ABSTRACT

Model transformation is a central artifact of software development in Model Driven Development (MDD). It can transform the developed model into other models automatically which may be done by forward and backward transformation. This is well-known as a bidirectional model transformation or BMT. A BMT process is claimed that its execution is correct if its end of process produce the desired result. One of the aspects that affect is consistency between elements of two models. This paper proposes a framework used to be a basic foundation in the system construction for the specification and the consistency verification of BMT. This framework will be specified using declarative approach.

Keywords: Model to Model Transformation, Bidirectional Model Transformation, Consistency Verification

1 INTRODUCTION

Model transformation is widely accepted in software development because it can produce the source code, other models and documentation automatically. Currently, model transformation plays a key role in the Model Driven Development (MDD) [1]. However, model transformation is still perceived to be complex and has a tendency for errors-prone. Obviously, a strong theoretical foundation is required to ensure and guarantee the output model of model transformation closer to or meets the desired result. Likewise, the involvement of the specification and the verification process especially in checking consistency assist in ensuring the output model to match the desired result.
The proposed framework is applied on the well-example transformation from the object-oriented class model to relational database model which is presented in section 3. Thereafter, the weaknesses and strengths of Triple Graph Grammar will be discussed in section 4 and followed by the related work in section 5.

2 THE BIDIRECTIONAL MODEL TRANSFORMATION (BMT)

BMT is perceived as a proper approach in maintaining consistency between original source model and target model with output model in two directions verification process. To simplify, BMT can be viewed as M ↔ N, where M is defined as a source model (or meta-model) and N is as a target model (or meta-model). It means that model transformation can be performed in two ways is that forward and backward transformation as shown in figure 1.

![Figure 1. The Basic Concept of BMT](image)

In order to a BMT process can achieve the target model compliant original model and satisfy to the desired result then the proper specification and checking verification of BMT is required.

2.1 The Specification of BMT

The specification of BMT can be written in the textual and graphical form [6]. Some literatures have mentioned that a graph-based transformation language has become a suitable approach for specifying and applying model transformation especially MOF-based Model [7]. It consists of graph transformation rules and graph transformation system [8]. The main idea of BMT on graph-based is to create a relationship between source and target graph by correspondence graph.

Basically, a graph consists of a set of vertices V (or nodes) and a set of edges E such that each edge e in E has a source s(e) and a target vertex t(e) in V, respectively. The instance relation between the vertices and edges and their types is similar to the relation between objects and classes in object-oriented software engineering. It means that a vertex or edge type can contain a set of specific attributes and operations. In graph-based structure each edge is associated with a source and a target node (vertex). Formally, a graph contains a type graph and graph morphisms where type graph set as abstraction representation of the class diagram while graph morphisms are structure and type-preserving mapping between two graphs.

**Definition 1. (Typed Graphs)** Let \( L_V \) be a set of vertex types and \( L_E \) be a set of edge types; type graph \( \mathcal{G} \) from the possible set of graph \( G \) over \( L_V \) and \( L_E \) is characterized by 5-tuple \( \langle V, E, \text{scr}, \text{tgt}, \text{type} \rangle \), with two finite sets \( V \) and \( E \) of nodes (or vertices) and edges, two function \( \text{type}_V : V \rightarrow L_V \) and \( \text{type}_E : E \rightarrow L_E \) which assigns a type to each edge and node and two function \( \text{scr} : E \rightarrow V \) and \( \text{tgt} : E \rightarrow V \) that assign to each edge a source and target node.

**Definition 2. (Graph Morphism)** Let \( G = \langle V, E, \text{scr}, \text{tgt}, \text{type} \rangle \) and \( G' = \langle V', E', \text{scr}', \text{tgt}', \text{type}' \rangle \) be two graphs; then a graph morphism \( m : G \rightarrow G' \) consist of a pair of mapping \( \langle m_V, m_E \rangle \), with \( m_V : V \rightarrow V' \) and \( m_E : E \rightarrow E' \).

