A Construction Approach of Model Transformation Rules Based on Rough Set Theory

Jin Li, Dechen Zhan, Lanshun Nie, Xiaofei Xu

A Construction Approach of Model Transformation Rules Based on Rough Set Theory

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. Model transformation rules are the central part of model transformation. Many model transformation approaches provide some mechanisms to construct transformation rules in industrial and academic research. However, transformation rules are typically created manually in these approaches. As far as we know, there are no complete solutions that construct transformation rules automatically. In this paper, we propose a rough set based approach to construct transformation rules semi-automatically. Construction approach of rough set is improved in order to support the transformations between different meta-models, then the corresponding algorithm to construct transformation rules is presented. We also provide the measurement indicators of transformation rules to support selecting proper rules from many rules which meet transformation requirement. Three kinds of experiments for problems with distinct complexity and size are given for the validation of the proposed method.

Keywords: Model transformation, Model transformation rules, Rough set theory

1 Introduction

Model-driven architecture (MDA) is an approach for the development of software systems. Model transformation is a core part of MDA and plays an indispensable role in many different application domains, for instance, to generate code from models, to derive higher-level models from legacy models, to support model driven interoperability, or to compose service models for enterprise interoperability.

However, model transformation involves many repetitive difficulties. 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 [2]. With increasing in number and size of models, the relationships implicated among models reflect gradually more and more uncertainty, incompleteness and inconsistency, etc. Therefore, the efficient design of transformation rules has become a major challenge to model transformation.

A lot of researchers in both academic and industry study on how to implement model transformation and construct transformation rules. For example, model transformation has been well achieved especially in database domain [3].Usually there are two kinds of approaches for constructing transformation rules. One is that domain experts and model transformation designers capture mapping relationships between source and target metamodel elements, then define transformation rules based on these mapping relationships. It depends on domain experts and model transformation designers to not only understand the knowledge of modeling and model transformation, but also discover the semantic relationships among source and target models, especially implicit relationships. The other is that matching relationships can be discovered from a set of input meta-models through data mining methods [4], and then transformation rules are designed according to these matching relationships [5]. It gives enough attention to meta-model transformations, however complex and uncertain relationships among source and target models are difficult to derive and selection from rules set is also a challengeable task.

We introduce rough set [6-8] into the construction process of transformation rules. In doing so, we try to discover both explicit and implicit matching relationships and specify transformation rules. Then we also provide measurement indicator for selecting the proper transformation rules [9]. 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 and main algorithms and illustrates the construction process of the example. Section 4 designs three kinds of experiments in order to respectively analyze three quantities' influence on transformation rules. Sect. 5 presents related work. Finally, Sect. 6 concludes the paper and further work.

2 Motivating Example

As the transformation from UML class diagram to relation database model is very common in the field of software system development, in this section, we introduce an example, entitled UCD2RDM, to construct transformation rules which describe how UML class diagram is transformed into relational database model. Both source and target metamodels are illustrated in Fig. 1.

UML class diagram consists primarily of *Class*, *Relationship* and *Property*. Each *Class* has zero or more *Properties*. The type of each *Property* can be defined by built-in type or another *Class*. The relation among *Classes* is specified by *Relationship*. *Relationship* can be extended to describe the more concrete relations, for example *Association*, *Composition*, *Aggregation*, *Generalization* and *Dependency*, etc. Relational database model is composed of *Table*, *FKey* and *Column*. A *Table* contains one or more *Column*, and zero or more *FKey*. An *FKey* is also a *Column*.

We define a source model conforming to UML class diagram and a target model conforming to relational database model. Both models describe the relationships among teachers, students and courses of some educational institution.

The class *Teacher* can teach one or more courses (*Course*). The class *Student* can select one or more courses (*Course*). According to the teacher's title information, *Teacher* has a subclass (*Professor*). *Student* has also a subclass (*Master*). Only the

class *Professor* can supervise the master (*Master*). The model of the educational institution is shown in Fig. 2.

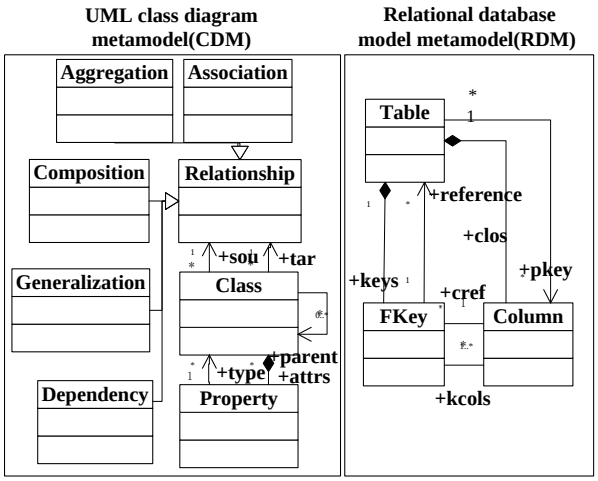

Fig. 1. Source and target metamodels of UCD2RDM.

