Overview in the eclipse model-driven architecture tools

Hamza Natek[1*], *Abdelali* Elmounadi[2], and *Fatima* Guerouate[1]

[1]Mohammed V University in Rabat, Higher School of Technologies of Salé, LASTIMI Laboratory
[2]Mohammed V University in Rabat, École Normale Supérieure of Rabat

Abstract. With the evolution of software engineering and the need to create multiplatform applications has increased in our daily life, but the fact to have a custom code for each platform is costly for companies, having a solution to save development time has become primordial, hence it comes to the model-driven architecture approach. This approach provides the ability to keep the business logic which is represented as the model and apply the transformation to generate these applications. Implementing and applying this transformation can be done by using some tools that can help generate code from models or do reverse engineering to go from code to models. This paper aims to shade light to the MDA approach and the used tools to implement the concepts of this approach to help the readers to save time and well understand the approach.

1 Introduction

With the increase in the need for software creation and the evolution of the software engineering field, finding a solution to generate fast and operational artifacts has become essential to save time and have an efficient result.
Model-Driven Engineering is an approach applied to software engineering that places the notion of models in the foreground. This approach implies a remarkable gain in productivity by maximizing compatibility between systems and simplifying the design process while facilitating communication between the development teams. Several variants of MDE exist, for example, Microsoft's software factories or OMG's MDA. This latter has succeeded to become a reference in the MDE field, therefore we will be concentrating on it in this work.
Therefore, in order to be able to implement the MDA concepts, it is necessary to employ a set of tools that allows us to carry out the required treatments throughout the model transformation lifecycle. The Eclipse platform integrates several tools that are used in the MDA context.
For this purpose, we decided to provide an overview of the several tools available in the Eclipse platform, in order to guide future researchers in the MDA field when selecting the appropriate tools to use in their research work.
This paper is organized as follows: in the second section, we explain the MDA approach and its relationship with the other OMG's standardization projects (OMG, Object Management Group). In the third section, we discuss the several types of model transformations, and then we discuss the several transformation approaches in the fourth section. In the fifth section, we present a set of existing model transformation tools that are widely used in the field of model transformations. The set of tools that have been chosen is specially Eclipse-based. Finally, the sixth section presents the conclusion and future works.

2 Model-driven architecture

The MDA approach was publicly presented by the OMG (Object Management Group, http://www.omg.org/) in November 2000, it is an extension of the MDE, combining all the standards proposed by the OMG. MDA is an approach used to build software from models, this approach has been proposed and supported by the OMG (Object Management Group).

The main goal of the MDA approach is to generate software artifacts from models by going through specific steps and using transformation languages dedicated to this subject, this helps to validate each step of transformation before moving to another, it allows to:
- Maintain the business requirements
- Reuse architecture and coding choices
- Ensure integrity and consistency between project phases.

A model is an abstraction of reality, it allows to represent a system. A model must conform to another model, hence the notion of the meta-model. Therefore, each model at a given level of abstraction must be described by a model at a higher level of abstraction, which would lead to an infinite number of hierarchical levels. To support this approach, the OMG proposes the MOF standard [2] (Meta-Object Facility) which allows defining the representation of meta-models and their manipulation. MOF has the particularity of being

* Corresponding author: hamzanatek@gmail.com

described reflexively. Therefore, the model-driven architecture has 4 layers of meta-modeling:

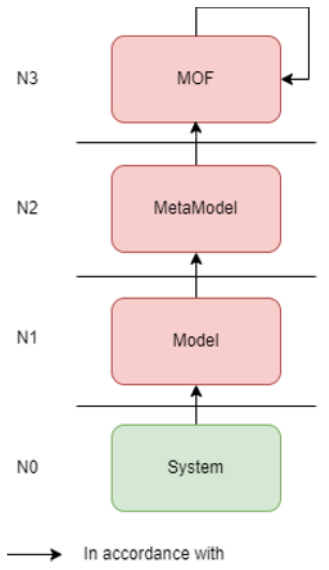

Fig. 1. The 4-level architecture of MDA [1].

