Model Driven Engineering Ontology: Increasing Value, Reusability and Accessibility of MDE Assets

Claudia T. Pereira, Liliana I. Martinez

Departamento de Computación y Sistemas
Universidad Nacional del Centro de la Provincia de Buenos Aires (UNCPBA)
Tandil, Buenos Aires, Argentina

{cpereira,lmartine}@exa.unicen.edu.ar

Abstract. *Model Driven Engineering is a relatively new software engineering paradigm that considers models as first-class entities where models can be used to assist engineering processes in any application domain. This paper presents an ontology to provide a degree of formality and a potential for automatic processing in MDE domain. The ontology has been designed to provide a common vocabulary of the field and allow the use of a common format to store and exchange data about MDE assets, increasing their value and reusability.*

1. Introduction

Model Driven Software Engineering (MDSE or MDE for short) is a software engineering paradigm that encompasses model-based tasks of the complete software engineering process. The MDE core concepts are models and transformations where models are considered first-class entities. [Brambilla et al. 2017, Rodrigues da Silva 2015].

The success of MDE approaches depends on the existence of CASE tools that make a significant impact on the automation degree of software processes [Favre et al. 2009] and also depends on the existence of repositories that provide significant support to research and education. The aim of repositories is to share MDE artifacts, not only models, metamodels, and transformations but also terminology, tools, projects and scientific research related to the field. Existing repositories containing both industrial and academic examples have been constructed with the aim of reusing, analyzing, and learning from them. However, not all proposals have been as successful as expected [Bucchiarone et al. 2020], due to storage issues (different format, missing metamodels, partial models) and information retrieval issues (keyword-based searches carried out in the current Web), among other things. Therefore, a common format to store and exchange data about MDE artifacts could be beneficial.

An ontology, defined as "a formal, explicit specification of a shared conceptualization" [Guarino et al. 2009], is a good tool for the formal representation of knowledge oriented to semantic analysis by machines. Hence its relationship with the Semantic Web, data on the Web defined and linked so that they can be used by machines to visualize, automate tasks, integrate and reuse data between applications [Berners-Lee et al. 2001].

In the light of the above mentioned, we propose an ontology to represent and share the common knowledge of the MDE domain and to allow the use of common format to store and exchange knowledge. If several different websites share and publish the same underlying ontology of the MDE domain terms they use, then, computer agents could

extract and aggregate information from these different sites such as, models, transformations, standards, modeling case studies, modeling success stories, and other forms of modeling knowledge and experience.

The paper is organized as follows. Section 2 presents an overview of the MDE domain. Section 3 presents and details the MDE ontology by describing the most important concepts and their relationships. Section 4 discusses related work. Finally, section 5 concludes the work and presents future directions.

2. MDE: Domain of Interest

The MDE core concepts are models and transformations where models are considered first-class entities. The key idea is to separate the specification of the system functionality from its implementation on specific platforms. The model abstraction level may vary widely according to its objectives, ranging from the domain model to the code model. A transformation is the automatic generation of a model from a source model, according to a transformation definition, which consists of a set of transformation rules that together describe how a model in the source language can be transformed into a model in the target language. Transformations between models aim to increase the automation degree in order to improve the productivity and quality of the whole software engineering process. In the MDE context everything is a model, thus [Brambilla et al. 2017]:

- Transformations can be considered as particular models of operation upon models.
- Models and transformations are expressed in modeling languages. The definition of a modeling language itself can be seen as a model, this procedure is known as metamodeling (modeling a model, modeling a modeling language).
- Processes, development tools, and resulting programs can also be defined as models.

3. MDE Ontology

The ontology was designed in the conceptual modeling tool ICOM (2021) that employs complete logical reasoning to verify the specification, infer implicit facts, devise stricter constraints, and manifest any inconsistency. Classes are represented by boxes and n-ary associations by diamonds. Role is used to denote the connection of an entity to a relationship and to express cardinality constraints of an entity in a relation: the minimum cardinality can be 0 or 1 (1 is represented by a black dot) and the maximum cardinality can be 1 (represented by an arrow) or n. *IsA* relationships specify that one class is a subclass of another and they are represented as a triangle with a disc in the middle. ICOM allows distinguishing between regular (white disc), total (bold disc), exclusive (cross disc), and total-exclusive (bold and cross disc) *isA* relationships [Fillottrani et al. 2012].

