Generating Preliminary Edit Lenses from Automatic Pattern Discovery in Business Process Modeling

Moisés Castelo Branco[1], Arif Wider[2]

[1] Generative Software Development Laboratory, University of Waterloo, Canada
WWW home page: http://gsd.uwaterloo.ca
mcbranco@gsd.uwaterloo.ca

[2] Humboldt-Universität zu Berlin
Department of Computer Science
Unter den Linden 6, D-10099 Berlin, Germany
wider@informatik.hu-berlin.de

Abstract. Business process models are often used to describe a single system at different levels of abstractions—for instance, a business workflow specification and its corresponding IT implementation—and have to be synchronized. Describing such model synchronizations with approaches to bidirectional model transformations (e.g., lenses) has several advantages from maintainability to verification. However, describing a synchronization from scratch by manually writing update operations can be tedious and error-prone. We show how to automatically generate such synchronizations by identifying refinement patterns in the relation between different models and by representing those patterns as edit lenses. Although these generated synchronization operations may not completely reflect all the refinement patterns that correlate the models in practice, a considerable part of manual work can be saved by generating them automatically first, and then only check and optimize them manually.

1 Introduction

Business Process Modeling (BPM) involves the participation and collaboration of many stakeholders (e.g. Business Analysts, Systems Analysts, IT Architects and Developers). The distribution of responsibilities and roles results in the creation of different models of the same business process. Specialized modeling languages have been developed to represent such models, for example the Business Process Modeling and Notation (BPMN). Business- and IT-level models evolve concurrently and periodically need to be synchronized [2]. Synchronizing the models means propagating changes in both directions, i.e., from business to IT and vice versa. Writing such synchronizations usually requires uncovering tacit knowledge, which may be lost entirely. IT personnel at the Bank of Northeast of Brazil (BNB), our industry partner, has faced this challenge as part of a regulatory compliance project. Without appropriate tool support, this task is very time-consuming and error-prone.

Aiming at mitigating this problem, there is a multitude of frameworks for bidirectional model transformations (BXs) [3–6,8,9]. Although these approaches provide the foundations of BXs in terms of properties and operations, few concrete implementations of such frameworks are available. It remains unclear how to generate such transformations in many problem domains. In particular, there is a lack of practical BX frameworks tailored to deal with consistency of business process models that target different levels of abstraction.

In this paper, we leverage previous work on process model matching [1] and present a practical approach for generating preliminary bidirectional model transformations in BPM based on edit lenses [9]. The implemented framework is tailored to process modeling at different abstraction levels. First, we start by providing some background in BPM and important concepts used throughout the paper (§ 2). Following, we discuss some background and rationale behind using edit lenses to deal with process models (§ 3). Then, we present technical details of the approach (§ 4) and show some evaluation based on industrial case studies (§ 5). Finally, we discuss related work (§ 6), conclusions, and future work (§ 7).

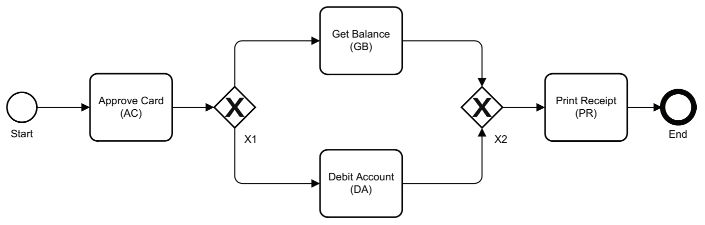
2 Business Process Models at Different Levels of Abstraction

2.1 Background

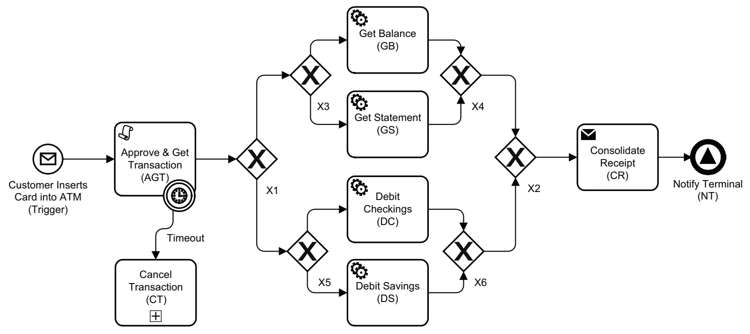
A business process is a collection of structured or ad hoc activities that produce a specific output, such as service or product. The activities of a process interact with IT assets to capture, transform, or report business data. In practice, a range of business to IT-oriented stakeholders create and use business process models for specific purposes, including requirements elicitation, documentation, simulation, and execution. Figure 1 shows two models in BPMN 2.0, each representing the process of using an *Automated Teller Machine* (ATM) system at different level of abstraction. We added shorter names in parentheses (e.g., *(AC), (GB)*) to avoid clutter when referring to the models. The first model (Fig. 1.a) represents a business-level process specification. The second one (Fig. 1.b) is an IT-level specification.

