The ISC Int'l Journal of Information Security

August 2019, Volume 11, Number 3 (pp. 97–103)
http://www.isecure-journal.org

Selected Paper at the ICCMIT'19 in Vienna, Austria

Aspect Oriented UML to ECORE Model Transformation☆

Muhammad Ali Memon [1,*], Zaira Hassan [2], Kamran Dahri [1], Asadullah Shaikh [3], and Muhammad Ali Nizamani [1]

[1] *Institute of Information & Communication Technology, University of Sindh, Pakistan*
[2] *School Of Information Technology, Shaheed Benazir Bhutto University, Pakistan*
[3] *Department of Information Systems, Najran University, Saudi Arabia*

ARTICLE INFO.

Keywords:
Model-driven engineering, Unified Modeling Language, Kermeta, EMF ECORE, Model transformation.

Abstract

With the emerging concept of model transformation, information can be extracted from one or more source models to produce the target models. The conversion of these models can be done automatically with specific transformation languages. This conversion requires mapping between both models with the help of dynamic hash tables. Hash tables store reference links between the elements of the source and target model. Whenever there is a need to access the target element, we query the hash table. In contrast, this paper presents an approach by directly creating aspects in the source meta-model with traces. These traces hold references to target elements during the execution. Illustrating the idea of model driven engineering (MDE), This paper proposes a method that transforms UML class models to EMF ECORE model.

1 Introduction

Model driven engineering (MDE) is an approach that uses the concepts of software development models for defining the complex systems at different levels of abstractions. MDE maps one model to another model with the help of transformation traceability that defines associations between source and target models. A model may be traversed several times depending on the complexity of transformation. Information flow between the source and target model have to be maintained by an intermediate mechanism. This mechanism can be handled by dynamic hash tables. Hash tables have two columns, first column stores source objects and second column stores newly created target objects. The target model creates the new object whenever the entries apply on hash table. These entries helps to create associations and properties for target model to access a direct reference. Problem with this approach is to retain the hash table each time the target element is searched, so this is a time consuming process. In this paper, we present an aspect-oriented approach to get rid of retaining and searching the hash table each time. With the help of the aspect-oriented approach, we have created aspects of classes related to the source model elements and weave these aspects into source meta-model. These aspects introduce direct references which are typed to the corresponding class of the target meta-model. Traversing a source object, we can directly access target objects by accessing their references. The ben-

☆ The ICCMIT'19 program committee effort is highly acknowledged for reviewing this paper.
* Corresponding author.
Email addresses: muhammad.ali@usindh.edu.pk, zairahassan@sbbusba.edu.pk, kamran.dahri@usindh.edu.pk, asshaikh@nu.edu.sa, ma.nizamani@usindh.edu.pk
ISSN: 2008-2045

efit of this approach is directly accessing the aspects/attributes without searching for target objects. We illustrate this approach by using a transformation which take UML class diagram as input and creating the corresponding ECORE model. The transformation have been implemented into Kermeta language. This approach also avoid the searching mechanism of dynamic Hash tables. To manage this traceability, we have used aspects. Every input and output model element confirm its Meta class into Meta- model respectively.as for as the input model elements we have created the aspects into meta-model for Meta classes. The code shown below mentions the traces that have been injected into these aspects that shows the references to the corresponding meta-classes of the output model.

```
aspect class inputMetaClass
{
    reference trace: outputMetaClass
}
```

These aspects created with the input Meta-model when transformation is executed. Aspects provide the additional functionality of the traceability mechanism. When the output model expands through traversing the input model, the trace will hold the reference to the output element for each input element. In the later pass, to transform more information about input element properties and links, the output element will be accessed directly through this reference.

We have organized the rest of the paper as follows. Section 2 presents the literature review. Section 3 explains the Traversing information between input and output model. Section 4 discusses the transformation with mapping. Section 5 provides the algorithm based on this approach. Section 6 define some techniques applied on this work. Section 7 provides an example of ECORE application. Section 8 elaborate some limitations associated with this work, and the paper finally concludes in Section 9.

