García-Holgado, A., & García-Peñalvo, F. J. (2018). Learning ecosystem metamodel quality assurance. In Á. Rocha, H. Adeli, L. P. Reis, & S. Costanzo (Eds.), *Trends and Advances in Information Systems and Technologies* (Vol. 1, pp. 787-796). Cham: Springer. doi:10.1007/978-3-319-77703-0_78

Learning ecosystem metamodel quality assurance

Alicia García-Holgado[1] and Francisco José García-Peñalvo [1],

[1] GRIAL Research Group, Research Institute for Educational Sciences
University of Salamanca, Salamanca, Spain
{aliciagh,fgarcia}@usal.es

Abstract. The learning ecosystem metamodel is a framework to support Model-Driven Development of learning ecosystems based on Open Source software. The metamodel must be validated in order to provide a robust solution for the development of this type of technological solutions. The first phase of the validation process has done manually, but to ensure the quality of the metamodel, the last phase should be made using a tool. The first version of the metamodel is an instance of MOF, the standard defined by the Object Management Group. There are not stable tools to support the definition and mapping of metamodels and models using the standards. For this reason, is necessary to transform the metamodel from MOF to Ecore in order to use the tools provided by Eclipse. This work describes the transformation process and the measures to ensure the quality of the learning ecosystem metamodel in Ecore.

Keywords: Metamodel, Model Driven Development, Learning ecosystems, Information systems, Software engineering, Ecore, Quality.

1 Introduction

Technological ecosystems are a set of software tools connected by information flows to provide new extra functionality than each tool separately and to support knowledge management processes inside any kind of institution [1]. Furthermore, these technological solutions have an important human factor represented through two input flows, the methodology and the management, not only by the users that use the ecosystem. Learning ecosystems are a kind of technological ecosystem focus on learning management both in companies and institutions.

This technological approach is the evolution of traditional information systems and offers advantages such as the ability to evolve in different dimensions [2, 3] or the reusing of heterogeneous tools already developed to build new systems. On the other hand, the definition, development and deployment of this type of software solutions is complex and involves several problems identified in previous works [4]. Based on this analysis, an architectural pattern has been defined [2] as an input to define a metamodel to support Model-Driven Development (MDD) of learning ecosystems. The basic idea of a metamodel is to identify the main concepts and their relations of a given problem domain used to describe the models of that domain [5].

The learning ecosystem metamodel [6] is a Platform-Independent Model (PIM) to define learning ecosystems based on Open Source software. It has been defined using the Model-Driven Architecture (MDA) proposed by the Object Management Group (OMG) to apply MDD using the OMG standards for visualizing, storing, and exchanging software designs and models [7]: Meta Object Facility (MOF), Unified Modeling Language (UML), XML Metadata Interchange (XMI) and Query/View/Transformation (QVT). The ecosystem metamodel is a M2-model in the four–layer metamodel architecture; it is an instance of the Meta Object Facility (MOF).

The validation of the metamodel has been carried out through the instantiation of conceptual models of learning ecosystems. Two Model-to-Model (M2M) transformations have been made in previous works to test that the metamodel allows to define a real learning ecosystem. In particular, the models instantiated from the learning ecosystem metamodel are learning ecosystems for knowledge management in two different contexts, PhD Programmes with an Open Access policy [8] and the Spanish Public Administration [2]. Both models fulfil the metamodel constraints verified from a theoretical point of view. These preliminary validations are available in [9, 10].

To complete the validation process is necessary verify that the instances of the learning ecosystem metamodel are reciprocated to the deployment of the learning ecosystem in a real context, in other words, it is necessary to transform the PIM model of a learning ecosystem in a PSM (Platform Specific Model) model. To ensure the validity of the process, the transformations should be done using a tool, not manually.

Although OMG provides several standards to support MDA, there are no stable tools to support the definition and mapping of metamodels and models using those standards. However, Eclipse has Eclipse Modelling Project (EMF), a set of Eclipse plugins that provide a framework to develop metamodels using Ecore and to support automatic Model-to-Model and Model-to-Text transformation through the definition of transformation rules. Ecore [11] is a meta-metamodel based on MOF focused on being simpler and more practical. Further, the designers of Ecore have participated in the definition of the core of MOF 2.0, Essential MOF or EMOF, so both are very similar. For this reason, the last phase of the validation process will be developed using EMF.

