Universal Journal of Computer Science and Engineering Technology
1 (1), 50-54, Oct. 2010.
© 2010 UniCSE, ISSN: 2219-2158

Model Transformations in Model Driven Architecture

Atif Aftab Ahmed Jilani
FAST-National University of Computer and Emerging Sciences,
Islamabad, Pakistan
atif.jilani@nu.edu.pk

Muhammad Usman
Mohammad Ali Jinnah University,
Islamabad, Pakistan
m_usman_99@hotmail.com

Zahid Halim
FAST-National University of Computer and Emerging Sciences,
Islamabad, Pakistan
zahid.halim@nu.edu.pk

Abstract—Transformation is one of the prominent features and the rising research area of Model Driven Architecture since last few years. There are many techniques which have been proposed as a Request for Proposal (RFP) in Query, View and Transformation (QVT). In this paper we have conducted a survey on transformation techniques. The surveyed techniques include pattern based approaches, transformation languages, transformation rules, Metamodel based approaches etc. This work has summarize, categorized and identified different analysis parameters of these techniques. On the basis of identified parameters we have presented an analysis matrix to describe the strength of different approaches. The major focus of the work is on model to model transformation techniques i.e. from PIM to PSM transformation.

Keywords- Model Driven Architecture; Transformation and analysis matrix

I. INTRODUCTION

Model transformations are increasingly gaining attention in software design and development. Model transformation plays a key role in Object Management group (OMG) Model Driven Architecture (MDA) initiative [1]. Transformation requires specialized support to fully present its prospective support from the end users as well as from the transformation developers. Different proposals have been issued with the combination of OMG Request for Proposal (RFP) for Query, View and Transformation (QVT) [5]. This paper briefly describes these techniques and performs analysis on them. Though the major emphasis of this paper is on model (PIM) to model (PSM) transformation, at the same time it also covers model to code transformation.

The paper is organized as follows: section II of this paper describes briefly the background of Model Driven Architecture (MDA), importance of model transformation in MDA and its current evolution. Section III outlines the architectural categorization of Transformations. Section IV covers different transformation techniques. Section V illustrates analysis on presented techniques. Conclusion and future work are presented in Section VI.

II. BACKGROUND

Due to rapid evolution of different technologies and platforms like distributed objects, components and web services etc. It has become very difficult for the developers to choose which platform is considered the best and which technology will be obsolete tomorrow and how to save investment on business logic. Model Driven Architecture (MDA) answers these entire queries. It raises a slogan *"Design once and build it on many"*.

OMG [1] defined model driven architecture (MDA) in 2000. MDA changes the software development style. It describes each artifact as a model. In MDA a model is the formal description of the system in well defined language. It shifts the developers from code oriented to model oriented approach. MDA separates the specification of system functionality from the specification of implementation. One specification of system functionality can be transformed into many specifications of implementation i.e. one specification model can be mapped on many implementation models. The artifact in MDA might be a "living document" or executable model.

MDA software development consists of three major types of model. Computation Independent Model (CIM) it involves specification of system functionality, platform Independent model (PIM) and platform specific model (PSM) [2]. Major focus of MDA is on PIM and its automated mapping on one or more PSM. Figure 1 shows general MDA Structure.

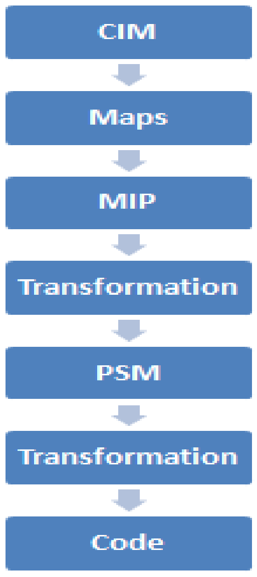

Figure 1: Major Structure of MDA

Computation independent Model (CIM) is mapped on Platform Independent Model (PIM). Transformation in bigger block describes the transformation of PSM from PIM. The second comparatively smaller transformation block is responsible for generating code from PSM.

