended for a domain within the framework for DSML development in the form of "lightweight" GPML extensions. Generators, implemented as part of the framework, automatically transform valid KM3eXtension models into extension specifications of the original GPMLs.

2.1 KM3eXtension: A Textual DSML for Language Embedding Specification

The motive for development of the KM3eXtension language has been to formulate and implement a simple mechanism for specification related to extension of valid KM3 models. KM3 (Kernel MetaMetaModel) [30] is a textual DSL intended for specification of the metamodel for a description related to a domain of domain specific languages. KM3eXtension has been designed as an extension of the KM3 language, from which it inherits the semantics, abstract and concrete syntax.

KM3eXtension has been defined through a set of coordinated models:

M1: KM3eXtension domain definition metamodel *(Metamodel)* is a self-defining (meta-metamodel) metamodel, obtained as a result of extending the original KM3 metamodel. An class model of KM3eXtension metamodel is shown in Fig. 1.

M2: Concrete syntax is in textual form. A definition of the textual syntax[1] is available under KM3eXtension Project[2]. It enables "lightweight" extension of an arbitrary metamodel to be specified in any textual editor.

M3: Semantics of the KM3eXtension language is defined by mapping elements of KM3eXtension metamodel and UML Profile specification. A definition of these transformations[3] in the ATL language [32] is available under KM3eXtension Project.

[1] **Formal Syntax of KM3eXtension -** http://km3e.ftn.uns.ac.rs/specification/?segment=KM3eXtension-Formal-Syntax.

[2] **KM3eXtensionProject is an infrastructure necessary for modeling process in Technical Space [31] of KM3eXtension meta-metamodel:** http://km3e.ftn.uns.ac.rs/.

[3] **ATL Definition of KM3eXtension Metamodel to UML Profile Transformation -** http://km3e.ftn.uns.ac.rs/specification/?segment=KM3eXtension2UMLProfile.

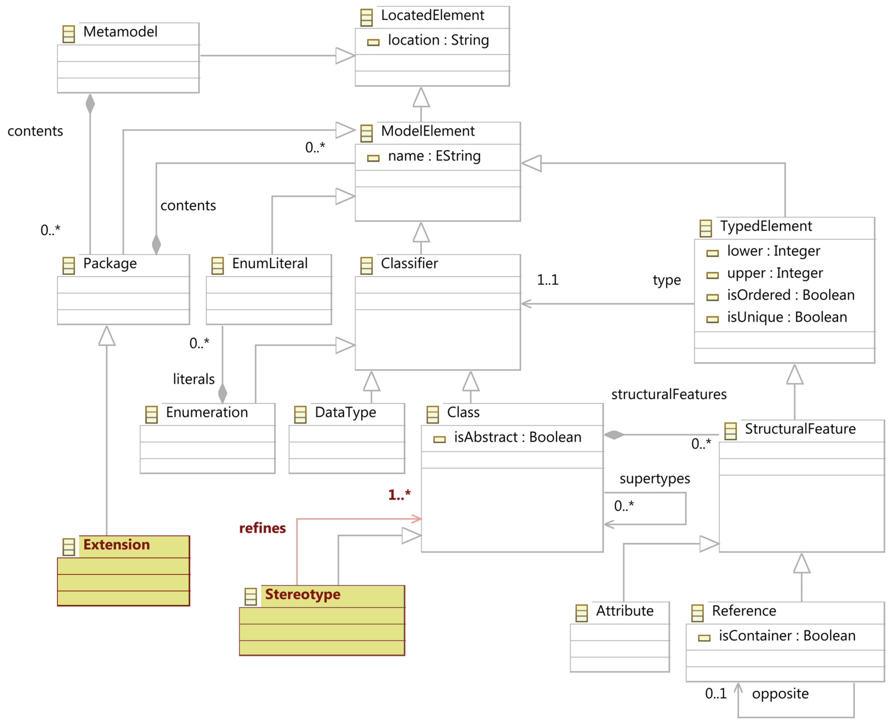

Fig. 1. KM3eXtension metamodel

In comparison with the base KM3 metamodel, KM3eXtension introduces three new metaelements (Extension and Stereotype classes and Refines relation). They are simplifications of concepts describing the UML metamodel extensions. The new metaelements are marked with different colors in Fig. 1.