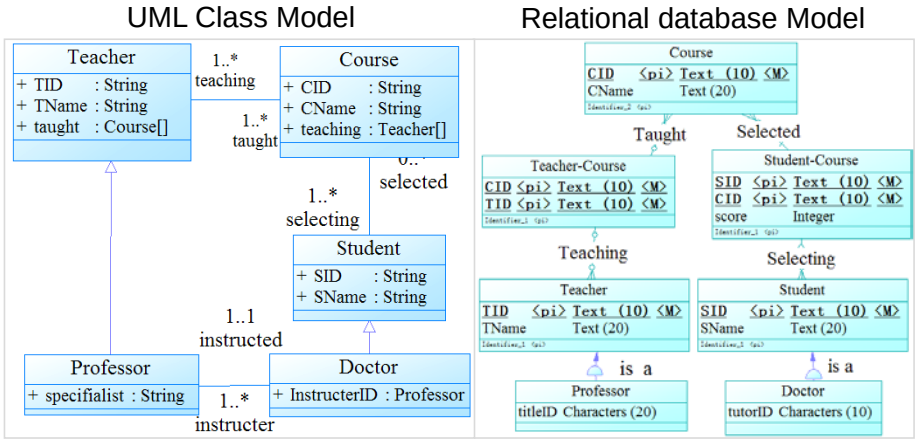

Fig. 2. Source and target models of UCD2RDM.

In Fig. 2, there are three kinds of transformation rules which are used to transform elements of UML class diagram into elements of relational database model.

- The first rule describes how to transform *Class* into *Table* through the mapping between the properties of *Class* and the columns of *Table*;
- The second rule expresses the correspondence between *Relation* and *Table*. It consists of the matching relationships between the *Relation*'s name and the *Table*'s name, and the mapping relationship between the crucial properties of *Relation* and the foreign column of *Table*;
- The last rule presents the mapping relationships between the identity property of *Class* and the important column of *Table*.

3 Rough Set and Discovered Rule Based on Rough Set

3.1 Rough Set Theory

Rough set theory is a mathematical tool to deal with vagueness and uncertainty. It has capability to effectively analyze uncertain, incomplete and inconsistent data, discover dependent and implicit relations, and construct mapping rules [10]. The process of constructing rule in rough set is composed of three steps. The first step is to classify the data according to the equivalence relations. The second step is to format the data in order to generate the decision table. The last step is to calculate the data belong to the decision table and construct the mapping rules [11].

This paper extends rough set theory to construct transformation rules with three major improvements. Firstly, we present more precisely domain using source and target metamodels in order to discover the mapping among source and target metamodels. Secondly, we redefine the traditional property set to get a new property set that has two kinds of properties: composed by source metamodels and decision properties composed by target metamodels. Finally, we extend the decision table and other correlative concepts. The work will help discover and construct transformation rules.

3.2 Basic Concept

For the convenience of description, we introduce some related definitions of model transformation [18]. We also extend some basic notions of rough set theory (i.e. decision tables and decision function) that will be useful in this paper.

Definition 3.1 (Directed-role graph): A directed-role graph is defined as

$$RG = (L, \otimes, \oplus) . \quad (1)$$

Where

- $L=N \cup R \cup U \cup D$ is a non-empty finite set of alphabets,
- N is a non-empty symbol finite set of nodes,
- R is a non-empty symbol finite set of relations,
- U is a non-empty symbol finite set of roles,
- D is a non-empty symbol finite set of domain,
- $\otimes(R)=[\ldots,U\times N,\ldots]$ is a relational function , and describes that the node plays a role in the relation,
- $\oplus(N,R,U)=(\min(N,R,U),\max(N,R,U))$ is a cardinality function, and it presents times that a node plays a special role in the relation. $\min(N,R,U)$ denotes the minimal times and $\max(N,R,U)$ denotes the maximal times.

Definition 3.2 (Model): A model is defined as

$$M=(RG, \gamma, \xi) . \quad (2)$$

Where $RG=(L, \otimes, \oplus)$ is a directed-role graph; γ is also a directed-role graph, and it can be denoted as $RG_\gamma=(L_\gamma, \otimes_\gamma, \oplus_\gamma)$; each L has a mapping function ξ: $L \rightarrow L_\gamma$, and ξ describes all of elements, such as node, relation, role and domain, come from the finite symbol set of RG_γ.

Definition 3.3 (Metamodel): Given two model $M_1=(RG_1, \gamma_1, \xi_1)$, $M_2=(RG_2, \gamma_2, \xi_2)$.
if $\gamma_1=M_2$, that is $RG_\gamma=(L_\gamma, \otimes_\gamma, \oplus_\gamma)=(RG_2, \gamma_2, \xi_2)$
then M_2 is called the metamodel of M_1, which denotes $M_1:M_2$.

Definition 3.4 (Model match): A model match is defined as

$$MMatch = \text{Match}(\{sm\}:MM_s, \{tm\}:MM_t) . \quad (3)$$

MMatch builds the relationships between the source$\{sm\}$ and target$\{tm\}$ metamodels

Definition 3.5 (Model transformation): A model transformation is defined as

$$MT = \cup \text{Match}(\{sm\}:MM_s, \{tm\}:MM_t) . \quad (4)$$

MT is a process in which elements of source models are transformed into elements of target models according to *MMatch.*

Definition 3.6 (Model transformation rule): A model transformation rule is defined as

$$MTR=\{sm\}:MM_s \rightarrow \{tm\}:MM_t . \quad (5)$$

If the conditions of the source metamodel are satisfied, the conditions of the target metamodel would be deduced.

Because model transformation language ATL provides simply structure, we use it to describe transformation rules.

