World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:5, No:2, 2011

Behavior Model Mapping and Transformation using Model-Driven Architecture

Mohammed Abdalla Osman Mukhtar*, Azween Abdullah**, and Alan Giffin Downe***

***Abstract*—**Model mapping and transformation are important processes in high level system abstractions, and form the cornerstone of model-driven architecture (MDA) techniques. Considerable research in this field has devoted attention to static system abstraction, despite the fact that most systems are dynamic with high frequency changes in behavior. In this paper we provide an overview of work that has been done with regard to behavior model mapping and transformation, based on: (1) the completeness of the platform independent model (PIM); (2) semantics of behavioral models; (3) languages supporting behavior model transformation processes; and (4) an evaluation of model composition to effect the best approach to describing large systems with high complexity.

***Keywords*—**MDA; PIM; PSM; QVT; Model Transformation

I. Introduction

DESPITE the growth of interest in model-driven architecture (MDA) there is still little agreement on how behavioral aspects should be supported with the approach. Considerable effort has been devoted recently to model mapping and the transformation from platform independent models (PIMs) to platform specific models (PSMs) in many application domains. Much of this work has focused heavily on behavioral aspects of PSMs. There is a need for broader consideration of behavior model mapping using either vertical mapping (refinement) or horizontal mapping (from PIM to PSM).

The central idea of Model Driven Architecture (MDA) which sponsored by Object Management Group (OMG) is that developer should develop models, not programs. That is not to privilege a graphical over a textual programming, but rather to make the developer to be enabled to work at as a high level of abstraction as is feasible. The general scenario of MDA is a single platform independent model (PIM) might be created and transformed, automatically, into various platform specific models (PSMs) by the systematic application of understanding concerning how applications are best implemented on each specific platform. The OMG's queries, views and transformations (QVT) standard [1] defines languages in which such transformations can be written.[2]

It is possible to analyze the current state of the development of procedures for the mapping and transformation of behavior according to four dimensions. The first of these relates to the semantics of behavioral (operational) models. The second relates to the completeness of behavior platform independent model (PIM). The third considers the languages that have been developed or proposed for application to behavior model mapping and transformation. The final dimension examines the suitability of new trends for model composition.

The purpose of this paper is to give more attention to the stage of describing the system requirement in high level abstraction especially in the PIM to add more details to make MDA as a framework for behavior model mapping.

II. Completeness of PIM

If one traces the development of MDA approaches, it can be seen that most research has focused attention on: (1) structural aspects of the PSM level; and (2) processes for generating code. Much less attention has been typically devoted to the PIM model level or to the behavior of modeled applications. One exception was found in the work of Daniele et al. [3], which presented an MDA-based approach that incorporated behavior modeling at the PIM level, but within a specific category of applications. These authors argued that behavior in a PIM can be divided to more than one layer of abstraction, the first one being more independent than subsequent layers, and the deeper ones, essentially, moving nearer to a PSM.

This approach was applied to a Mobile System (M-MUSE DSL), in which the platform-independent design phase was decomposed in the service specification and platform-independent service design steps [3]. PIM design, it was argued, should be a refinement of the service specification, which implies that correctness and consistency particularly of behavioral issues must be addressed in the refinement transformation. However, when trying to realize this refinement transformation, the gap between service specification and platform-independent service design can become rather wide, such that correctness and consistency becomes hard to guarantee in a single refinement transformation, T1. Therefore, an intermediate step in which the service specification behavior is refined (see figure1) may be necessary.

*Mohammed Abdalla Osman Mukhtar is a PhD Student in the Universiti Teknologi PETRONAS, Department of Information Science & Technology (phone: +60 1472 30771; e-mail: abofatima92@gmail.com).

**Assoc.Prof.Dr. Azween Bin Abdullah is with the Department of Information Science & Technology, Universiti Teknologi PETRONAS, Bandar Seri Iskandar, Perak 31750 MALAYSIA (e-mail: azweenabdullah@petronas.com.my).