The main purpose of this work is to ensure the quality of the learning ecosystem metamodel. More specifically, the objectives are to describe the transformation of the learning ecosystem metamodel from MOF to Ecore in order to use the tools provided by EMF, and to apply a quality framework to both MOF and Ecore version.

The paper has been organized in the following way. Section 2 describes the methodology used to transform the metamodel from MOF to Ecore. Section 3 describe the transformation of the learning ecosystem metamodel and the differences between both versions. Section 4 analyses the quality of the learning ecosystem metamodel both MOF and Ecore. Finally, Section 5 summarizes the main conclusions of this work.

2 Methodology

In order to guarantee the quality of the learning ecosystem metamodel, a series of analysis and transformations have been performed. First, the metamodel instantiated from MOF has been analysed following the quality framework provided by López-Fernández, Guerra and de Lara [12]. The objective has been to know the quality problems of the metamodel to solve them in the Ecore version.

After the analysis, the learning ecosystem metamodel in MOF has been transformed in an instance of Ecore. Both metamodels are M2-model in the four–layer metamodel architecture provided by MDA (Fig. 1). Both MOF and Ecore support the use of XMI enabling the interchange of models and model instances through XML based on DTDs/XMLSchemas generated from the corresponding models [13]. However, in this work the transformation has been made manually because of several problems with the tool used to define the metamodel in MOF. This one was made with a UML class diagram in Visual Paradigm and it has not been possible to import it into Eclipse using XML Metadata Interchange (XMI). The instance of Ecore has been made using the Graphical Modelling for Ecore included in EMF.

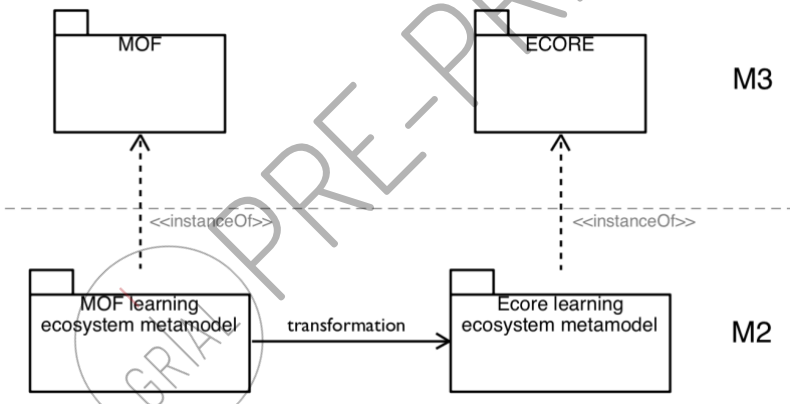

Fig. 1. Different abstraction levels of the learning ecosystem metamodel

The first version of the metamodel has a set of constraints defined with Object Constraint Language (OCL) and included in the metamodel as annotations. During the transformation process, these constraints have been reviewed and finally twelve constraints have been included in the Ecore metamodel. Moreover, the constraints in the Ecore metamodel are included in each element using the OCLinEditor provided by EMF.

Finally, the quality of the learning ecosystem metamodel instantiated from Ecore has been checked. It has been used the same metamodel quality framework than in the MOF version.

3 Learning ecosystem metamodel

The first version of the learning ecosystem metamodel instantiated from MOF is published in García-Holgado and García-Peñalvo [6] and is available in high resolution in http://doi.org/10.5281/zenodo.829859.

This section describes the mapping between the MOF version of the metamodel to an instance of Ecore, as well as a set of improvements to ensure the quality of the learning ecosystems instantiated from the final version of the metamodel.

3.1 Ecore metamodel

The transformation from MOF to Ecore has made manually. To prevent confusion, this work uses the "MOF" prefix for concepts in MOF and the "E" prefix for concepts in Ecore.

