Verification of Bidirectional Model to Model Transformation in Model Driven Development

Lusiana
Sekolah Tinggi Manajemen Informatika & Komputer (STMIK) – AMIK Riau Pekanbaru
Jl. Purwodadi Indah Km. 10 Panam - Pekanbaru
Riau - Indonesia
lusi_dl@yahoo.co.id

Wan M.N. Wan-Kadir & Radziah Muhammad
Software Engineering Department
Faculty of Computer Science and Information Systems
Universiti Teknologi Malaysia
81300 UTM Skudai, Johor, Malaysia.
wnasir@utm.my, radzhia@utm.my

ABSTRACT
One of the prominent elements in MDD is model transformation where the developed model can be automatically transformed into other models. Many approaches were proposed to simplify the model transformation processes. However most of the processes focus on unidirectional transformation. In order to achieve full benefit of MDD, a pragmatic bidirectional model transformation (BMT) with a strong theoretical foundation is required. BMT must address reversible transformation in deducing forward and backward transformation automatically. This paper proposes a novel approach to BMT by introducing a pragmatic conceptual framework as a guideline in the specification and verification of BMT. In the proposed framework, metamodel and verification processes will be respectively specified using descriptive and procedural formal specification.

Categories and Subject Descriptors
D.2.4: Software/Program Verification
F.3.1 Specifying and Verifying

General Terms
Verification, Model-to-Model Transformation

Keywords
Model Driven Development, Bidirectional Model Transformation, Verification

1. INTRODUCTION
Model transformation is a key element in the implementation of the Model Driven Development (MDD). It is widely accepted in software development due to the automated generation of source code, other models and documentation by model transformation process [1]. However, model transformation is still perceived to be complex and has a tendency for error-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 specification and verification process also assists in ensuring the output model to match the desired result.

Many approaches were proposed to simplify the model transformation process, but most of approaches focus on unidirectional transformation instead of bidirectional transformation [2, 3]. In order to achieve full benefits of MDD (e.g. round-trip engineering), a pragmatic bidirectional model transformation (BMT) is required. BMT is a proper approach to reversible transformation and able to figure out forward and backward transformation automatically [4-6]. In the actual practice, users hardly focus on implementing specification and verification process at early stage. As a result, problems during implementation stage can not be avoided and lead to budget and schedule overrun. To overcome this problem, a framework is proposed to represent the specification of bidirectional model transformation and the consistency verification between two element models during development phase. The proposed framework is used as a foundation to build a Verification of Bidirectional Transformation (VoBiT) approach.

VoBiT consists of the metamodel and the verification process which is respectively specified by using descriptive and procedural formal specification. VoBiT aims to ensure and guarantee the output model of model transformation meets the desired result and matches the original source model, to check the consistency between elements of source metamodel and elements of target metamodel as well as to maintain the coherency between models by using graphical consistency condition (GCC).

The rest of the paper is structured as follows: section 2 presents the fundamental concept of the bidirectional model transformation which is equipped with the specification of bidirectional transformation and the verification of bidirectional transformation (VoBiT). The practical applicability of 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. Lastly, conclusion and future work will be presented in section 6.

2. THE BIDIRECTIONAL MODEL TRANSFORMATION (BMT)
Model transformation consists of transformation rules used to map input model as a source model to an output model as a target model. Unlike bidirectional model transformation, unidirectional model transformation is often used to generate most of model
transformations that exist today. However, BMT outperforms unidirectional model transformation in guaranteeing the reliability of output model since it implements two directions forward and backward transformation which verifies the output model against target model and original source model. Hence, 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 metamodel) and N is as a target model (or metamodel). It means that model transformation can be performed in two ways is that forward and backward transformation as shown in figure 1. Whilst, unidirectional transformation is represented as M → N where transformation is only one direction usually called forward transformation as shown in figure 2.

![Figure 1. Bidirectional Transformation](image)

![Figure 2. Unidirectional Transformation](image)

### 2.1 The Specification of BMT

The specification of BMT can be written in the textual and graphical form [7]. 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 [8]. It consists of graph transformation rules and graph transformation system [9]. 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 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 $TG$ from the possible set of graph $G$ over $L_V$ and $L_E$ is characterized by 5-tuple $(V, E, scr, tgt, type)$, with two finite sets $V$ and $E$ of nodes (or vertices) and edges, two function type $type: V → L_V$ and $type: E → L_E$ which assigns a type to each edge and node and two function $scr: E → V$ and $tgt: E → V$ that assign to each edge a source and target node.

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

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 (RHS) to source model (LHS). 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 $p = (GLHS, GI, GRHS, ml, mr)$ consists of three directed type graphs $GLHS$ as left-hand side graph, $GI$ set as interface graph and $GRHS$ as right-hand side graph. GI is just an auxiliary graph. The morphisms $m_l: GI → GLHS$ and $m_r: GI → GRHS$ are used to describe the correspondence between these graphs and maps the elements of the interface graph to either the LHS or RHS graph.

