Model Transformation Languages: State–of–the-art

Madhavi Karanam
Computer Science & Engineering
Gokaraju Rangaraju Institute of Engineering & Technology, Hyderabad, india
bmadhaviranjan@yahoo.com

Lavanya Gottemukkala
Computer Science & Engineering
Gokaraju Rangaraju Institute of Engineering & Technology, Hyderabad, india
lavanya.cvsr@gmail.com

Abstract: **Model Transformation is a core activity of all model-driven approaches. The success of model driven approaches like Model-Driven Development, Model-Driven Engineering etc depends on the model transformations. Hence, Model Transformations has become an emerging approach. Number of appropriate languages and tools are available for Model transformation. In literature, there are several studies are available which encounter comparison of Model Transformation Languages. These articles focused on limited number of languages, related work, motivation and comparison specifications. Hence, this paper aims to focus on all available model transformation languages and their comparison specifications considered here which are not yet focused. The main purpose of this paper to present how timely evolution of model transformation languages have materialized and also the languages are discriminated in various aspects.**

Keywords- Model Transformation, Transformation Languages, Model Driven Engineering, Tools.

I. INTRODUCTION

Transforming models is a main activity of model driven approaches like Model Driven Engineering (MDE), Model-Driven Development (MDD), as it is a process of converting one model to another model of the same system. There are various definitions for Model Transformation are available in [1,2].Model Transformations are used for different activities like creating, modifying, merging, mapping, etc. ,for models in Model Driven Development.

The central concept of model-driven development approaches is Model transformation. It provides a mechanism which automatically does the manipulation of models in various ways. This paper deals with survey of existing model transformation languages. Classif cation used in this paper presents how model transformation languages are evolved, and the mechanism, i.e. characteristics of the model transformation languages. Selection of model transformation languages, tools, and techniques are introduced in this paper by using our classif cation scheme.

Model transformations play an important role in Model Driven Engineering (MDE) approach. It is expected that writing model transformation definitions will become a common task in software development [13]. Software engineers needs support from Model transformation tools and techniques in performing the task as similar to support given now by IDEs, compilers, and debuggers in their everyday work. As a result a number of transformation languages have been proposed. It is observed that, even though the problem domain of these languages is fixed, they still differ in programming paradigm. For programming languages like declarative, functional, object-oriented, imperative, etc supporting model transformation languages are already developed and currently available which usually expose a synthesis of programming paradigms. Different approaches are suitable for different tasks. It is very clear that a single approach may be provided in future. But understanding and more experience is base for identifying non trivial problems which is still necessary.

One class of transformation problems may be solved by using one type of paradigm like declarative and other class of problems may be required imperative. Hence, it is essential to understand various model transformation languages and approaches used, and tools available. For this purpose this paper focuses on review of existing and available Model transformation languages and also comparing them in various aspects. Section 2 of this paper represents various transformation languages available. Section 3 presents classification and comparison, and Section 4 concludes the paper.

II.RELATED WORK

Available model transformation languages are described in this paper and mentioned as below.

A. ATL: (Atlas Transformation Language)

ATL is a domain -specific language for specifying model-to-model transformations. It is a part of the AMMA (ATLAS Model Management Architecture) platform.ATL is a hybrid of declarative and imperative. The preferred style of transformation writing is declarative, which means simple mappings can be expressed simply. However, imperative constructs are provided so that some mappings too complex to be declaratively handled can still be specified. An ATL transformation program is composed of rules that define how source model elements are matched and navigated to create and initialize the elements of the target models.ATL supports only unidirectional transformations. [8],[9],[10],[13], [15], [17]

Query/View/Transformation (QVT)

OMG is the root for Query/View/Transformation (QVT) which is a standard language for Model Transformation.QVT having Three Languages for Model-to-Model Transformations.1.QVT Relational 2.QVT Operational, 3.QVT Core.