First, each MOFClass has been mapped in a EClass and three new classes have been defined. On the one hand, two children have been added to the hierarchy of *Infrastructure*: *IndexingService* element was detected during the preliminary validation when the metamodel was instantiated in the learning ecosystem for knowledge management in the Spanish Public Administration; and *OtherSystemTool* replaces the MOFClass "…" because ellipsis are forbidden symbols in EClass names. On the other hand, the MOFClass *InformationFlow* has been divided in two EClasses, one with the same name that represents the communication between software tools either through human interaction or through the development of software mechanisms; and one for representing the implementation of the information flow, *CommunicationMechanism*.

Secondly, each MOFAttribute has been mapped in a EAttribute. In this process, some new EAttributes have been defined in order to fulfil one of the EMF best practices, all classes must have a unique identifier attribute. Specifically, a EAttribute *name* or *title* has been added to: *InformationFlow*, *CommunicationMechanism*, *ServiceInterface* and *ServiceOperation*.

Moreover, other EAttributes have been included to have the required information to transform PIM models to PSM models. In particular, *ExternalTool* has two new EAttributes of type EString related to the connection between the ecosystem and the external tool (*id*, *key*); *InternalTool* has three new EAttributes of type EBoolean to determine some features related to information needs - complexity of the contents (*complexContentType*), use of questionnaires or surveys (*questionnaire*) and use for teaching (*teaching*) -; and *User* has a new EAttribute to distinguish his/her role in the institution, a EAttribute of a EEnum defined in the metamodel, *userType*.

Finally, each MOFAssociation has been mapped in a EReference. This process has been more difficult because in the learning ecosystem metamodel the MOFAssociations had not defined the navigability. Instead, Ecore support uni-directional and bi-directional references and it is mandatory define the navigability and a unique name for each EReference. Also, the upper and lower bounds of EReferences have been reviewed and some changes have been made: the lower bound of the EReference *configConsumer* is 0 instead 1; and the lower bound of the EReference *establishedMethodology* is 0 instead 1.

Fig. 2 show the result of the mapping process from MOF to Ecore and the changes made to support the M2M transformations in EMF. The final version of the learning ecosystem metamodel in Ecore can be seen in better resolution on the following link https://doi.org/10.5281/zenodo.1066369.

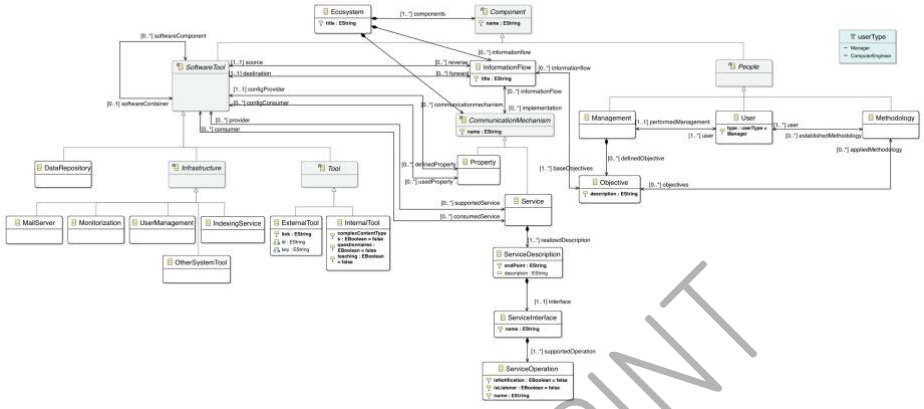

Fig. 2. Learning ecosystem metamodel in Ecore

3.2 OCL constraints

The MOF version of the learning ecosystem metamodel has four OCL constraints [6]. During the mapping from MOF to Ecore some new OCL constraints has been defined to guarantee the correct instantiation of the metamodel. In particular, eight new OCL constraints have been defined in the metamodel. Moreover, two constraints defined in the MOF metamodel have been modified.

The twelve constraints have been included in the Ecore metamodel using the OCLinEditor provided by EMF.

