A Segment-based Approach for Reconcilable Model Transformation

Xin Zhou
IBM China Research Laboratory
Diamond Building, Zhong Guan Cun
Software Park, 100094, P.R.China
86-10-58748465
zhouxin@ cn.ibm.com

Ying Liu
IBM China Research Laboratory
Diamond Building, Zhong Guan Cun
Software Park, 100094, P.R.China
86-10-58748465
aliceliu@ cn.ibm.com

Jun Zhu
IBM China Research Laboratory
Diamond Building, Zhong Guan Cun
Software Park, 100094, P.R.China
86-10-58748465
zhujun@ cn.ibm.com

ABSTRACT
Model transformation transforms a high level source model into the skeleton of a low level target model, thereafter developers continue to refine and concretize the skeleton. With the changing requirements, the source model will be consequently modified and such modification should be reflected incrementally in the refined target model. This paper presents an end-to-end segment-based reconcilable model transformation approach to identify the conflict between source model changes and target model changes and generate a new target model which accommodates all the changes if no conflict exists. The approach is experimented in three real transformation cases.

Categories and Subject Descriptors
D.2.10 [Design] : Model Transformation

General Terms: Design

Keywords: Model Driven Development, Model transformation, Reconcilable Model Transformation, Model Segment, Transformation Linkage

1. INTRODUCTION
Since its inception by OMG (Object Management Group) in 2001, MDA [9] (Model-Driven Architecture) has been gaining more and more adoption from both academic and field. While practitioners are proving the advantages of employing layered models linked by transformation [10], some key challenges are met that prevents it from being widely deployed into large scale scenarios. In model driven software development, high level source models are transformed into low level target models when they are ready. Then, software developers might change the transformed low level models for refinement, enhancement, and etc. The changes on low level models are usually not reflected in high level models. When later changes happen at the higher level model, how to correctly and effectively get the changes down and reconcile it with the low level changes thus becomes a problem. For some cases, the problem is not so difficult if the high and low level models are similar in structure, such as the transformation from class diagram to Java code. However, there are multiple more complex cases, like the transformation from flow-style process model to structural-style process model, that high and low level models are different in structure. The difference between high and low level models' structure brings major challenges: How to judge whether the changes on high level models conflict with those existing in the low level models? How to reconcile the non-conflictive high level changes and low level changes?

Presently, complex reconcilable model transformation is usually solved by software developers manually in an ad-hoc manner, which is neither efficient nor reliable when we are building large scale software. In this paper, we will introduce an end-to-end segment based approach to address this problem. Models at different level are mapped to their corresponding segment models, which are the basis for conflict detection and reconciliation.

The structure of the remaining of this paper is as follows. Section 2 defines model segment and some other pertinent concepts. Section 3 presents our approach for reconcilable model transformation. Section 4 introduces and discusses the application of the approach to three real cases. Section 5 briefs some related works and section 6 concludes and discusses future works.

2. DEFINITIONS
We use the well accepted concepts (model, model transformation, and model transformation rule) from [5] [8] as the basis for our further definition of our concepts in the context of reconcilable model transformation.

Definition 1. A ***model segment*** is a subset of a model, which is accepted by a ***model transformation rule instance*** as its valid source or target.

Definition 2. A ***model segment property*** is a transformation related characteristic attribute possessed by a model segment, which is manipulated or used by transformation rules.

We differentiate two kinds of model segment properties: major properties and minor properties. Major property is the identity of a model segment, while minor property is the secondary characterization of a model segment.

Definition 3. A ***segment relationship*** is the structural or semantic inter-relationship between two segments of the same model.

Definition 4. A ***transformation linkage*** is an inter-relationship between the source model segment and the target model segment if the later is transformed from the former.

Definition 5. A ***Segment model*** is a set of segments and the inter-segment structural and/or semantic relationships.

Definition 6. ***Segment model isomorph*** is a characteristic between two segment models denoting that they share common structure.

ESEC/FSE'07, September 3–7, 2007, Cavtat near Dubrovnik, Croatia.
ACM 978-1-59593-812-1/07/0009.