Extension is a Package specialization and defines a constrained extension of a base metamodel. *Stereotype* metaelements are defined as components of Extension, through the package content which Extension inherits and additionally constrains. Stereotype is the principal extension mechanism of an initial metamodel. Each Stereotype can refine one or more classes via its *Refines* relations. Refines is a relation which introduces extensions of structural class properties using the Stereotype concept.

2.2 Architecture of DSM Environment for Embedding DSMLs into GPMLs

Figure 2 shows the architecture model of the DSM environment for embedding DSMLs into GPMLs, as well as its relations with an external environment.

The DSM environment architecture from Fig. 2 consists of:

Web Editor – An editor for KM3eXtension model. It enables creation, maintenance, validation and formatting of textual KM3eXtension models. Web editor is

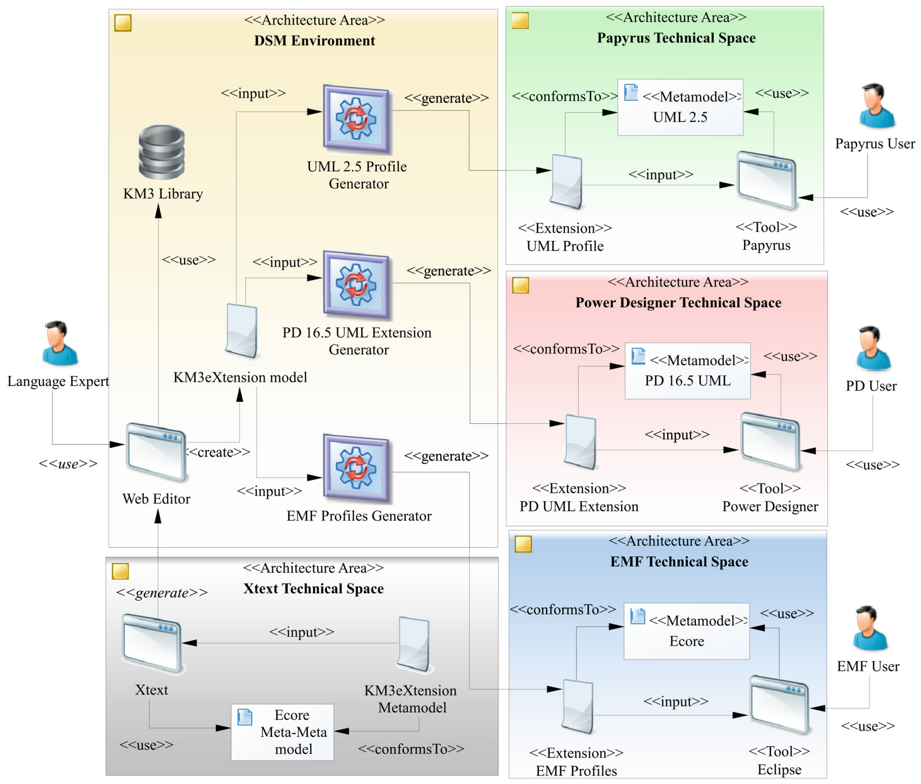

Fig. 2. Architecture model of the DSM environment for language embedding specification

automatically generated, based on the KM3eXtension Abstract Syntax Tree, from the Xtext[4] framework for DSL development;

Generators of metamodel extensions – Program components which transform valid KM3eXtension models into extension specifications of original GPMLs. The generators implement model transformations [33], that is, automatically generate a target model (metamodel extension specification) from a source model (KM3eXtension model) according to a definition of transformation. For including new GPMLs in the framework, it is sufficient to define a transformation which maps the elements of a KM3eXtension metamodel into the metamodel elements of a new GPML. The framework currently supports generation of UML Profiles for Papyrus[5], UML Extensions for Power Designer[6], as well as EMF Profiles for the EMF[7] framework;

[4] Xtext - https://eclipse.org/Xtext/.

[5] Papyrus - https://eclipse.org/papyrus/.

[6] SAP Power Designer - http://go.sap.com/product/data-mgmt/powerdesigner-data-modeling-tools.html.

[7] Eclipse Modeling Framework (EMF) - https://eclipse.org/modeling/emf/.

Library of valid KM3 models (*KM3 Library*) - A repository of models conforming to base KM3 metamodel. The library contains metamodel definitions for some of the most popular GPMLs, such as various versions of UML modeling languages.