Definition 3.7 (Decision table of transformation rules): A decision table of transformation rules is defined as

$$RT = (M, V, \Gamma) . \quad (6)$$

Where

- $M=SM \cup TM$ is a non-empty set of finite model elements,
- $SM=\{sm_1, sm_2, \ldots, sm_i\}$ is the finite element set of the source model,
- $TM=\{tm_1, tm_2, \ldots, tm_j\}$ is the finite element set of the target model,
- $V=V\gamma_{SM} \cup V\gamma_{TM}$ is the set of values that associate for every element of metamodel, $\gamma_{SM}=\{smm_1, smm_2, \ldots, smm_k\}$ is the finite element set of the source metamodel and $\gamma_{TM}=\{tmm_1, tmm_2, \ldots, tmm_l\}$ is the finite element set of the target metamodel, in general, γ_{SM} is called the decision property set and γ_{TM} is called the decision property,
- Γ: $M \times \gamma_M \rightarrow V$ is a determine function, $\gamma_M=\gamma_{SM} \cup \gamma_{TM}$ is the finite element set of the source and target metamodel, and Γ determines the metamodel element of each model element.

For $\forall mm \in \gamma_{SM}$, $sm \in SM$, then has

$$V_{\gamma_{SM}} = \bigcup_{mm \in \gamma_{SM}} V_{mm}, \Gamma_s : SM \times \gamma_{SM} \rightarrow V_{\gamma_{SM}} . \quad (7)$$

For $\forall mm \in \gamma_{TM}$, $tm \in TM$, then has

$$V_{\gamma_{TM}} = \bigcup_{mm \in \gamma_{TM}} V_{mm}, \Gamma_T : TM \times {}_{TM} \to V_{\gamma_{TM}} . \quad (8)$$

In decision table decision properties can be unique or not. When some condition properties are satisfied, decisions, operations and actions in decision properties will be executed.

Definition 3.8 (Indiscernibility relation): Let $RT=(M, V, \Gamma)$ be a decision table, and $B \subseteq \gamma_{SM}$. Then an indiscernibility relation Δ_B is generated from B and SM with the form

$$\Delta_B = \{(w_1,w_2) \in SM \times SM : \Gamma(w_1,smm) = \Gamma(w_2,smm), \forall smm \in B\} \quad (9)$$

if $(w_1,w_2) \in \Delta_B$, then w_1 and w_2 are called as the indiscernibility relation of B.

Indiscernibility relation is the main concept of rough set theory. The source model is divided into the equivalence class according to indiscernibility relation, which is denoted as SM/Δ_B.

Definition 3.9 (Indiscernibility relation of the decision value): Let $RT=(M, V, \Gamma)$ be a decision table, and $B \subseteq \gamma_{SM}$. Then an indiscernibility relation of the decision value Δ_B is generated from B and RT with the form

$$\begin{aligned} &\delta_B : SM \to \rho(V\gamma_{TM}) \\ &\delta_B(w) = \{ i : \exists w' \in SM \text{ s.t. } w\Delta_B w', t(w) = i, 1 \le i \le r(\gamma_{TM}) \} . \end{aligned} \quad (10)$$

Where $\rho(V\gamma_{TM})$ denotes the power set of $V\gamma_{TM}$, and $t(w)$ denotes the value of model element w in the decision property set γ_{TM}.

Definition 3.10 (Indiscernible decision table): Let $RT=(M, V, \Gamma)$ be a decision table, and $B \subseteq \gamma_{SM}$. δ_B is an indiscernible decision function; $SM/\Delta\gamma_{TM}$ is the indiscernible partition over the source model SM,

$$\text{if } \theta(\delta_B) = \{ \forall (p, q) \in SM \times SM, \text{ then } \delta_B(p) = \delta_B(q) \} .$$

Where RT is called the indiscernible decision table; $SM/\theta(\gamma_{SM})=\{W_1, W_2,\ldots,W_n\}$, W_i (i=1,2,…,n) is the equivalence class of the condition property and i=1 to n; $TM/\theta(\gamma_{TM})=\{X_1, X_2, X_m\}$, X_j(j=1,2,…,m) is the decision class of transformation rules and j=1 to m.

Definition 3.11 (Property reduction): Let $RT=(M, V, \Gamma)$ be a decision table, and $B \subseteq \gamma_{SM}$, $smm_j \in B$

(1) if $\Delta_B = \Delta_B - \{smm_j\}$ is true, the metamodel smm_j is redundancy to the set B, else smm_j is necessary to B;
(2) if all properties of B are necessary, B is independent;
(3) set $B' \in B$, if B' is independent and $\Delta'_B = \Delta_B$, B' is one of simplified set of B.

Definition 3.12 (Decision matrix of transformation rules): Let $RT= (M, V, \Gamma)$ be a decision table, then a decision matrix $CM(RT)$ is defined as

$$CM(RT) = (m_{ij})_{n \times n} . \quad (11)$$

Where $SM/\theta(\gamma_{SM})=\{W_1,W_2,\ldots,W_n\}$, $\tau(W_i)$ is the value of the equivalence class W_i in the set of condition properties; $m_{ij} = \{ \tau \subseteq \gamma_{SM} : \tau(W_i) \neq \tau(W_j) \text{ 且 } \delta_B(W_i) \neq \delta_B(W_j) \}$

Decision matrix of transformation rules is a symmetric matrix, namely $CM(RT)$ is an upper triangular matrix.