MOF is then described as a "meta-meta-model".

Since the main objective of the MDA approach is the separation of business concerns from technical constraints, it is essentially based on the following two models:

- **PIM (Platform Independent Model):** The PIM is a model that describes the design, this model is independent of any technology used. This model presents the business logic in a UML (Unified Modeling Language) diagram.
- **PSM (Platform Specific Model):** The PSM is a model that respects the technical constraints of the target platform.

Also, we should mention the **CIM (Computational Independent Model)** which is a model that describes the functional requirements of the application as well as the situation in which the system is used. This model is rarely changed, it is only modified if the business or application domain needs change.

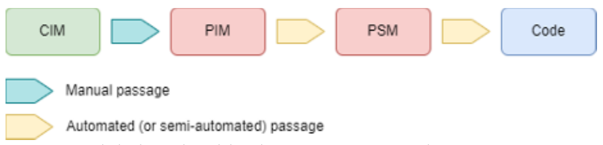

Fig. 2. Models involved in the MDA approach [1]

Model transformation in an MDA framework consists of transforming PIM models into PSM models. It is performed thanks to a transformation engine that applies a set of rules to the PIM provided as input to produce the PSM as output.

The notion of meta-model is omnipresent in this case, insofar as each model (PIM or PSM) relies on a meta-model which is used to describe it. When both source and target models are based on the same meta-model, we talk about an "endogenous" transformation, otherwise, we talk about an "exogenous" transformation [3].

Table 1. Orthogonal dimensions of model transformations with examples [3].

	Horizontal	**Vertical**
Endogenous	Refactoring	Formal refinement
Exogenous	Language migration	Code generation

MDA is basically based on a set of standards proposed by the OMG, like MOF. These standards have evolved to be mutually enriching:

- **UML (Unified Modeling Language):** UML [4] [5] is a graphical modeling language that is widely used by systems based on object-oriented programming. It is a language based on the MOF specifications established by the OMG. UML provides a set of diagrams to model and designs the systems to be developed.
- **QVT (Query/View/Transformation):** [6] It is a language standardized by the OMG to allow model transformation. It proposes a formal framework to describe the transformation rules that will be applied to the models in question.

QVT is the technological backbone of model transformation and its syntax is based on the OCL (Object Constraint Language) standard.

- **XMI (XML Metadata Interchange):** [1] It is a format for the exchange of metadata information UML based on XML. It has also been proposed by the OMG as a standard and represents a method of serialization process for MOF objects that allows objects to be described in XML format.

3 Model transformations

In figure 2 (fig.2) above, we can observe that the passage from the PIM to the PSM is governed by an automated (or semi-automated) transformation, we also distinguish an automated (or semi-automated) passage from the PSM to the code. We can then detect two types of model transformations:

- **M2M transformations (Model to Model):** These transformations concern all the tasks to be carried out in order to obtain a model that respects the technical specifications of the target environment from another model. The technological base that technically represents this type of transformation in the MDA approach is the MOF 2.0 QVT standard [1].
- **M2T transformations (Model to Text):** these transformations are aimed at generating code or documentation. M2T transformations are the subject of the MOFM2T project which is one of the parts of the MDA project [15].

It should also be noted that these model transformations are characterized by certain properties [16] including:

• **Reversibility:** a model transformation is reversible if there is a transformation that allows it to go in the opposite direction.

• **Reusability:** consists of the ability to extend existing transformations in order to produce new model transformations. The modularization aspect of the transformation rules plays an important role at this level.

• **Rule ordering:** this aspect determines the order of execution of the transformation rules, in fact, this impacts the representation of the nesting levels of the latter.

• **Traceability:** specifies the way in which the stages of a model transformation are logged and the links allowing the correspondence of the elements of the source model to the elements of the target model are followed.

4 Model transformation tools

In this section, we list a set of tools that allow the implementation of the various model transformation approaches. Generally, the model transformation tools adopting the MDA approach are organized into two categories: commercial tools and open-source tools. In this study, we focus on describing open-source projects-based tools that are the most widespread.

