An Approach for Validating Semantic Consistency of Model Transformation Based on Pattern

Jin Li, Dechen Zhan, Lanshun Nie, Xiaofei Xu

An Approach for Validating Semantic Consistency of Model Transformation Based on Pattern

Jin Li[1,2], Dechen Zhan[1], Lanshun Nie[1], Xiaofei Xu[1]

[1] School of Computer Science and Technology, Harbin Institute of Technology, 92 West Dazhi Street, Harbin 150001, China

[2] School of Computer Science and Technology, Harbin Engineering University, 145 Nan Tong Street, Harbin 150001, China

miaookok@163.com, { dechen, nls, xiaofei }@hit.edu.cn

Abstract. The correctness of model transformation is an import research field in model-driven architecture. Syntactic correctness and semantic consistency are hot topics in the field of model transformation. Syntactic correctness has many mature solutions. However the validation of semantic consistency has some problems. Therefore, how to validate semantic consistency of model transformation is a major problem in model-driven development. In this paper, we propose a validation approach for semantic consistency of model transformation, which is based on pattern. We analyze some patterns in models and make these patterns as transformation pattern. We define transformation rule with transformation pattern and analyze three parts of semantic transformation. We present two theorems to validate semantic consistency of model transformation. Finally, we give a case to illustrate the effectiveness of our approach.

Keywords: Model transformation, Transformation rule, Transformation pattern

1 Introduction

The correctness of model transformation is an important research field in model-driven architecture (MDA) [1-2]. The research mainly focuses on syntactic correctness and semantic consistency [3]. Syntactic correctness has some mature solutions [4], e.g. planning algorithm [5]. However the validation of semantic consistency has some problems, e.g. effective theory. Therefore, how to validate semantic consistency of model transformation is a major problem in model-driven development of software systems.

Model transformation consists of transformation rules which describe how a set of elements of the source model are transformed into a set of elements of the target model through transformation relationships [6]. Semantic consistency of model transformation is for maintaining consistency between source model and target model in the semantics. So, the validation problem for semantic consistency of model transformation is equivalent to the formulization proof of semantic consistency in the process of model transformation.

Many methods to solve semantic consistency of model transformation have emerged from industrial and academic research. Varró [7] defined and validated the model constraints to preserve semantic consistency of model transformation. Jinkui Hou [8] proposed a semantic description framework, and promoted category theory to describe and validate semantics of model transformation. Caplat [9] extended formal language to describe and validate model semantics. Engles [10] provided the relationships of semantic objects of UML-RT to describe the consistency required among models, and proved these relationships through static analysis. XiaoHe [11] extended QVT Relations with three new concepts and discussed the semantics of the mapping pattern and creating model.

Different model transformation methods may need different methods to preserve semantic consistency of model transformation. The paper proposes a validation approach for semantic consistency of model transformation based on pattern. We make some patterns, e.g. sequence pattern, branching pattern and loop pattern, in models as transformation patterns and use these transformation patterns to define transformation rule. There are three parts of the validation process of semantic consistency: (1) the semantic mapping from source model to source transformation pattern; (2) the semantic mapping from source transformation pattern to target transformation pattern; (3) the semantic mapping from target transformation pattern to target model. Then, the validation problem for semantic consistency of model transformation is equivalent to the problem about the three semantic mappings.

The rest of this paper is structured as follows. In Sect. 2, we propose the motivating example which will be used throughout the paper. Section 3 provides the core concepts. Section 4 presents the validation theory of semantic consistency of model transformation. Section 5 illustrates the validation theory. Sect. 6 concludes the paper and further work.

2 Motivation Example

There are some basic patterns in models, e.g. sequence pattern, branching pattern, and loop pattern, which belong to business process model. The three patterns are shown in Fig.1 (a). We use the model transformation from UML Activity Diagram Model (UADM) to Java Business Process Model (JBPM) to describe how to preserve model semantics during model transformation. The UADM and JBPM are shown in Fig.1(b). The UADM describes a business process of submitting sale order. The process is: firstly query sale data, secondly fill these data into a sale order, thirdly audit the sale order, and finally submit the sale order. The activity about auditing the sale order has a judging condition, i.e. if the sale data is less than 1000, the sale order should be submitted directly. Otherwise the manager should audit the sale order. If the manager agrees the sale order, he submits the sale order. Otherwise the sale data will be queried again.

