Measuring Model Transformation in Model Driven Development

Motoshi Saeki[1] and Haruhiko Kaiya[2]

[1] Dept. of Computer Science, Tokyo Institute of Technology
Ookayama 2-12-1, Meguro-ku, Tokyo 152, Japan
[2] Dept. of Computer Science, Shinshu University
Wakasato 4-17-1, Nagano 380-8553, Japan
saeki@se.cs.titech.ac.jp, kaiya@cs.shinshu-u.ac.jp

Abstract. In this paper, we propose the technique to define the metrics of model transformation using a meta-modeling technique and a graph rewriting techniques in a Model Driven Development (MDD) context.

1 Introduction

In Model Driven Development (MDD), model transformation is one of the key technologies [1] and the transformations that can improve the quality of models are significant. One of the problems in MDD is how to identify what transformations can improve the quality of models. If a metrics value can express the quality of a model, increasing the metrics values before and after a model transformation can show the improvement of the model quality. It means that the formal definition of a transformation should include the definition of metrics of model quality so that the metrics can be calculated during the transformation. Although we could find excellent techniques to formalize model transformation of MDD until now, there are quite few arguments on the significance of clarifying and defining the quality metrics of model transformation [5].

In this short paper, we propose a technique to solve the problem on how to define metrics of model transformation. To realize this technique, we should have a technique to specify metrics according to models, i.e. model-specific metrics, because the metrics of model transformation can be defined from the metrics of the models that appear in the transformation. We can summarize the approaches that we adopt as follows; 1) Using a meta modeling technique to specify model-specific metrics and 2) Using a graph rewriting system to formalize model transformation, whose essential points will be explained in sections 2 and 3 respectively.

2 Meta Modeling and Defining Metrics

A meta model specifies the structure or data type of the models and in this sense, it can be considered as an abstract syntax of the models. In our technique, we adopt a class diagram of UML for specifying meta models and Object Constraint Language (OCL) for constraints on models. The example of the meta model of the simplified version of class diagrams is shown in Figure 1 (a). As shown in the figure, it has the concepts

"Class", "Operation" and "Attribute" and all of them are defined as classes and these concepts have associations representing logical relationships among them. For instance, the concept "Class" has "Attribute", so the association "has_Attribute" between "Class" and "Attribute" denotes this relationship.

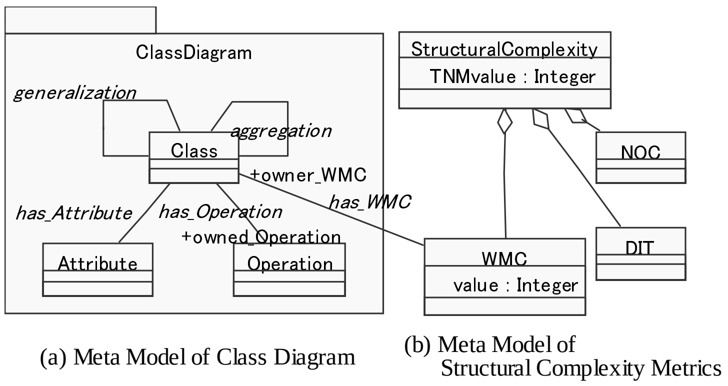

Fig. 1. Meta Model with Metrics Definitions

We can embed metrics and their calculation methods into a meta model in the same way. More concretely, metrics such as WMC (Weighted Methods per Class), DIT (Depth of an Inheritance Tree) and NOC (Number of Children) of CK metrics [3] are defined as classes having the attribute "value" in the meta model as shown in the Figure 1 (b). The "value" has the metrics value and its calculation is defined as a constraint written with OCL. For example, WMC is associated with each class of a class diagram through the association "has_WMC" and the role names "owner_WMC" and "owned_Operation" are employed to define the value of WMC with OCL. Intuitively speaking, the value of WMC is the number of the methods in a class when we make weighted factors 1 and we take this simple case. It can be defined as follows.

context WMC::value : Integer

derive: owner_WMC.owned_Operation -> size()

WMC and the other CK metrics are for a class not for a class diagram, and in this example, we take the sum total of WMCs for the class diagram, and the attribute TNMvalue of StructuralComplexity holds it as shown in Figure 1 (b). The technique of using OCL on a meta model to specify metrics was also discussed in [2, 4].

3 Metrics on Model Transformation

The model following its meta model is represented with an attributed typed graph and it can be transformed by applying the rewriting rules. We call this graph instance graph in the sense that the graph is an instance of the meta model. We can design graph rewriting rules considering the nodes of the metrics and their values. Figure 2 depicts the overview of a model transformation process based on graph rewriting. After a model is converted into an instance graph notation following its meta model, it is transformed into an instance graph of a new model by applying a rewriting rule. At that time, the rule specifies the metrics value m1 is transformed into m2 in the figure, and the value 0.5 of the model is changed into 0.8 at the new model. That is to say, the metrics values of the model after the transformation can be related to the value of the older model before the transformation.