Definition 3.13 (Decision function of transformation rules): Let $CM(RT) = (m_{ij})_{n\times n}$ be a decision matrix, a decision function of the condition equivalence class is defined as

$$B_i = \bigwedge_{j} \bigvee_{smm\in m_{ij}} smm = (\bigvee_{k<i} \bigwedge_{smm\in m_{kj}} smm) \wedge (\bigvee_{l<i} \bigwedge_{smm\in m_{il}} smm) \quad \cdot \tag{12}$$

Where i=1 to n, and let B_i be minimal disjunctive normal form (DNF) according to the idempotent rule $a\wedge a=a$, the absorption rule $a\wedge(a\vee b)=a$, and the distributive rule $a\wedge(b\vee c)=(a\wedge b)\vee(a\wedge c)$.

Decision matrix of transformation rules describes the relationships between equivalence classes($SM/\theta(\gamma_{SM})$) and decision classes($SM/\theta(\gamma_{TM})$). These relationships can be presented using the form $W_i\rightarrow X_j$. W_i, called as the precondition of transformation rules, is the minimal disjunctive normal form. X_j, called as the decision of transformation rules, is the elements of the target metamodel. We use three measurement indicators[10], i.e. support, accuracy and coverage, in rough set theory to evaluate transformation rules. Support, denoted as support ($W_j\wedge X_i$), means the number of objects which satisfy both X_i and W_j in decision table. Accuracy, denoted as accuracy ($W_i\rightarrow X_j$)=support($W_i\wedge X_j$)/support(W_i), is the confidence of the decision of transformation rules. Coverage, denoted as coverage ($W_i\rightarrow X_j$)=support($W_i\wedge X_j$)/ support(X_j), means the applicability of transformation rules.

3.3 Algorithm for Transformation Rules

In this section, we present a construction algorithm, entitled rsCRT, based on the generation algorithm of rough set theory using UCD2RDM in Sect. 2. There are four sub-algorithms in rsCRT. The main construction process of transformation rules is the following:

(1) Firstly, decision table is generated by Def. 6. Its construction algorithm is shown in Sub-algorithm 1;
(2) After generating the decision table, the compatibility of the decision table should be detected according to Def. 8. If the decision table is incompatible, it should be redesigned through indistinguishable function δ_B defined by Def. 9. The specific detection and operation is written in Sub-algorithm 2;
(3) Before building the decision matrix, the decision table should be rewritten according to equivalence class and decision class. The decision matrix is built according to Sub-algorithm 3;
(4) Finally, the decision function B_i, established based on the decision matrix, is converted into minimal disjunctive normal form to construct transformation rules. The rules with the same source and target metamodels are merged and calculated their measure indicators. The creation algorithm of transformation rules is written in Sub-algorithm 4.

Sub-algorithm 1. Generating Decision Table

```
Input: MS=(RGS,γS,ξS), MT=(RGT,γT,ξT)
Output: model_tran_rule_dec_table
model_tran_rule_dec_table(schema)={n,r,u,c,t};
List queryRelList,queryUseList,queryCarList;
void queryRUCList(String tempNodeInstance){
   for(m=1 to |{⊗(r)}|)
      if(⊗(r)[m].n==tempNodeInstance){
         queryRelList.add(⊗(r)[m].r);
         for(n=1 to |⊕(n,u,r)|)
            if(⊕(n,u,r)[n].n==tempNodeInstance)&
                  ⊕(n,u,r)[n].u==⊗(r)[m].u)&
                  ⊕(n,u,r)[n].r==⊗(r)[m].r))
               queryCarList.add(⊗(r)[m].u);} }
      for(i=1 to |RGS(L)|){
        meta_model_Element=ξS(RGS(Li));
        queryRUCList(meta_model_Element);

  Value=queryNodeSchemaValue(meta_model_Element);
      for(j =1 to queryRelList.length()){
         rValue=queryRelSchemaValue(queryRelList[j]);
         uValue=queryUseSchemaValue(queryUseList[j]);
         cValue=queryCarSchemaValue(queryCarList[j]);
         model_tran_rule_dec_table.add(row)=({nValue,
rValue, uValue, cValue, ξT(RGT(tempNodeInstance }); }
}
```

In Sub-algorithm 1, to reduce decision table of UCD2RDM, we use *n* to describe the node of source metamodel (i.e. *Class*,*Relationship*,*Propery*), and *r* to denote the relation of source metamodel (i.e. *Association*,*Inheritance*,*Composition*), and *u* to present the node's role participated in the relation, such as *attr*, *type* and *parent* etc, and *c* to indicate the number of the role, and *t* to reveal the elements of target metamodel(i.e. *Table*,*Fkey* and *Column*).