The UADM contains these three patterns above. For example, the operations of querying sale data and filling the sale order correspond to the sequence pattern; the operation of checking the sale data corresponds to the branching pattern; the operation of auditing the sale order corresponds to the loop pattern, and the auditing loop pattern contains the sequence patter.

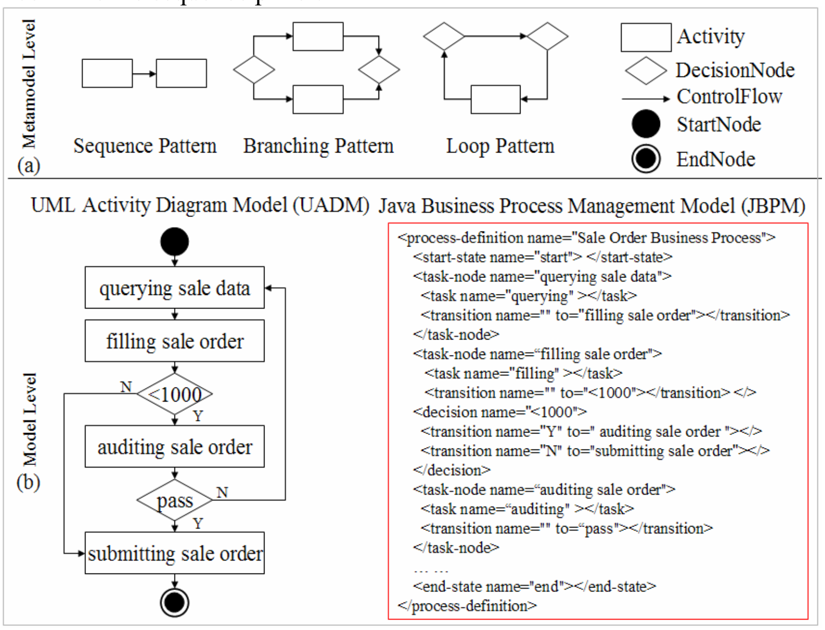

Fig. 1. Motivation example

The approaches of model transformation define transformation rules with the patterns. A pattern is either basic pattern or user-defined pattern. The user-defined pattern is generally defined according to domain business requirements. The patterns are called transformation patterns in transformation rules. These transformation patterns can simplify transformation definition, raise transformation efficiency, and improve transformation quality.

3 Transformation Pattern and Transformation Rule

Transformation rule, which is defined in model transformation based on pattern, contains two patterns: left pattern and right pattern. The two patterns also consist of some basic patterns and user-defined patterns. These basic patterns and user-defined patterns are called transformation pattern in the paper. Transformation patterns must be formed in pairs transformation rule, namely if left pattern contains a transformation pattern TP_s, right pattern should contain another transformation pattern TP_t. The semantics of TP_s and TP_t must be equivalence. The metamodel of transformation rule is shown in Fig.2.

The class *TransformationRule* consists of *LeftPattern*, *RightPattern* and *Constraint*. *LeftPattern* and *RightPattern* are composed of *Element* and *TransformationPattern*. *Element* has three subclasses, and they are *TransformationPattern*, *NodeElement*, and *RelationElement*. *TransformationPattern* is composed of *NodeElement*, *RelationElement*, and *Constraint*. *Constraint* has two subclasses: *interConstraint* and *exterConstraint*. *interConstraint* describes the internal relationships of *LeftPattern* and *RightPattern*, and *exterConstraint* describes the external relationships of *LeftPattern* and *RightPattern*. *interConstraint* can be described by OCL, while *exterConstraint* is described according to the constraint relationships of *NodeElement*, *RelationElement* and *TransformationPattern*. We focus on the semantic information of *exterConstraint* in the paper. We firstly analyze the external relationships of transformation pattern, which are illustrated in Fig. 3.

