CVSM 2013 Challenge: Recognizing High-level Edit Operations in Evolving Models

Timo Kehrer
Software Engineering Group
University of Siegen
kehrer@informatik.uni-siegen.de

Christian Gerth
Department of Computer Science
University of Paderborn
gerth@uni-paderborn.de

Abstract

In Model-Driven Engineering (MDE), models are subject to continuous change and typically exist in many versions during their lifetime. To support an efficient MDE, fully featured development environments are a necessity that in particular must enable the comparison of different versions of models. However, currently available comparison tools operate on low-level, sometimes tool-specific model representations and result in elementary difference representations. The resulting difference representations are often not understandable. Tool users rather prefer differences to be reported in terms of high-level edit operations, e.g. as offered by modern graphical editors, to present and handle model differences and to analyze the evolution of models. In this paper, we present two particular challenges for the detection of high-level edit operations between model versions in the absence of a change-log.

1 Introduction

The comparison of models is an essential part in Model-Driven Engineering (MDE) of software systems. In MDE, models are refined in team environments and several different versions of a single model are created. In order to merge the different model versions into an integrated version, differences between the models have to be considered. In scenarios where applied changes are not recorded in a change log, the differences between versions are identified by comparing the model versions. The identified differences are then represented and visualized appropriately that a human user can inspect the differences and can select a subset of the differences that shall be resolved by applying edit operations in an integrated model version [7].

For that purpose, a difference representation must be able to represent all possible differences that can occur between two models in order to transform a source model into a target model. In addition, based on a representation it must be possible to resolve a difference. Furthermore, similar to the merging of textual documents, models may also be merged by users with a limited background and knowledge regarding the model representation. These users shall be supported to inspect and resolve certain differences in order to obtain an integrated model version. For that purpose, the representation of differences between model versions has to fulfill specific requirements concerning user-friendliness. For instance differences must be displayed in a form that is understandable by a business user, e.g. by grouping related differences that need to be resolved together.

As a consequence, the granularity of the difference representation is of uttermost importance for model comparison tools. In the remainder of this paper, we present two challenges for the recognition of high-level edit operations between model operations for two different types of model. In Section 2, we consider the evolution of model transformation rules and in Section 3, we address the detection of high-level operations between process models.

2 Evolution of Transformation Rules

The modeling language which is subject of the first challenge is the graphical model transformation language provided by the EMF Henshin toolset [4]: Transformation rules are represented as EMF models [2] and can be comfortably edited using the graphical "Henshin integrated editor".

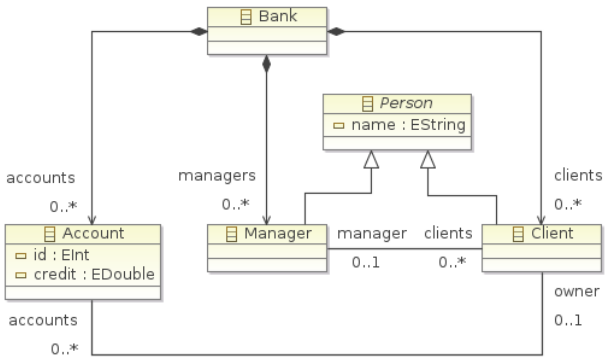

Figure 1: Bank metamodel serving as "type graph" of the sample transformation rules

The evolution scenario itself is based on the "Bank Accounts" example from the Henshin online