B. QVT Relational

QVT-Relations is a declarative language designed to permit both unidirectional and bidirectional model transformations to be written. A transformation embodies a consistency relation on sets of models. Consistency can be checked by executing the transformation in check only mode; the transformation then returns true if the set of models is consistent according to the transformation and False otherwise. The same transformation can be used in enforce mode to attempt to modify one of the models so that the set of models will be consistent. The QVT-Relations language has both a textual and a graphical concrete syntax.[2],[8],[10]

C. QVT Operational

QVT Operational is an imperative model transformation language that extends QVT Relational with imperative constructs. The transformations are unidirectional. It uses implicit trace models.[2],[8],[10]

D. QVT Core

QVT Core is a simple, low-level declarative model transformation language. It serves as a foundation for QVT Relational and is equally expressive. It supports pattern matching over a flat set of variables, where the variables of source, target and trace models are treated symmetrically. Trace models must be defined explicitly.[2],[8][10]

E. Henshin

EMF Henshin is a continuation of the EMF Tiger transformation language. It uses triple graph grammars (TGG) for model-to-model transformation. It is based on the Eclipse Modeling Framework EMF. The transformations are described in the form of model consisting of a left-hand-side graph, a right-hand-side and a list of correspondence mappings. The graph nodes are model element instances of the source metamodel and the target metamodel, respectively. Due to this higher-order transformations are possible with EMF Henshin. [1], [8],[16]

F. GReAT

The Graph Rewriting and Transformation (GReAT) language, which is a graphical language for the specification of graph transformations between domain-specific modeling languages (DSMLs)? It consists of three distinct parts: (1) Pattern Specification language, (2) Graph transformation language, and (3) Control flow language. Input and output transformations are defined in terms of meta-models. Set of transformation rules are used for model transformation. It is not a stand-alone tool.[27], [28].

G. Kermeta

Kermeta is a meta modeling language which allows describing both the structure and the behaviour of models. The characteristics of Kermeta are Model oriented, Imperative, Kermeta is a meta modelling language which allows describing both the structure and the behaviour of models. The characteristics of Kermeta are Model oriented, Imperative, Object-Oriented, Aspect-Oriented and Statically Typed (100% type safe).Various versions of Kermeta are available. [21],[29]

H. ETL

Epsilon family (Kolovos et al., 2006) is a model management platform that provides transformation languages for model-to-model, model-to-text, update-in-place, migration and model merging transformations. Epsilon Transformation Language (ETL) provides all the standard features of a transformation language. ETL can transform many inputs to many output models. Both source and target model scan query/navigate/modify.ETL is a Hybrid model-to-model transformation language. It is part of Epsilon model management infrastructure. Several source and target models can be handled by ETL. [33]

I. JTL

Janus Transformation Language (JTL), is a declarative model transformation language. Distinctive characteristics of JTL are non-bijectivity, model approximation. It also supports bi directionality and change propagation. Answer Set Programming (ASP) is used to represent the semantics of JTL. [19],[20]

J. MT

Model Transformation (MT) is low-cost transformation language and have the same common features as other transformation languages. MT is recognized as vital part of MDD. It also provides additional set of new features compared to others. MT is a unidirectional model transformation language. MT is also capable of doing change propogation. By the MTL ie. Model-to-model, model-to-code. Code-to-model, model-to-text is given in third coloumn. Input/output to any MTL may be one model to one or many which is mentioned as cardinality. How MTLs are used to perform transformation, and how to use is given as technical space. For each MTL a specific tool is available for usage purpose that is also mentioned. When we consider model transformation, in which language models are generated is more important. This information is given as modelling language dependency. [20],[30].