4.1 ATL (Atlas Transformation Language):

ATL [7] [8] [9] is designed to express model transformations as described by the hybrid approach in an MDA framework. It is the result of research by the Atlas INRIA and the LINA research group in response to the OMG MOF/QVT Request for Proposal.

ATL is specified both as a meta-model and as a concrete textual syntax that relies on the syntax of the OCL (Object Constraint Language) formalism for the expression of transformation rules. It is a hybrid language adopting the declarative and imperative approach.

Declarative style is recommended for writing the transformation. Imperative constructs are provided to further specify certain transformations that are too complex to be handled in a declarative manner alone. The ATL transformation program consists of a set of rules that define how the elements of a source model will match the elements of the target model. The implementation of ATL-based transformations is provided through a number of standard development tools integrated into the Eclipse IDE [1].

4.2 Acceleo:

Acceleo is a tool written in Java, developed by the «Obeo» company, and deployed on the Eclipse platform. It is an implementation of the MOFM2T standard for transforming models into text. It adopts the Templates approach. Acceleo is considered as a very powerful tool in view of the richness of the features it offers. Indeed, it does not only offer an editor for editing code, but it consists of a debugger for monitoring code generation too, a profiler for quantifying the transformation of the model in terms of execution time, and finally, a tracer that allows to trace the elements that are involved in the performed transformations [10]. The expressions of the Acceleo language are inspired by the OCL standard for navigating between model elements and extracting information from them.

4.3 Modisco:

MoDisco is an Eclipse Generative Modeling Tool (GMT), which provides an extensible and customizable MDRE framework to develop model-driven tools supporting different model driven reverse engineering scenarios such as legacy migration or modernization, quality assurance, redocumentation, etc. The main purpose of MoDisco is to offer an open source, generic and extensible MDRE framework. Considering as inputs miscellaneous legacy artifacts (source code, databases, configuration files, documentation, etc.), MoDisco aims to provide the required functionalities for creating models and allowing their handling, analysis and computation. Afterwards, the framework targets the production of different types of artifacts as outputs, depending on the selected MDRE objectives (source code, data, metrics, documentation, etc.). Furthermore, MoDisco is an Eclipse-based project that provides and uses concrete implementations of three OMG standard meta-models: KDM [18], SMM [19] and ASTM [20].

4.4 Eclipse QVTo:

The QVT operational mapping language is an imperative language to define transformations. It extends OCL but includes all the necessary machinery that is needed to write in a comfortable way complex transformations. The imperative expressions in QVT realizes a compromise between functional features in OCL and the more traditional constructs that we found in general purpose languages like Java [24]. The Eclipse QVT Operational component is an implementation of the Operational Mappings Language defined by Meta Object Facility™ (MOF™) 2.0 Query/View/Transformation (QVT). It aims to provide a complete implementation of the operational part of the standard.

4.5 JET [15]:

JET is typically used in the implementation of a "code generator". A code-generator is an important component of Model Driven Development (MDD). The goal of MDD is to describe a software system using abstract models (such as EMF/ECORE models or UML models), and then refine and transform these models into code. Although it is possible to create abstract models, and manually transform them into code, the real power of MDD comes from automating this process. Such transformations accelerate the MDD process, and result in better code quality. The transformations can capture the "best practices" of experts, and can ensure that a project consistently employs these practices.

However, transformations are not always perfect. Best practices are often dependent on context - what is optimal in one context may be suboptimal in another. Transformations can address this issue by including some mechanism for end-user modification of the code generator. This is frequently done by using "templates" to create artifacts, and allowing users to substitute their own implementations of these templates if necessary. This is the role of JET.
This component provides:

1. Expand the JET language to support custom tags (which are distributed in "tag libraries"). (The language specification will be described in a separate document.)
2. Define Java interfaces and Eclipse Extension points for declaring custom tag libraries.
3. Provide Standard JET tag libraries that make it possible to create entire transformations without recourse to Java and the Eclipse APIs. (These tag libraries will be described in a separate document.)
4. Provide Eclipse API and UI for invoking such transformations.
5. Provide a JET template editor.