***Dr. Alan Giffin Downe is with the Management & Humanities Department, Universiti Teknologi PETRONAS, Bandar Seri Iskandar, Perak 31750 MALAYSIA, (e-mail: alan_downe@petronas.com.my).

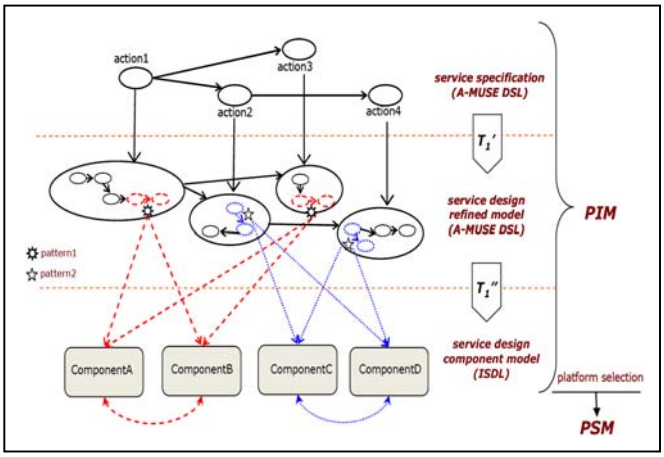

Fig. 1 PIM levels and transformation between these levels [3]

III. SEMANTICS OF BEHAVIORAL MODELS

Key to the development of the MDA approach has been the extensive work aimed at defining the description semantics of Object Constraints Language (OCL), the programming or modeling languages. For that, metamodeling, since the beginning of this decade, has become a widely used tool to describe the abstract syntax of modeling languages. There are two generally acknowledged approaches to describing OCL constraints semantics [4]. For instance, this constraint eval:

CONSTRAINT × STATE → {true,false,undefined}

Can be defined either mathematically by using structural induction over CONSTRAINT (refer to [5]), or logically like using Isabelle/High-Order Logic (HOL).

These two approaches have a good manner to evaluate OCL constraints in a formal and non-ambiguous method, but they still have some disadvantages. First disadvantage is this gap between OCL's official syntax definition which is given as metamodel, and the OCL's syntax which is given in structural induction. Second, which is the main drawback is the understandability.

The main technique to heal the rift of this gap and to get good understandability is metamodeling. Metamodels are already used to define abstract syntax with very expressive and easy to understand. It is already used to define the semantics of class diagrams. This technique is sponsored by OMG using Evaluation-Metaclasses [4], and this approach is provided using transformation rules written in QVT. Figure2 shows metamodel for OCL abstract syntax, and figure3 shows metamodel for the semantics of OCL. In [6] also applying graph transformation to OCL constraints semantics.

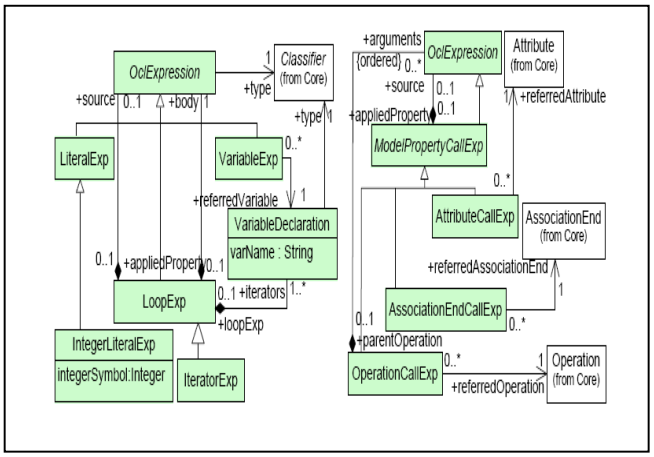

Fig. 2 MetaModel for OCL – Syntax [4]

Now, in recent years, OCL becomes a constraint language that is applied to various modelling languages, instead of just it is a language used to constrain UML models. This includes Domain Specific Languages (DSLs), and meta-modelling languages like MOF or Ecore.