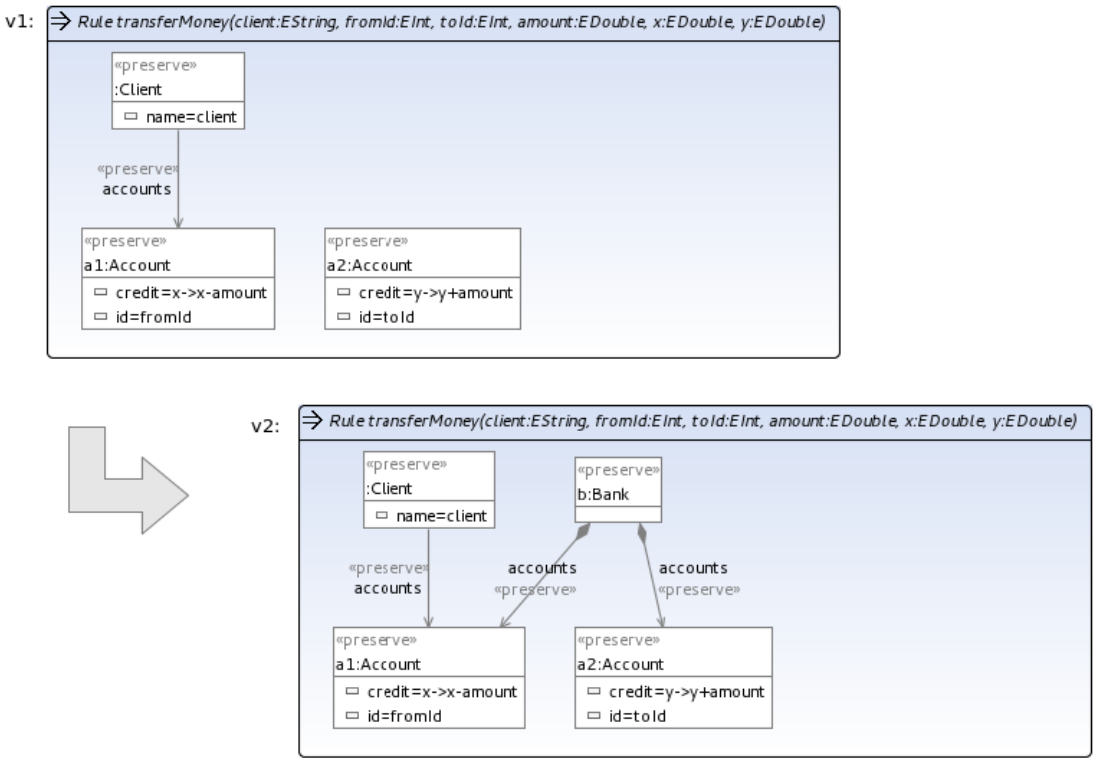

Figure 2: Transformation rule *transferMoney*; revision v1 (top) and v2 (bottom)

tutorials[6]. The tutorial shows some of the basic concepts of the Henshin transformation language and tool environment. Basically, the example defines

- a simple metamodel for representing clients and accounts of a bank (s. Figure 1), and
- shows how to realize a couple of sample transformations as Henshin transformation rules.

A more detailed introduction into the bank example as well as in-depth descriptions of the transformation rules are provided on the Henshin website.

As an example, Figure 2 shows two revisions of the Henshin transformation rule *transferMoney*. The two transformation rules represent a typical evolution step of the underlying transformation system. Revision v1 serves as base revision of the example. In revision v2, transferring money is restricted to bank internal money transfers, i.e. the accounts `a1` and `a2` must be contained by the same bank instance. Syntactically, v1 has been refine into v2 as follows:

1. `createPreserveNode`("transferMoney", "b", Bank): Create a `preserve`-node (i.e., a node being part of the intersection of the LHS- and the RHS-graph of the rule "transferMoney") which is of type `Bank` and which is named "b".
2. `createPreserveEdge`(b, a1, accounts): Create a `preserve`-edge of type `accounts` from the `preserve`-node `b` created in step 1 to the existing `preserve`-node `a1`.
3. `createPreserveEdge`(b, a2, accounts): Create a `preserve`-edge of type `accounts` from the `preserve`-node `b` created in step 1 to the existing `preserve`-node `a2`.

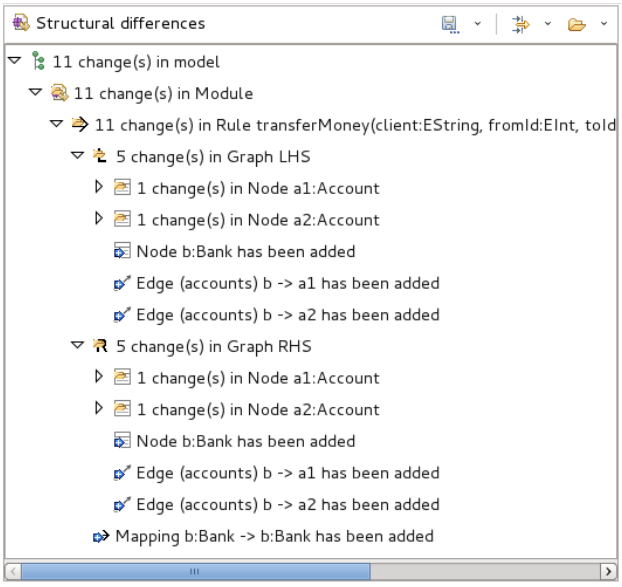

Figure 3: Results for comparing *transferMoney* v1 and v2 using EMF Compare

Consequently, this sequence of editing steps represents the **expected result** of a comparison of both revisions of *transferMoney*. Of course, editing steps

