Mapping of Behavior Model using Model-Driven Architecture

Mohammed Abdalla Osman Mukhtar
Azween Bin Abdullah
Department of Computer & Information Science
Universiti Teknologi PETRONAS
Tronoh, Malaysia

ABSTRACT
Mapping and transformation is a twin process in a high level system abstraction, which they playing corner stone of model driven architecture (MDA) technique. But the researchers on this field gave most attention for the static systems abstraction, while we find that almost systems in the world are dynamic with high frequency behavior changing. In this paper we will focus on what is the work has been done on behalf to behavior model transformation depending on four aspects; firstly is the semantics of behavior model, secondly is about the completeness of platform independent model (PIM), thirdly is about some language which are supporting behavior model transformation process, and the last aspect is the model composition which can be the perfect approach to deal with describing the large system that definitely has high complexity and hard understanding scale.

General Terms
Model Driven Architecture, Model Transformation, Software Engineering and Modeling.

Keywords
MDA; QVT; PIM; PSM; OCL; Model Transformation.

1. INTRODUCTION
For MDA approach there is still no agreement on how behavior aspects should be supported. There are a lot of effort has been done on model mapping and transformation from PIMs to PSMs in many application domains. Much works which have been done using MDA approach give attention on behavior aspects in PSMs. In this paper, we will provide a good monitoring for behavior model mapping either using vertical mapping (refinement) or horizontal mapping (from PIM to PSM).

The central idea of the OMG's Model Driven Architecture (MDA) is that developer should be used to 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].
Depending on four aspects, we can trace the progress in development of behavior models mapping and transformation. First aspect is about the semantics of behavioral (Operational) models, second aspect is the completeness of behavior platform independent model (PIM), beside the languages that are working on the field of behavior model mapping and transformation as a third aspect, and the last aspect which is about the new trends for model composition.

2. SEMANTICS OF BEHAVIORAL MODELS
All work which has been done in this field is about defining the description semantics of Object Constraints Language (OCL) or the programming or modeling languages. For that, metamodeling became in the beginning of this decade has a widely useful tool to describe the (abstract) syntax of modeling languages. As [3] said there is already two approaches to describe OCL constraints semantics (like this constraint eval: CONSTRAINT x STATE → {true,false,undefined}), which can be defined either mathematically by using structural induction over CONSTRAINT (refer to[4]), 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 one 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 also used to define the semantics of class diagrams that technique is provided by OMG using Evaluation-Metaclasses [3], which this approach is provided using transformation rules written in QVT. Figure1 shows metamodel for OCL abstract syntax, and figure2 shows metamodel for the semantics of OCL. In [5] also applying graph transformation (Model Transformation) to OCL constraints semantics.

Now, in recent years, OCL becomes a constraint language that is applied to various modeling 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. The new trend is going on providing firstly variability to OCL parsers to work with different modeling languages, second variability concentrate on the technical space which models are implemented in (like Java, Ecore, or a specific model repository).