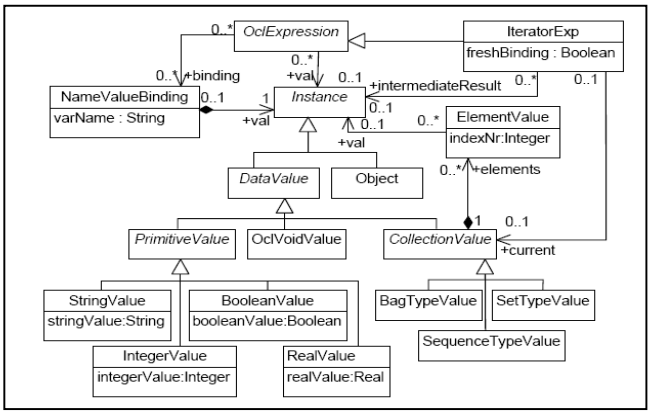

Fig. 3 MetaModel for OCL – Semantics [4]

The new trend is going on providing variability to OCL parsers to work with different modeling languages; variability concentrate on the technical space which models are implemented in (like Java, Ecore, or a specific model repository). In [7] the authors argued that all OCL tools support variability at the model level (OCL compilers), for that they said we can support variability at the model instance level (OCL interpreter) and proposed a generic adaptation architecture for OCL interpreters that hides models and model instances behind well-defined interfaces. This enables reuse of the complete OCL infrastructure including the OCL parser, standard library and interpreter. There is also some work done for modeling operational semantics of domain specific modeling language (DSML) as presented in [8], which

applyed this approach to petri nets as well as for a stream - oriented language in the domain of earthquake detection.

IV. Supporting Languages for Mapping of Behavior Models

There is a lot of transformation languages working as a tool to make the transformation operation full automated, we have chosen a three types of these languages depend on some criterion. First one is Query, View, Transformation (abbreviated by QVT) which is most standardized, which is sponsored by Object Management Group (OMG). The second one is KerMeta (abbreviation of Kernel Metamodel), it is domain specific language, it is building basically on Object Oriented Programming, and it can be plugged on Eclipse. The third one is MATA (abbreviation of Modeling Aspects using a Transformation Approach), from its' name we can see that it is building on Aspect Oriented Programming. Now we need to take each language individually, and focusing the light on some its' features, and making technical comparison.

A. QVT

QVT (Query/Views/Transformation) is the OMG standard language for specifying model transformations in the context of MDA. It is regarded as one of the most important standards since model transformations are proposed as major operations for manipulating models [8].

The three concepts that are used in the name of the QVT language as defined by OMG documents are: [9]

Query: A query is an expression that is evaluated over a model. The result of a query is one or more instances of types defined in the source model, or defined by the query language.

View: A view is a model which is completely derived from another model (the base model). There is a 'live' connection between the view and the base model.

Transformation: A model transformation is a process of automatic generation of a target model from a source model, according to a transformation definition.

QVT languages are arranged in a layered architecture shown in Figure 4. The languages Relations and Core are declarative languages at two different levels of abstraction. The specification document defines their concrete textual syntax and abstract syntax. In addition, Relations language has a graphical syntax. Operational Mappings is an imperative language that extends Relations and Core languages. Relations language provides capabilities for specifying transformations as a set of relations among models. Core language is a declarative language that is simpler than the Relations language. One purpose of the Core language is to provide the basis for specifying the semantics of the Relations language. The semantics of the Relations language is given as a transformation *RelationsToCore*. This transformation may be written in the Relations language.

Sometimes it is difficult to provide a complete declarative solution to a given transformation problem. To address this issue the QVT proposes two mechanisms for extending the declarative languages Relations and Core: a third language called Operational Mappings and a mechanism for invoking transformation functionality implemented in an arbitrary language (Black Box implementation).