Table 1. Decision table of transformation rules.

decision class	equivalence class	number of source metamodel	condition property				decision property
			n	r	u	c	t
X1	W1	2	0	0	6	3	0
	W2	2	0	1	0	1	0
	W3	2	0	0	6	3	0
	W4	1	0	1	6	3	0
	W5	1	0	0	6	3	0
	W6	1	0	0	6	3	0
	W7	1	0	1	6	3	0
	W8	6	1	0	6	3	0,2
	W9	4	1	1	6	1	0
X2	W10	3	2	2	2	1	1,2
X 3	W11	2	2	2	1	1	2

Sub-algorithm 2. Redesigning Compatible Decision Table

```
Input: model_tran_rule_dec_table
Output: model_tran_rule_dec_table
for(i=1 to |model_tran_rule_dec_table(row).length|)
  row_i= model_tran_rule_dec_table(row)[i];
  for(j=i+1 to
  |model_tran_rule_dec_table(row).length|)
      row_j= model_tran_rule_dec_table(row)[j];
      if(row_i.n==row_j.n&row_i.r==row_j.r &
         row_i.u==row_j.u&row_i.c==row_j.c){
            if(row_i.t<>row_j.t){
               row_i.t=row_i.t+row_j.t ;
               row_j.t=row_i.t;} }
```

We apply Sub-algorithm 2 in UCD2RDM and get the incompatible information which is shown as follows:

$\delta_B(TID)= \delta_B(TName)=\{1,2\}$， that is $|\delta_B(TID)|=|\delta_B(TName)|\neq 1$

similarly， $\delta_B(CID)= \delta_B(CTName)= \delta_B(SID)= \delta_B(SName)==\{1,2\}$

So the decision table should be redesigned through Sub-algorithm 2. The result is shown in the right part of Table 1 (i.e. condition property and decision property).

Sub-algorithm 3. Building Decision Matrix

```
Input: the equivalence class Wi (i=1,2,…,n),
the decision class Xj(j=1,2,…,m),
model_tran_rule_dec_table
Output: the decision matrix CM
String[] TM/θ(γ_TM)={X_1,X_2,…,X_n},SM/θ(γ_SM)={W_1,W_2,…,W_m};
for(i=1 to |SM/θ(γ_TM)|){W_i= SM/θ(γ_TM)[i];
   for(j=1 to |TM/θ(γ_TM)|){X_j= TM/θ(γ_TM)[j];
      if(ξ_S(RG_S(W_i))∈X_j){
         X_j.add(RG_S(W_i));
```

```
            Support(W_i,X_j)=|W_i∩X_j|;//support
            SI(W_i,X_j)=|W_i∩X_j|/|W_i|;//accuracy
            CI(W_i,X_j)=|W_i∩X_j|/|X_i|;//coverage   } }
    String[][] m_ij={smm∈γ_S:smm(X_i)≠smm(X_j)&Γ(X_i,ξ_S(RG_S(X_i))
           = Γ(X_j,ξ_S(RG_S(X_j)}
        CM(model_tran_rule_dec_table)=(m_ij)_n×n;
```

In Sub-algorithm 3, the decision class $SM/\Delta_{\gamma_{TM}}$ is defined as following:

$X_1 = Teacher \cup Professor \cup Course \cup Student \cup Master \cup$
$Teaching \cup Selecting \cup Supervising$

$X_2 = TID \cup CID \cup SID \cup TName \cup CName \cup SName$

$X_3 = title \cup tutor$

We apply this algorithm and obtain the decision matrix. The result is shown in Table 2. The equivalence class W_{10} is defined by Def.3.10 and def.3.11 based on the results of incompatibility.

Sub-algorithm 4. Constructing Transformation Rules Set

```
    Input: the decision matrix CM
    Output: {MTR}
    List rulefirstList, rulelastList, ruleList;
    for(i=1 to CM.col.length){
        row_i= model_tran_rule_dec_table(row)[i];
        String rulefirst, rulelast;//save a rule
        for(j=1 to i)
            if(i==j)
                for(k=i to CM.row.length)
                    rulefirst= rulefirst∧m[i][k];
            else
                for(k=1 to i)
                    rulefirst= rulefirst∧m[k][i];
            rulelast= rulelast∨row_i.t;
            rulefirstList.add(disformat(rulefirst));
            rulelastList.add(rulelast); }
        for(i=1 to rulefirstList.length)
            for(j=i+1 to rulefirstList.length){
                if(rulefirstList[i]==rulefirstList[j])
                    if(rulelastList[i]<>rulelastList[j])
    rulelastList[i]=rulelastList[i]∨rulelastList[j];
    rule=rulefirstList[i]→rulelastList[i];
    rule.SI=   rulefirstList[i].SI+   rulefirstList[j].SI;
    rule.CI= rulefirstList[i].CI+ rulefirstList[j].CI;
        ruleList.add(rule);
        rulefirstList.del[j];
        rulelastList.del[j]; }
        ruleset_rulepara(schema)={rule,rule.SI,rule.CI};
        for(m=1 to ruleList.length){
    MTRuleSet.add(row)[m]=({ruleList[m].rule,
```

```
ruleList[m].SI,ruleList[m].CI});
   out(MTRuleSet);
```

Table 3. Set of transformation rules and measurement indicator.

order	transformation rules	support	accuracy	coverage
1	(n,0)∧(r,0) → (t,0)	5	1.00	(1.0,0.25)
2	(n,0)∧(c,1) → (t,0)	2	1.00	(1.0,0.10)
3	(r,1)∧(c,3) → (t,0)	1	1.00	(1.0,0.05)
4	(n,0)∧(r,1) → (t,0)	1	1.00	(1.0,0.05)
5	(r,1)∧(c,3) → (t,0)	1	1.00	(1.0,0.05)
6	(n,0)∧(r,1)∧(u,6) → (t,0)	1	1.00	(1.0,0.05)
7	(n,1)∧(r,0) → (t,0)∨(t,2)	6	1.00	(1.0,0.30) (1.0,0.55)
8	(n,1)∧(c,3) → (t,0)∨(t,2)	6	1.00	(1.0,0.30) (1.0,0.55)
9	(n,1)∧(r,1) → (t,0)	4	0.50,0.50	(1.0,0.20)
10	(n,1)∧(c,1) → (t,0)	4	0.50,0.50	(1.0,0.20)
11	(u,6)∧(c,1) → (t,0)	4	0.50,0.50	(1.0,0.20)
12	(u,2) → (t,0) (t,0)∨(t,2)	3	0.50,0.50	(1.0,1.00) (1.0,0.27)
13	(u,1) → (t,2)	2	1.00	(1.0,0.18)