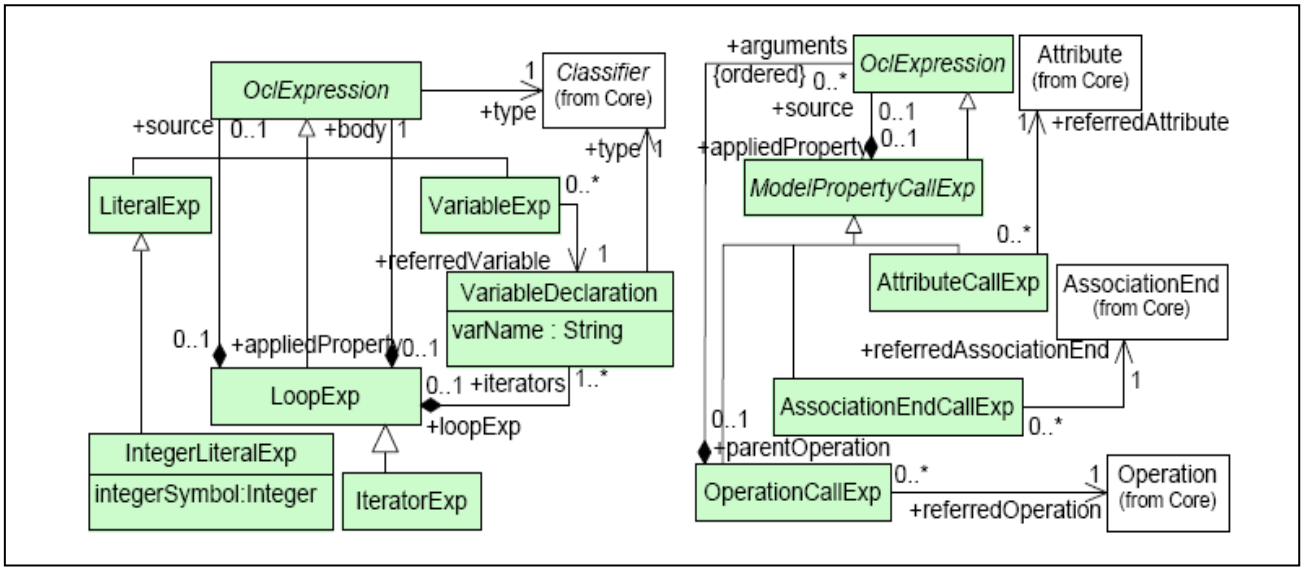
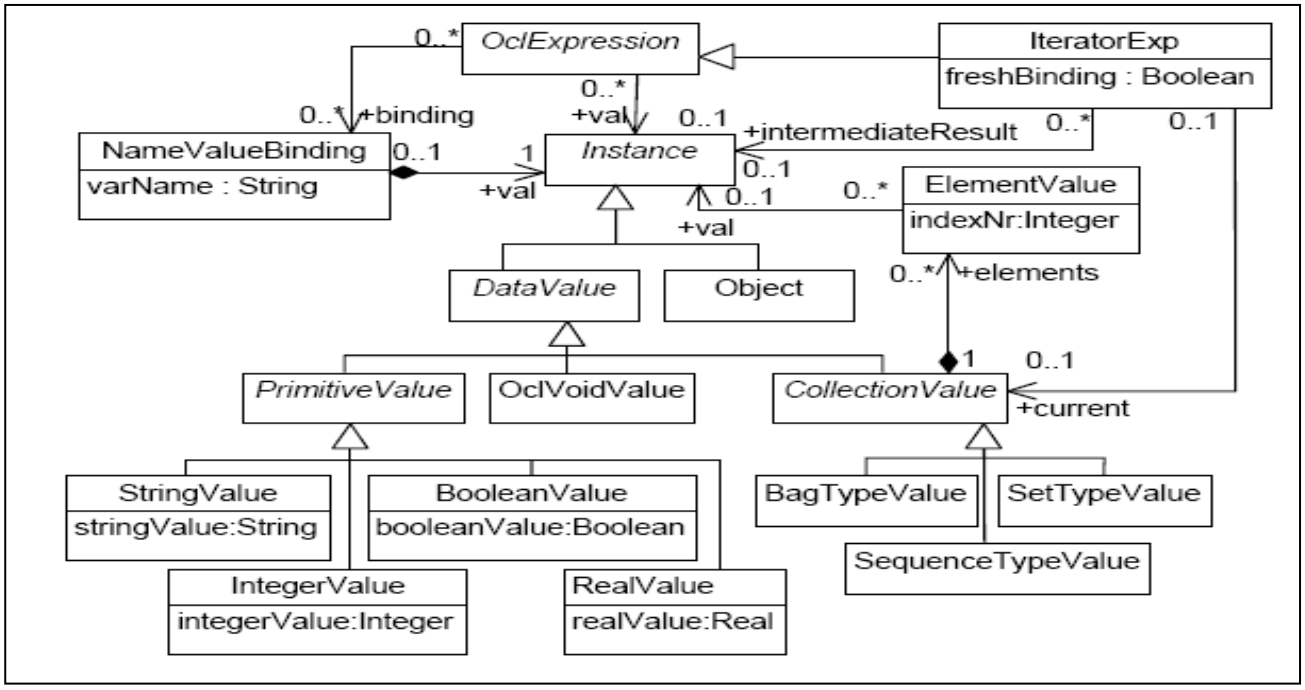


Figure 1: MetaModel for OCL – Syntax [3]

Figure 2: MetaModel for OCL – Semantics [3]



In [6] argue that all OCL tools support variability at the model level (OCL compilers), for that he 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 what is doing in [7], which is applying this approach to petri nets as well as for a stream - oriented language from the domain of earthquake detection.