4.6 Papyrus (GMF) [17]:

Eclipse Papyrus is a graphical editing tool for UML 2 as defined by OMG. Eclipse Papyrus targets to implement 100% of the OMG specification.
Eclipse Papyrus provides editors for all the UML diagrams:

- Class Diagram
- Object Diagram
- Package Diagram
- Composite Structure Diagram
- Component Diagram
- Deployment Diagram
- Profile Diagram
- Use case Diagram
- Activity Diagram
- State machine Diagram
- Communication Diagram
- Sequence Diagram
- Timing Diagram
- Interaction overview Diagramquations and mathematics

5 Related works

In [11], Esbai & al. show how to design and apply transformation rules to migrate from an SQL relational database to a Big Data solution within NoSQL. in order to achieve this, the authors used the Model Driven Architecture (MDA) and the transformation languages like MOF 2.0 QVT (Meta-Object Facility 2.0 Query-View-Transformation) and Acceleo which define the meta-models for the development of the transformation model. The transformation rules defined in this work can generate, from the class diagram, a CQL code for the creation column-oriented NoSQL database.
In [12], Elmounadi & al. propose a new model discovery tool intended for PHP language as an extension for the Modisco framework that allows managing the application's assets written in PHP language. The proposed work aims to enhance the Modisco platform capabilities in managing more software development technologies.
Another work has been provided in [13] where the authors developed two meta-models, one for UML and another for CodeIgniter. The authors proposed an algorithm to generate the CodeIgniter model, which has been expressed in QVTo language.
In [14], Srai & al. aims to respond to the problem of generating NoSQL MongoDB databases by applying an approach based on model-driven engineering (Model Driven Architecture Approach). The authors provide Model to Model (using the QVT model transformation language), and Model to Code transformations using Acceleo. They also propose vertical and horizontal transformations to demonstrate the validity of their approach on NoSQL MongoDB databases. They have studied in this article the PSM transformations towards the implementation. And as a perspective, they mentioned that PIM to PSM transformations is the subject of another work.
Another work related to the same topic [21] where the authors present a methodology based on the Model Driven Architecture (MDA) to develop mobile applications according to the principle "develops once, use everywhere". The proposed approach exploits UML modeling and Acceleo to generate specific code in order to accelerate and facilitate the development of mobile applications.
In [22], The authors propose a Model-Driven Architecture (MDA) approach for the development of data warehouses independently of any execution platform, to allow the facilitation of the development of these data warehouses as well as the migration of information systems based on relational DBMS to systems NoSQL.
In [23], the authors propose an application of ADM principles to provide relevant model-based views on legacy systems. They describe a framework to reverse engineering models from object-oriented code. The authors show how to recover UML sequence diagrams from Java code.

From the above presented works, we may conclude that recent research work related to MDA mainly uses the QVTo tool and Acceleo in the framework of generative engineering, ATL remains a very powerful tool too.
Regarding reverse engineering, we may observe that Modisco is the only tool able to offer possibilities, by providing a formal framework for moving from code to models. Finally, we may observe that the Papyrus or JET tools are generally not used.

6 Conclusion & perspective

From the researchers related to this work, we can conclude that most of these researches use the QVTo tool and Acceleo in the framework of generative engineering and the ATL remains a very powerful tool as well, Modisco is the only used tool to do the reverse engineering transformations, and we notice some tools that generally not used as Papyrus or JET.

As a perspective, currently, we are working on a way to generate NoSQL databases starting from a UML diagram using M2M/M2T transformations which will be the subject of our next work.