We apply Sub-algorithm 3 and get the transformation rules, as shown in Table 3. The transformation rules can be selected according to their measurement indicators, for example, we can select the rule through coverage. The highest coverage of the rule between *Class* and *Table* is 0.25, and the coverage of the rule between *Relation* and *Table* is 0.30, and the coverage of the rule between *Property* and *Column* is 0.27. The three kinds of transformation rules described using ATL are shown in Fig. 3. Note that according to the inclusion relationship between *Class* and *Property*, like *Table* and *Column*, the rule between Property and Column is consisted in the rule *Class2Table* and *Relationship2Table*.

These sub-algorithms are central components of rsCRT, so they determine the time complexity of rsCRT. Sub-algorithm 1 is used to generate the decision table through searching elements of the source and target model according to the mapping relationships, so the decision table is created in $O(N^2)$, and N is equivalent of $\max\{|\{\otimes(r)\}|,|\oplus(n,u,r)|\}$; in Sub-algorithm 2, the decision table is detected through searching itself. The number of cycles is equal to the record number of items in the decision table, so the compatibility problem is solved in $O(N^2)$; the number of the decisional class($|TM/\theta(\gamma_{TM})|$) is less than the number of the equivalence class ($|SM/\theta(\gamma_{TM})|$), and the number of the equal class is less than the number of the decision table in Sub-algorithm 3, so the decision matrix is built in $O(N^2)$; in Sub-algorithm 4, the number of transformation rules is equal to the number of the equivalence class, so these transformation rules are constructed in $O(N^2)$. Although the same rules are merged in Sub-algorithm 4, the number of cycles is not more than $|SM/\theta(\gamma_{TM})|^2$. Therefore the time complexity of rsCRT is $O(N^2)$, and $N=\max\{|\{\otimes(r)\}|, |\oplus(n,u,r)|\}$.

4 Experiment

To validate the proposed approach, we designed three kinds of experiments in order to respectively analyze three quantities' influence on transformation rules (i.e. Class2Table, Relationship2Table and Property2Column), i.e. the sample's number, the number of the source metamodels and the property reduction. In each experiment, seven intervals of the sample's number are investigated. They are [0, 50], (50,100], (100, 150], (150, 200],…,(300, 350]. For each interval, 350 instances which are randomly generated are executed and averages of the three measure indicators are calculated. Java is used for programming in all of simulation.

```
 MTR 1 Class is transformed into
Table
  rule Class2Table{
    from source: CDM!Class
    to   target: RDM!Table
      column ← source.attrs
      fkey ← source.attrs   }
 MTR 2 Relation is transformed
into Table
  rule Relationship2Table{
    from source: CDM!Relationship)
    to   target: RDM!Table
      fkey ← source.sou.attrs
      column ← source.sou.attrs
      fkey ← source.tar.attrs
      column ← source.tar.attrs }
```

Fig. 3. Transformation rules of the example.

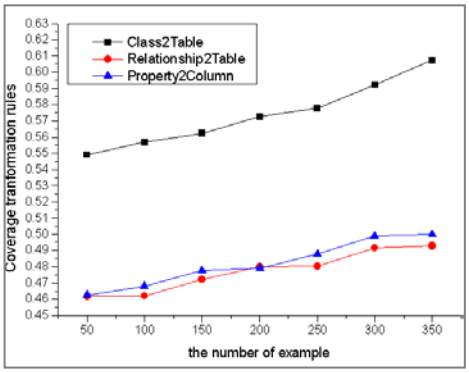

Fig. 4. Comparisons of the number of samples.

Experiment 1 Number of samples

The average Coverage (Class2Table, Relationship2Table, and Property2Column) is shown in Fig. 4. The average Coverage of Class2Table varies from 0.5489 to 0.6072, and the average Coverage of Relationship2Table rises from 0.4619 to 0.4928 and the average Coverage of Property2Column increases from 0.5243 to 0.5522.

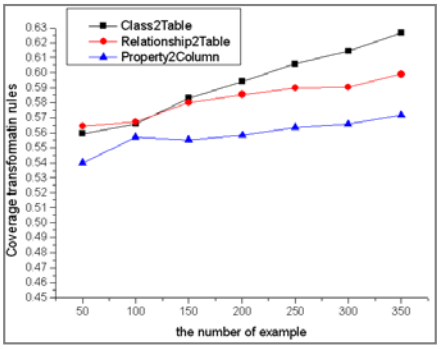

Fig. 5. Comparisons of the number of source metamodels.