Figure 1 partially shows the ontology, it captures the MDE core concepts, relationships among them and constraints. There are different types of *Models*, each one *conformsTo* its respective *Metamodel* and *isSpecifiedBy* a *Language*. There are different types of *Languages* such as *ModelingLanguage, TransformationLanguage* and *ProgrammingLanguage*. A *Transformation isBasedOn* a *TransformationDefinition* and has at least one source model and one target model. These concepts are linked to concepts defined in other ontologies such as DBpedia (2021), DBLP (2021), and Dublin Core (2021) in order to reutilize them. DBpedia is a project aiming to extract structured content from the information created in the Wikipedia project. Our ontology reuses concepts related to software engineering such as *computing platform, programming language*, and

software. The DBLP ontology describes concepts such as *Person* and *Publication* and relationships such as *author*. The Dublin Core ontology provides vocabularies of elements and terms for describing generic metadata. The terms *isReferencedBy* and *creator* are related to the term *MDEconcept* of our ontology.

Some of the competence questions that the proposed ontology could answer are: "Does model X conform to metamodel Y? "Is the definition of transformation X written in language Y?", and "What type of transformation does the transformation language X support?"

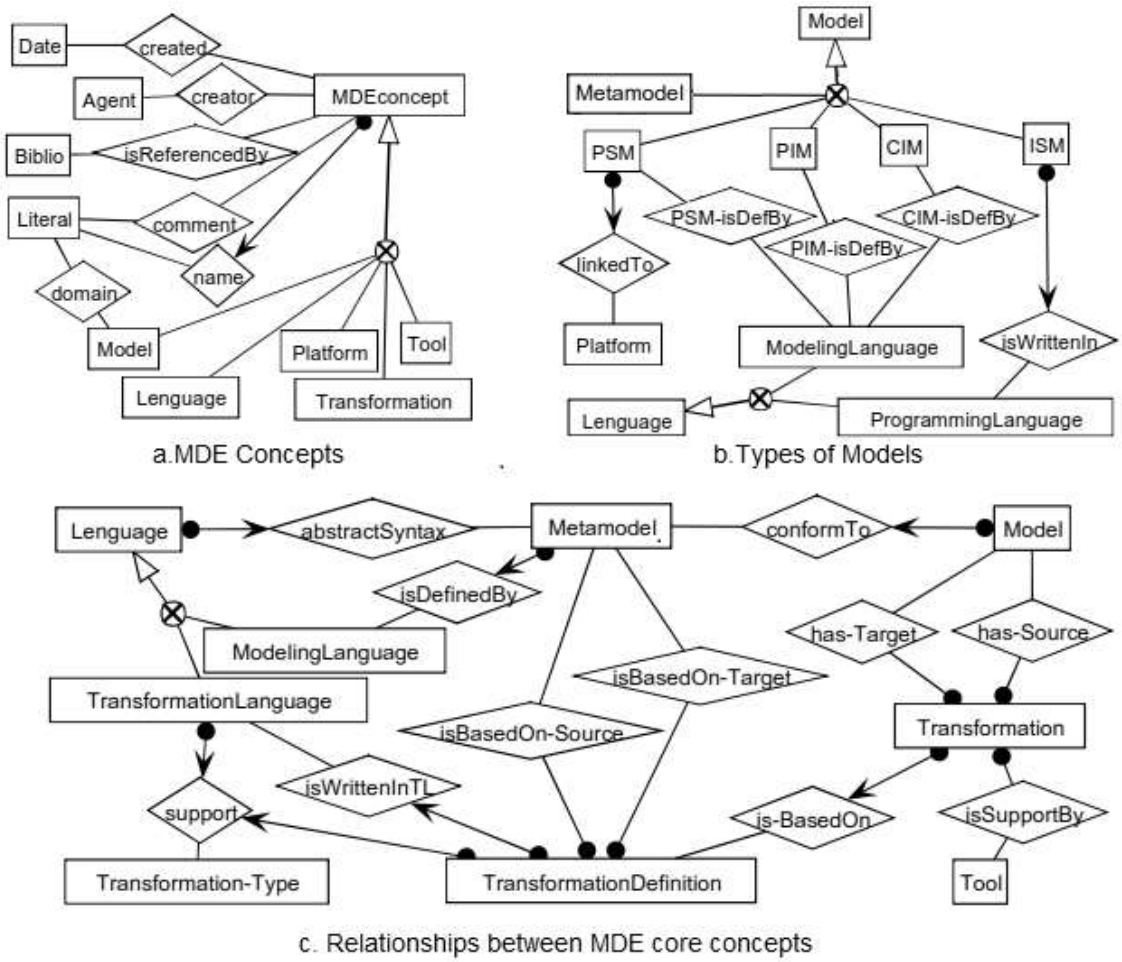

Figure 1. MDE Ontology

3.1 Concepts and properties

The most relevant concepts included in the ontology and their most important properties are described below. Cardinality constraints are enclosed in square brackets ([minimum cardinality..maximum cardinality]). Sources of MDE concepts are MDA (2014), Brambilla et al. (2017), Rodrigues da Silva (2015), and Kleppe (2003).

MDEConcept: A concept that owns the common properties of the components that build up the MDE ontology. Its properties are:

- name [1..1]: The name of the concept.
- creator [0..n]: Agents or entities responsible for making the concept.

- isReferencedBy [0..n]: Related resources that refer to the concept described.
- created [0..1]: The date of creation of the concept.
- comment [0..n]: Comments may be used to provide a human-readable description that helps clarify the meaning of the concept.

Model: A specification of the function, structure and/or behavior of an application or system, expressed in a language. Three subclasses of models are distinguished, computer independent model (CIM), platform independent model (PIM), and platform specific model (PSM). The model properties are:

- subclass of: MDEConcept.
- conformTo [1..1]: The metamodel to which the model conforms.
- domain [0..n]: Domains that the model represents.
- isSpecifiedBy [1..n]: Languages used to specify the model.

Metamodel: A special kind of model that defines the language for specifying a model. A metamodel specifies the abstract syntax of the language, the modeling concepts, their attributes, and their relationships as well as the structural rules that restrict the possible elements of valid models. The metamodel properties are:

- subclass of: Model.
- isDefinedBy [1..1]: The Modeling Language used to define the metamodel.

Transformation Definition: Transformations are defined at metamodel level and applied at model level. Source and target models must conform to their respective metamodels. Transformation definitions can be classified into: Text to Model (T2M), Model to Model (M2M), and Model to Text (M2T). The properties of the transformation definition are:

- subclass of: Model.
- isWrittenInTL [1..1]: The language used to write the transformation definition.
- isbasedOn_SourceMetamodel [1..n]: Metamodels that source models must conform to.
- isbasedOn_TargetMetamodel [1..n]: Metamodels that target models must conform to.
- support [1..1]: The transformation type (M2M, M2T, T2M).

Transformation: The automatic generation of target models from source models, according to a transformation definition. The transformation properties are:

- subclass of: MDEConcept.
- isBasedOn [1..1]: The Transformation Definition that specifies the transformation.
- isSupportBy [1..n]: Tools that support the transformation.
- has_SourceModel [1..n]: Input models of the transformation.
- has_TargetModel [1..n]: Output models of the transformation.

Language: A tool to define a concrete representation of a conceptual model that can be automatically interpreted by a computer. The language properties are:

- subclass of: MDEConcept.
- abstractSyntax_definedBy [1..1]: The metamodel that defines the concepts of the language and their relationships, independently of any particular representation.
- has_ConcreteSyntax [1..n]: The concrete syntax type that describes specific representations of the modeling language (“textual”, “graphical” or “mixed”).

- has_Semantics [1..1]: The semantics type used to describe the meaning of the elements defined in the language and the meaning of the different ways of combining them ("denotational", "translational" or "operational").

Modeling Language: Language that lets designers specify the models for their systems. These languages are classified in Domain-Specific Modeling Languages (DSL) and General-Purpose Modeling Languages (GPL). The properties are:

- subclass of: Language.
- type [1..1]: The modeling language type ("DSL", "GPL") .
- specification [0..1]: The standard that specifies the modeling language.

Transformation Language: Language for the definition of transformations. Its properties are:

- subclass of: Language.
- has_style [1..1]: The style such as "declarative", "imperative" and "multi-paradigm".
- support [1..1]: The transformation type ("M2M", "M2T", "T2M").
- specification [0..1]: The standard that specifies the transformation language.

4. Related Work

Some ontologies have been developed in the Software Engineering (SE) domain, however, MDE specific ontologies have not been developed yet. Some related works about existing ontologies that define concepts with which our ontology could be linked are presented below.