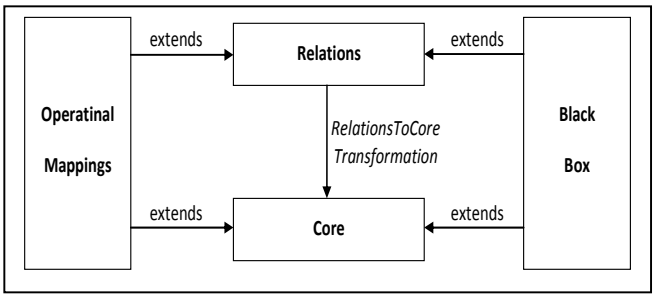

Fig. 4 Layered Architecture of QVT Languages [9]

B. KerMeta

KerMeta is a meta-language for specifying the structure and behavior of models. It has been also been developed as a core language for Model Driven Engineering (MDE) platform. KerMeta is an executable metamodelling language implemented on top of the Eclipse Modeling Framework (EMF) within the Eclipse development environment. Figure 5 shows three main windows in KerMeta Graphical Interface. The first one is the metamodel using class diagram(which is a subset from UML class diagram MOF metamodel), the second widows is the KerMeta code to describe the class diagram, and the last one is the summarization for the class diagram.

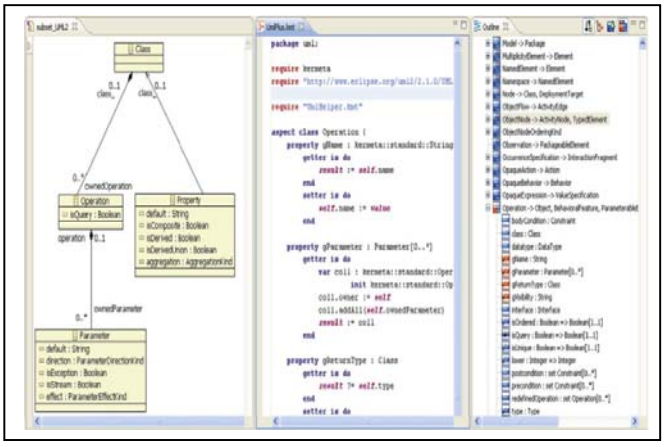

Fig. 5 KerMeta Graphical Interface [10]

Kermeta is a language for specifying metamodels, models,and model transformations that are compliant to the Meta Object Facility (MOF) standard [11]. The object-oriented meta-language MOF supports the definition of metamodels in terms of object-oriented structures (packages, classes, properties, and operations). It also provides model-specific constructions, such as containments and associations between classes [10].

C. MATA

MATA takes a different approach to aspect-oriented modeling (AOM) since there are no explicit join points. Rather, any model element can be a join point, and composition is a special case of model transformation. The

graph transformation execution engine, AGG, is used in MATA to execute model compositions, and critical pair analysis is used to automatically detect structural interactions between different aspect models. MATA has been applied to a number of realistic case studies and is supported by a tool built on top of IBM Rational Software Modeler.

Figure 6 [12] shows the base model slice which is composed of a set of base models. Similarly, an aspect model slice is composed of a set of aspect models. Base models are written in standard UML. Aspect models are written in the MATA language and are defined as increments of the base models or other aspect models. Each aspect model describes the set of model elements affected by the aspect (i.e. the joinpoints) and how the base model elements are affected. Note that an aspect model can only be defined as an increment of a model of the same type; for example, sequence diagram aspects can extend base sequence diagrams but not base state diagrams.

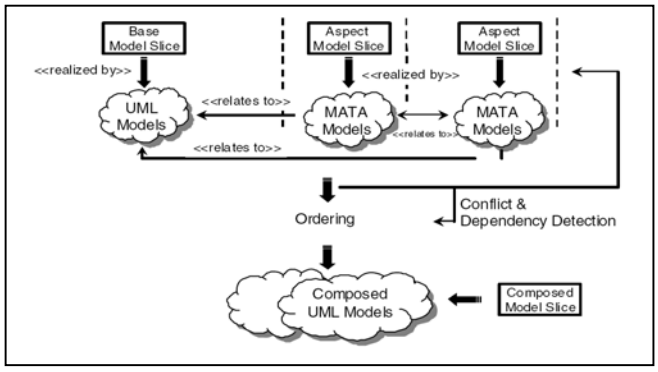

Fig. 6 An Overview of MATA [12]

V. Model Composition

Model composition is a technique, which used with behaviour models for building bigger models from smaller models, thus allowing system designers to control the complexity of a model-driven design process. But many these model composition techniques are themselves very complex because they compose the internal member of participating models in non-simple manner.