Unified Modeling Language (UML) [3] and Meta Object Facility (MOF) [4], another standard of OMG, are usually used to express PIM and PSM. The approaches proposed in OMG Request for Proposal (RFP) in Query/View/Transformation (QVT) [5] contributed a lot in identifying and categorizing transformation techniques.

III. ARCHITECTURAL CATEGORIZATION OF TRANSFORMATION

We can categorize transformation into three major types as proposed by S. Sendall and W. Kozaczynski [6]. These categorizations are:

Direct Manipulation: In direct manipulation the internal model representation is available and user can manipulate the representation using any traditional Application Programming Interface (API). The API in direct manipulation is usually a general purpose language API is like Visual Basic or Java. The usage of general API will give ease to the developers, but due to the API's limited ability the API restricts the some kinds of transformations. Direct Manipulation lacks high level of abstraction due to the use of general purpose language API. Tools like Rational Rose and Rational XDE follow the same technique.

Intermediate representation: In this category model is exported to into a standard form usually Extensible Markup Language (XML) or XML Metadata Interchange XMI. This approach is difficult to perform when changes are frequent as this does not support cross model synchronization. Intermediate representation is not considered user friendly as compare to direct manipulation approach as direct manipulation approach use general purpose language API. Many UML tools are available that export and import model to and fro from XML or XMI.

Language Support: This approach emphasize that instead of using general purpose API, a language specific to transformation nature should be used or formed. Language provides all the necessary constructs, properties and rules for expressing, composing and applying transformation. The language can either be declarative, procedural or both depend on the nature of transformation.

IV. TECHNIQUES AND APPROACHES

This section describes transformation approaches and techniques that exist in literature or proposed by different authors. Brief introductions about different transformation techniques are stated below.

A. Model to Code Transformation:

Model to code transformation technique generates code from PSM. The commonly used model to code transformation techniques include:

- VB: Visitor Based Approach

K. Czarnecki and S. Helsen [7] describe visitor based approach. In this approach source model is traversed and code is generated for each model element. Meta-Concept defines which model is traversed first and which is to be traversed next. Example of this approach is Jamda tool that generates java code. Jamda [8] is based on object oriented framework. It is an open source OMG compliant tool.

- TB: Template Based Approach

Template based approach uses template to generate code. A template consists of rules which are mapped on source model. As compare to visitor base code generation, template is more specific for code generation. The structure of template resembles more closely to the generated code results in more accurate and precise code generation. This technique is quite famous and many famous tool used template based code generation. Tools include JET [9], AndroMDA [10], OptimalJ [11] etc.

B. Model to Model Transformation:

Model to Model transformation includes PIM to PSM transformation and mapping of CIM to PIM. Model to model transformation i.e. PIM to PSM includes.

- CWM: Common Warehouse Metamodel Transformation

CWM [12] is defined by OMG. CWM introduces the concept of white box and black box transformation. Black Box transformation only provides the relationship between source and target model element but does not identify what the resulting target will consist of. White Box transformation is associated with procedure expression which allows the language to describe the implementation. Due to limited capability of CWM, it is not very popular in model to model transformation.

- GT: Graph Transformation

Graph Transformation due to easy use is the most admired and famous technique for conducting transformation. The transformation consists of L.H.S and R.H.S rule which is to be mapped on source and target model. A L.H.S consists of subgraph and condition which is to be searched in source model and replaced by the graph in R.H.S. It is a non deterministic technique for rule selection and for applying the rule. Graph transformation, due to its nature of in-place, does not support state full transformation. Tool is responsible for state management of model in graph transformation. The common available tools and languages that follow this technique are Graph Rewriting and Transformation Language (GREAT) [13], Bidirectional Object Transformation Language (BOTL) [14], Visual Language for defining transformation VIATRA [15] etc.

- Relational Approach

Relational approach is a declarative approach, which mainly focuses on mathematical relations and on source and target model relationship [16]. Relationships are composed of complex mathematics. Predicates and constraints are used to define mathematical relationships. Due to mathematical foundation Relational approach is bidirectional and also supports backtracking. As compare to Graph transformation it does not allow in-place transformation.