Regarding the modifications, the constraint to ensure the required components of a learning ecosystem has been changed to allow more than one monitorization tool, because sometimes there are several monitorization tools that are part of other components and combined provide the monitorization of the ecosystem.

```
context Ecosystem invariant requieredComponents:
   self.components -> select(c |
      c.oclIsTypeOf(MailServer)) -> size() = 1
   and self.components -> select(c |
      c.oclIsTypeOf(Monitorization)) -> notEmpty()
   and self.components -> select(c |
      c.oclIsTypeOf(UserManagement)) -> size() = 1
   and self.components -> select(c |
      c.oclIsTypeOf(InternalTool)) -> notEmpty()
   and self.components -> select(c |
      c.oclIsTypeOf(Management)) -> notEmpty()
```

```
   and self.components -> select(c |
      c.oclIsTypeOf(Methodology)) -> notEmpty()
   and self.components -> select(c |
      c.oclIsTypeOf(User)) -> notEmpty();
```

An information flow always involves two different software tools, both for services and properties. The constraint for ensuring that a software tool cannot consume a service provided by itself has been redefined and a new constrain for properties has been defined.

```
context SoftwareTool invariant differentService:
   self.consumedService -> forAll(k |
      k.provider -> forAll(j | j <> self));

context SoftwareTool invariant differentConfig:
   self.usedProperty -> forAll(k |
      k.configProvider <> self);
```

Five of the new constraints are focused on the relationships among the components. First, a software tool cannot be contained itself directly or by transitivity:

```
context SoftwareTool invariant ownContainer:
self.softwareComponent -> forAll(k | k <> self);
```

An external tool cannot contain or be container of other software tools and a data repository cannot be a component of other software tool:

```
context DataRepository
invariant independentExternalTool1:
   self.softwareComponent -> forAll(k |
      not k.oclIsTypeOf(ExternalTool));

context InternalTool
invariant independentRepo_ExternalTool2:
   self.softwareComponent -> forAll(k |
      not k.oclIsTypeOf(DataRepository))
   and self.softwareComponent -> forAll(k |
      not k.oclIsTypeOf(ExternalTool));

context ExternalTool
invariant independentExternalTool2:
   self.softwareComponent -> forAll(k |
      not k.oclIsKindOf(Infrastructure)) and
   self.softwareComponent -> forAll(k |
      not k.oclIsTypeOf(DataRepository)) and
   self.softwareComponent -> forAll(k |
      not k.oclIsTypeOf(InternalTool));
```

```
context Infrastructure
invariant independentRepo_ExternalTool1:
   self.softwareComponent -> forAll(k |
      not k.oclIsTypeOf(DataRepository))
   and self.softwareComponent -> forAll(k |
      not k.oclIsTypeOf(ExternalTool));
```

Finally, when a software tool consumes a service or a property must be at least an information flow between it and the service or property provider:

```
context SoftwareTool
invariant servicewithInformationFlow:
   self.consumedService -> isEmpty() or
   self.consumedService -> forAll(k |
      k.informationFlow -> exists(j |
         j.source = self and k.provider -> exists(m |
            m = j.destination)));

context SoftwareTool
invariant propertywithInformationFlow:
   self.usedProperty -> isEmpty() or
   self.usedProperty -> forAll(k |
      k.informationFlow -> exists(j |
         j.source = self and
         j.destination = k.configProvider));
```


4 Quality of the metamodel

The quality of the learning ecosystem metamodel has been checked using the metamodel quality framework proposed by López-Fernández, Guerra and de Lara [12]. This framework is composed by thirty features that metamodels should follow (Table 1). The features are divided in four categories: (1) design, properties signalling a faulty design (an error); (2) best practices, basic design quality guidelines (a warning); (3) naming conventions, questions related to the use of verbs, nouns, etc.); (4) metrics, measurements of metamodel elements and their threshold value [14].

Table 1. Features of the metamodel quality framework [12]

	Design
D01	An attribute is not repeated among all specific classes of a hierarchy.
D02	There are no isolated classes (i.e., not involved in any association or hierarchy).
D03	No abstract class is super to only one class (it nullifies the usefulness of the abstract class).
D04	There are no composition cycles.
D05	There are no irrelevant classes (i.e., abstract and subclass of a concrete class).
D06	No binary association is composite in both member ends.
D07	There are no overridden, inherited attributes.
D08	Every feature has a maximum multiplicity greater than 0.