3. COMPLETENESS OF PLATFORM INDEPENDET MODEL (PIM)

If we trace the development of MDA approaches, we will find that most researchers had gave their attention on structural aspects of platform specific model (PSM) level and in generating code, but they had gave less attention to the platform independent model (PIM) level and the behavior of the modeled applications. We did not find a lot of work about this aspect, but we only aware of one paper, which present an MDA based that incorporates behaviour modelling at the PIM level in the development of a specific category of applications [8]. In this paper they mentioned that behavior PIM can be divided to more than one layer of abstraction, the first one is more independent than the other layers, while the deep one can be near more to PSM.

In [8] they had applied their approach to a Mobile System (called M-MUSE DSL), therefor we find that the platform-independent design phase has been decomposed in the service specification and platform-independent service design steps. The platform-independent service design model should be a refinement of the service specification, which implies that correctness and consistency particularly of behavioral issues have to be addressed in the refinement transformation. However, when trying to realize this refinement transformation, they noticed that the gap between service specification and platform-independent service design is rather wide, so that correctness and consistency was hard to guarantee in a single refinement transformation T1. Therefore, they introduced an intermediate step in which the service specification behavior is refined (see figure 3). This intermediate step results in an intermediate design called service design refined model and their final PIM is renamed to service design component model, the refinement transformation T1 has been consistently decomposed in two transformations T1' and T1".

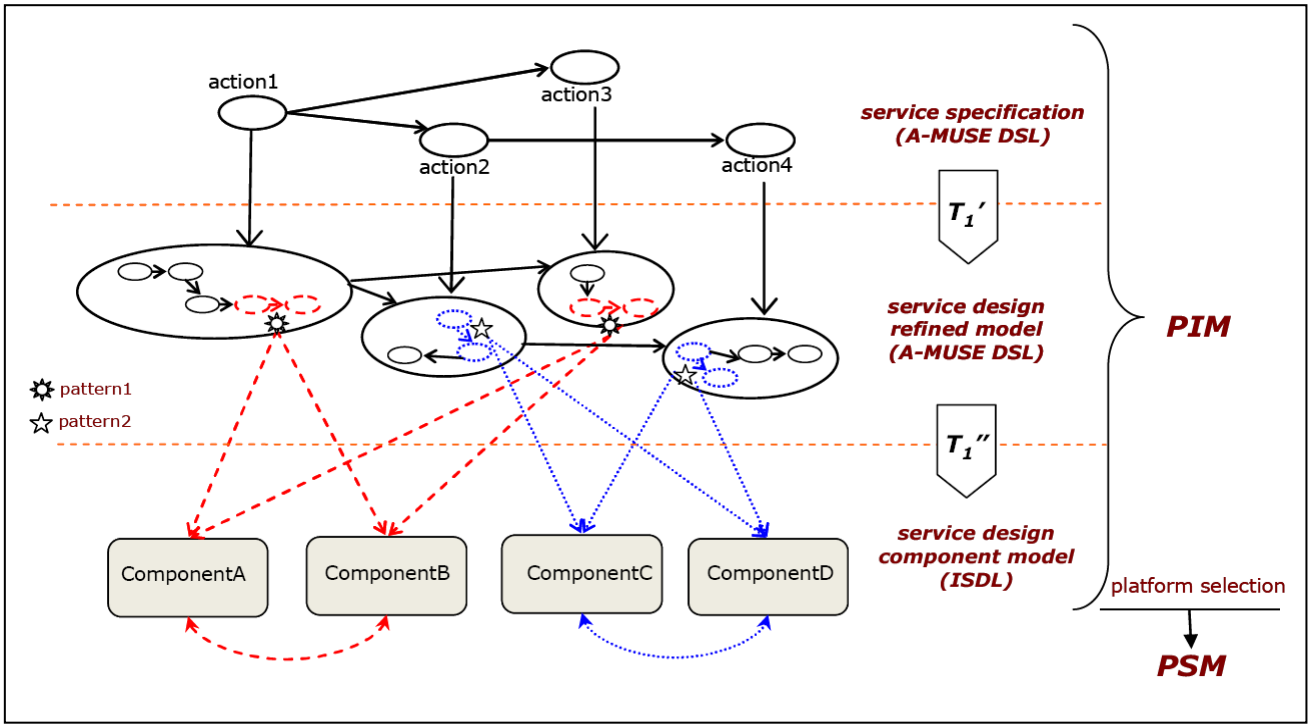


Figure 3: PIM levels and transformation between these levels [8]

4. 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.