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 in the early nineties. TGG is a technique in the declarative way to define the correspondence between two different types of models [4, 9, 10]. Each TGG rule contains three graphs production; one 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 $RGG = (SG, CG, TG)$, called source, connection and target graphs, with two graph morphism $s_G : CG → SG$ and $t_G : CG → TG$ form a triple graph $G = (SG ← CG → TG)$. G is called empty, if $SG$, $CG$ and $TG$ are empty graphs. A triple graph morphism $m = (s, c, t) : G → H$ between two triple graphs $G = (SG ← CG → TG)$ and $H = (SH ← CH → TH)$ consists of three graph morphisms $s : SG → SH$ and $c : CG → CH$ and $t : TG → TH$ such that $s ∘ s_G = s_H ∘ c$ and $t ∘ t_G = t_H ∘ c$. it is injective if morphism $s$, $c$, $t$ are injective. All three graph grammar rule should be specified in one rule diagram.

After the explanation of theoretical foundation above, next section will discuss about how to check the consistency between element of source model and element of target model through model transformation. Consistency problems arise because there is no definition of relationship between models preserving consistency. Checking consistency between elements of two models in BMT is called VoBiT which is derived from the conceptual framework of BMT as shown in figure 3.
2.2 The Verification of Bidirectional Transformation (VoBiT)

The verification of bidirectional Transformation (VoBiT) is essential in MDD [11] to check the consistency between elements of two model through model transformation. 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 [7].

BMT or bidirectional transformation should be formalized as M ↔ N or f: M ↔ 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 → M’ ↔ N’ ← N approach proposed by Steven in [5]. VoBiT framework presents as shown in figure 4 as below:

In detail, M → M’ ↔ N’ ← N algorithm can be described as follows:

Firstly, we define all possibly relations.
- p: M → M’
- g: N → N’
- r: N’ → M’
- s: M’ → N’

Secondly, generated a new mapping from combination between two mapping processes:
- p ∧ s = M → N’
- g ∧ r = N’ → M’

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 5. M → M’ of checking consistency](image)

![Figure 6. N → N’ of checking consistency](image)

![Figure 7. M’ → N’ of checking consistency](image)

![Figure 8. N’ → M’ of checking consistency](image)

![Figure 9. N → M’ (a) and M → N’ (b)](image)
3. EXAMPLE

The practical applicability of TGG in the specification of BMT and VoBiT framework are viewed by using the well-known 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 Modelling Environment (GME) [12] is shown in Figure 10.

Figure 10. Metamodel Including the Mapping Relation

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 (or VoBiT) 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 package that is already connected to a schema by a P2S object.

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 Datatype 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 Datatype (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.
Based on the literature’s point of view, TGG can generate three transformation rules (Scufl, winter, 1995) 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 a 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.

4. DISCUSSION

The transformation example given that used TGG to specify the BMT process and TGG are suitable to define and to specify the simplification of inter-model transformation. However, we still find some problem when implementing TGG approach in GME. Currently, TGG approach can be found in FUJABA Tools, but in this paper, we used GME as modeling tools. Since it is a new tool and less guidelines are available regarding the use of GME for creating transformation models, we were facing difficulties in drawing transformation models. Consequently, it is necessary to optimize the rule matching algorithms in the specification of BMT especially using GME.

One of requirements that can make a task of specification of BMT rule easier is to provide an optimal support (guidelines) to undergo transformation model. This includes a tool that guides the user within the specification of the transformation that contains less constraint. Based on few of literatures, Fujaba can help the users more in specification of BMT but, it still need to be improved by providing a guideline to use it. Up to now, there is no theoretical concept to apply inheritance to graph grammar rules.

Based on the fact of the existing work, TGG still have a number of weaknesses. Although TGG are a suitable approach to define BMT for keeping the coherency between two models in similar structure, it is still not so well-suited. We believe that TGG can be combined with the other approaches such as templates, other operational approaches or hybrid transformation languages in order to be more flexible.

5. RELATED WORK

VIATRA-Visual Automated model Transformation [13]. It is a 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. However, in VIATRA2 [14] some initial ideas on incremental graph transformation for increasing the performance of the transformation mechanism have been presented.

GRReAT-Graph Rewriting and Transformation framework [15] is a transformation system for domain specific language (DSL) built on metamodeling 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 pattern matcher returns all the possible matches to avoid the inherent non-determinism in the matching process. 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 postcondition are not supported. In addition, GRReAT 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 [16] 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., [17] 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. [18, 19] 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 [20] 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. [21] 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.

6. 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.

7. REFERENCES