D09	No class can be contained in two classes, when it is compulsorily in one of them.
D10	No class contains one of its superclasses, with cardinality 1 in the composition end (this is not finitely satisfiable).
	Best practices
BP01	There are no redundant generalization paths.
BP02	There are no uninstantiable classes (i.e., abstract without concrete children).
BP03	There is a root class that contains all others (best practice in EMF).
BP04	No class can be contained in two classes (weaker version of property D09).
BP05	A concrete top class with subclasses is not involved in any association (the class should be probably abstract).
BP06	Two classes do not refer to each other with non-opposite references (they are likely opposite).
	Naming conventions
N01	Attributes are not named after their feature class (e.g., an attribute paperID in class Paper).
N02	Attributes are not potential associations. If the attribute name is equal to a class, it is likely that what the designer intends to model is an association.
N03	Every binary association is named with a verb phrase.
N04	Every class is named in pascal-case, with a singular-head noun phrase.
N05	Element names are not too complex to process (i.e., too long).
N06	Every feature is named in camel-case.
N07	Every non-boolean attribute has a noun-phrase name.
N08	Every boolean attribute has a verb-phrase (e.g., isUnique).
N09	No class is named with a synonym to another class name.
	Metrics
M01	No class is overloaded with attributes (10-max by default)
M02	No class refers to too many others (5-max by default) – a.k.a. efferent couplings (Ce).
M03	No class is referred from too many others (5-max by default) – a.k.a. afferent couplings (Ca).
M04	No hierarchy is too deep (5-level max by default) – a.k.a. depth of inheritance tree (DIT).
M05	No class has too many direct children (10-max by default) - a.k.a. number of children (NOC).

The first version of the metamodel did not comply the features D03 and BP03. The MOF version of the metamodel has an abstract class, *InformationFlow*, that was a superclass of only one class, *Service*. In the Ecore version of the metamodel, in order to comply the feature D03, the *Property* class has been included in the hierarchy of *InformationFlow*. Furthermore, the *InformationFlow* class has been divided in two classes, one with the same name that represent the communication between two tools and another one named *CommunicationMechanism* to describe the software mechanism used to establish that communication, in case there was.

Regarding the feature BP03, there is a class in the metamodel in MOF, *Ecosystem*, that contains all classes except two, *Property* and *InformationFlow*. The Ecore version of the metamodel has two new composition associations, one between the root class and *InformationFlow*, and other between the root class and the new class *CommunicationMechanism*.

The learning ecosystem metamodel instantiated from Ecore (Fig. 2) fulfils with the thirty features that compose the framework. Highlight the metrics:

- M01. Maximum number of attributes in a class of the metamodel is 4.
- M02. The classes with more references to others are *InformationFlow*, *SoftwareTool* and *Ecosystem* with a Ce value of 3.
- M03. The classes more referred from others are *InformationFlow* with a Ca value of 4, and *SoftwareTool* and *Objective* with a Ca value of 3.
- M04. The deepest hierarchy has a DIT value of 4, where the root class is *Component*.
- M05. The class with more children is *Infrastructure* with a NOC value of 5.

5 Conclusions

The learning ecosystem metamodel is a M2-model in the four–layer metamodel architecture provided by MDA. The main objective of this metamodel is to provide a Computing Independent Model (CIM) for describing learning ecosystems building from software components, human elements and information flows between them.

The validation of the metamodel is necessary to provide a robust solution for the development of this type of technological solutions. In previous works a first phase has been carried out; two M2M transformations have been made to test that the metamodel allows to define real learning ecosystems. These preliminary validations have been made manually because there are no stable tools that support the standards defined by OMG.

The transformation from MOF to Ecore of the learning ecosystem metamodel represents an important step in the validation process because of the Ecore version can be an input in the different modelling tools provided by Eclipse. Furthermore, the metamodel instantiated from Ecore (Fig. 1) is a quality metamodel according to the quality framework proposed by López-Fernández, Guerra and de Lara [12].

In future works the validation process will be completed defining a set of transformation rules to transform a PIM model instantiated from the learning ecosystem metamodel to a PSM model that represent the deployment of the learning ecosystem in a real context.