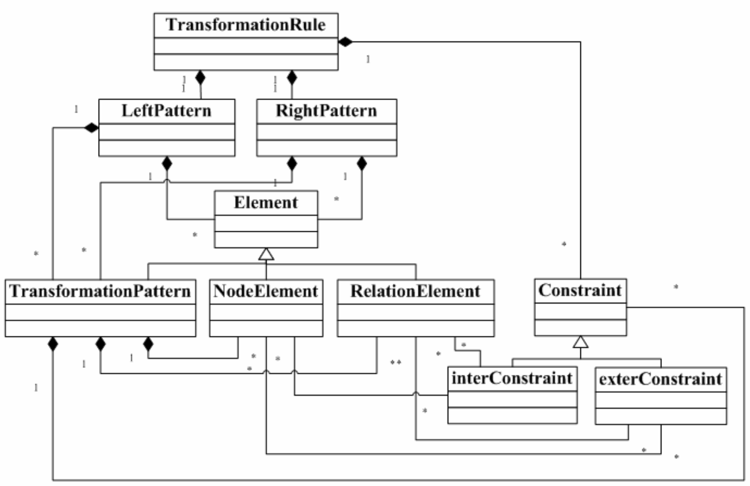

Fig. 2. Metamodel of transformation rule

In Fig. 3, there are two kinds of constraint relationships: Element-Pattern and Pattern-Pattern. In the first constraint relationship, there is a relationship r_1 between the element n_1 and the transformation pattern TP_1. The end element of r_1 is one of elements of TP_1. Because TP_1 contains two elements (n_2 and n_3), there are three conditions of the mapping between n_1 and TP_1: (1) from n_1 to n_2; (2) from n_1 to n_3; (3) from n_1 to n_2 and n_3. There needs an external constraint relationship to accurately describe the mapping condition between n_1 and TP_1. In the second constraint relationship, the identifier r_2 is a relationship between the transformation patterns TP_1 and TP_2. Because the start element of r_2 comes from TP_3 and the end element of r_2 comes from TP_2, we divide the constraint relationship (Pattern-Pattern) into two constraint relationships (Element-Pattern): the relationship between r_2 and TP_2, and the relationship between r_2 and TP_3. The two relationships are similar to the relationship between r_1 and TP_1.

We firstly propose the model definition, the definition of transformation pattern with the external constraint relationship, and the definition of transformation rule.

Definition 3.1 (Model): A model is defined as

$$M=<E, L, rel, meta> \tag{1}$$

Where

- $E=\{e_1,e_2,\ldots,e_s\}$ denotes a finite set of model elements,

- $L=\{l_1,l_2,...,l_t\}$ denotes a finite set of the relationships among model elements,
- $rel(l_k)=[e_i,e_j]$ denotes a relational function between model elements, and describes that e_i is a start element of l_k and e_j is an end element of l_k, $e_i,e_j\in E$, $1\leq i,j\leq s$, $l_k \in L$, $1\leq k\leq t$,
- $meta(e_u)$ denotes an instance function, and it describes the metamodel of e_u, $1\leq u\leq s$.

Note that we use "." to describe the element of model, e.g. *M.E* describes the element set of *M*, and *M.L* denotes the relationship set of *M*.

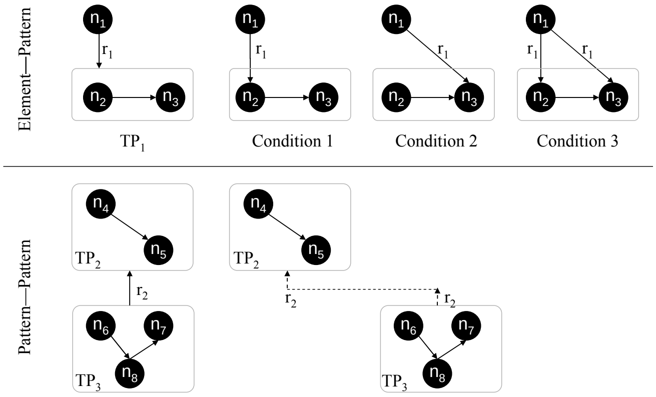

Fig. 3. The external relationship of transformation pattern

Definition 3.2 (Transformation Pattern): A transformation pattern is defined as

$$TP=<N, R, rel, interC, exterC> \qquad (2)$$

Where

- $N=\{n_1,n_2,...,n_p\}$ denotes a finite set of nodes, and its instance element set is *M.E*;
- $R=\{r_1,r_2,...,r_q\}$ denotes a finite set of relationships, and its instance element set is *M.L*;
- $rel(r_k)=[n_i,n_j]$ denotes an element-relationship function, and describes that n_i is a start element of r_k and n_j is an end element of r_k, n_i, $n_j\in N$, $1\leq i,j\leq p$, $r_k \in R$, $1\leq k\leq q$,
- $interC=\{iC_1,iC_2,...,iC_m\}$ denotes a finite set of the internal constraint relationships;
- $exterC=\{eC_1,eC_2,...,eC_n\}$ denotes a finite set of the external constraint relationships, $eC_x=<r,TP, \{n_x,n_{x+1},...,n_l\}>$ denotes the constraint relationship between r and TP, $r \notin R$, $1\leq x,l\leq p$. Every element n_k is either a start element or an end element of r, $n_k\in N$, $1\leq k\leq p$.