2 Related Works

In recent years, there are some computational science models that have been observed in the field of model driven engineering. Such as scientific simulation workflows, DSML and chemical markup language [1]. The model driven engineering also developed some rich ecosystem modeling techniques to reduce the complexity of software systems. The main cause of the complexity of the system is high-level and low-level abstractions provided by general-purpose programming languages. This gap is so costly in terms of time and effort. In model-driven engineering, aspects of systems are described by various models through different languages [2].

Lots of work has been done on different modeling languages that includes the behavioral semantics, static and abstract syntax [3]. It is difficult to combine different meta-languages for the design principles. Kermeta works well for the meta-languages as it permits us to gather various combinations of languages. It also handles the specific issues associated with Meta languages that cannot be properly solved by generic software engines [4]. Kermeta provides a mechanism through aspect keywords that can be compared with open class mechanism that is useful for Kermeta as it reuses the classes in meta models also use the powerful meta-modeling language EMOF to avoid the problems and annotations. A number of engineering tools provide the strong ecosystem around the EMOF platform that generates the java code to handle the meta-modeling problems.

Kermeta provides model-oriented and aspect-oriented capabilities: OCL-like lexical closures, native support of open-classes, model typing feature, and the ability to load and save EMF models. In recent years, there are some computational science models and frameworks that have been observed in the field of software development. One of the related work from [5] presented an extendable framework known as CrossECORE that uses OCL. The modeling framework CrossECORE supports Typescripts, C#, JavaScript and ECORE model with OCL. OCL provides all functions through API that can be used across various different target platforms. It also maximizes the use of traceability and supports portability across target platforms. [6] proposed a framework that is based on custom-built model transformation known as T-Core framework. The main purpose to use this framework is to analyze the reality of MTLS into the most primitive constructs. Another work presented by [7], they provide the concepts of a novel approach known as variability-based graph transformation. This approach was designed to improve the reusability and performance in Model-transformation systems because there are so many problems associated with large model transformation systems as they often use similar transformation rules. Variability-graph based transformation works well for this kind of problems. A work mention by [8], presents the meta-model packages to analyze the validation, transformation and comparison of UML models and also determine the behavior and structure of software system with a particular domain specific language. Further they describe that when transformation applies from UML to Ecore or MOF then the elements mapped in to Meta-meta- models. Because there is no direct association between elements of meta-meta

model and the elements of UML.

[9] proposed a mechanism that analyzed and execute UML state machines through DEVS.It provides a mechanism to bridge the gap between two official disputes with different technological bases and generally defines the model transformation that travers the elements from UML SCs to elements of DEVS model.

3 Traversing Information between Source/Input and Target/Output model

Model transformations are implemented by traversing input model for number of times. As Figure 1b shows the information traversing between input and output model. This traversing is often referred as a pass. First time when input model is traversed, it is called 1st pass, traversing second time is called 2nd pass and so on. Number of passes depends on the hierarchy and complexity of input model. In every pass partial information is acquired from input model and transformed in to the output model.

In Figure 1b, we have presented the intermediate mechanism known as traceability. This traceability is classically implemented by dynamic hash tables. Hash tables have two columns <Object1, Object2>. Object1 refers to the input model element whereas Object2 refers to the corresponding output model element. Hash table updates its entries whenever the information acquires by input elements, we have assumed that in the first pass the hash table fill up with entries for input model elements, then in the second pass each element of the input model is to be searched to acquire information and then link with other elements in the input model. This search can be time consuming depending on the size of input model.

4 Transformation

4.1 Mapping

For every translation first step is always to map elements of source model and target model. In translation we have mapped elements and relationships for both UML and ECORE meta- models. Table 1 shows some direct and some indirect complex mappings. These ECORE elements are created directly whenever UML elements are encountered in UML model. For the indirect mappings they are complex mappings to be handled. We have also encounter some problems associated with complex direct and indirect mappings.

4.1.1 Data types

Problem: It is difficult to handle data types because UML basic data types are dissimilar from EMF ECORE basic data types.
Solution: We have created our own set of data types for ECORE model and made a reference with a java data types. Reference to Java data types will make these data types compatible to Kermeta source files and Java code files.