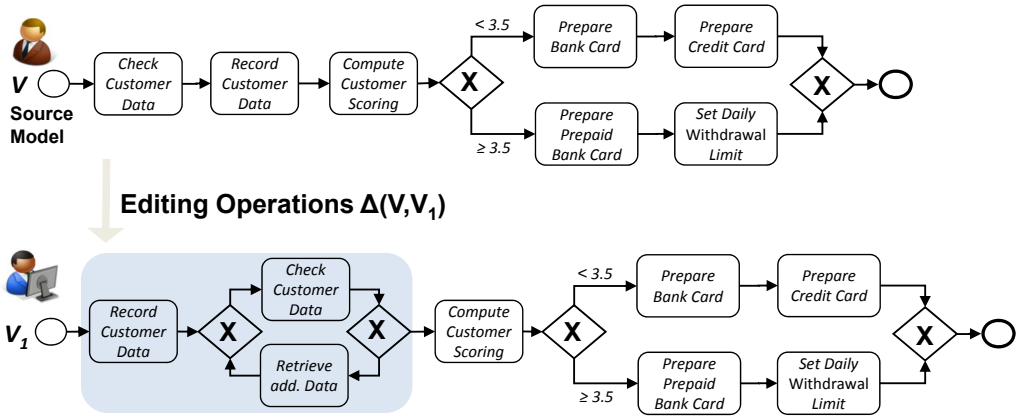

Figure 4: Two Versions of a Business Process Model

2. and 3. may be reported in inverse order as they are sequentially independent of each other.

Figure 3 shows the result comparing *transferMoney* v1 and v2 using EMF Compare [3]. Although a precise matching has been obtained based on persistent IDs, the low-level changes derived from the matching are far apart from reflecting the original sequence of editing steps that have been actually performed.

To participate in this challenge, we invite submissions that address the recognition of applied high-level edit operations as available in the graphical Henshin editor.

3 Evolution of Business Process Models

As a second challenge for the recognition of applied high-level edit operations, we consider the concurrent development of business process models in a team environment, e.g. during the development of a service-oriented software system. Figure 1 visualizes such a multi-user scenario. There, two process models in the Business Process Model and Notation (BPMN) language are shown that describe the necessary steps to open a banking account for a customer in a bank. The source model V was created first and afterwards V was refined into the version V1.

The intention of the refinement of version V into $V1$ was as follows:

1. `InsertCyclicFragment`(V, f_{Loop}, "Record Customer Data", "Compute Customer Scoring"): Insert a cyclic fragment f_{Loop} into the process model V between the tasks "Record Customer Data" and "Compute Customer Scoring".

2. `MoveTask`(V, "Check Customer Data", *Start Event*, "Record Customer Data", $Exclusive - Merge_{fLoop}$, $Exclusive - Decision_{fLoop}$): Move task "Check Customer Data" in process model V from its original position between the *Start Event* and "Record Customer Data" into the cyclic fragment f_{Loop} at the specified position between $Exclusive - Merge_{fLoop}$ and $Exclusive - Decision_{fLoop}$.

3. `InsertTask`(V, "Retrieve additional Data", $Exclusive - Decision_{fLoop}$, $Exclusive - Merge_{fLoop}$): Insert a task "Retrieve add. Data" into the cyclic fragment f_{Loop} at the specified position between $Exclusive - Decision_{fLoop}$ and $Exclusive - Merge_{fLoop}$.

This sequence of high-level compound operations represents a result of a comparison of both process model versions[1], which we claim is understandable by a user. The understandability of the difference representation could be improved by providing visual feedback in terms of highlighted model elements in the process model whenever an edit operation is selected.