In Fig. 3, the three mapping condition of the constraint relationship between n_1 and TP_1 can be defined as the following:

$eC_1=<r,TP,\{n_2\}>$, $eC_2=<r,TP,\{n_3\}>$, $eC_3=<r,TP,\{n_2,n_3\}>$.

When a transformation rule contains an element, we make the element as a transformation pattern. Therefore, the left and right pattern of transformation rule can contain one or more transformation patterns.

Definition 3.3 (Transformation Rule): A transformation rule is defined as

$$TR=<LP, RP, C> \qquad (\mathbf{3})$$

Where

- $LP=\{tp_{s1},tp_{s2},...,tp_{su}\}$ denotes left pattern, and it contains a finite set of source transformation patterns,
- $RP=\{tp_{t1},tp_{t2},...,tp_{tv}\}$ denotes right pattern, and it contains a finite set of target transformation patterns;
- C denotes constraint relationship of LP and RP.

The left and right pattern are formed pairs in transformation rule, i.e. if left pattern contain the transformation pattern tp_{si}, right pattern should contain another transformation pattern tp_{tj}. The semantics tp_{si}, is similar to the semantics of tp_{tj}, $1 \le i \le u$, $1 \le j \le v$.

4 The Validating of Semantic Consistency of Model Transformation

The goal of preserving semantic consistency of model transformation is that the semantics of source and target model is equivalence. During a process of model transformation based on pattern, the semantic transformation process from source model to target model is an implementation process of transformation rules. The processes contain three parts of model semantic mappings: the semantic mapping from left pattern to source model, the semantic mapping from left pattern to right pattern, and the semantic mapping from right pattern to target model. Because the left and right patterns have the equivalent semantics, the problem of preserving model semantic consistency is similar to two the semantic mapping problem: the mapping from left pattern to source model, and the mapping from right pattern to target model. We will propose the theorems to solve the problem.

4.1 Semantic Mapping from source transformation pattern to source model

According to the definition 3.3, the left pattern of transformation rule is a set of source transformation patterns. Every element of source transformation pattern is either automatic element or another transformation pattern. So we describe the semantic mapping from left pattern to source model according to two mapping conditions. The identifiers M and TP denote a model and a transformation. If the semantics of M and TP is equivalence, we called the equivalence relationship as $TP \trianglelefteq M$. We provide a theorem to validate the semantic consistency of the mapping between transformation pattern and model.

Theorem 4.1 Let M be a Model, the node set of M is $M.E=\{ e_1,e_2,...,e_s\}$, TP is a transformation pattern, the node set of TP is $TP.N=\{n_1,n_2,...,n_p\}$. $TP \cong M$ if and only if Satisfying the following three conditions:

(1) every node $n_i \in TP.N$, then $n_i \in \cup$ meta($M.E$);

(2) every relationship $r_j \in TP.R$, then $r_j \in \cup$ meta($M.L$);

(3) every external constraint relationship $eC \in TP.exterC$ preserves the semantics.

Proof.

(1) Every node of transformation pattern is a certain metamodel element of model element. n_i is an element of TP, $1 \le i \le p$,

<i> if n_i is an automatic element, there exists a node $e_k \in M.E$ which satisfies n_i=meta(e_k). So $n_i \in \cup$ meta($M.E$); ①

<ii> if n_i is a transformation pattern, it should semantic map with a submodel M_i' of M. The node set of M_i' is a sub set of M nodes, i.e. $M_i'.E \subseteq M.E$. Every element n_l of n_i, there exists a node $e_o \in M_i'.E$ and it satisfies n_l=meta(e_o). So $n_i \in \cup$ meta($M.E$); ②

(2) Every relationship of transformation pattern is a certain metamodel relationship of model element. r_j is a relationship of TP, $1 \le j \le q$. According to the definition 3.2, $rel(r_j)=[n_{j1},n_{j2}]$, n_{j1} and n_{j2} are the elements of TP, i.e. $n_{j1},n_{j2} \in TP.N$. And according to ①②, because $n_{j1},n_{j2} \in \cup$ meta($M.E$), there exists two elements $e_u,e_v \in M.E$. They satisfy n_{j1}=meta(e_u) and n_{j2}=meta(e_v). To the relationship l_w between e_u and e_v, there exists $rel(l_w)=[e_u, e_v]$. So $r_j \in \cup$ meta($M.L$); ③