- XSLT: Extensible Style Sheet Language Transformation

XSLT is inspired by intermediate model representation technique which has been discussed earlier. This technique uses XML document and specifically XMI to perform transformation [17]. It operates on textual representation of models. Rules are used to manipulate the textual document. Similarity between graph and XML structure helps to define transformation in XSLT. To manipulate XSLT is not simple. Frequent changes and cross model synchronization is not admired in XSLT.

- Structural Approach

This approach is based on the assumption that source and target model have similar structure. By assuming this it generate target element for each source element, it creates a hierarchal structure for target model, sets attribute and references in the target model. User has to provide rule and schedule for transformation. The OptimalJ [11] is the common example that also supports this approach.

- QVT: Query, View and Transformation Approach

This approach follows Hybrid approach as describe in QVT in 2002 [5]. QVT approach promotes combination of declarative, structure and imperative techniques. According to OMG transformation language recommendation [18], a transformation language should follow declarative relational approach to specify transformation and use imperative mapping patterns for implementation.

- Other Approaches

There exist other techniques as well that are proposed by different authors. These approaches use the combination of above mentioned approaches:

- DUPA: A declarative reusable pattern Approach

K. Duddy et al proposed a language [19] that based on the requirement of request for proposal (RFP). They purposed a declarative pattern based approach. The language support patterns and tracking. They also proposed that the pattern used in the language must be reusable.

- MDAMT: A model driven approach to model transformation

B. Appukuttan et al proposed a approach [20] according to them a framework must exists to express models. They define transformation as a super class of relation and mapping .The approach based on Metamodel of class diagram and Metamodel of XML element. They define relationship class Metamodel and XML Metamodel and generate XML Code. The technique is simple but supports only class diagram to XML code transformation.

- MT: The MT model transformation language

L. Tratt proposed MT model transformation language [21]. MT is a unidirectional technique. MT focuses on defining Domain Specific Language which works together with patterns (rules). A source model is passed to a function (which consists of rules). This function determines either the transformations should carry or not if it carry the target source is generated.

- MAMT: A Metamodeling Approach to Model Transformation

S. R. Judson et al proposed an approach [22].The approach emphasizes that source model is the instance of Source Pattern and target model is the instance of target pattern and there exists a transformation pattern which is responsible for carrying transformation from source model to target model.

- MOLA: Basic of Model Transformation Language MOLA

A. Kalnins, et al proposed an easy readable graphical transformation language [23]. The language built on Metamodel and patterns. Pattern consists of rule to create target model. Other languages like GREAT use recursive calls as a central structure but MOLA incorporate loops.

- BOTL: Model transformation for the MDA with BOTL

Marschall and Braun proposed BOTL [14] .MOLA, GReAT have no in-built mechanism to validate their target model. They did not guarantee that invalid model cannot generate in transformation. BOTL is bidirectional language build on object oriented paradigm has overcome the above mentioned problem. It generates valid OMG standardize target model and have an ability to validate the generated model.

- GReAT: A Metamodel Based Model transformation Language

A. Agrawal proposed GReAT [13].GReAT follows graph transformation approach. Agrawal focuses that Domain Specific Modeling Environment (DSME) must exist to carry out transformation. This environment is used to develop domain specific Platform Independent Model (DSPIM). Development time reduces due to the use of DSME and DSPIM. GReAT uses the combination of three sub languages pattern specification language, rewriting & transformation language and control flow language.

V. ANALYSIS

Analysis consists of two sub section. The proceeding sections describe the analysis parameters and on the basis of these parameters analysis matrix is created.

A. Analysis Parameters

On reviewing different techniques discussed earlier in this paper, certain parameters have been identified. On the basis of these parameters an analysis matrix has been created, which is described in proceeding section.