Analogously to the high-level edit operation detection between transformation rules, we compared the process model versions using a comparison approach, which results in low-level, elementary edit operations, e.g. by using EMF Compare [3]. The resulting change log $\Delta(V, V1)$ that represents the differences between the process model versions is shown in Figure 5.

Based on elementary edit operations it is difficult to grasp the actual high-level logical/structural change that was applied to obtain the process model $V1$. In general, even simple differences result in a multitude of elementary operations. In addition, based on elementary operations it is difficult to identify, which differences can be resolved independently, i.e. without the necessity to resolve other differences in advance. As a consequence, merging two process model versions using a difference representation based on elementary edit operations becomes a trial-and-error approach.

To participate in this challenge, we invite submissions that address the recognition of applied high-level differences in the evolution of process models and rep-

[1] Please note, the editing operations 2. and 3. are sequentially independent of each other and may be reported in inverse order.

$\Delta(V, V1)$

$DeleteEdge(V, e_2,$ "Record Customer Data", "Compute Customer Scoring")

$InsertNode(V, Exclusive - Join_{fLoop})$

$InsertNode(V, Exclusive - Split_{fLoop})$

$InsertEdge(V, e_{11},$ "Record Customer Data", $Exclusive - Join_{fLoop})$

$InsertEdge(V, e_{12}, Exclusive - Split_{fLoop},$ "Compute Customer Scoring")

$DeleteEdge(V, e_0, Start,$ "Check Customer Data")

$DeleteEdge(V, e_1,$ "Check Customer Data", "Record Customer Data")

$InsertEdge(V, e_{13}, Start,$ "Record Customer Data")

$InsertNode(V,$ "Retrieve additional Data")

$InsertEdge(V, e_{14}, Exclusive - Join_{fLoop},$ "Check Customer Data")

$InsertEdge(V, e_{15},$ "Check Customer Data", "$Exclusive - Split_{fLoop}$)

$InsertEdge(V, e_{16},$ "Retrieve additional Data", "$Exclusive - Join_{fLoop}$)

$InsertEdge(V, e_{17},$ "$Exclusive - Split_{fLoop},$ "Retrieve additional Data")

Figure 5: Change Log $\Delta(V, V1)$ consisting of Change Operations that represent the Elementary Differences between the Process Models V and $V1$

resent these differences in terms of intuitive, high-level edit operations.

4 Conclusion

In this proposal for a CVSM 2013 challenge, we address the problem of how to semantically lift low-level differences, which are produced by currently available differencing engines for models, towards user-comprehensible, intuitive edit operations. In both of our examples, only a small number of high-level edit operations has been applied. However, each of these applied high-level editing steps causes a considerable amount of low-level differences in terms of elementary edit operations.

As a consequence, to provide fully-featured development environments for Model-Driven Engineering (MDE), there is strong necessity for adaptable approaches that are able to detect high-level edit operations on evolving models. The example can be downloaded from the CVSM 2013 webpage: http://pi.informatik.uni-siegen.de/CVSM2013

Acknowledgement

This work was partially supported by the DFG (German Research Foundation) under the Priority Programme SPP1593: Design For Future - Managed Software Evolution.

References

[1] Biermann, E., Ermel, C., Taentzer, G.; Lifting Parallel Graph Transformation Concepts to Model Transformation based on the Eclipse Modeling Framework; ECEASST 26; 2010

[2] EMF: Eclipse Modeling Framework; http://www.eclipse.org/emf; 2013

[3] EMF Compare Project; http://www.eclipse.org/emf/compare/; 2013

[4] EMF Henshin; http://www.eclipse.org/modeling/emft/henshin/; 2013

[5] Heckel, R.; Graph Transformation in a Nutshell; in: Electronic Notes in Theoretical Computer Science (ENTCS), Volume 148, Issue 1, February, 2006

[6] Krause, C.; Henshin Example: Bank Accounts; Available from the Henshin website [4]; 2013

[7] Küster, J. M., Gerth, C., Förster, A., Engels, G. Detecting and Resolving Process Model Differences in the Absence of a Change Log; in: Proceedings of the 6th International Conference on Business Process Management (BPM'13), Springer (Berlin/Heidelberg), LNCS, vol. 5240, pp. 244-260, 2008