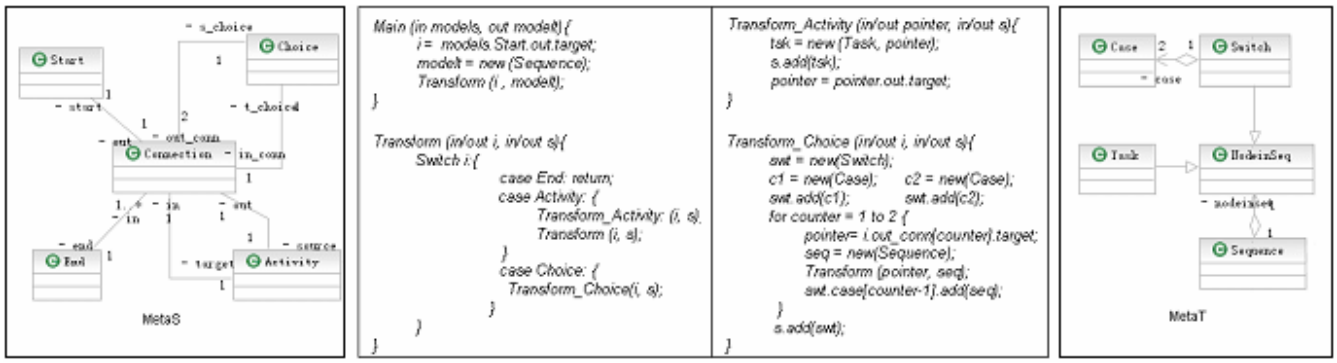

Figure 1: MetaS, MetaT and Transformation TR

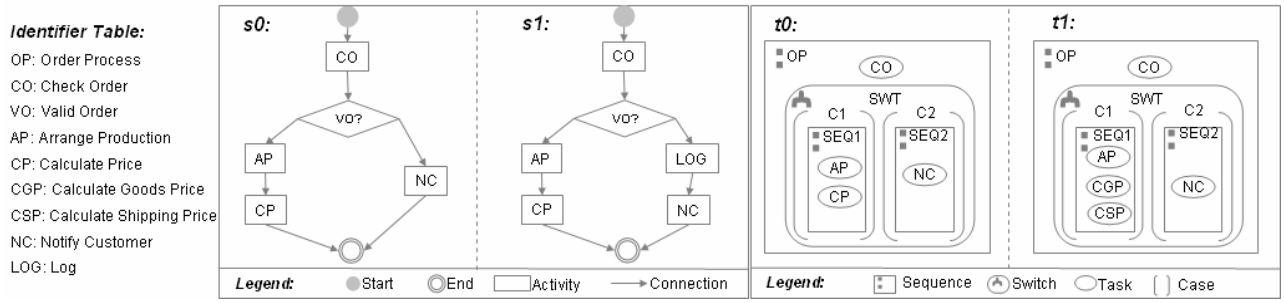

Figure 2: Model s0, s1, t0, t1

3. APPROACH

3.1 Illustrative Example

In order to illustrate our approach, we make up two fictional meta-models: MetaS and MetaT, and a transformation TR (as shown in Figure 1). A model s0 is built complying MetaS to describe a simple "Order Processing" process. s0 is transformed by TR to t0 (satisfying MetaT). To simulate target model change, task 'CP' in t0 is split to two tasks: 'CGP' and 'CSP', and implementation is added to 'CO', 'AP', 'CP' and 'NC'. The post-change t0 is recorded as t1. Also, s0 is changed to s1 by adding a step 'LOG' before 'NC' to simulate source model change caused by changing requirement. Figure 2 shows s0, s1, t0, t1.

3.2 The Overall Process

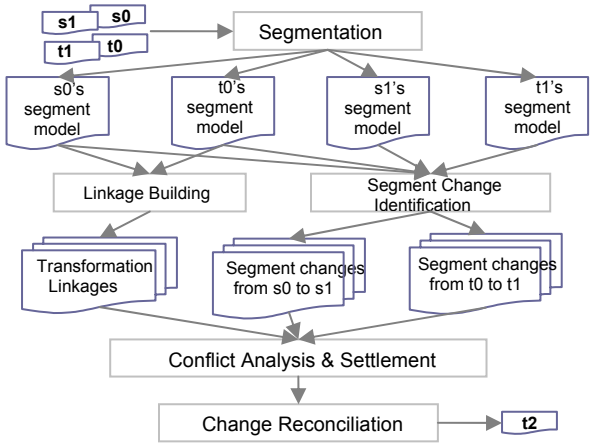

Figure 3: Process for Reconcilable Model Transformation

The overall process of the segment-based approach is sketched in Figure 3, which will be illustrated in detail in sub-sections below.