3 Example of Use

In the process of embedding, a guest language inherits the concrete syntax of a host language, while semantics is, as a general rule, defined translationally[8], by mapping the elements of an original DSML into elements of the GPML extension. For an embedding specification it is sufficient to establish a correspondence between the host language concepts and guest language concepts, that is, it is necessary to integrate their metamodels.

A specific example of using KM3eXtension model is illustrated using the specification of embedding a DSML for the description of a WikiTable in the UML Component Model. WikiTable[9] metamodel is used for representing a simple Table in Wiki. The Ecore metamodel of the WikiTable is shown in Fig. 3.

The embedding process starts by selecting the GPML which will be subjected to extension. A metamodel of the selected GPML has to be represented by a valid KM3 model, which is also a base metamodel in the KM3eXtension model. A segment of the UML2 metamodel represented in KM3 format is specified in Listing 1. Due to the limited space, only a minimum part of the UML2 metamodel is shown.

After the base model is created, the DSML metamodel is translated into a base model extension. The WikiTable metamodel classes (LocatedElement, Table, Caption, Row and Cell) are translated into stereotypes with identical names. All structural properties of classes (attributes and references) are translated into structural properties of corresponding stereotypes. Enumeration (BorderStyle) and data types (Boolean and String) are represented by corresponding KM3eXtension elements.

A correspondence definition between the host language concepts and guest language concepts is established by a refinement relation between an extension stereotype and base metamodel classes. The establishment of refinement relations depends on the needs of a modeler, as well as on the level of his knowledge about the language metamodels involved in the process of embedding.

[8] Domain-Specific Languages from Javier Canovas - http://modeling-languages.com/useful-presentations-model-driven-engineering-dsls-uml-eclipse-modeling-technologies-2/#dsl.

[9] https://meta.wikimedia.org/wiki/Help:Table.

```
package uml2 {
        abstract class NamedElement extends Element {...}
        class Class extends EncapsulatedClassifier,BehavioredClassifier {...}
        class Component extends Class{...}
        class Property extends StructuralFeature,ConnectableElement,DeploymentTarget{...}
        ...
}
extension WikiTable {
        abstract stereotype LocatedElement refines NamedElement {
             attribute location : String;
             attribute commentsBefore[*] : String;
             attribute commentsAfter[*] : String;
        }
        stereotype Table extends LocatedElement refines Component{
             attribute border[1-1] : BorderStyle;
             attribute style[1-1] : String;
             attribute "class"[1-1] : String;
             reference caption container : Caption;
             reference rows[*] container : Row;
        }
        stereotype Caption extends LocatedElement refines Property {
             attribute content[1- 1] : String;
        }
        stereotype Row extends LocatedElement refines Component{
             reference cells[*] container : Cell;
        }
        stereotype Cell extends LocatedElement refines Component{
             attribute isHeading[1-1] : Boolean;
             attribute align[1-1] : String;
             attribute style[1-1] : String;
             attribute content[1-1] : String;
        }
        datatype Boolean;
        datatype String;
        enumeration BorderStyle{
             literal solid;
             literal dotted;
             literal double;
             literal dashed;
        }
}
```

Listing 1. KM3eXtension model: WikiTable to UML2 Specification

Generators implemented as part of the DSM environment transform a valid KM3eXtension model to an extension specification for the selected GPML. Specifications of WikiTable UML Profile for Papyrus[10] and WikiTable Extension for PowerDesigner[11] are available as part of the KM3eXtension Project.

[10] http://km3e.ftn.uns.ac.rs/examples/WikiTable/?segment=WikiTableUMLProfile.

[11] http://km3e.ftn.uns.ac.rs/examples/WikiTable/?segment=WikiTablePDExtension.