2.2 Process Structure Tree (PST)

Any BPMN model can be uniquely decomposed into single-entry single-exit (SESE) regions [10]. Let $G = (N, E)$ be a workflow graph, where N is the set of nodes and E the set of edges. A SESE region $R = (N', E')$ is a nonempty subgraph of G, i.e., $N' \subseteq N$ and $E' = E \cap (N' \times N')$ such that there exist edges $e, e' \in E$ with $E \cap ((N \backslash N') \times N') = \{e\}$ and $E \cap (N' \times (N \backslash N')) = \{e'\}$; e and e' are called the *entry* and the *exit* edge of R, respectively. The Process Structure



(a) Business Specification



(b) Technical Specification

Fig. 1: BPMN Models

Tree (PST) for a BPMN process model is a tree representing the decomposition of the model into SESE regions [10]. Figure 2 shows the PSTs corresponding to the BPMN process models. There is a unique PST for each BPMN model. The root represents the whole process model. Leaves represent model elements, i.e., tasks, gateways and events. Inner nodes represent SESE regions.

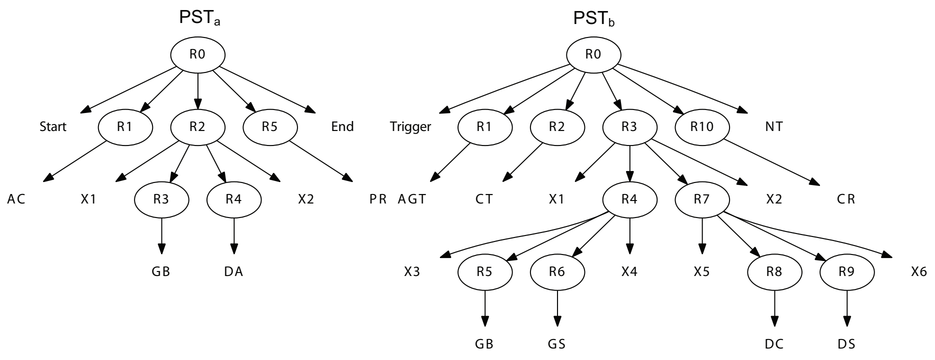


Fig. 2: PSTs representation of the business process models

2.3 Differences between Business and IT process models

We have compiled a catalog of 11 recurrent patterns used to refine business-level models into IT-level models [2]. These patterns include (i) adding or modifying properties of model elements, such as changing the name or type of an activity, and (ii) changing the flow structure. An example from category (i) is the renaming and retyping of the empty start event *Start* (Fig. 1.a) into the message-driven event *Customer inserts card into ATM* (Fig. 1.b). An example from category (ii) is the refinement of the task *Debit Account* (Fig. 1.a) into the block consisting of the gateways *X5* and *X6* and two other tasks *Debit Checkings* and *Debit Savings* (Fig. 1.b).

3 Edit Lenses

A *lens* is a bidirectional transformation between a pair of connected data structures, X and Y, capable of translating an edit on one structure into an appropriate edit on the other. Each lens is a pair of functions—**to** and **from**—one mapping X updates to Y updates and the other mapping Y updates to X updates. Although many varieties of lenses exist in the literature, only *edit lenses* [9] offer a satisfactory treatment of how editing operations are *represented.*

Our approach employs edit lenses because of two reasons. First, in business process modeling, changes (including refinement patterns [2]) can be distilled into atomic editing operations, such as *inserting, deleting or moving* an activity and *updating* its attributes. Second, edit lenses are intuitive. Human users can inspect them easily to review, add, discard or select specific operations before synchronizing the models.

4 Generating Edit Lenses from Correspondences between Process Models

In previous work [1], we presented an algorithm to automatically detect non-trivial correspondence patterns [2] between BMPN process models across levels of abstraction. The algorithm identifies attribute and structural correspondences over the PSTs of the input models. Table 1 shows the correspondences identified by the matching algorithm on the PSTs shown in the Fig. 2.