3.3 Model Segmentation

Model segmentation is the process to represent a model from model segment's perspective. With analyzing the transformation algorithm, we can define a set of source/target model segmentation rules, source/target segment relating rules and their execution logic. Actually, a segmentation rule can be regarded as a query on model, which is expressed with meta-model elements. In Figure 4, we show the result source segment model SegMdl_S and target segment model SegMdl_T by separately applying rules related to TR to s0 and t0. In the figure, one box denotes a segment and the segment's type is written on its top-left. The directed line from one segment to the other segment denotes the *Include* relationship between two segments.

3.4 Transformation Linkage Building

Building the transformation linkages between a source segment and a target segment is the prerequisite for later identifying the conflict between source side changes and target side changes. Still, how to link a source segment with a target segment depends on the actual transformation algorithm.

It's interesting that *SegMdl_S* and *SegMdl_T* becomes *isomorphic* after applying above four criteria to linking s0 and t0's segments. The characteristic is very important for merging the source side segment changes and target side segment changes to one base and identifying the conflict.

3.5 Segment Change Identification

This step is to identify the segment level changes between source model s0 and s1, as well as between target model t0 and t1. A basic sub-step of segment change identification is to compare two segments. Given two arbitrary segments seg1 and seg2, if all of their major properties and minor properties are equal, they are the same (denoted as '=').

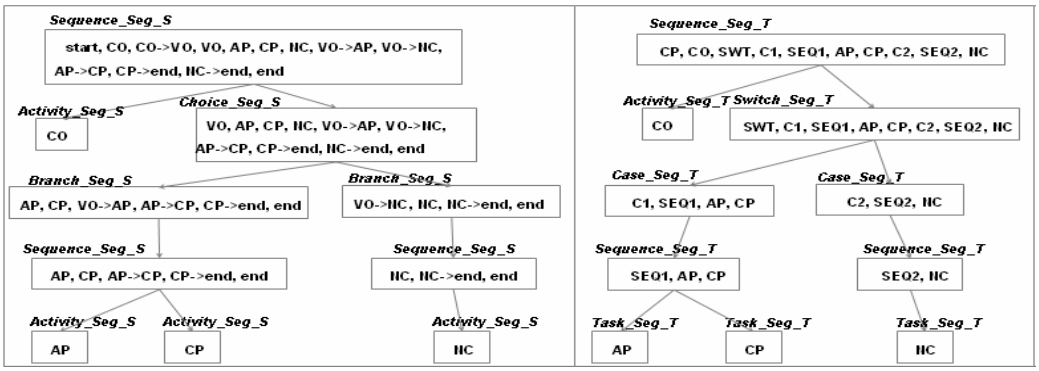

Figure 4: SegMdl_S and SegMdl_T

If all of their major properties are equal but at least one of their minor properties is unequal, they are slightly different (denoted as '≈'). Otherwise, if at least one of their major properties is unequal, they are totally different (denoted as '≠'). In the example, we select task name as the major property of a task segment based on the transformation, so two task segments are the same if the tasks in each have the same name.

Based on the segments comparing approach, we can further identify a model segment changes. Given model M is changed to M', Table 1 shows the criteria for identifying segment changes.

Table 1: Criteria for Identifying Segments Change

Result	*Identifying Criteria*
seg is not changed	seg ∈ SegMdl_M ∧ (∃ seg' ∈ SegMdl_M' seg = seg')
Seg is added	seg ∈ SegMdl_M' ∧ (∀ seg' ∈ SegMdl_M !(seg = seg' | seg ≈ seg'))
seg is deleted	seg ∈ SegMdl_M ∧ (∀ seg' ∈ SegMdl_M' !(seg = seg' | seg ≈ seg'))
seg is modified	seg ∈ SegMdl_M ∧ (∃ seg' ∈ SegMdl_M' seg ≈ seg')