Graph transformation system uses rewriting technique to manipulate graphs which is defined by a set of graph production rules that contains a left-hand side (LHS) graph and a right-hand side (RHS) graph. LHS is associated with source model while RHS is associated with target model. As mentioned before, BMT is defined as a pair function of forward and backward transformation. Forward process encompasses the connection flows from source model (LHS) to target model (RHS). Meanwhile, backward process is depicted by a connection from target model to source model. In the graph transformation process, the forward transformation affects the target model which implicitly defines how to modify RHS so that it relates to LHS, whereas, the backward transformation affects the source model which implicitly defines how to modify LHS so that it relates to RHS. A graph transformation rule can be defined as follows:

**Definition 3. (Graph Transformation Rule)** A graph transformation rule \( p = (G_{\text{LHS}}, G_i, G_{\text{RHS}}, m_l, m_r) \) consists of three directed type graphs \( G_{\text{LHS}} \) as LHS of graph, \( G_i \) set as interface graph and \( G_{\text{RHS}} \) as right-hand side graph. \( G_i \) is just an auxiliary graph.

The morphisms \( m_l : G_i \rightarrow G_{\text{LHS}} \) and \( m_r : G_i \rightarrow G_{\text{RHS}} \)
In the context of model transformation, almost all formal work on BMT are based on graph grammars, especially triple graph grammars (TGGs) as introduced by Andi Schurr at the early nineties. TGG is a technique in the declarative way to define the correspondence between two different types of models [4, 8, 9]. Each TGG rule contains three graphs production; one operates on a source graph (SG), one on a target graph (TG) and one on a correspondence graph (CG). The CG describes a graph to graph mapping that relates elements of the source graph to elements of the target graph. Formally, TGG rules can be defined as follow:

**Definition 4. (Triple Graph Transformation Rule and Triple Graph Morphism)** Let \( tgg = \langle SG, CG, TG \rangle \), called source, connection and target graphs, with two graph morphism \( sG : CG \rightarrow SG \) and \( tG : CG \rightarrow TG \) form a triple graph \( G = \langle SG \leftarrow CG \rightarrow TG \rangle \). \( G \) is called empty, if \( SG \), \( CG \) and \( TG \) are empty graphs. A triple graph morphism \( m = \langle s, c, t \rangle : G \rightarrow H \) between two triple graphs \( G = \langle SG \leftarrow CG \rightarrow TG \rangle \) and \( H = \langle SH \leftarrow CH \rightarrow TH \rangle \) consists of three graph morphisms \( s : SG \rightarrow SH \), \( c : CG \rightarrow CH \) and \( t : TG \rightarrow TH \) such that \( s \circ sG = sH \circ c \) and \( t \circ tG = tH \circ c \). It is injective if morphism \( s \), \( c \), \( t \) are injective. All three graph grammar rule should be specified in one rule diagram.

### 2.2 The Consistency Framework for Verification of Bidirectional Transformation (VoBiT)

The verification of bidirectional Transformation is essential in MDD [10] to check the consistency between elements of two model through model transformation. Consistency problems arise because there is no definition of relationship between models preserving consistency. This research uses a declarative approach to define VoBiT. Declarative approaches tend to be easier to write and understand in software engineering community due to the use of graphic set.

In practice, checking consistency is rarely implemented. It happens normally when the output result model does not match with the input model and as a result the process of transformation will be re-run. Nevertheless, re-work will escalate such emergence costs of development. To avoid this unexpected circumstance, we proposed an approach to check the consistency between models and restore the inconsistency within model transformation especially bidirectional transformation. For this purpose, we use an descriptive approach to describe the result based on an approach proposed in [6].

BMT or bidirectional transformation should be formalized as \( M \leftrightarrow N \) or \( f: M \leftrightarrow N \) where there are two models such \( M \) is as a source model and \( N \) is as a target model. The verification of bidirectional transformation or VoBiT in shortly, contains checking consistency using GCCs. VoBiT algorithm is generated based on \( M \rightarrow M' \leftrightarrow N' \rightarrow N \) approach proposed by Steven in [5]. VoBiT framework presents as shown in figure 2 as below:

In detail, \( M \rightarrow M' \leftrightarrow N' \rightarrow N \) algorithm can be described as follows:

Firstly, we define all possibly relations.