Table 1. Complex mapping between elements.

ECORE Model	UML Model
EPackage	Package
Eclass	Class
EOperation	Operation
EParameter	Parameter
EAttribute	Property (Primitive Data types)
EEnum	Enumeration
EEnumLiteral	EnumerationLiterel

4.1.2 Association

Problem: No Direct mapping between both UML and ECORE meta-meta-model.
Solution: Association relationship create association properties in concerned classes of UML model. In ECORE model, EReference element in concerned Eclass with the same name as association property.

4.1.3 Multiplicities

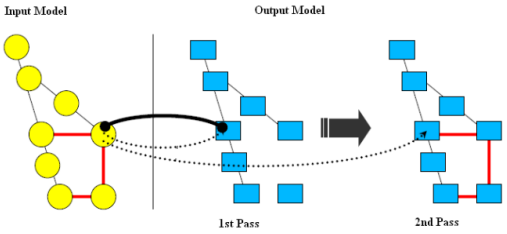
Problem: Multiplicities are handled as lower and upper bounds as properties in both UML and ECORE meta-meta-models and both properties in both meta-meta-models have different data types. These data types are fixed and cannot be changed.
Solution: local temporary variables of UML basic(Primitive) data types and type caste these variables in ECORE meta-meta-model.

4.1.4 Opposite Navigability

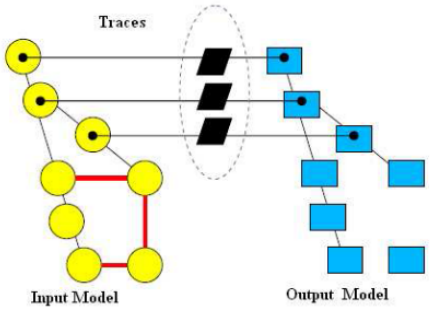
Problem: Opposite Navigability is not a new element to be added to the ECORE model, but it is bidirectional relationship in Association.
Solution: To handle the navigability, we set the EOpposite Property of EReference element in the EClass with the EReference element in the opposite EClass that share common Association.

5 Algorithm

ECORE elements (e.g., data types, classes, attributes, operations, and parameters) are created by traversing each sub package in the UML model. Traversing can be done in the four passes as described in Algorithm 1.



(a) Figure 1a



(b) Figure 1b

Figure 1. Here needs to be filled with a caption

Algorithm 1 Traversing Algorithm

1: In the first pass, main package and its sub packages are generated within main package.
2: In the second pass, data types, classes, enumerations and its enumeration literals are generated.
3: In the third pass, operations in the classes, parameters in the operations, attributes of the classes, generalization and association relationships are generated.
4: In the fourth pass, relationships between properties are generated for both side of navigability. The opposite [property] defines an association (bidirectional link) between two entities.

The Model traversed in four passes because we cannot create all the elements in one pass. Traversing in four passes prevents all different conflicts.

6 Techniques applied on this work.

In this section we have combined two approaches for implementing the translation. The first technique is visitor pattern that allows class hierarchy to add one or more behaviors. Typically these design patterns form a tree structure without modifying the classes. These classes use accept method to take the visitor as an argument, as shown in Figure 2. These design patterns enables the developers to discrete the base functionality from set of classes because they only use for specific situations and further they encapsulated into the visitor classes [10].

The second technique is aspect oriented that decompose the programs into aspects and classes. It is a software decomposition unit that implements transversal property into split classes of applications. So many problems can be solved with combined composition in aspects and classes like code scattering and code tangling problems [11]. This approach can also increase the modularity by enabling improved separation of concerns. Where each concern represents some different feature of the system and these concerns can be implemented independently with each other. In the translation aspects of each class for the UML generic Meta-model required for Class Diagram are created to introduce some new visitor methods for traversing. These aspects are restricted only to this translation and they will be used only in combination with UML generic classes whenever this translation is applied to any UML model or meta-model.