3.6 Segment Conflict Analysis and Settlement

Segment conflict analysis detects the potential conflict at segment level according to the segment change types and their transformation linkages. More specifically, we can first mark the source segment changes on the pre-change source segment model and mark the target segment changes on the pre-change target segment model. As the pre-change source segment model and pre-change target segment model are isomorphic, the change marks can be merged into a common base - Unified Change Marking Graph (UCMG). Figure 5 shows the UCMG in our example. UCMG abstracts the common structure of source and target segment model and merges the source and target segment change marks. With the UCMG, conflict analysis can be performed by navigating the structure and checking each node's change marks. Criteria can be defined to clarify which situations are really conflictive and which are not. Several of the criteria used in our example are shown in Column I, II and III of Table 2. By applying these criteria on the nodes from bottom to up, multiple conflicts are detected between the source segment changes and target segment changes in our example.

In case of conflicts, some policies are needed to settle them before further processing. Either they can be batch settled based on some pre-defined rules or be settled case by case manually. Column I, II and IV of Table 2 depicts the illustrative predefined policy for conflict settlement. A node's settlement result should also be updated to its parent node marked as 'M.sub_seg'.

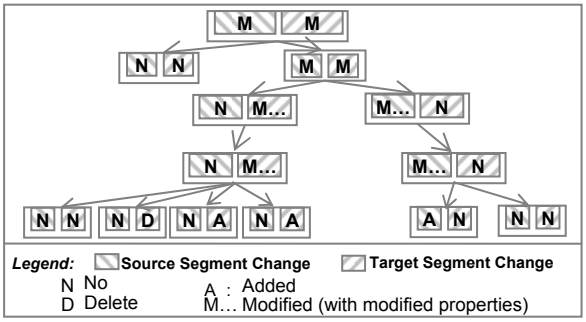

Figure 5: Unified Change Marking Graph Example

Table 2: Criteria Example for Detecting Conflict

Source Change	Target Change	Conflict?	Settlement
N	A/D/M…	Yes	Keep target change
A/D/M…	N	Yes	Keep source change
M…	D	Yes	Keep target change
D	M…	Yes	Keep source change
…	…	**…**	…

3.7 Segment Change Reconciliation

Segment change reconciliation is to construct the segments of t2, which are reused from t1's segment, or modified based on t1's segment or transformed from s1's segment following the guidance of the post-settlement UCMG.

Let node.seg_s1 denotes a node's corresponding segment from s1, and node.seg_t1 denotes a node's corresponding segment from t1, below policies are applied to constructing the segments of t2:

1. if the node's source mark 'N' and target mark 'N' are both kept, copy *node.seg_t1* to t2;
2. if the node's source mark 'A' is kept, transform *node.seg_s1* and add the resulting segment to t2

3. if the node's target mark 'A' is kept, copy *node.seg_t1* from t1 to t2;
4. if a node's source mark is 'M.sub_seg' and target marks is 'M.sub_seg', transform *node.seg_s1.sub_seg* and use the resulting segment to replace *node.seg_t1.sub_seg*, and copy the resulting *node.seg_t1* to t2;
5. if only the node's source mark 'M.sub_seg' is kept, transform *node.seg_s1.sub_seg* and use the resulting segment to replace *node.seg_t1.sub_seg*;
6. if only the node's target mark 'M.sub_seg' is kept, copy *node.seg_t1* from t1 to t2;
7. if a node's target mark 'D' is kept, copy nothing to t2;

4. VALIDATION AND EVALUATION

We applied the approach to three real cases to evaluate its feasibility and effect. The first case is the reconcilable transformation from Model Blue [12] Process Model to WID BPEL Process Model [1], the second is the reconcilable transformation from WBIM Process Model [6] to WID BPEL Process Model, and the third case is the reconcilable transformation from WBI Modeler Information Model to RSA Class Diagram. We come to some initial findings in the practice. The first finding is that the approach can be applied to enable various reconcilable transformation cases without special requirement for the structural similarity of source and target model. For instance, the flow-style Model Blue process model and structural-style BPEL process model in case one are significant different in structure. The second finding is that to apply our segment-based approach, correct and deep understanding of original transformation policy is the most critical requisite, and it's difficult. The third finding is that the big investment on applying the approach will be rewarded when the resulting reconcilable transformation is used to facilitate any transformation instances. It saves the developers' work effort by avoiding boring and error-prone manual work. In the second phase of a model driven business integration project for a TaiWan bank, we applied the prototype to *Case One*. Compared to the first phase of the project, developers' work effort on analyzing the conflict and merging two-side changes decreases 85%.