- p: \( M \rightarrow M' \) \hspace{1cm} (a)
- g: \( N \rightarrow N' \) \hspace{1cm} (b)
- r: \( N' \rightarrow M' \) \hspace{1cm} (c)
- s: \( M' \rightarrow N' \) \hspace{1cm} (d)

Secondly, generated a new mapping from combination between two mapping processes:

- \( p \land s = M \rightarrow N' \) \hspace{1cm} (e)
- \( g \land r = N \rightarrow M' \) \hspace{1cm} (f)

In order to get easy understand how to check consistency in BMT based on the VoBiT algorithm, we simply explain them using Graphical Consistency Conditions (GCCs) as follows:

**Figure 2. Conceptual Framework of VoBiT**

**Figure 5. M \rightarrow M' of checking consistency**
The practical applicability of TGG in the specification of BMT and VoBiT framework are viewed by using the well-know example of the transformation from an object-oriented class model to a relational database model. Three graph grammar rules of TGG will be represented into one rule diagram within Generic Modeling Environment (GME) [11] is shown in Figure 10.

To transform an instance of the object-oriented meta-model into the relational model have the main rules which are used:

a. **Package-to-Schema**: every package in the class model corresponds to a schema with the same name as the package.

b. **Class-to-Table**: Every persistent class should be mapped to a table with the same name as the class. Furthermore, the table should have a primary-key where key is identical to the id of the class.

c. **Attribute-to-Column**: The Class attributes have to be appropriately mapped to columns, and some column may need to be related to other tables by foreign key definitions.

d. **Association-to-Fkey**: Association should be corresponded to foreign keys (Fkey). Fkey is created with refer to key which corresponding to the target class and is linked to the source class.

To implement the specification and checking consistency between the object-oriented class model and the relational database model a set of TGG rules must be created for each law. First law, the mapping from package to schema is described as follow (Figure 11). Package-to-schema rule matches package and creates the corresponding schema while each rule of the package-to-schema correspondence object i.e. instance of P2S has negative application conditions (NACs) which is assumed to be its RHS. Because of NACs, no additional schema objects will be created as a

Figure 6. $N \rightarrow N'$ of checking consistency

Figure 7. $M' \rightarrow N'$ of checking consistency

Figure 8. $N' \rightarrow M'$ of checking consistency

Figure 9. $N \rightarrow M'$ (a) and $M \rightarrow N'$ (b)

**3 EXAMPLE**

The practical applicability of TGG in the specification of BMT and VoBiT framework are viewed by using the well-know example of the transformation from an object-oriented class model to a relational database model. Three graph
package that is already connected to a schema by a P2S object.

Figure 11. Mapping package-to-schema (Forward)

Second law is the mapping from classes to tables is given in figure 12. The rule diagram shows in figure demonstrated a mapping between classes in a class diagram and tables in a relational database.

Figure 12. The Mapping Classes to Table (Forward)

The precondition of this rule is drawn in the bottom of the diagram. To apply this rule to its precondition, there must be a package mapped via a mapping to schema. The diagram shows an object of type class is mapped to an object of the type of table which has link to a key object or vice versa where the id of class is stored as name of the table’s key.

The next rule is to relate each data type in the object-oriented model and each type in the relational model the same way as it is done for the first law. To store the attributes in the columns and vice versa, we need to distinguish between attributes of a simple type and attributes that are classes. Therefore, we should specify two different TGG-rules. The first rules searches for all attributes of a class that are typed by a data type (Fig. 13). Subsequently, it creates a column for each attribute and assigns the column to the table of the class and to the type of the corresponding attribute. The second rule tries to match all attributes of a class that refer to another class. If this rule finds such a match, it creates a new column and assigns it to the Table of the Class to which the Attribute belongs. Setting the correct type of the Attribute of the Table of the Class is carried out by the mapping relation and the association that refers to the key within the Table.

Figure 13. The First TGG rules for attributes and column

Based on the literature’s point of view, TGG can generate three transformation rules which are forward transformation, backward (reverse) transformation and a connection rule that check the consistency between models. The forward transformation rules are described if a mapping from class diagram to a relational database exists and if the class diagram contains a class, a new table and a new key is created and its attributes are set according to the TGG rule. The newly created objects are marked by mapping to the class using a new mapping node. The reverse rule (backward rule) will create a class for every table in relational model. The matching and creation of object is undergone in the same manner as the forward rule. The last rule is the connection rules that needs a class diagram and a database and tries to relate them. This task will produce a consistency checking between two models. Note that only object created in these rules are involved in the mapping process.