Wongthongtham et al. (2009) propose an ontology model for representing the SE knowledge. The main purpose is to enable communication between computer systems or software engineers to understand common SE knowledge and to perform certain types of computations. The key elements that make up the ontology are a vocabulary of basic terms of the SE domain and a precise specification of what those terms mean.

In [Ruy et al. 2016], authors present a SE Ontology Network (SEON), which provides a network of ontologies in the SE domain and mechanisms to derive and incorporate new subdomain ontologies into the network. SEON includes core ontologies for software and software processes, as well as domain ontologies for the main technical software engineering subdomains, such as requirements, design, coding, and testing.

In [Dietrich and Elgar 2007], authors present an ontology to formally define design patterns and some related concepts such as pattern participant, pattern refinement, and pattern instance. The WOP (WebOfPatterns) is a toolkit that facilitates the sharing of knowledge about software design.

5. Conclusions

This research proposes an ontology with the aim that the common knowledge of the MDE domain can be shared and reused across people and computers throughout Semantic Web. Thus, heterogeneous information such as models, transformations, languages, tools, projects, and academic papers could be expressed and handled by a single system of expression of knowledge based on the proposed ontology.

The presented ontology is a starting point that provides a degree of formality and a potential for automatic processing in the MDE domain. It is expected that the use of the ontology provides benefits to the community by increasing the value, reusability and accessibility of MDE assets.

We foresee expanding the conceptual model by adding concepts and relationships linked to other tasks or subdomains within MDE, such as refactoring and model evolution.

References

Berners-Lee, T., Hendler, J. and Lassila, O. (2001). The semantic web: a new form of Web content that is meaningful to computers will unleash a revolution of new possibilities. In *Scientific American*, Volume 284 (5), pages 34-43.

Brambilla, M., Cabot, J. and Wimmer, M. (2017). Model-Driven Software Engineering in Practice, Morgan & Claypool Publishers, Second Edition.

DBLP Ontology (2021). Available at: http://swat.cse.lehigh.edu/resources/onto/dblp.owl

DBpedia Ontology (2021). Available at: http://wiki.dbpedia.org/

Dietrich, J. and Elgar, C. (2007). Towards a Web of Patterns. In *Journal of Web Semantics*. Volume 5, Issue 2, June 2007, pages 108-116. ISSN: 1570-8268.

Dublin Core Ontology (2021). Available at: https://dublincore.org/specifications/dublin-core/dcmi-terms/

Favre, L., Martinez, L. and Pereira, C. (2009). "MDA based reverse engineering of object-oriented code". In: Proceedings EMMSAD 2009, Lecture Notes in Business Information Processing, Volume 29, p. 251-263. Berlin: Springer-Verlag.

Fillottrani, P., Franconi, E., and Tessaris, S. (2012) The ICOM 3.0 intelligent conceptual modelling tool and methodology. In *Semantic Web Journal*, volume 3, pages 293–306.

Guarino, N., Oberle, D. and Staab, S. (2009). What Is an Ontology? Handbook on Ontologies (2nd ed.). Springer Publishing Company, Incorporated, pages 1-17.

ICOM (2021). A Tool for Intelligent Conceptual Modelling. https://www.inf.unibz.it/~franconi/icom/

Kleppe, A., Warmer, J. and Bast, W. (2003). MDA Explained: The Model Driven Architecture: Practice and Promise. Addison-Wesley.

MDA Guide (2014). MDA Guide Revision 2.0. Document: ormsc/14-06-01. https://www.omg.org/cgi-bin/doc?ormsc/14-06-01

Rodrigues da Silva, A. (2015). Model-driven engineering: A survey supported by the unified conceptual model. In *Computer Languages, Systems & Structures*, Volume 43, pages 139-155.

Ruy, F.B., Falbo, R.A., Barcellos, M.P., Costa, S.D., Guizzardi, G. (2016). SEON: A Software Engineering Ontology Network. In *Knowledge Engineering and Knowledge Management,* LNCS, Volume 10024. Springer, Cham, pages 527-542.

Wongthongtham, P., Chang, E., Dillon, T. and Sommerville, I. (2009). Development of a Software Engineering Ontology for Multisite Software Development. In *IEEE Transactions on Knowledge and Data Engineering*, Volume. 21(8), pages 1205-1217.