(3) The constraint relationship of transformation pattern satisfies semantic consistency. There are two parts of the constraint relationship: the internal constraint relationship and the external constraint relationship. Preserving the internal constraint relationship can be validated by OCL, while preserving the external constraint relationship is validated according to the definition of the external constraint relationship. There is an external relationship r between the element n_i and the transformation patter TP:

<i> if n_i is an automatic element, there exists an instance element $l_k(l_k \in M.L)$ of r and $rel(l_k)=[e_i,e_j]$ $(e_i,e_j \in M.E)$. According to ①②, n_i satisfies either n_i=meta(e_i) or n_i=meta(e_j), then the external constraint relationship $eC=<r,TP,\{n_i\}>$ preserves the semantics of r and TP; ④

<ii> if n_i is a transformation pattern TP_i, according to ②, there exists a sub model M_j' of M, and $n_i \cong M_j'$, then a certain element n_k of TP_i may be the start or end element of r, $n_k \in TP_i.N$. According to ④, the external constraint relationship $eC=<r,TP_i,\{n_k\}>$ preserves the semantics of r and TP. ⑤

4.2 Semantic Mapping from target transformation pattern to target model

In the session, we will propose the construction process from target transformation pattern to target model, and provide a theorem to validate whether the process preserve the semantics.

TP_s and TP_t are the source and target transformation patterns, and their corresponding mapping models are M_s and M_t. The construction process from TP_t to M_t is the following:

(1) Constructing element. n_t is an element of TP_t, $n_t \in TP_t.N$,

<i> if n_t is an automatic element, according to the instance relationship of the metamodel and model, constructing a new model element e_i and make n_t=meta(e_i); ⑥

<ii> if n_t is a transformation pattern, according to the corresponding element n_s of TP_s:

a) if n_s is an automatic element, constructing a new element e_i, and make $n_t=meta(e_i)$; ⑦

b)if n_s is a transformation pattern, constructing a sub model M_t', and make $n_t.N=meta(M_t'.E)$; ⑧

(2) Constructing relationship. r_k is a relationship of TP_t, $r_k \in TP_t.R$. There exists two elements (n_i,n_j) and they satisfy $rel(r_k)=[n_i,n_j]$, $n_i,n_j \in TP_t.N$,

<i> if n_i and n_j are two automatic elements, according to ①, their instance elements (e_{ti}, e_{tj}) are the automatic elements of M_t. n_i and n_j satisfy $n_i=meta(e_{ti})$ and $n_j=meta(e_{tj})$. Then, constructing a relationship l_{tk} from e_{ti} to e_{tj}, and make $rel(l_{tk})=[e_{ti},e_{tj}]$, $l_{tk} \in M_t.L$; ⑨

<ii> if n_i is a transformation pattern and n_j is an automatic element. According to ②,n_i corresponds to a sub model M_{ti}' of M_t, $M_{ti}'.E=\{e_{ti1},e_{ti2},\ldots,e_{tim}\}$. If there exists an external constraint relationship $eC=<r,n_i,\{e_{tj}\}>$, the model element e_{tk} will be found in $M_{ti}'.E$ Then constructing a relationship l_k from e_{tk} to e_{tj}, and make $rel(l_k)=[e_{tk},e_{tj}]$. In the same way, if n_i is an automatic element and n_j is a transformation pattern, there exists $M_{tj}'.E=\{e_{tj1},e_{tj2},\ldots,e_{tjn}\}$. According to the external constraints relationship $eC=<r,n_j,\{n_i\}>$, there exists an element e_{tk}' in $M_{tj}'.E$, constructing a relationship l_k' from e_{ti} to e_{tk}', make $rel(l_k')=[e_{ti}, e_{tk}']$; ⑩