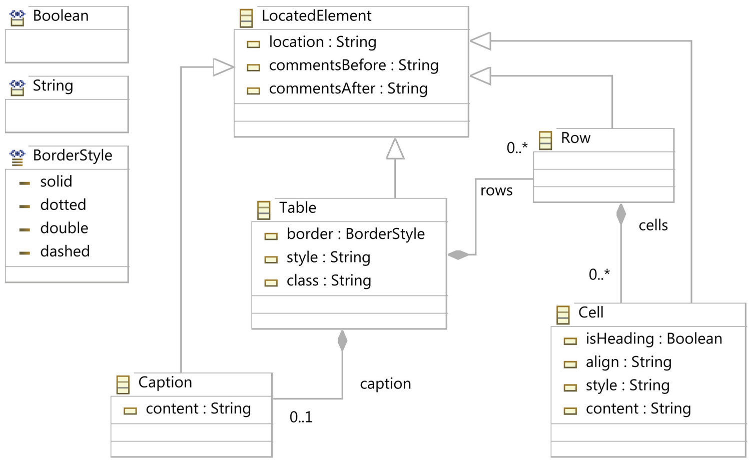

Fig. 3. Ecore definition of a WikiTable metamodel

4 Results and Conclusion

DSM environment for DSML development in the form of lightweight GPML extensions was successfully tested through fast and uniform development of several DSMLs (from various application domains) in the form of an extension of several different GPMLs. Initial versions of the KM3eXtension language and framework for an automatic translation of the KM3eXtension model into UML and EMF Profiles were implemented in the period 2013-2014 at the Computing and Control Department of the Faculty of Technical Sciences, University of Novi Sad. The first versions of the framework had exclusively educational purpose. Students had the opportunity to develop simple domain languages in the form of UML Profiles, without the need for expert level knowledge of the complex UML language metamodel.

In cooperation with the MD&Profy[12] company, the framework has been subsequently upgraded with a generator of metamodel extensions for SAP Power Designer tool. In the period 2015–2016, the environment was commercially used during the development of several DSMLs intended for various industry branches. The last example is the Bussines Process Transformation Framework implementation (BPTF) [34] in the form of the PowerDesigner metamodel extension [35].

The current drawback of KM3eXtension is the inability to introduce constraints on metamodels of languages included in embedding. The plan is to extend the KM3eXtension to include specifications of the OCL constraints [36], as well as the language editor and language parser extensions to provide a mechanism for validating the entered constraints.

[12] http://www.mdprofy.com/en/home/.

References

1. Schmidt, D.C.: Model-driven engineering. IEEE Comput. **39**(2), 25–31 (2006)
2. Favre, J.M.: Towards a basic theory to model model driven engineering. In: 3rd Workshop in Software Model Engineering, WiSME, pp. 262–271 (2004)
3. France, R., Rumpe, B.: Model-driven development of complex software: a research roadmap. In: 2007 Future of Software Engineering. IEEE Computer Society, pp. 37–54 (2007)
4. Watson, A.: A brief history of MDA. Upgrade Eur. J. Inform. Prof. **9**(2), 7–11 (2008)
5. Kelly, S., Tolvanen, J.P.: Domain-Specific Modeling: Enabling Full Code Generation. Wiley, New York (2008)
6. Wegeler, T., Gutzeit, F., Destailleur, A., Dock, B.: Evaluating the benefits of using domain-specific modeling languages: an experience report. In: Proceedings of the 2013 ACM Workshop on Domain-Specific Modeling, pp. 7–12 (2013)
7. Völter, M., Stahl, T., Bettin, J., Haase, A., Helsen, S.: Model-Driven Software Development: Technology, Engineering, Management. Wiley, New York (2013)
8. Selic, B.: The theory and practice of modeling language design for model-based software engineering—a personal perspective. In: Fernandes, João M., Lämmel, R., Visser, J., Saraiva, J. (eds.) GTTSE 2009. LNCS, vol. 6491, pp. 290–321. Springer, Heidelberg (2011). doi:10.1007/978-3-642-18023-1_7
9. Guizzardi, G.: On ontology, ontologies, conceptualizations, modeling languages, and (meta) models. Front. Artif. Intell. Appl. **155**, 18–39 (2007)
10. Kelly, S., Tolvanen, J.P.: Visual domain-specific modeling: benefits and experiences of using metaCASE tools. In: International Workshop on Model Engineering, at ECOOP, vol. 2000 (2000)
11. Fowler, M.: UML Distilled: A Brief Guide to the Standard Object Modeling Language. Addison-Wesley Professional, Boston (2004)
12. Petre, M.: UML in practice. In: Proceedings of the 2013 International Conference on Software Engineering, pp. 722–731 (2013)
13. Lange, C.F., Chaudron, M.R., Muskens, J.: In practice: UML software architecture and design description. IEEE Softw. **23**(2), 40–46 (2006)
14. France, R.B., Ghosh, S., Dinh-Trong, T., Solberg, A.: Model-driven development using UML 2.0: promises and pitfalls. Computer **39**(2), 59–66 (2006)
15. Fowler, M.: Domain-Specific Languages. Addison Wasley, Boston (2010)
16. Mernik, M., Heering, J., Sloane, A.: When and how to develop domain-specific languages. ACM Comput. Surv. (CSUR) **37**(4), 316–344 (2005)
17. Gray, J., Fisher, K., Consel, C., Karsai, G., Mernik, M., Tolvanen, J.P.: DSLs: the good, the bad, and the ugly. In: Companion to the 23rd ACM SIGPLAN Conference on Object-Oriented Programming Systems Languages and Applications, pp. 791–794. ACM (2008)
18. Fuentes-Fernández, L., Vallecillo-Moreno, A.: An introduction to UML profiles. UML Model Eng. **2**(2), 6–13 (2004)
19. Evans, A., Maskeri, G., Sammut, P., Willans, J.S.: Building families of languages for model-driven system development. In: Proceedings of WiSME, Joint with UML (2003)
20. Bruck, J., Hussey, K.: Customizing UML: which technique is right for you. White paper, Eclipse UML Project (2007)
21. Hudak, P.: Building domain-specific embedded languages. ACM Comput. Surv. (CSUR) **28** (4), 196–202 (1996)