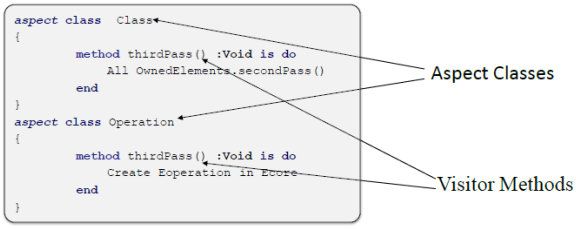


Figure 2. Visitor Pattern

7 Application

In this section we have presented the translation on UML model of a simple case study of University Management System as shown in Figure 3. This Model consists of five classes: `Student`, `University`, `Person`, `Instructor`, and `Course`.

Where class `University` have been put further in sub package `sub`. Class `Student` and `Instructor` have the generalization relationship with class `Person`, and associations with the `University` class. Associations between class `Student`, `Instructor`, `University` have bidirectional links. Class `Course` has a composition relationship with the class `Instructor` and directed association with `University` class.

Here we have mentioned the examples of first, second, third, and fourth passes. Figure 4 shows first pass where the main package and sub packages have been

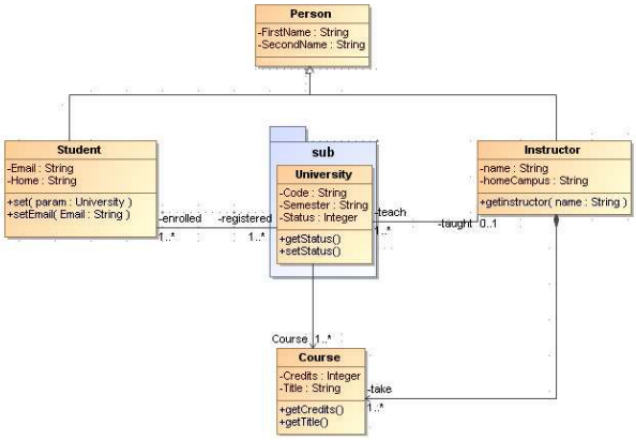


Figure 3. Scenario of university management system

created. Figure 5 mentions the second pass where data types, classes, enumerations and its enumeration literals are created. Figure 6 shows the third pass of UML model to ECORE model.