When elements of two models are related by model transformation which unavoidable risk should be arise during the development of BMT called inconsistency. Consistency verification is required to ensure that the BMT executed correctly and implement ability. There are two levels of consistency in the language specification. First,
inconsistency may be occurred if a model does not conform to the meta-model. Second, the output model does not satisfy with well-formalness rules.

In order to get the correctness of BMT process that compliant the original source or model and satisfied with the expected result then throughout the entire development of BMT will be performed checking verification process by the implementation of consistency management. There are three strategies in the consistency management: consistency by construction, consistency by monitoring and consistency by analysis. All of them will be defined in the future work.

4 RELATED WORK

VIATRA-Visual Automated model Transformation [12] and VIATRA2 [13] are framework for transformation based on V&V to improve the quality of system design within UML by automatically checking consistency, completeness, and dependability requirement. The general purposes of its framework are to define and to implement the transformation process within a mathematically precise paradigm. The frameworks do not support bidirectional transformation, traceability and consistency maintenance between transformed models.

GReAT-Graph Rewriting and Transformation framework [14] is a transformation system for domain specific language (DSL) built on meta-modeling and graph rewriting concept. It contains the control structure to specify an initial context for the matching to reduce the complexity of the general matching case. The attribute transformation is specified by a proprietary attribute mapping language, whose syntax is close to C. LHS of the rules can contain OCL constraint to refine the structure but post condition are not supported. In addition, GReAT support only unidirectional transformation in batch-oriented way. No further traceability information about the current transformation is provided.

A framework for testing model transformation proposed in [15] focuses on the concept of model difference and mapping. Their framework provides number facilities such construction of test cases, test execution, test comparison and visualization of test result. However, test models are manually developed in their work. On the other hand, Prakash et al., [16] described the classification framework for model transformation into three different views or dimensions that comprise the transformation technique view, the transformation language support view and the generic view/dimension. They did not focus on how to transform one or more model to another model in detail what unidirectional or bidirectional.

Kuster et al. [17, 18] proposed ideas that Validation model transformation is necessary in developed model transformation. The authors focused on a concept of rule-based model transformation with control condition and provided a set of criteria to ensure termination and confluence as well as on syntactic correctness of rule-based model transformation. These approaches are concentrated on the functional behavior and syntactic correctness of the model transformation. In [19] presented rule-level verification approach to verify the semantic properties of business process transformation using CSP. The proposed approach did not support bidirectional transformation and traceability.

Ehrig et al. [4] defined the information preserving bidirectional model transformation. They used TGG to define BMT, which can be inverted without specifying a new transformation. We adopted their approaches especially to define the BMT using algebraic graph transformation. Although the approach sufficiently offered the concept of BMT, it still did not provide a way for checking consistency and traceability between two models.

5 CONCLUSION AND FURTHER WORK

This paper discussed TGG as a technology to define the specification of bidirectional model transformation. The proposed approach is to use declarative way to describe and to specify the bidirectional transformation by using graph grammars rules which are comprised pre- and post-condition. One of the most important features of TGG rules is the implicit creation of a correspondence graph between models. To present the practical applicability of TGG for model to model transformation, we used the well-known example of the transformation from an object-oriented class model to a relational database model. One of the benefits of TGG is to be able to check the consistency between two models and the ability to graphically specify transformations. This proves that these graphical transformation rules are really easier to specify and to maintain the consistency between models in BMT. For that purpose, we proposed a theoretical foundation to checking consistency using graphical consistency condition (GCC) called VoBiT. One way to validate this proposed framework to be appropriate
to use for maintaining coherency is to conduct empirical study.

TGGs are a well-suited technique to define the specification of BMT. However, it has still a number of weaknesses in order to make it to be more flexible thus it should be combined with other approaches. In future work, we will construct an appropriate framework to specify the BMT from the combination of TGG with other technique. Thus, it can ease the implementation of TGG in GME. Subsequently, after we have obtained the appropriate specification for BMT then we will proceed with the implementation of VoBiT in the industrial case study.

REFERENCE