For each correspondence, the lenses generator produces a pair of functions, **to** and **from**, composed of edit operations for bidirectional transformations in both directions: *business* $\hookrightarrow$ *IT* (*to*) and *business* $\hookleftarrow$ *IT* (*from*). Regions and model elements without correspondences, such as PST_a.DA and PST_b.GS, are treated as individual *inserts* or *deletes.* The edit operations are generated according to the following heuristic:

- ***insert(l,y,k)***; insert a new PST node l as the kth child of node y. For example, in Fig. 2, the region PST_b.R6 is inserted as the 3rd child of PST_b.R4: *insert*(PST_b.R6,PST_b.R4,3).

Table 1: Correspondences

i	PST_a.R0	≘	PST_b.R0	(Root)
ii	PST_a.Start	≘	PST_b.Trigger	(Structure)
iii	PST_a.R1	≘	PST_b.R1	(Structure)
iv	PST_a.AC	≘	PST_b.AGT	(Structure)
v	PST_a.R2	≘	PST_b.R3	(Attribute)
vi	PST_a.R3	≘	PST_b.R5	(Attribute)
vii	PST_a.GB	≘	PST_b.GB	(Attribute)
viii	PST_a.R4	≘	PST_b.R7	(Structure)
ix	PST_a.R5	≘	PST_b.R10	(Structure)
x	PST_a.PR	≘	PST_b.CR	(Structure)
xi	PST_a.End	≘	PST_b.NT	(Structure)

- ***delete(x)***; delete PST node x from its parent. In the example, PST_a.DA is deleted from PST_a.R4: *delete*(PST_a.DA).
- ***move(x,y,k)***; node x becomes the kth child of y. In the example, the task PST_a.GB is moved from PST_a.R3 to PST_b.R5: *move*(PST_a.GB,PST_b.R5,1)
- ***update(x,v)***; update value of x with v. For example, the value of the node PST_a.PR was updated to PST_b.CR: *update*(PST_a.PR,PST_b.CR)
- ***refine(r,s)***; the region r is refined into the region s: the atomic changes are also presented by *inserts* and *deletes*. In the example, the region PST_a.R4 is refined into the region PST_b.R7. The corresponding *inserts* and *deletes* are presented hierarchically, such as *delete*(PST_a.DA), *insert*(PST_b.R8,PST_b.R7, 2), and so on.

Node positions are shown as parameters using absolute paths. A path like "/0/4/3/0" means: the unique child of the 3rd child of the 4th child of the root node. The output of the lenses generator for the correspondences *vii* and *viii* previously shown are as follows:

vii PST_a.GB ≘PST_b.GB

to (*business* $\hookrightarrow$ *IT*)
```
move PSTNode="GB" destination="/0/4/2/2" origin="/0/3/2"
```
from (*business* $\hookleftarrow$ *IT*)
```
move PSTNode="GB" destination="/0/3/2" origin="/0/4/2/2"
```

viii PST_a.R4 ≘PST_b.R7

to (*business* $\hookrightarrow$ *IT*)
```
refine PSTNode="R4(DA..DA) to R7(X5..X6)" composed of:
  delete PSTNode="DA" source="/0/3/3/0"
  insert PSTNode="X5" destination="/0/4/3/0"
  insert PSTNode="X6" destination="/0/4/3/1"
  insert PSTNode="R8(DC..DC)" destination="/0/4/3/2"
  insert PSTNode="R9(DS..DS)" destination="/0/4/3/3"
```

```
    insert PSTNode="DC" destination="/0/4/3/2/0"
    insert PSTNode="DS" destination="/0/4/3/3/0"
```
from (*business* $\hookleftarrow$ *IT*)
```
 refine PSTNode="R7(X5..X6) to R4(DA..DA)" composed of:
    delete PSTNode="X5" source="/0/4/3/0"
    delete PSTNode="X6" source="/0/4/3/1"
    delete PSTBranch="R8(DC..DC)" source="/0/4/3/2"
    delete PSTBranch="R9(DS..DS)" source="/0/4/3/3"
    insert PSTNode="DA" destination="/0/3/3/0"
```