Figure 4. Pass1

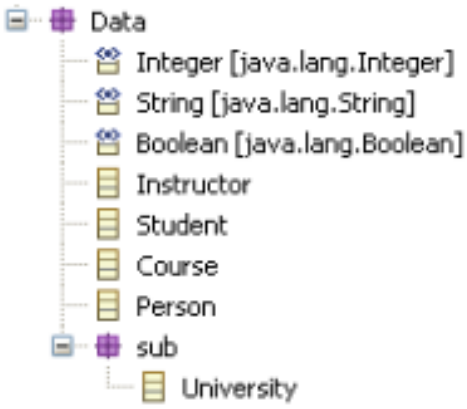


Figure 5. Pass2

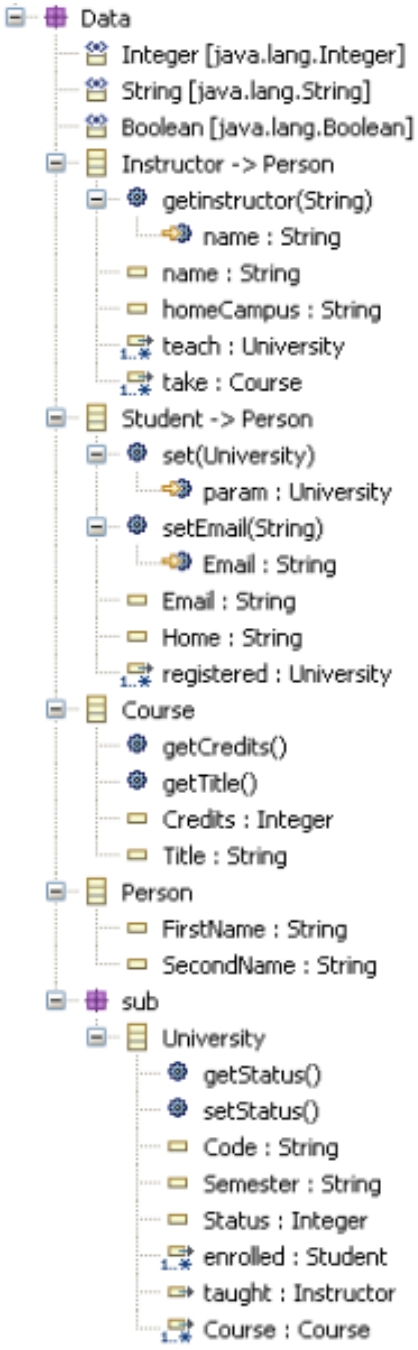


Figure 6. Pass3

Figure 7 shows the fourth pass of UML model to ECORE model. When we apply our fourth pass the EOpposite is set for respective opposite properties in `University`, `Student` and `Instructor` classes in ECORE. Opposite Property is not a new element to be added in ECORE, so ECORE model remains the same as shown in Figure 6. While Opposite property is set in the property window of ECORE model, for example opposite property in `Student` class is shown in Figure 5 for the bidirectional link between class `Student` and class `University`.

8 Issues

In this section we discuss two issues, limitations and complexities faced during implementing this translation.

8.1 Limitations

Here we mention some limitations of this translation:

(1) This translation does not support automated synchronization. When we make any change in

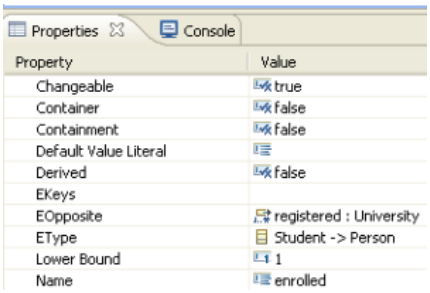


Figure 7. Pass3

UML model then these changes are not reflected automatically to ECORE model, but we have to apply the complete translation to acquire changed ECORE model.

(2) There is no reverse translation from ECORE model to UML model.

(3) ECORE meta-models has its own set of data types for example (EInt, EString) but we did not restrict our translation to ECORE data types. We created our own data types for which we maintain reference to java data types Integer, String and Boolean to make ECORE model compatible with java and Kermeta source files.

(4) Our translation does not cover all the aspects of the UML, but it is restricted only to one part of the UML .This restriction is due to the ECORE model that have compatibility only with the elements of Class Diagram. There are still some of the more concepts to be covered Class Association, Aggregation etc.

9 Conclusion

In this paper we have presented an aspect oriented approach for mediating information flow from input to output model transformation. We have created aspects in the source meta-model that hold direct references to the objects for target model objects. These references will be used to create relationships in target model objects in subsequent passes. Mediate Information classically maintained through dynamic hash tables. Meanwhile UCORE is a tool for translating UML models and meta-models designed in the form of class diagram to ECORE models and meta-models. Translation is implemented in Kermeta language and application of this translation is shown with an example. This translation will help us to easily transform our models designed in UML, from UML to EMF ECORE, which will again facilitate many features of EMF tools family, for example simulation of models ,applying OCL constraints, writing aspects or transforming to any other platform specific model . This tool is under testing for its stability verification and soon it will be available in the Kermeta MDK for community around Kermeta users. Future work includes removing limitations as mentioned above.

References

[1] Andreas Bender, Angela Poschlad, Stefan Bozic, and Ivan Kondov. A service-oriented framework for integration of domain-specific data models in scientific workflows. Procedia Computer Science, 18:1087 – 1096, 2013. 2013 International Conference on Computational Science.

[2] Jean-Michel Bruel, Benoit Combemale, Ileana Ober, and HÃľlÃĺne Raynal. Mde in practice for computational science. Procedia Computer Science, 51:660 – 669, 2015. International Conference On Computational Science, ICCS 2015.