<iii> if n_i and n_j are two transformation patterns, according to ②, n_i and n_j are corresponds to the sub model M_{ti}' and M_{tj}', and these sub models satisfy $M_{ti}'.E=\{e_{ti1},e_{ti2},\ldots,e_{tim}\}$ and $M_{tj}'.E=\{e_{tj1},e_{tj2},\ldots,e_{tjn}\}$.According to the external relationships $eC_1=<r,n_i,\{e_{tj1},e_{tj2},\ldots,e_{tjn}\}>$ and $eC_2=<r,n_j,\{e_{ti1},e_{ti2},\ldots,e_{tim}\}>$, if there exists two elements e_{tk} and e_{tk}' in $M_{ti}'.E$ and $M_{tj}'.E$, constructing a relationship l_k from e_{tk} to e_{tk}', and make $rel(l_k)=[e_{tk}, e_{tk}']$. ⑪

Theorem 4.2. Let TP_s and TP_t be two transformation patterns, if they correspond to the source M_s and target model M_t, the semantics of M_s and M_t is equivalence.

Proof.

There are three parts of the semantic transformation of source model: the semantic mapping from M_s to TP_s, the semantic mapping from TP_s to TP_t, and the semantic mapping from TP_t to M_t. According to definition 3.3, the semantics of TP_s and TP_t is equivalence. According to the theorem 4.1, the semantics of M_s and TP_s is equivalence. So the preserving semantic problem of M_s and M_t is equivalent to the preserving semantic problem of TP_t and M_t. Because M_t is constructed by TP_t, the preserving semantic problem only validates the semantic equivalence of element and relationship of M_t. The validation of preserving the semantic equivalence of element and relationship are the following:

(1) Element equivalence. n_t is an element of TP_t, $n_t \in TP_t.N$,

<i> if n_t is an automatic element, according to ⑦, the instance element e_i of n_t satisfies n_t=meta(e_i), then $TP_t.N \cong M_t.E$;

<ii> if n_t is a transformation pattern, according to ⑧, the instance sub model M_t' of n_t satisfies $n_t.N=meta(M_t'.E)$, then $TP_t.N \cong M_t.E$;

(2) Relationship equivalence. r_k is a relationship of TP_t, $r_k \in TP_t.R$. There exists two elements (n_i, n_j) and they satisfy $rel(r_k)=[n_i,n_j]$, $n_i,n_j \in TP_t.N$,

<i> if n_i and n_j are two automatic elements, according to ③⑥, their instance elements are e_{ti} and e_{tj} which are two elements of M_t. According to ⑨,there exists the instance relationship l_{tk} of l_k, and $rel(l_{tk})=[e_{ti},e_{tj}]$, So $TP_t.R \cong M_t.L$;

<ii> if n_i is a transformation pattern and n_j is an automatic element. According to ③⑩, there exists an element e_{tk} in the instance model M_{ti}' of n_i, and a relationship l_k from e_{tk} to e_{tj}. The relationship satisfies $rel(l_k)=[e_{tk},e_{tj}]$, so $r_k \cong M_{ti}$'; In the same way, if n_i is an automatic element and n_j is a transformation pattern, there exists a relationship l_k', so $TP_t.R \cong M_t.L$;

<iii> if n_i and n_j are two transformation patterns, according to③⑪, if there exists e_{tk} and e_{tk}', the relationship l_k between e_{tk} and e_{tk}', and $rel(l_k)=[e_{tk}, e_{tk}']$, so $TP_t.R \cong M_t.L$.

5 Experiment

We validate the semantic equivalence of UADM and JBPM. There are three parts of the validation process: (1) preserving semantic equivalence of UADM and TP_s; (2) preserving semantic equivalence of TP_s and TP_t; (3) preserving semantic equivalence of TP_t and JBPM. The validation process is the following:

(1) Preserving semantic equivalence of UADM and TP_s

The metamodel of UADM contains *Activity*, *Decision*, *ConstrolFlow*, *Start* and *End*. There are three basic patterns in UADM: sequence pattern, branching pattern and loop pattern. We firstly define three transformation patterns according to these basic patterns.

(a) A source sequence pattern is composed of *querying sale data*, *filling sale order* and their relationship in UADM. It is defined as TP_{SS}=<N_{SS}, R_{SS}, rel_{SS}, $interC_{SS}$, $exterC_{SS}$>, where N_{SS}={*Activity*, *ControlFlow*}, $R_{SS}=\{r_1,r_2\}$, $rel_{SS}(r_1)$=[*Activity*, *ControlFlow*], $rel_{SS}(r_2)$ =[*ControlFlow*, *Activity*], $exterC_{SS}=\{eC_1\}$, eC_1=<r_{s1}, TP_{SS},{*Activity*}>.