Acknowledgments. This research work has been carried out within the University of Salamanca PhD Programme on Education in the Knowledge Society scope (http://knowledgesociety.usal.es) and was supported by the Spanish *Ministry of Education, Culture and Sport* under a FPU fellowship (FPU014/04783).
This work has been partially funded by the Spanish Government Ministry of Economy and Competitiveness throughout the DEFINES project (Ref. TIN2016-80172-R) and the Ministry of Education of the Junta de Castilla y León (Spain) throughout the T-CUIDA project (Ref. SA061P17).

References

1. García-Peñalvo, F.J., García-Holgado, A. (eds.): Open Source Solutions for Knowledge Management and Technological Ecosystems. IGI Global (2017)

2. García-Holgado, A., García-Peñalvo, F.J.: Architectural pattern to improve the definition and implementation of eLearning ecosystems. Science of Computer Programming 129, 20-34 (2016)
3. Alspaugh, T.A., Asuncion, H.U., Scacchi, W.: The Role of Software Licenses in Open Architecture Ecosystems. In: IWSECO@ ICSR. (Year)
4. García-Holgado, A., García-Peñalvo, F.J.: The evolution of the technological ecosystems: an architectural proposal to enhancing learning processes. Proceedings of the First International Conference on Technological Ecosystem for Enhancing Multiculturality (TEEM'13) (Salamanca, Spain, November 14-15, 2013), pp. 565-571. ACM, New York, NY, USA (2013)
5. Gómez-Sanz, J.J., Pavón, J.: Meta-modelling in agent oriented software engineering. In: Garrijo, F.J., Riquelme, J.-C., Toro, M. (eds.) Ibero-American Conference on Artificial Intelligence 2002, vol. 2527, pp. 606-615. Springer, Heidelberg (2002)
6. García-Holgado, A., García-Peñalvo, F.J.: A Metamodel Proposal for Developing Learning Ecosystems. In: Zaphiris, P., Ioannou, A. (eds.) Learning and Collaboration Technologies. Novel Learning Ecosystems. LCT 2017. Lecture Notes in Computer Science, vol. 10295. Springer, Cham (2017)
7. Frankel, D.: Model Driven Architecture: Applying MDA to Enterprise Computing. John Wiley & Sons, Inc. (2002)
8. García-Holgado, A., García-Peñalvo, F.J., Rodríguez-Conde, M.J.: Definition of a Technological Ecosystem for Scientific Knowledge Management in a PhD Programme. Proceedings of the Third International Conference on Technological Ecosystems for Enhancing Multiculturality (TEEM'15) (Porto, Portugal, October 7-9, 2015), pp. 695-700. ACM, New York, NY, USA (2015)
9. García-Holgado, A., García-Peñalvo, F.J.: Definición de ecosistemas de aprendizaje independientes de plataforma. IV Congreso Internacional sobre Aprendizaje, Innovación y Competitividad (CINAIC 2017) (Zaragoza, Spain, October 4-6, 2017), (2017)
10. García-Holgado, A., García-Peñalvo, F.J.: Preliminary validation of the metamodel for developing learning ecosystems. In: Dodero, J.M., Ibarra Sáiz, M.S., Ruiz Rube, I. (eds.) Proceedings of the 5th International Conference on Technological Ecosystems for Enhancing Multiculturality (TEEM 2017) (Cádiz, Spain, October 18-20, 2017). ACM, New York, NY, USA (2017)
11. Eclipse Foundation, http://download.eclipse.org/modeling/emf/emf/javadoc/2.11/org/eclipse/emf/ecore/package-summary.html
12. López-Fernández, J.J., Guerra, E., de Lara, J.: Assessing the Quality of Meta-models. In: Boulanger, F., Famelis, M., Ratiu, D. (eds.) MoDeVVa, vol. 1235, pp. 3-22. CEUR Workshop Proceedings, Valencia, Spain (2014)
13. Gerber, A., Raymond, K.: MOF to EMF: there and back again. In: Proceedings of the 2003 OOPSLA workshop on eclipse technology eXchange, pp. 60-64. ACM, (Year)
14. Chidamber, S.R., Kemerer, C.F.: A metrics suite for object oriented design. IEEE Transactions on Software Engineering 20, 476-493 (1994)