Table 1.Comparision of MTLs

LANGUAGE	Model transformation paradigm	Type of transformation	Technical space	Input/ output cardinality	Supporting tools	Direction	Modeling language dependency	Source
ATL(atlas transformation language)	Hybrid(mixer of imperative and declarative)	Model to model(M2M) `	EMF(Eclipse modeling framework)	M to N	ATL Tool kit	Unidirectional	UML	13,15, 17
QVT Relational	High-Level Declarative	Model to model(M2M)	SmartQVT	M to N	ModelMorf EclipseM2M	Both Unidirectional & Bidirectional	UML	2, 8,10
QVT Core	Low level Declarative	Model to model(M2M)	EMF(Eclipse modeling framework)	M to N	OptimalJ	Bidirectional	UML	2, 8,10
QVT Operational	Imperative	Model to model(M2M)		M to N	EclipseM2M MagicDraw SmartQVT	Unidirectional	UML	2,8,10
Henshin	Graph Transformation	Model to model(M2M)	EMF(Eclipse modeling framework)	M to N	Henshin Plug-in	Bidirectional	UML class diagram State macjine	2,8,16
GreAT	Graph Transformation	Model to model(M2M)	GME	M to N And 1 to 1	GME & GR-Engine	Unidirectional	UML Class Diagram	27,28
Kermeta	Imperative	Model to model(M2M)	EMF(Eclipse modeling framework)	1 to 1	Kermeta	Multidirectional	UML,DSML	21,28,29
ETL	Hybrid	Model to model(M2M)	Epsilon	M to N	ETL plug-in	Unidirectional	-	33
JTL	Declarative	Model to model(M2M)	EMF & AMMA		JTL Engine	Bidirectional	UML State Machine	19.20
MT	Declarative	Model to model(M2M)	QVT	M TO N	MT	Unidirectional	UML	20,30

III..PROPOSED WORK

In this section model transformation languages (MTL) are distinguished in various aspects. The correlated aspects presented in Table 1 are useful for right model transformation. From literature various Model transformation languages have considered for distinguishing. This kind of comparison of number of MTLs is not available. Hence, this paper targeted to provide such comparison in which features of languages are focused. In Table 1, first column represent name of the MTL. Second One is about Model transformation paradigm gives whether MTL is declarative, imperative and hybrid. What kind of transformation provided by the MTL ie. model-to-model, model-to-code. code-to-model, and model-to-text is given in third column. Input/output to any MTL may be one model to one or many which is mentioned as cardinality. How MTLs are used to perform transformation, and how to use this transformation is given as technical space. For each MTL a specific tool is available for usage purpose that is also mentioned. When we consider model transformation, in which language models are generated is more important. This information is given as modelling language dependency.

IV.CONCLUSION

In this paper, an overview of model transformation languages have given, and outlined how they can compare in different aspects. A review of current model transformation languages is presented here. Identified common and unexplored MTLs. Based on this MTLs are capable of providing a flexible, efficient, and practical platform for creating model transformations. These transformations are facilitated by tool integration. Distinguish ion provided in this paper handle experimentation and also model transformations. Choosing a right transformation made easy by the comparison available in this paper.