22. Hofer, C., Ostermann, K., Rendel, T., Moors, A.: Polymorphic embedding of DSLs. In: Proceedings of the 7th International Conference on Generative Programming and Component Engineering, pp. 137–148 (2008)
23. Vallecillo, A.: On the combination of domain specific modeling languages. In: Kühne, T., Selic, B., Gervais, M.-P., Terrier, F. (eds.) ECMFA 2010. LNCS, vol. 6138, pp. 305–320. Springer, Heidelberg (2010). doi:10.1007/978-3-642-13595-8_24
24. Selic, B.: A systematic approach to domain-specific language design using UML. In: 10th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC 2007), pp. 2–9 (2007)
25. Lagarde, F., Espinoza, H., Terrier, F., Gérard, S.: Improving UML profile design practices by leveraging conceptual domain models. In: Proceedings of the Twenty-Second IEEE/ACM International Conference on Automated Software Engineering, pp. 445–448 (2007)
26. Giachetti, G., Marín, B., Pastor, O.: Using UML as a domain-specific modeling language: a proposal for automatic generation of UML profiles. In: International Conference on Advanced Information Systems Engineering, pp. 110–124 (2009)
27. Langer, P., Wieland, K., Wimmer, M., Cabot, J.: EMF profiles: a lightweight extension approach for EMF models. J. Obj. Technol. **11**(1), 1–29 (2012)
28. OMG: OMG Meta Object Facility (MOF) Core Specification Version 2.5. OMG Document formal/2015-06-05 (2015)
29. OMG: OMG Unified Modeling Language (OMG UML), Superstructure, Version 2.4.1, OMG Document formal/2011-08-06 (2011)
30. Jouault, F., Bézivin, J.: KM3: a DSL for metamodel specification. In: Gorrieri, R., Wehrheim, H. (eds.) FMOODS 2006. LNCS, vol. 4037, pp. 171–185. Springer, Heidelberg (2006). doi:10.1007/11768869_14
31. Bézivin, J., Kurtev, I.: Model-based technology integration with the technical space concept. In: Metainformatics Symposium, vol. 20, pp. 44–49 (2005)
32. Jouault, F., Kurtev, I.: Transforming models with ATL. In: International Conference on Model Driven Engineering Languages and Systems, pp. 128–138 (2005)
33. Czarnecki, K., Helsen, S.: Classification of model transformation approaches. In: Proceedings of the 2nd OOPSLA Workshop on Generative Techniques in the Context of the Model Driven Architecture, vol. 45(3), pp. 1–17 (2003)
34. Mercer, T., Groves, D., Drecun, V.: Part III – Practical BPTF Application, Business Process Trends (2010). http://www.bptrends.com/publicationfiles/FOUR%2011-02-10-ART-BPTF%20Framework–Part%203-Mercer%20et%20al%20–final1.pdf
35. Maruna, V., Mercer, T., Zečević, I., Perišić, B., Bjeljac, P.: The business process transformation framework implementation through metamodel extension. In: Proceedings of the 6th International Conference on Information Society and Technology, (ICIST 2016), pp. 11–17 (2016)
36. OMG: OMG Object Constraint Language (OCL) Version 2.4, OMG Document formal/2014-02-03 (2014)