In [13] they applied some of the ideas from modular programming to reduce the complexity of model compositions, trying to provide a model composition technique with a proposed modular that treats the participating models as black boxes. They argue that it will be simple, it does not require a separate language for expressing the composition, and the resulting composed model will be easy to understand by the modular nature of the model composition.

There are a lot of approaches been proposed depending on different components. Feature model composition [14] is one of these approaches, where Model-Based Engineering (MBE) and Aspect-Oriented Modeling (AOM) communities have developed a set of model composition techniques and tools. For that there is an interest in determining how these techniques perform with feature model composition and which techniques are the most suitable.

Aspect model composition is another approach of combining two models, MB and MA, where an aspect model MA is said to crosscut a base model MB. As such, aspect model composition is a special case of the more general problem of model fusion. A number of techniques and languages have been developed to specify how MA crosscuts MB, and, in particular, how MA and MB should be composed [12].

VI. Conclusion

In this paper we are focusing on behavior model transformation in order to push the wheel of behavior model transformation development, and to be aware about some aspects that we can contribute on to participate in these developing. These aspects are Completeness of Platform Independent Model (PIM), Semantics of Behavior Models, Supporting Languages for Mapping of Behavior Models, and Model Composition.

References

[1] O.M.G. (OMG), "Meta Object Facility (MOF) 2 . 0 Query / View / Transformation Specification," Transformation, 2008.
[2] P. Stevens, "Bidirectional model transformations in QVT: semantic issues and open questions," Software & Systems Modeling, vol. 9, Dec. 2008, pp. 7-20.
[3] L.M. Daniele, L.F. Pires, and M.V. Sinderen, "An MDA-Based Approach for Behaviour Modelling of Context-Aware Mobile Applications" Behaviour, 2009, pp. 206-220.
[4] S. Marković and T. Baar, "Semantics of OCL specified with QVT," Software & Systems Modeling, vol. 7, Mar. 2008, pp. 399-422.
[5] O.M.G. (OMG), "OMG Object Constraint Language," Management, vol. 03, 2010.
[6] P. Bottoni, M. Koch, F. Parisi-presicce, and G. Taentzer, "Consistency Checking and Visualization of OCL Constraints," Constraints, 2000, pp. 294-308.
[7] C. Wilke, M. Thiele, and C. Wende, "Extending Variability for OCL Interpretation," 2010, pp. 361-375.
[8] G. Wachsmuth, "Modelling the Operational Semantics of Domain-Specific Modelling Languages," Structure, 2008, pp. 506-520.
[9] I. Kurtev, "State of the Art of QVT : A Model Transformation Language Standard," Data Engineering, 2008, pp. 377-393.
[10] N. Moha, S. Sen, C. Faucher, O. Barais, and J.-M. Jézéquel, "Evaluation of Kermeta for solving graph-based problems," International Journal on Software Tools for Technology Transfer, vol. 12, Apr. 2010, pp. 273-285.
[11] O.M.G. (OMG), "Meta Object Facility (MOF) Core Specification," Management, 2006.
[12] J. Whittle, P. Jayaraman, A. Elkhodary, and A. Moreira, "MATA : A Unified Approach for Composing UML Aspect Models Based on Graph Transformation *," 2009, pp. 191-237.
[13] P. Kelsen and Q. Ma, "A Modular Model Composition Technique," 2010, pp. 173-187.
[14] M. Acher, P. Collet, P. Lahire, and R. France, "Comparing Approaches to Implement Feature Model Composition," Springer-Verlag Berlin Heidelberg 2010, 2010, pp. 3-19.