Application of Model Comparison Techniques to Model Transformation Testing

Matthew Stephan and James R. Cordy
Queen's University, Kingston, Ontario, Canada
{*stephan, cordy*}*@cs.queensu.ca*

Keywords: Model Transformations: Model Comparison: Model Transformation Quality Assurance: Model Transformation Testing: Metamodel Comparison

Abstract: In this paper, we discuss model-to-model comparison techniques that can be used to assist with model-to-model transformation testing. Using an existing real-world model transformation, we illustrate and qualitatively evaluate the comparison techniques, highlighting the associated strengths and weaknesses of each in the context of transformation testing.

1 INTRODUCTION

Although the area of *model transformation testing* (MTT) is relatively new, there already is a consensus that *model comparison* (MC) is a key and fundamental component of it (Baudry et al., 2010; Kolovos et al., 2006; Mazanek and Rutetzki, 2011). MC entails identifying similarities and differences between a pair, or multiple, models. There are many different MC approaches intended for a variety of different purposes (Stephan and Cordy, 2013). However, there are relatively few that are explicitly intended to facilitate MTT. Furthermore, there is no consensus as to which techniques are best suited. For example, current transformation testing implementations, such as *Transformation Judge*[1], simply use comparison tools based on the fact that they are "the most widely used" (Mazanek and Rutetzki, 2011).

In this paper we discuss and qualitatively evaluate the available model comparison approaches that can be used as part of a MTT oracle, including some approaches that may not have been intended for this purpose. We focus on model-to-model (M2M) transformation testing because the verification of these transformations is more challenging than model-to-code (Lin et al., 2005) and the literature contains more M2M comparison approaches. Also, M2M transformations have been identified as one of the main road blocks necessary to overcome in facilitating change evolution in MDE (Sendall and Kozaczynski, 2003).

Our main contribution is an exploration of which MC approach(es) are the most promising for a MTT oracle. To demonstrate and discuss the comparison approaches in the transformation testing context, we use a heterogeneous M2M transformation from the literature and discuss both homogeneous and heterogeneous comparisons.

[1]http://sites.google.com/site/transformationjudge/

2 BACKGROUND

Model Transformation Testing involves executing a set of transformations on source models and comparing the output of the transformations with the expected output (Selim et al., 2012). If a generated output model corresponds to expected output, then one can infer that the transformation has been performed correctly. Otherwise, the transformation needs to be updated. Oracle implementation is an outstanding challenge in accomplishing MTT (Baudry et al., 2006) and can be done in a variety of ways (Mottu et al., 2008), many involving forms of MC.

Model Comparison involves discovering similarities and differences among different models. Some applications include model versioning, model clone detection, model merging, and, to a lesser extent, MTT (Stephan and Cordy, 2012). In this paper we consider all of the approaches intended for MTT and others that we believe can be adapted. We discuss both the calculation and visualization capabilities of each approach.

Automatic MC is crucial in achieving MTT because manual comparison for testing is not practical, nor are character- or line- based comparisons. There are three different forms of MC being used in the context of an oracle (Mottu et al., 2008): 1] Comparing an output model with a reference model generated by a reference transformation; 2] Comparing an input model with a model that was transformed once by the transformation and again by the inverse transformation, if the transformation is injective; 3] Comparing an output model with an expected output model, if available. This list refers to the steps preceding the comparison. For each of these, the comparisons themselves do not differ that much: They are likely

homogeneous, or endogenous, meaning they are performing comparisons of models from the same metamodel. In this paper, we consider both homogeneous and heterogeneous, or exogenous, comparisons, the latter being useful for comparing the input and output model of a heterogeneous transformation.

3 MODEL COMPARISON TECHNIQUES

Using a recent survey on MC (Stephan and Cordy, 2013), we include any approaches that are intended to deal with MTT explicitly, meaning the comparisons are specifically tailored to provide information that can help with transformation testing.

C-Saw[2] is a general-purpose model transformation engine implemented as a Generic Modeling Environment (GME)[3] plugin, originally intended as an aspect weaver for modeling languages. It facilitates the execution of a model transformation specification and includes a basic model comparison algorithm (Lin et al., 2005).