REFERENCES

[1] Adaptability of Model Transformations , Ivan Kurtev, PhD Thesis, University of Twente, 2005 ISBN 90-365-2184-X
[2] Literature Study on Model Transformations, Matthias Biehl. Royal Institute of Technology, Tech. Rep. ISRN/KTH/MMK
[3] Verifying Metamodel Coverage of Model Transformations, Junhua Wang, Soon-Kyeong Kim and David Carrington, Proceedings of the 2006 Australian Software Engineering Conference (ASWEC'06) 1530-0803/06 $20.00 © 2006 IEEE
[4] OMG, UML 2.0 Infrastructure - Final Adopted Specification, OMG document ptc/03-09-15. http://www.omg.org/cgi-bin/doc?ptc/2003-09-15. 2003
[5] OMG, MDA Guide Version 1.0.1. http://www.omg.org/docs/omg/03-06-01.pdf. 2003
[6] From UML/SPT Models to Schedulability Analysis: a Metamodel-based Transformation AbdelouahedGherbi and FerhatKhendek. Proceedings of the Ninth IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing 0-7695-2561-X/06 $20.00 © 2006 IEEE
[7] Model-driven Development of Complex Software: A Research Roadmap, Robert France, Bernhard Rumpe, Future of Software Engineering(FOSE'07) 0-7695-2829-5/07 $20.00 © 2007,IEEE
[8] A taxonomy of model Transformation.TMens, P Van Gorp - Electronic Notes in Theoretical Computer Science, 2006 – Elsevier pp 125-142
[9] Model Driven Development:Integrating Tools with Practices , Proceedings of the 1997 Workshop on Engineering of Computer-Based Systems (ECBS '97) 0-8186-7889-5/97 $10.00 © 1997 IEEE
[10] Model Transformation: The Heart and Soul of Model-Driven Software Development, Shane Sendall, Swiss Federal Institute of Technology in Lausanne WojtekKozaczynski, Microsoft.
[11] A Survey on Incremental Model Transformation Approaches∗ Angelika Kusel1, Juergen Etzlstorfer1, Elisabeth Kapsammer1, Philip Langer2, Werner Retschitzegger1, Johannes Schoenboeck3, Wieland Schwinger1, and Manuel Wimmer2 ACM/IEEE 16th International Conference on
[12] A Survey on Incremental Model Transformation Approaches∗ Angelika Kusel1, Juergen Etzlstorfer1, Elisabeth Kapsammer1, Philip Langer2, Werner Retschitzegger1, Johannes Schoenboeck3, Wieland Schwinger1, and Manuel Wimmer2 ACM/IEEE 16th International Conference on Model Driven Engineering Languages and Systems September 30, 2013 – Miami, Florida (USA).
[13] Transforming Models with ATL1 FrédéricJouault, Ivan Kurtev
[14] https://wiki.eclipse.org/ATL/User_Guide_-_The_ATL_Language.
[15] http://www.eclipse.org/atl/
[16] Arendt, Thorsten, et al. "Henshin: advanced concepts and tools for in-place EMF model transformations." International Conference on Model Driven Engineering Languages and Systems. Springer Berlin Heidelberg, 2010.
[17] Jouault, Frédéric, Freddy Allilaire, Jean Bézivin, and Ivan Kurtev. "ATL: A model transformation tool." Science of computer programming 72, no. 1 (2008): 31-39.
[18] Kalnins, Audris, Janis Barzdins, and Edgars Celms. "Model transformation language MOLA." Model Driven Architecture. Springer Berlin Heidelberg, 2005. 62-76.
[19] Cicchetti, Antonio, et al. "JTL: a bidirectional and change propagating transformation language." International Conference on Software Language Engineering. Springer Berlin Heidelberg, 2010.
[20] Tratt, Laurence. "The MT model transformation language." Proceedings of the 2006 ACM symposium on Applied computing. ACM, 2006.
[21] Jézéquel, Jean-Marc, Olivier Barais, and Franck Fleurey. "Model driven language engineering with kermeta." International Summer School on Generative and Transformational Techniques in Software Engineering. Springer Berlin Heidelberg, 2009.
[22] https://code.google.com/archive/p/synclib /
[23] Review of Model to model Transformation approaches and Technology ModelWriter,Text & Model-Synchronize Engineering Platform. Project number: ITEA 2 13028 Edited by: FerhatErata, Moharram Challenger, Geylani Kardas
[24] http://mola.mii.lu.lv/
[25] http://jtl.di.univaq.it/
[26] Huber, Philipp. The model transformation language jungle: an evaluation and extension of existing approaches. na, 2008.
[27] Daniel Balasubramanian, Anantha Narayanan, Chris vanBuskirk, and Gabor Karsai,The Graph Rewriting and Transformation Language: GReAT, Proceedings of the Third International Workshop on Graph Based Tools (GraBaTs 2006)
[28] GReAT: A Metamodel Based Model Transformation Language AdityaAgrawal. Proceedings of Automated Software Engineering, 18th IEEE International Conference, 2003.

[29] www.kermeta.org/documents/tutorials/.../mosser_transformation_with_kermeta.
[30] http://www.tcs-trddc.com
[31] Model transformations and tool integration, Laurence Tratt, Journal of Software and Systems Modelling, 4(2):112-122, May 2005
[32] http://www.eclipse.org/epsilon/doc/etl/

AUTHORS PROFILE

G.Lavanya working as a Asst.Prof in Computer Science and Engineering Department, Gokaraju Rangaraju Institute of Engineering and Technology, Hyderabad. She has completed her B.Tech in 2010, M.Tech from JNTUH in 2012.She has 4 years of teaching experience. Her research interest includes software engineering, and Model Driven Engineering.

Dr.K.Madhavi, working as a Professor in Computer Science and Engineering Department, Gokaraju Rangaraju Institute of Engineering and Technology. She has completed her B.E in 1997, M.Tech from JNTUA in 2003 and awarded Ph.D from JNTUA in 2013. She has 19 years of teaching experience. She has published several papers in reputed international journals and international conferences. Her research interest includes software engineering, Model Driven Engineering, Data Mining, and Mobile software engineering.