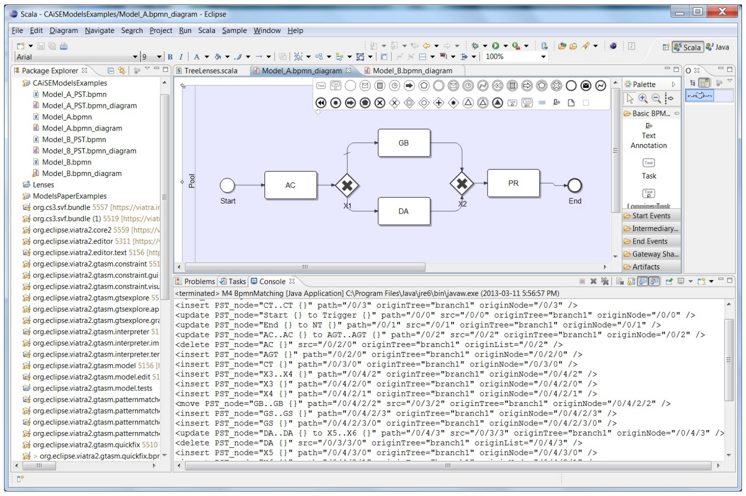
Users can review this preliminary set of edit operations to add, remove, group and discard specific ones. The final lens (revised by the user) is executed when something is changed in either of the two models to generate a consistent version of the other model. The synchronization is fully automatic: our implementation provides a module on top of VIATRA2 to synchronize any number of individual operations selected by the user. Transformations are first performed on the PSTs, and afterwards they are transformed back into BPMN. Occasional malformed BPMN models after the transformations (e.g., an added task without incoming or outgoing flows) need to be fixed manually by the user. To deal with this issue we are currently working on integrating the approach with a quick fix generator [7] that guides the user in fixing post-synchronized models, when needed.

5 Evaluation

Implementation. We have implemented the lenses generator framework in Java as an Eclipse feature (Fig. 3), on top of the SOA Tools Platform BPMN Modeler.

Evaluation. The quality of generated lenses directly depends on the quality of the matching algorithm, whose recall varies between 40-70% [1]. Thus, the users always need to review the initial lenses to capture the correct refinement patterns and create a baseline of operations that ensure proper business-IT synchronization. We wanted to know how much manual work is needed to update baselines of lenses over time, in the presence of typical model changes. We inspected 48 real changes made in three BPM projects over a period of one year, and counted how many individual operations would need to be manually changed in each baseline to meet those changes. The results are shown in the Table 2.

A large number of operations needs to be manually revised to cope with changes. Nevertheless, we believe that keeping baselines of lenses is useful, instead of the burden of periodically rebuilding all synchronizations from scratch. Approximately 50% of the work over the analysed period would be saved by basically maintaining the lenses incrementally, in pace with the changes.



```
<terminated> M4 BpmnMatching [Java Application] C:\Program Files\Java\jre6\bin\javaw.exe (2013-03-11 5:56:57 PM)
<insert PST_node="CT..CT {}" path="/0/3" originTree="branch1" originNode="/0/3" />
<update PST_node="Start {} to Trigger {}" path="/0/0" src="/0/0" originTree="branch1" originNode="/0/0" />
<update PST_node="End {} to NT {}" path="/0/1" src="/0/1" originTree="branch1" originNode="/0/1" />
<update PST_node="AC..AC {} to AGT..AGT {}" path="/0/2" src="/0/2" originTree="branch1" originNode="/0/2" />
<delete PST_node="AC {}" src="/0/2/0" originTree="branch1" originList="/0/2" />
<insert PST_node="AGT {}" path="/0/2/0" originTree="branch1" originNode="/0/2/0" />
<insert PST_node="CT {}" path="/0/3/0" originTree="branch1" originNode="/0/3/0" />
<insert PST_node="X3..X4 {}" path="/0/4/2" originTree="branch1" originNode="/0/4/2" />
<insert PST_node="X3 {}" path="/0/4/2/0" originTree="branch1" originNode="/0/4/2/0" />
<insert PST_node="X4 {}" path="/0/4/2/1" originTree="branch1" originNode="/0/4/2/1" />
<move PST_node="GB..GB {}" path="/0/4/2/2" src="/0/3/2" originTree="branch1" originNode="/0/4/2/2" />
<insert PST_node="GS..GS {}" path="/0/4/2/3" originTree="branch1" originNode="/0/4/2/3" />
<insert PST_node="GS {}" path="/0/4/2/3/0" originTree="branch1" originNode="/0/4/2/3/0" />
<update PST_node="DA..DA {} to X5..X6 {}" path="/0/4/3" src="/0/3/3" originTree="branch1" originNode="/0/4/3" />
<delete PST_node="DA {}" src="/0/3/3/0" originTree="branch1" originList="/0/4/3" />
<insert PST_node="X5 {}" path="/0/4/3/0" originTree="branch1" originNode="/0/4/3/0" />
```