These parameters include:

- Stateless: State of the model is or is not preserved during the transformation. Weather transformations performed as in-place (stateless) or as a state full way i.e. the original model is modified or new model is created.
- Automatable: Transformation can be automated or it can only be applied manually.
- Understandability: Transformation syntax can easily understand by user or not.
- Bidirectional: The model can be reconverted from one Model type to other or not i.e. from PIM to PSM or retransform from PSM to PIM. The transformation can be bidirectional or unidirectional.

- Amendable: Transformation support incremental updates or not.
- Tracking: Transformation track changes or not i.e. the transformation technique took care of changes and change can be track
- UML Artifact: Which UML artifacts are supported by the transformation? Class diagram, Activity Diagram etc
- Model Validity: Transformation produces valid model or not. The generated model followed OMG standard or not.
- Tool Support: Tool support available for transformation technique or not.

B. Analysis Matrix

This section presents an analysis matrix on the basis of analysis parameters described above. The analysis matrix in Table: 1 describes the behavior of different approaches/languages on the identified parameters. The matrix in Table: 1 shows both PIM to PSM and PSM to code transformations.

In Analysis Matrix all the values are assigned on the basis of subjective judgment by viewing surveyed paper. The techniques like Visitor Base Approach (VB) and Template Base Approach (TB) have some don't care (DC) values reflect that the technique does not check that parameter.

CONCLUSION AND FUTURE WORK

MDA was defined by OMG in 2000. MDA gained great importance from last few years. A lot of research has been conducted in MDA. The heart of model driven architecture (MDA) is its transformation. There are two types of transformations in MDA, i.e. Model (PIM) to Model (PSM) and the Model (PSM) to Code transformation.

This paper has explained different approaches of Model transformation. Different parameters have been identified and were applied on the existing techniques and found that transformation approaches lack in term of tool support. Very few models to model transformation approaches have tool support. Besides this, most of the presented approaches are stateless, unidirectional and do not support model validity and backtracking. Most of the proposed approaches use UML class diagram only for specifying transformation.

In future work we hope to provide a transformation technique for model to model transformation. The technique convert PIM to PIM transformation from one paradigm to other

TABLE I. ANALYSIS MATRIX

S #	**Technique Name**	**Analysis Parameter**								
		Stateless	**Automatable**	**Understand -ability**	**Amendable**	**Tracking**	**UML Artifact**	**Validity**	**Direction**	**Tool Support**
1	**VB**	DC	Yes	Yes	No	No	DC	No	Unidirectional	Yes
2	**TB**	DC	Yes	Yes	No	No	DC	No	Unidirectional	Yes
3	**CWM**	No	Yes	No	No	No	Class Diagram	No	Unidirectional	No
4	**GT**	Yes	Yes	No	No	Yes	DC	No	Unidirectional	Yes
5	**RA**	No	Partial	No	Yes	Yes	DC	Yes	Bidirectional	No
6	**XSLT**	No	Yes	No	No	No	Class Diagram	No	Unidirectional	Yes
7	**SA**	No	Yes	No	No	No	DC	Yes	Unidirectional	Yes
8	**DUPA**	No	yes		Yes	Yes	Class Diagram	No	Unidirectional	No
9	**MDAMT**	No	Yes	Yes	No	No	Class Diagram	Yes	Bidirectional	No
10	**MT**	No	Yes	Yes	Yes	Yes	Class Diagram	No	Unidirectional	No
11	**MAMT**	No	Yes	No	No	No	Class Diagram	No	Unidirectional	No
12	**MOLA**	No	Yes	yes	No	No	Class, State Diagram	No	Unidirectional	Yes
13	**BOTL**	No	Yes	Yes	No	No	Class Diagram	Yes	Bidirectional	No
14	**GReAT**	Yes	Yes	No	No	No	DC	No	Unidirectional	No

VB: Visitor Base Approach
TB: Template Base Approach
CWM: Common Warehouse Metamodel
GT: Graph Transformation
RA: Relational Approach
SA: Structural Approach
MOLA: Model transformation language.