4.1 Query/View/Transformation (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).

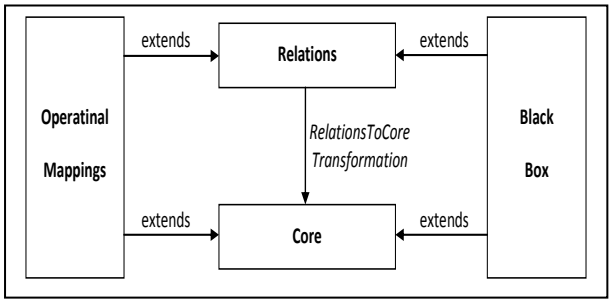


Figure 4: Layered Architecture of QVT Language

4.2 Kernel MetaModel (KerMeta)

KerMeta is a meta-language for specifying the structure and behavior of models; it has been also 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 show three main windows in KerMeta Graphical Interface, which 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.

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].

4.3 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.

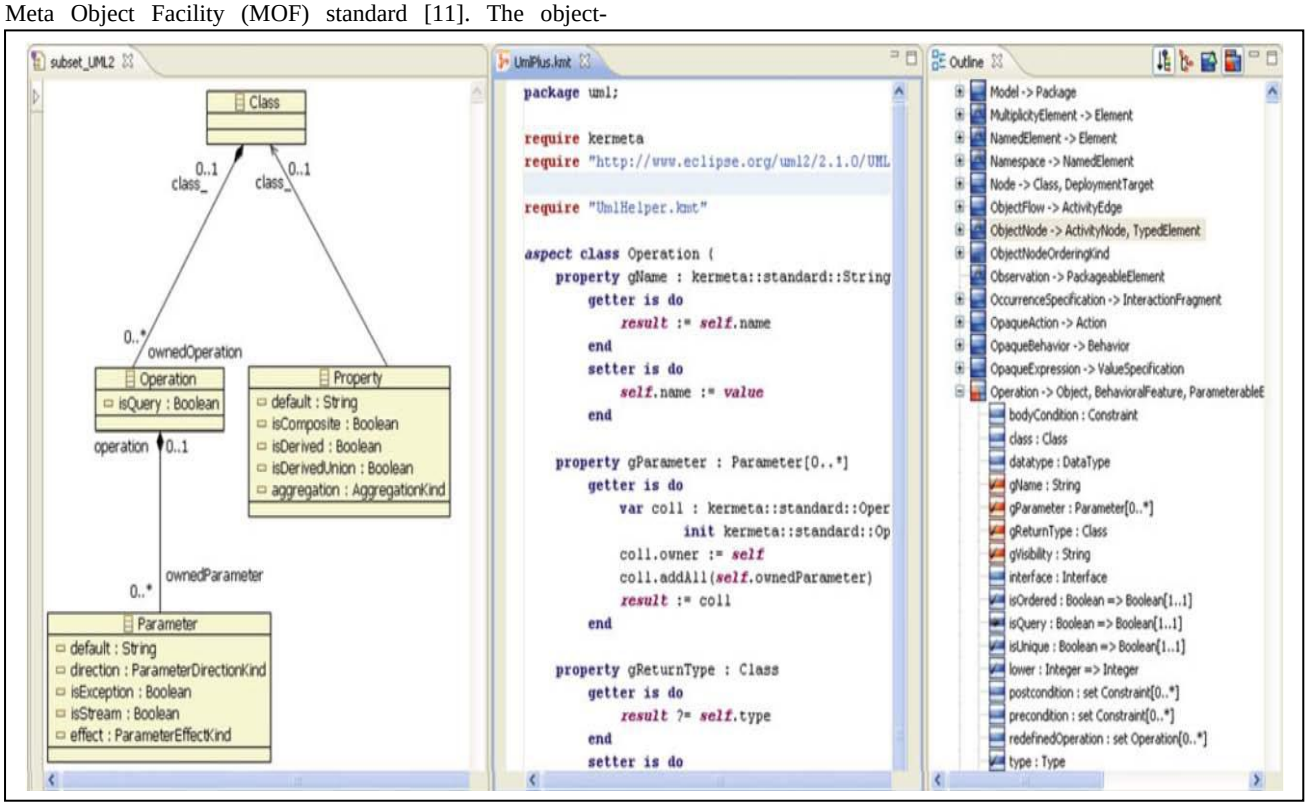


Figure 5: KerMeta Graphical Interface [10]