Note that r_{s1} is an external relationship, $r_{s1} \notin rel_{SS}$.

(b) A source branching pattern is composed of *auditing sale order*, *pass*, and TP_{SS}. It is defined as TP_{SB}=<N_{SB}, R_{SB}, rel_{SB}, $interC_{SB}$, $exterC_{SB}$>, where N_{SB}={*Activity*, *Decision*, *ControlFlow*, TP_{SS}}, $R_{SB}=\{r_3,r_4,r_5,r_6\}$, $rel_{SB}(r_3)$=[*Decision*, *ControlFlow*], $rel_{SB}(r_4)$=[*ControlFlow*, *Activity*], $rel_{SB}(r_5)$=[*Activity*, *ControlFlow*], $rell_{SB}(r_6)$= [*ControlFlow*, *Decision*], $exterC_{SB}=\{eC_2\}$, eC_2=<r_{s2}, TP_{SB},{ *Decision*}>

Note that r_{s2} is an external relationship, $r_{s2} \notin rel_{SB}$.

(c) A source loop pattern is composed of less 1000, *auditing sale order*, *pass*, and TP_{SS}. It is defined as TP_{SL}=<N_{SL}, R_{SL}, rel_{SL}, $interC_{SL}$, $exterC_{SL}$>, where N_{SL}={*Activity*, *Decision*, *ControlFlow*, TP_{SS}}, R_{SL}={r_7,r_8,r_9,r_{s1}}, $rel_{SL}(r_7)$=[*Decision*, *ControlFlow*], $rel_{SL}(r_8)$=[*ControlFlow*,TP_{SS}], $rel_{SL}(r_{s1})$=[TP_{SS}, *ControlFlow*], $rel_{SL}(r_9)$ =[*ControlFlow*, *Decision*], $exterC_{SL}$={eC_3}, eC_3=<r_6, TP_{SL},{*Activity*}>.

According to the theorem 4.1, we validate the semantic equivalence of source transformation patterns and UADM. The instance model of TP_{SS} contains the elements *querying sale data* and *filling sale order*. When the auditing order is error, the sale data should be queried again. So there is a relationship between TP_{SS} and r_{s1}. TP_{SS} contains an external relationship eC_1=<r_{s1}, TP_{SS},{*Activity*}> to describe the relationship. When the auditing order is ok, there exists a relationship between TP_{SL} and TP_{SB}. Then there is a relationship r_6 between TP_{SL} and TP_{SB}. TP_{SL} contains an external relationship eC_3=<r_6, TP_{SL},{*Activity*}> to describe the relationship. So the transformation patterns (TP_{SS}, TP_{SB}, TP_{SL}) preserve the semantic equivalence of M_s.

(2) Preserving semantic equivalence of TP_s and TP_t

We firstly define three target transformation patterns. JBPM contains *TaskNode*, *DecisionNode*, and *Transition*.

(d) Target sequence pattern contain two *TaskNodes* and a *Transition*. Target sequence pattern TP_{TS}=<N_{TS}, R_{TS}, rel_{TS}, $interC_{TS}$, $exterC_{TS}$>, where N_{TS}={*TaskNode*, *Transition*}, R_{TS}={r_{10},r_{11}}, $rel_{TS}(r_{10})$=[*TaskNode*, *Transition*], $rel_{TS}(r_{11})$=[*Transition*, *TaskNode*], $exterC_{TS}$={eC_4}, eC_4=<r_{t1}, TP_{TS},{*TaskNode*}>

(e) Target branching pattern contain a *TaskNode*, a *DecisionNode*, a *Transition* and a TP_{TS}. It is defined as TP_{TB}=<N_{TB}, R_{TB}, rel_{TB}, $interC_{TB}$, $exterC_{TB}$>, where N_{TB}={*TaskNode*, *DecisionNode*, *Transition*, TP_{TS}}, R_{TB}={r_{12},r_{13},r_{14},r_{t1}}, $rel_{TB}(r_{12})$= [*DecisionNode*, *Transition*], $rel_{TB}(r_{13})$=[*TaskNode*, *Transition*], $rel_{TB}(r_{14})$=[*TaskNode*, *DecisionNode*], $rel_{SB}(r_{t1})$=[*DecisionNode*, TP_{TS}], $exterC_{TB}$={eC_5}, eC_5=<r_{t2}, TP_{TS}, {*TaskNode*}>