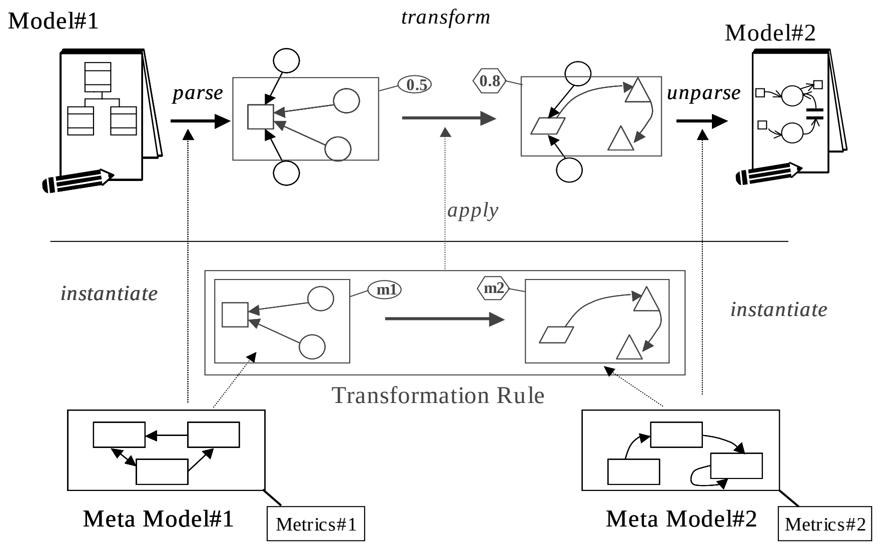

Fig. 2. Model Transformation Process

See an example of a transformation rule shown in Figure 3. Two conditions $x2 > a$ and $x3 < y3$ are attached to the rule for rewriting the graph G1 with G2 and these conditions should be satisfied before the rule is applied. This rule includes two nodes for metrics; one is the metrics for G1 and another is for G2. The first condition $x2 > a$ expresses that the rule cannot be applied until the value of the metrics m2 before the rewriting is greater than a certain value, i.e. “a”. It means that this model transformation is possible when the model has a quality higher than a certain standard. The second condition $x3 < y3$ specifies monotonic increasing of the metrics m3 in this transformation. This formula has both values of metrics before and after the transformation as parameters and it can specify the characteristics of the transformation, e.g. a specific metrics value is increasing by the transformation. As shown in the figure, the calculation of the metrics n2 uses the metrics m1 of the model before the transformation, and this calculation formula of n2 shows that the metrics value of G1 is propagated to G2. The quality of a transformation can be formally specified by using this approach. In Figure 3, we can calculate how much the quality could be improved with the transformation by using the metrics values of the model before the transformation and those after the transformation. The function g in the figure calculates the improvement degree of the quality. This is a basic idea of the quality metrics of model transformation.

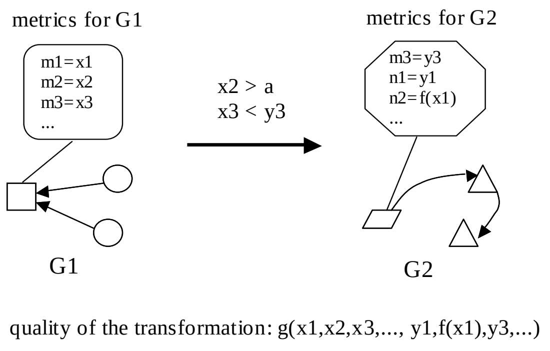

Fig. 3. Metrics and Model Transformation

4 Conclusion and Future Work

In this paper, we propose the technique to specify the metrics of model transformations based on graph rewriting systems. In addition to the development of supporting tools, one of the future research agenda can be collecting useful definitions of metrics. Although the aim of this research project is not to find or collect useful and effective quality metrics, making a kind of catalogue of metrics definitions and specifications is important in the next step. The assessment of the collected metrics is also a research agenda.

References

1. OMG Model Driven Architecture. http://www.omg.org/mda/.
2. F. B. Abreu. Using OCL to Formalize Object Oriented Metrics Definitions. In *Tutorial in 5th International ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering (QAOOSE 2001)*, 2001.
3. S. Chidamber and C. Kemerer. A Metrics Suite for Object-Oriented Design. *IEEE Trans. on Software Engineering*, 20(6):476–492, 1994.
4. M. Saeki. Embedding Metrics into Information Systems Development Methods: An Application of Method Engineering Technique. In *Lecture Notes in Computer Science (Proc. of CAiSE 2003)*, volume 2681, pages 374–389, 2003.
5. M. Saeki and H. Kaiya. Model Metrics and Metrics of Model Transformation: *Materials of 1st Workshop on Quality in Modeling: MoDELS2006 Conference*. http://www.ituniv.se/ miroslaw/QiM.htm, 2006.