5. 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 apply 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

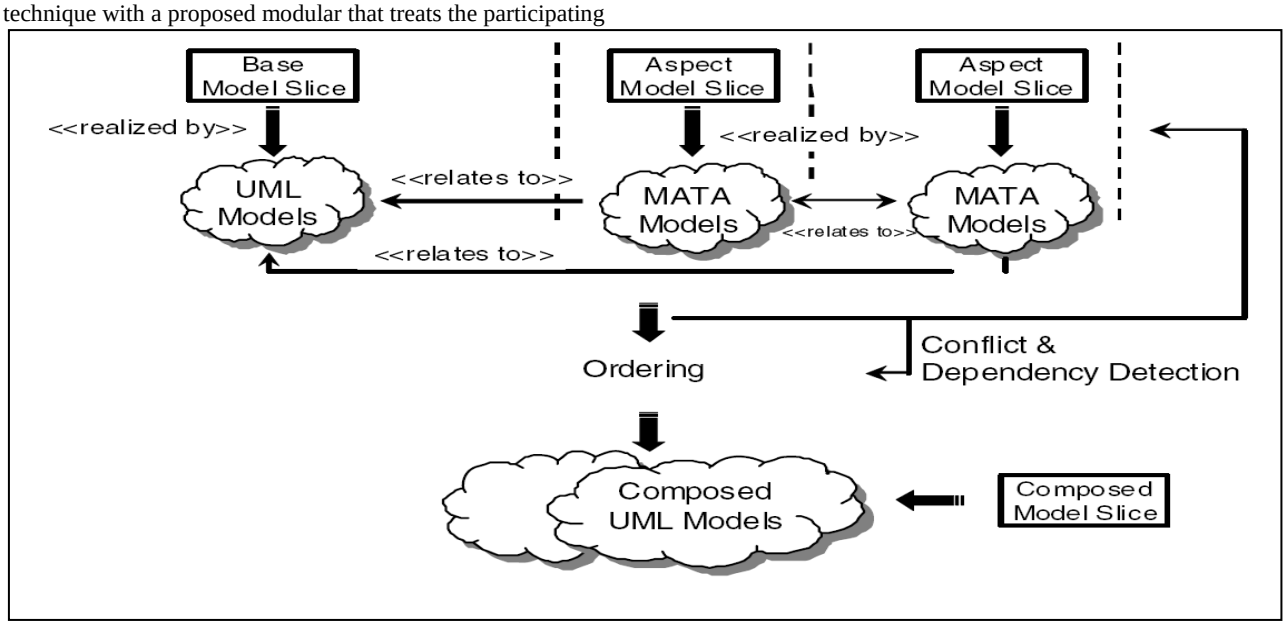


Figure 6: An Overview of MATA [12]

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 is a lot of approaches have been proposed depending on different components. Feature model composition [14] is a one of these approaches, therefore 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].

CONCLUSION

In this paper we are focusing on behavior model transformation 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 Semantics of Behavior Models, Completeness of Platform Independent Model (PIM), Model Composition, and Supporting Languages for Mapping of Behavior Models.

6. 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] S. Marković and T. Baar, "Semantics of OCL specified with QVT," Software & Systems Modeling, vol. 7, Mar. 2008, pp. 399-422.

[4] O.M.G. (OMG), "OMG Object Constraint Language," Management, vol. 03, 2010.

[5] P. Bottoni, M. Koch, F. Parisi-presicce, and G. Taentzer, "Consistency Checking and Visualization of OCL Constraints," Constraints, 2000, pp. 294-308.

[6] C. Wilke, M. Thiele, and C. Wende, "Extending Variability for OCL Interpretation," 2010, pp. 361-375.

[7] G. Wachsmuth, "Modelling the Operational Semantics of Domain-Specific Modelling Languages," Structure, 2008, pp. 506-520.

[8] 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.

[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.