(f) Target loop pattern contain two *DecisionNodes*, a TP_{TS}, and a *TaskNode*. It is defined as TP_{TL}=<N_{TL}, R_{TL}, rel_{TL}, $interC_{TL}$, $exterC_{TL}$>, where N_{TL}={*TaskNode*, *DecisionNode*, *Transition*, TP_{TS}}, R_{TL}={r_{15},r_{16},r_{17},r_{18},r_{t1}}, $rel_{TL}(r_{15})$= [*DecisionNode*, TP_{TS}], $rel_{TL}(r_{16})$=[TP_{TS}, *TaskNode*], $relT_L(r_{17})$= [*DecisionNode*, *TaskNode*], $rel_{TL}(r_{18})$=[*TaskNode*, *DecisionNode*], $exterC_{TL}$={eC_6}, eC_6=<r_{t3}, TP_{TL}, {*TaskNode*}>

The semantics of TP_s and TP_t is equivalence. This is not the focus of this paper, and therefore, we will not describe it here.

(3) Preserving semantic equivalence of TP_t and JBPM

M_t is constructed by TP_t. According to the theorem 4.2, the semantics of the constructed elements and relationships is equivalence is equivalent to the semantics of M_s. So the mapping between TP_t and M_t preserves the semantic equivalence.

As noted above, the semantics of UADM to JBPM is equivalence.

6 Conclusions and Future Work

In this paper we propose an approach for validating semantic consistency of model transformation. We analyze some basic patterns in models, e.g. sequence pattern, branching pattern, and loop pattern, and use these basic patterns to define transformation rules. Therefore, the semantic transformation of source model has been divided three parts: (1) the semantic mapping from source transformation pattern to source model; (2) the semantic mapping from source transformation pattern to target transformation pattern; (3) the semantic mapping from target transformation pattern to target model. The validation problem for semantic consistency of model transformation is equivalent to the problem about the three semantic mappings. The motivation example illustrates the effectiveness of our approach.

Future work is to optimize transformation rules constructed through our approach transformation. For this reason, we plan to analyze the typical business patterns in models and compose some transformation rules using these patterns to improve the efficiency of model transformation.

Acknowledgments. Research works in this paper are supported by the National Natural Science Foundation of China (60773064, 60904080), the National High-Tech Research and Development Program of China (2009AA04Z153, 2008GG1000401028).

References

1. Miller J, Mukerji J. MDA Guide Version 1.0.1 [EB/OL]. OMG Document number omg/2003-06-01. http://www.omg.org/docs/omg/03-06-01.pdf, 2003.
2. Brent Hailpern, Peri Tarr. Model-driven development: The good, the bad, and the ugly [J]. IBM System Journal, 2006, 45(3): 451-461.
3. Beydeda S, Book M, Gruhn V. Model-Driven Software Development-Volume of Research and Practice in Software Engineering [M]. Berlin: Springer, 2005.
4. J.H.Hausmann, R.Heckel, and S.Sauer. Extended model relations with graphical consistency conditions. In UML 2002 Workshop on Consistency Problems in UML-based Software Development, 2002:61–74.
5. Varró D, Varró G, Pataricza A. Designing the automatic transformation of visual languages. Science of Computer Programming, 2002, 44 (2):205–227.
6. Kleppe A, Warmer J, Bast W. MDA Explained: The Model Driven Architecture: Practice and Promise, Addison-wesley, Boston, 2003.
7. Varró D, Pataricza A. Automated Formal Verification of Model Transformation. In: Proceedings of Workshop on Critical Systems Development with UML (CSDUML 2003), Technische Unviersitat Munchen, 2003, 63-78.
8. Jinkui Hou, Haiyang Wang, Jun Ma et al. Semantic Description Framework for Architecture-Centric Model Transformation. Journal of Software, 2009, 20(8):2113-2123.
9. Caplat G, Sourrouille JL. Model Mapping Using Formalism Extensions [J]. IEEE Software, 2005, 2(22):44-51.
10. Engels G, Heckel R, Kuster JM, et al. Consistency-Preserving Model Evolution Trough Transformations [C]. In: Proceedings of UML’02, LNCS2460, Heidelberg: Springer-Verlag, 2002, 212-227.
11. Xiao He, Zhiyi Ma, Yan Zhang, Weizhong Zhang. Extending QVT Relations for business process model transformation. Journal of Software, 2011, 22(2):195-210.