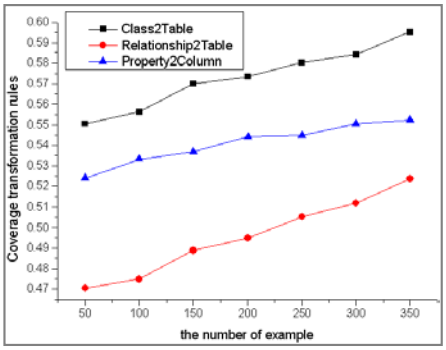

Fig. 6. Comparisons of the property reduction.

Table 2. Decision matrix of transformation rules.

decision class	Equivalence class	decision matrix of transformation rules											transformation rules		measurement indicator
		X1									X2	X3	precondition	decision	(SI,CI)
		W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	W11			
X1	W1		ruc		r	c		r	n	nrc	nruc	nruc	(n,0)∧(r,0)	(t,0)	(1.0,0.10)
	W2			ruc	uc	ruc	ruc	uc	nruc	nu	nruc	nru	(u,0), (n,0)∧(c,1)	(t,0)	(1.0,0.10)
	W3				r	c		r	n	nrc	nruc	nruc	(n,0)∧(r,0)	(t,0)	(1.0,0.10)
	W4					rc	r		nr	nc	nruc	nruc	(r,1)∧(c,3), (n,0)∧(r,0)∧(u,6)	(t,0)	(1.0,0.05)
	W5						c	rc	nc	nrc	nru	nruc	(n,0)∧(r,0)	(t,0)	(1.0,0.05)
	W6							r	n	nrc	nruc	nruc	(n,0)∧(r,1)	(t,0)	(1.0,0.05)
	W7								nr	nc	nruc	nruc	(r,1)∧(c,3), (n,0)∧(r,1)∧(u,6)	(t,0)	(1.0,0.05)
	W8									rc	nruc	nruc	(n,1)∧(r,0), (n,1)∧(c,3)	(t,0)∨ (t,2)	(1.0,0.30) (1.0,0.55)
	W9										nruc	nru	(n,1)∧(r,1), (n,1)∧(c,1), (u,6)∧(c,1)	(t,0)	(1.0,0.20)
X2	W10											uc	(u,2)	(t,1)∨ (t,2)	(1.0,1.00) (1.0,0.27)
X3	W11												(u,1)	(t,2)	(1.0,0.18)

Experiment 2 Number of the source metamodels
In order to verify whether the detailed information of the source and target metamodels affects the construction and measurement of transformation rules, we expand the role elements of UCD2RDM, such as *sou*, *tar* and *child*. The result of the measurement indicators is shown in Fig. 5.

Experiment 3 Property reduction
The property reduction plays an important role in rough set theory. In this experiment, the condition properties, such as the property c, are reduced to observe the change of the average Coverage The result is shown in Fig. 6.

The first experiment demonstrates that the more the number of the mapping relationships is, the larger the average of the measurement indicators of the transformation rules is. For instance, the average Coverage of Class2Table varies from 0.2500 to 0.6072. In the second experiment, the more the number of the source metamodels is, the larger the average of the measurement indicators of the transformation rules is, in particular Relationship2Table and Property2Column. The third experiment denotes the average of the measure indicators become large when some properties whose values fluctuate are eliminated. For example, the average Coverage of Property2Column increases observably after eliminating property c in the source metamodels.

5 Related Work

Shane [12] presents a transformation language based on graph transformation and defines transformation rules through a group of production rule, while Karsai [13] proposes another transformation language to describe the mapping between the source model into target model using the theory of graph transformation and graph rewriting. Dhamanka [14] proposes an approach to create complex links in the beginning of a mapping. However, the complex mapping should be divided into smaller parts to produce transformation rules. David [15] gives emphasis to use set and relational algebra to describe transformation rules. The definition of relation is too strict to expand the set of transformation rules. So the coverage of the method is not widely. The solution in [16] provides a mechanism which name is Clio to produce transformations based on a set of relationships. Clio has a shortage that the definition of the relationships is not extended, so creating complex kinds of model transformation has become a challenge. Similarity Flooding has evolved in [17] to define transformation rules for difference metamodels. The major work gives an algorithm which is the basis of metamodel matching transformations. Design Pattern [18] is one of the first solutions to integrate modeling units to generate transformation rules. It can discover the relation between the source and target models. It is not adapted to match elements in metamodel-level, so it can not support the production of more complex mappings.

The work from [19] uses matching transformations and weaving models to semi-automate the development of model transformation. The weaving model can be used to discover and save the different kinds of relationships among metamodels. Our approach can not only construct transformation rules but also evaluate these

transformation rules. In a paper of Zoltán[20], their approach is applied to derive transformation rules on the basis of the prototype of the source and target models using the inference logic method. Wimmer [21] use an object-based idea to derive ATL rules for model transformation. Because both approaches use mappings between terminal models to construct transformation rules, the rules will have the influence of the size of terminal models.

The INTEROP [22] approach is generally used to solve the transformations from and to the enterprise model level and usage of ontologies. In paper [23], starting from the ATHENA interoperability architecture, Bernhard Bauer presents a methodical approach to transform ARIS into UML and BPDM. However, the source and target models are conforming to UML2 and the approach doesn't support the other metamodels.

6 Conclusions and Future Work

In this paper we use rough set to semi-automate the production of transformation rules and present an approach to construct transformation rules. The approach supports the matching of different metamodels. Transformation rules constructed by our approach are accompanied by measurement indicators supporting selection of transformation rules.