BOTL: Bidirectional Object transformation language
Great: Graph Rewriting and transformation language
DUPA: Declarative usable pattern Approach
MDAMT: Model driven Approach to model transformation

MT: Model transformation language
MAMT: Metamodeling approach to model transformation

Values:
DC: Don't Care

i.e. Data Flow Diagram (DFD) from Structure paradigm could be converted to Class diagram of Object Oriented Paradigm. That approach provides a MDA platform for structure paradigm. This could include proposing such a declarative language that follows the OMG standard.

References

[1] J. Miller, J. Mukerji, (2001). Model Driven Architecture (MDA). OMG Document available at http://www.omg.org on 03-04-2001

[2] MDA Guide (2003). Version 1.0, Object Management Group, omg / on 01-05-2003.

[3] Object Management Group, The Unified Modeling Language 1.5, OMG Document: formal/ on 03-03-2001

[4] OMG, Meta Object Facility 1.4, OMG Document: formal/on 02-04-2003

[5] Object Management Group, MOF 2.0 Query / Views / Transformations RFP, OMG Document: ad/2002-04-10, revised on April 24, 2002

[6] S. Sendall and W. Kozaczynski, Model Transformation: the heart and soul of MSD, Software, IEEE 2003; 20(5): 42-45

[7] K. Czarnecki and S. Helson. Classification of Model Transformation Approaches. In OOPSLA'03 Workshop on Generative
Techniques in the Context of Model-Driven Architecture, 2003.

[8] Jamda: The Java Model Driven Architecture 0.2, on May 2003,
http://sourceforge.net/projects/jamda/

[9] Java Emitter Templates (JET). Part of the Eclipse Modeling Framework; see JET Tutorial by Remko Pompa at on 31 May 2004 at http://eclipse.org/articles/Article-JET2/jet_tutorial2.html

[10] AndroMDA 2.0.3, on July 2003, http://www.andromda.org

[11] OptimalJ 3.0, User's Guide, http://www.compuware.com/products/optimalj

[12] OMG, The Common Warehouse Model 1.1., OMG Document: formal/2003-02-03,

[13] A. Agrawal,"Metamodel based model transformation language to facilitate domain specific model driven architecture**"**. In SIGPLAN 2003 Anaheim, CA, USA.118-119**.**

[14] P. Braun and F. Marschall. The Bi-directional Object-Oriented Trans-formation Language. Technical Report, Technische Universität München, TUM-I0307, May 2003

[15] D. Varro, G. Varro and A. Pataricza. Designing the automatic transformation of visual languages. Science of Computer Programming, vol. 44(2):pp. 205--227, 2002.

[16] D. H. Akehurst, S.Kent. A Relational Approach to Defining Transformations in a Metamodel. In J.-M. Jézéquel, H. Hussmann, S. Cook (Eds.): UML 2002 – The Unified Modeling Language 5th International Conference, Dresden, Germany, September 30 - October 4, 2002. Proceedings, LNCS 2460, 243-258, 2002.

[17] W3C. XSL Transformations(XSLT) v1.0. W3C Recommendation: http://www.w3.org/TR/xslt, on Nov.1999.

[18] QVT-Merge Group. MOF 2.0 Query/Views/Transformations. Object Management Group, Apr. 2004. on 01-04-2004, Revised Submission.

[19] K. Duddy, et-all, **"**Model Transformation: A declarative, reusable patterns approach". In IEEE EDOC'03**;** 174-185

[20] B. Appukuttan, et-all; "A model driven approach to model transformation". In MDAFA 2003, Holland, June 2003."

[21] L. Tratt; *"The MT model transformation language".* In Proceedings of the 2006 ACM symposium on Applied computing Dijon, France; 1296 - 1303

[22] S. R. Judson, D. L. Carver, R. B. France; "A Metamodeling Approach to Model Transformation". In OOPSLA'03, Anaheim, California, USA; 326 - 327

[23] A. Kalnins, J. Barzdins, E. Celms;" Basics of Model Transformation Language MOLA". In Proceedings of WMDD, 2004.