Fig. 3: Edit Lenses Generator Tool

Table 2: Evaluation

	Number of		
Project	Baseline Operations	Changes	Operations Revised
Customer Registration	273	23	106 (39%)
Credit Backoffice	356	16	163 (46%)
Procurement	161	9	83 (52%)

6 Related Work

Bidirectional transformation frameworks originate from the lenses framework proposed by Foster et al. [6] which assumes one model being an abstract view of the other. Inspired by the lenses framework, researchers proposed state-based framework for symmetric synchronization [8]. As a more general case, symmetric synchronization allows neither of the model to be a view of the other. However, as Diskin et al. [3] point out, state-based bidirectional transformations actually mix two different operations—delta discovery and delta propagation—leading to several semantic problems. To fix these problems, several approaches [3, 5, 9] propose delta-based frameworks, where deltas are taken as input and output. Typical delta-based frameworks include delta lenses [4] for the asymmetric cases, and symmetric delta lenses [5] and edit lenses [9] for the symmetric cases.

7 Conclusions

We have presented a practical approach that generates preliminary edit lenses in BPM, tailored to maintaining consistency of process models at different abstraction levels. A prototype tool is implemented on top of Eclipse and SOA Tools Platform. We performed a preliminary evaluation of the tool based on real-world models. As for future work, we aim to perform a qualitative assessment of the approach, obtaining feedback from BPM practitioners in industry. We also hope that this work may encourage developers to evolve the approach and provide tool support to more elaborated BX scenarios in BPM.

Acknowledgment

We would like to thank the Bank of the Northeast of Brazil (Banco do Nordeste – BNB) for providing the case study as well as valuable requirements and feedback on the design of the tool.

References

1. Branco, M.C., Troya, J., Czarnecki, K., Küster, J., Völzer, H.: Matching Business Process Workflows Across Abstraction Levels. In: Proceedings of 15th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems. MODELS 2012, ACM/IEEE (2012)
2. Branco, M.C., Xiong, Y., Czarnecki, K., Küster, J., Völzer, H.: A case study on consistency management of business and IT process models in banking. Journal of Software and Systems Modeling SoSyM (2013)
3. Diskin, Z., Xiong, Y., Czarnecki, K.: From state- to delta-based bidirectional model transformations. In: Proceedings of the Third international conference on Theory and practice of model transformations. pp. 61–76. ICMT'10, Springer-Verlag, Berlin, Heidelberg (2010)
4. Diskin, Z., Xiong, Y., Czarnecki, K.: From State- to Delta-Based Bidirectional Model Transformations: the Asymmetric Case. Journal of Object Technology 10 (2011)
5. Diskin, Z., Xiong, Y., Czarnecki, K., Ehrig, H., Hermann, F., Orejas, F.: From State- to Delta-Based Bidirectional Model Transformations: the Symmetric Case. In: Proceedings of the 14th international conference on Model driven engineering languages and systems. pp. 304–318. MODELS'11, Springer-Verlag, Berlin, Heidelberg (2011)
6. Foster, J., Greenwald, M., Moore, J., Pierce, B., Schmitt, A.: Combinators for bidirectional tree transformations: A linguistic approach to the view-update problem. ACM Transactions on Programming Languages and Systems (TOPLAS) 29(3), 17 (2007)
7. Hegedüs, A., Horváth, A., Ráth, I., Branco, M.C., Varró, D.: Quick fix generation for DSMLs. In: Proceedings of IEEE Symposium on Visual Languages and Human-Centric Computing VLHCC 2011. IEEE (2011)
8. Hofmann, M., Pierce, B., Wagner, D.: Symmetric lenses. In: Proceedings of the 38th annual ACM SIGPLAN-SIGACT symposium on Principles of programming languages. pp. 371–384. POPL '11, ACM, New York, NY, USA (2011)
9. Hofmann, M., Pierce, B.C., Wagner, D.: Edit lenses. In: POPL. pp. 495–508 (2012)
10. Vanhatalo, J., Völzer, H., Leymann, F.: Faster and More Focused Control-Flow Analysis for Business Process Models Through SESE Decomposition. In: ICSOC 2007. pp. 43–55. LNCS, Springer-Verlag, Berlin, Heidelberg (2007)