We analyzed the construction theory of rough set from several aspects related to the complexity of transformation rules. Firstly, we have improved the construction approach of rough set to support the mapping between different metamodels. Secondly, we have presented the algorithm rsCRT to construct transformation rules. We provided the measurement indicators of transformation rules according to rough set. These indicators consist of support, accuracy and coverage.

To validate the construction approach, we designed three kinds of experiments in order to respectively analyze three quantities' influence on transformation rules, i.e. the sample's number, the number of the source metamodels and the property reduction. The result denote that the more the number of the mapping relationships and the source metamodels is, the larger the average of the measurement indicators of the transformation rules is, and the average of the measure indicators become large when some properties whose values fluctuate are eliminated. Future work is to optimist transformation rules constructed through our approach transformation. For this reason, we plan to analysis the transformation rules and compose some transformation rules to improve the efficiency of model transformation.

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

References

1. Mukerji J, Miller J.: MDA guide version1.0.1. OMG (2003). http://www.omg.org/cgi-bin/doc? omg/03-06-01.pdf.
2. Kleppe A, Warmer J, Bast W.: MDA Explained: The Model Driven Architecture: Practice and Promise. Addison-Wesley, Boston (2003).
3. R. Agrawal and R. Srikant: Quest Synthetic Data Generator. http://www.almaden .ibm.com/cs/ quest/syndata.html.
4. Jiawei Han, Micheline Kamber: Data Mining: Concepts and Techniques. Morgan Kaufmann (2006).
5. Varró, D.: Model transformation by example. In: Proceedings of Model Driven Engineering Languages and Systems (MODELS 2006). LNCS 4199, pp. 410–424. Springer (2006).
6. Pawlak Z, Skowron A.: Rough sets rudiments. Bulletin of IRSS 3(3): 67-70 (1999).
7. Chang Liyun, Wang Guoyin, Wu Yu.: An Approach for Attribute Reduction and Rule Generation Based on Rough Set Theory. Journal of Software 10(11): 1206-1211 (1999).
8. Roman W. Swiniarski, Skowron A.: Rough set methods in feature selection and recognition. Pattern Recognition Letters 24: 833-849 (2003).
9. Øhrn A.: Discernibility and rough sets in medicine: tools and applications[C]. Trondheim, Norway (1999).
10. Pawlak A, Slowinski R. Rough set approach to multi-attribute decision analysis. European Journal of Operational Research 72(3): 443-459 (1994).
11. Pawlak Z, Skowron A.: Rough sets and Boolean reasoning. Information Sciences 177(1): 41-73 (2007).
12. Shane S.: Combining generative and graph transformation techniques for model transformation: An effective alliance? In: Proc. of the 2nd OOPSLA Workshop on Generative Techniques in the context of Model Driven Architecture. Anaheim: ACM Press (2003). http://cui.unige.ch/~sendall/files/sendall-mda-workshop-OOPSLA03.pdf
13. Karsai G, Agrawal A.: Graph transformations in OMG's model-driven architecture. In: Applications of Graph Trans. with Industrial Relevance. LNCS 3062, pp. 243-259. Springer-Verlag (2004).
14. Dhamanka, R., Lee Y., Doan, A., Halevy, A., Domingos, P.: iMAP: discovering complex semantic matches between database schemas. In: Proceedings of ACM SIGMOD 2004, pp. 383–394. ACM (2004).
15. David K, Stuart A. A relational approach to defining transformations in a metamodel. In: Hussmann H., Jezequel J.M., Cook S. (eds.) UML 2002 - The Unified Modeling Language, 5th Int'l. Conf. LNCS 2460, pp. 243-258. Springer-Verlag (2002).
16. Miller, R.J., Hernandez, M.A., Haas, L.M., Yan, L.-L., Ho, C.T.H.,Fagin, R., Popa, L.: The Clio Project: Managing heterogeneity. SIGMOD Record 30 (1): 78–83 (2001).
17. Melnik, S.: Generic Model Management: Concepts and Algorithms. Ph.D. Dissertation, University of Leipzig, LNCS2967. Springer (2004)
18. Zhang Tian, Zhang Yan, Yu Xiaofeng et al.: MDA Based Design Patterns Modeling and Model Transformation. Journal of Software 19(9): 2203-2217 (2008).
19. Maros Didont Del Fabro, Patrick Valduriez: Towards the efficient development of model transformations using model weaving and matching transformations. Software and System Modeling 8(3): 305-324 (2009).
20. Zoltán Balogh and Dániel Varró: Model Transformation by Example Using Inductive Logic Programming. Software and System Modeling 8(3): 347-364 (2009).
21. Wimmer, M., Strommer, M., Kargl, H., Kramler, G.: Towards model transformation generation by-example. In: Proceedings of HICSS-40 Hawaii International Conference on System Sciences, p.285. IEEE Computer Society (2007)

22. Hervé Panetto, Monica Scannapieco, and Martin Zelm. INTEROP NoE: Interoperability Research for Networked Enterprises Applications and Software. OTM Workshops 2004.LNCS3292, Dresden:Springer-Verlag,2004, pp:866-882.
23.Bernhard Bauer, Jörg P. Müller, Stephan Roser. A Model-Driven Approach to Designing Cross-Enterprise Business Processes. OTM Workshops 2004.LNCS3292, Dresden:Springer-Verlag,2004, pp:544-555.