References

1. A. Elmounadi, n. Berbiche, n. Sefiani Architecture dirigée par les modèles : Méthodes et outils de transformation de modèles
2. Q. Omg, “Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification,” Final Adopt. Specif. (November 2005), no. January, 2008
3. A. Srai, F. Guerouate, H. Lahsini ‘A generation of a Multi-Layered Application by Applying the MDA Approach for Online Learning Platforms’ International Journal of Innovative Technology and Exploring Engineering 10(3):2278-3075 DOI:10.35940/ijitee.C8408.0110321
4. O. M. G. D. Number and I. A. Files, “OMG Unified Modeling Language TM (OMG UML), Infrastructure,” no. February, 2009.
5. O. M. G. D. Number, S. Associated, and N. M. Files, “OMG Unified Modeling
6. A. Abdullah and A. G. Downe, “A Proposed Compiler to Integrate Model Driven Architecture with Web Services – Road Map,” Int. J. Comput. Appl., vol. 15, no. 7, pp. 1–7, 2011.
7. N. Cuong and X. Qafmolla, “Model transformation in web engineering and automated model driven development,” Int. J. Model. Optim., vol. 1, no. 1, 2011.
8. S. Diaw, R. Lbath, and B. Coulette, “Etat de l’art sur le développement logiciel basé sur les transformations de modèles,” 2009. DOI:10.3166/tsi.29.505-536
9. M. Rahmouni and S. Mbarki, “MDA- BASED ATL TRANSFORMATION TO GENERATE MVC 2 WEB MODELS,” Int. J. Comput. Sci. Inf. Technol., vol. 3, no. 4, 2011.
10. Acceleo http://www.eclipse.org/acceleo/.
11. R. Esbai, F. Elotmani, F. Belkadi ‘Toward Automatic Generation of Column-Oriented NoSQL Databases in Big Data Context’ International Journal of Online and Biomedical Engineering (iJOE) 15(09):4 DOI:10.3991/ijoe.v15i09.10433
12. A. Elmounadi, N. El Moukhi, N. Berbiche, N. Sefiani ‘A New PHP Discoverer for Modisco’ International Journal of Advanced Computer Science and Applications 10(1) DOI:10.14569/IJACSA.2019.0100122
13. A. Srai, F. Guerouate, N. Berbiche, H. Drissi ‘MDA approach for CodeIgniter PHP Framework’ The 2 nd Scientific Day on Computer Science, Optimization and Systems’ Modelization (CSOSM’17), March 9, 2017-Faculty of science, Kenitra.
14. A. Srai, F. Guerouate, N. Berbiche ‘The Integration of the MDA Approach in Document-Oriented NoSQL Databases, the case of MongoDB’ International Journal of Engineering and Advanced Technology 10(3):115-122 DOI:10.35940/ijeat.C2235.0210321
15. “Model To Text.” http://www.eclipse.org/modeling/m2t/.
16. M. Dehayni and K. Barbar, “Some Model Transformation Approaches: a Qualitative Critical Review,” J. Appl. …, vol. 5, no. 11, pp. 1957–1965, 2009.
17. “Eclipse Papyrus - Modeling environment” https://www.eclipse.org/papyrus/.
18. “KDM - Knowledge Discovery Metamodel” https://www.omg.org/spec/KDM/1.4/About-KDM/
19. “SMM - Structured Metrics Metamodel” https://www.omg.org/spec/SMM/1.2/About-SMM/
20. “ASTM - Abstract Syntax Tree Metamodel” https://www.omg.org/spec/ASTM/1.0/About-ASTM/
21. H. Benouda, R. ESSBAI, M. Azizi, M. Moussaoui ‘Modeling and Code Generation of Android Applications Using Acceleo’ International Journal of Software Engineering and its Applications 10(3):83-94 DOI:10.14257/ijseia.2016.10.3.08
22. A. Srai, F. Guerouate, N. Berbiche, H. Drissi ‘An MDA approach for the development of data warehouses from Relational Databases Using ATL Transformation Language’ International Journal of Applied Engineering Research 12(12):3532-3538
23. Martinez, L.; Pereira, C. and Favre, L. (2014). Recovering Sequence Diagrams from Object-oriented Code - An ADM Approach. In Proceedings of the 9th International Conference on Evaluation of Novel Approaches to Software Engineering - ENASE, ISBN 978-989-758-030-7; ISSN 2184-4895, pages 188-195. DOI: 10.5220/0004894201880195
24. OMG, Meta Object Facility (MOF) 2.0 Query/View/Transformation, V1.1, 2011.