[3] Benoît Combemale, Xavier Crégut, Pierre-Loïc Garoche, and Xavier Thirioux. Essay on semantics definition in mde - an instrumented approach for model verification. Journal of Software, 4:943–958, 11 2009.

[4] Jean-Marc JÃľzÃľquel, Benoit Combemale, Olivier Barais, Martin Monperrus, and FranÃğois Fouquet. Mashup of meta-languages and its implementation in the kermeta language workbench. Software & Systems Modeling, 14, 06 2013.

[5] S. Schwichtenberg, I. Jovanovikj, C. Gerth, and G. Engels. Poster: Crossecore: An extendible framework to use ecore and ocl across platforms. In 2018 IEEE/ACM 40th International Conference on Software Engineering: Companion (ICSE-Companion), pages 292–293, May 2018.

[6] Eugene Syriani, Hans Vangheluwe, and Brian LaShomb. T-core: a framework for custom-built model transformation engines. Software & Systems Modeling, 14(3):1215–1243, Jul 2015.

[7] Daniel Strüber, Julia Rubin, Marsha Chechik, and Gabriele Taentzer. A variability-based approach to reusable and efficient model transformations. In Alexander Egyed and Ina Schaefer, editors, Fundamental Approaches to Software Engineering, pages 283–298, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

[8] S. JÃďger, R. Maschotta, T. Jungebloud, A. Wichmann, and A. Zimmermann. An emf-like uml generator for c++. In 2016 4th International Conference on Model-Driven Engineering and Software Development (MODELSWARD), pages 309–316, Feb 2016.

[9] Ariel Gonzalez, Carlos Luna, Roque Cuello, Marcela PÃľrez, and Marcela Daniele. Towards an automatic model transformation mechanism from uml state machines to devs models. CLEI electronic journal, 18:3:1–3:27, 08 2015.

[10] Matthias Springer, Hidehiko Masuhara, and Robert Hirschfeld. Classes as layers: Rewriting design patterns with cop: Alternative implementations of decorator, observer, and visitor. pages 21–26, 07 2016.

[11] Ouafa Hachani and Daniel Bardou. Using aspect-oriented programming for design patterns implementation. In Reuse in Object-Oriented Information Systems Design., 07 2008.

Muhammad Ali Memon is currently serving University of Sindh in IICT(Institute of Information & Communication Technology), Jamshoro, Pakistan as an Associate Professor. He finished his PhD from National institute of Polytechnic Toulouse (INPT), University of Toulouse, France with specialization in Information Systems. His research interests are Software interoperability, Transportation scheduling systems, Ontologies, Information systems, Enterprise Engineering, ,Supply chain solutions.

Zaira Hassan is currently working as a lecturer in the Department of Information Technology, Shaheed Benazir Bhutto University, Shaheed Benazirabad, Pakistan. She has done BS in IT from University of Sindh. Her Master's work based on social network and her research interests also focuses on natural User experience (NUX/NUI), artificial intelligence and data security.

Kamran Dahri is currently working as an Assistant Professor in the Institute of Information & Communication Technology, University of Sindh, Jamshoro, Pakistan. Currently, he is pursuing his PhD in Information Systems, with focus on use of the Blockchain technology for healthcare systems. His area of research interest include data clustering, data minning and blockchain technologies.

Asadullah Shaikh is PhD in software engineering from University of Southern Denmark. Dr. Shaikh is currently working as an Associate Professor and head of research in the college of Computer Science and Information Systems Najran University, Najran, Saudi Arabia. His current research interests are: UML Model Verification, UML Class Diagrams Verification with OCL Constraints for Complex Models, formal Verification, feedback Technique for Unsatisfiable UML/OCL Class Diagrams and Networks.

Muhammad Ali Nizamani is PhD in Artificial Intelligence from INSA Toulouse, France. He is currently working as an Assistant Professor in the Institute of Information & Communication Technology, University of Sindh, Jamshoro, Pakistan. His area of research interests are: Artificial Intelligence, Human Computer Interaction, Human Robot Interaction, Intelligent Systems, Semantic Software Engineering, Knowledge Integration.