5. RELATED WORKS

Rational Software Architect (RSA) [11] enables the transformation from UML design to corresponding code skeleton that will be further modified and implemented. When the design is changed and re-transformed, RSA allows the user to manually decide which parts of the code should be kept un-unchanged by the re-transformation and which parts of the code can be replaced by the newly generated code skeleton. With this feature, important changes and implementation works can be kept, and the repeated implementation works can be reduced. However, for middle or large scale software system, manually deciding which parts to be kept and which parts to be replaced is still a time-consuming and error-prone task. Besides, the underlying mechanism supporting this feature, i.e. recording the one-to-one traceability between design element and code element, is incapable to be applied to the more complex case that multiple source elements correspond to multiple target elements.

Eclipse Modeling Framework [4] allows users to select whether to protect some existing code from being overwritten when regenerating changed Ecore model to code. Still, the user should judge ahead which part of the code is not conflict with the changed Ecore model, unless this feature helps little.

Johann et al [7] addresses incremental model transformation from source model to target model, i.e. only those changed source model elements are re-transformed and merged to previous target model so as to save effort. However, this approach doesn't consider the situation that previous target model might be changed before re-transformation. Actually, the target models are often changed in model-driven software development practice.

6. CONCLUSION AND FUTURE WORKS

This paper presented a segment-based approach for reconcilable model transformation, which uses segment model as an abstraction of the model structure for change identification, conflict analyzing, change reconciling, etc. The approach reduces the efforts and difficulties of model change management in model driven software development. We validated the approach in real cases.

Plans for future research include extending the current two-layer reconcilable model transformation approach to support multiple-layer reconcilable model transformation. We also intend to enhance the prototype so that more complex business processes and BPEL processes from real projects can be dealt with. We will use the experimental results generated from the real cases to show the percentage of efforts saved by using our approach.

7. REFERENCES

[1] Andrews, T., et. Al, Business Process Execution Language for Web Services (BPEL4WS), http://www-106.ibm.com/developerworks/library/ws-bpel/, 2003.

[2] Boehm, B., Egyed, A., and etc. Using the WinWin Sprial Model: A Case Study. IEEE Computer, 1998, 33-44.

[3] Czarnecki, K. and Helsen, S. Classification of Model Transformation Approaches. OOPSLA 2003 Workshop on Generative Techniques in the Context of Model-Driven Architectures. 2003.

[4] Eclipse Modeling Framework. http://www.eclipse.org/emf/

[5] Hailpern, B., Tarr, P. Model-driven development: The good, the bad, and the ugly. IBM System Journal, Volume 45, Number 3, 2006.

[6] IBM WebSphere Business Integration Modeler, http://www-306.ibmcom/software/integration/wbimodeler/.

[7] Johann, S., Egyed, A., Instant and Incremental Transformation of Model. Proceedings of the 19th IEEE Conference of Automated Software Engineering (ICASE), Linz, Austria, Sep 2004.

[8] Kalnins, A., J. Barzdins and E. Celms, Model transformation language MOLA, in: Proc. Model-Driven Architecture: Foundations and Applications, 2004, pp. 14--28.

[9] OMG Model-Driven Architecture Home Page: http:// www.omg.org/mda/index.html

[10] Sendall, S. and Kozaczynski, W., Model Transformation: The Heart and Soul of Model-Driven Software Development, In IEEE Software, vol. 20, no. 5, Sept./Oct. 2003

[11] Switchinbank, P., et. Al. Patterns: Model-Driven Development Using IBM Rational Software Architect. http://www.redbooks.ibm.com/redbooks/pdfs/sg247105.pdf

[12] Zhu, J. et. Al. "Model-driven Business Process Integration and Management: A case study with Bank SinoPac regional service platform", IBM Journal of Research and Development, vol 48, issue 5-6, p649-p670.