DSMDiff (Lin et al., 2007) is a MC tool that is intended to work with domain-specific models. It is built on top of the GME, a tool intended to assist with developing and managing domain-specific models. Although there are attempts to bridge the gap between GME and EMF (Bézivin et al., 2005), there is nothing yet that can convert instance models from EMF to GME instance models. Thus, we do not include it in our case study.

EMF Compare (Brun and Pierantonio, 2008) is an Eclipse project that implements MC for EMF [4] models. It avoids using static unique identifiers and relies more on similarity based-matching to add flexibility and usefulness in more contexts.

The *Epsilon Comparison Language*(ECL)[5] is a rule-based approach to MC. It provides modelers the ability to utilize language-specific information to describe semantics and to make MC more accurate and efficient (Kolovos, 2009). It allows comparison of models subscribing to different metamodels.

SmoVer (Reiter et al., 2007) is a technique that can likely be adapted to perform MC for MTT. Intended to assist with model versioning for EMF-based models, *SmoVer* performs both syntactic and semantic comparisons. Semantic comparisons are evaluations of models that have undergone a model transformation into *semantic views*. Unfortunately, much of the described tool has yet to be implemented including a lack of adequate visualization of differences[6] and having its comparison engine inseparable from its version control system. As such, we do not include this approach in our case study.

[2]http://www.gray-area.org/Research/C-SAW/

[3]http://www.isis.vanderbilt.edu/Projects/gme/

[4]http://www.eclipse.org/emf/

[5]http://www.eclipse.org/gmt/epsilon/doc/ecl/

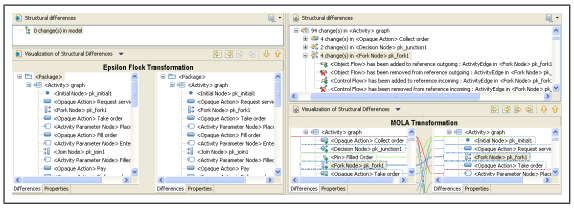

Figure 1: Homogeneous MC using *EMF Compare*.

4 CASE STUDY

Our case study for comparing *EMF Compare* and *ECL* involves testing transformations for model migration. Specifically, we consider migration from UML 1.4 activity diagrams to UML 2.2 activity diagrams (Rose et al., 2010b). We believe this is a representative and sufficient example: 1] It was used in a transformation tool contest in 2010[7], meaning it was viewed as a fair and representative set of transformations; 2] The transformation is challenging and realistic because the differences between the source and target metamodels are significant; and 3] As a result of it coming from a tool contest, we have a relatively large set of solutions and output models to work with via the contest website.

4.1 Homogeneous Comparison

The comparison techniques should perform well as homogeneous MC is much simpler than heterogeneous. Any discrepancies between the expected and actual output models mentioned are not, necessarily, a reflection of the transformation tools, but of the transformations that were submitted to the transformation tool contest.

4.1.1 EMF Compare

In order to use *EMF Compare* for MTT, we configure it to ignore unique identifiers. In Figure 1, we show the comparison results of the expected output with the output models that resulted from transformations from two M2M tools, *Flock* (Rose et al., 2010a) and *MOLA* (Kalnins et al., 2005) on the left and right of the figure, respectively. The output model from *Flock* is identical to our expected output. From a testing perspective, this means that *EMF Compare* is able to show when a test case is executed as expected. For the *MOLA* transformation, there seems to be a symmetric list of changes under many of the model elements. For example, for every **control flow** or **object**

[6]http://smover.tk.uni-linz.ac.at/prototype.php

[7]http://planet-mde.org/ttc2010/index.php

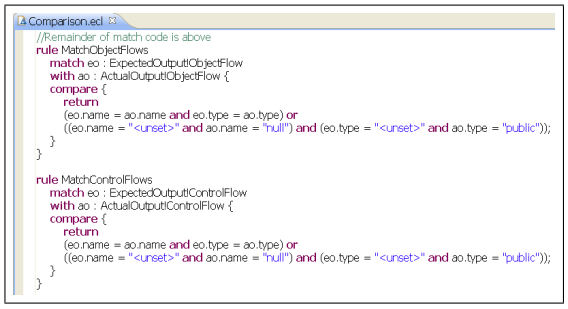

Figure 2: ECL rules to remove false positives.

flow reference missing when compared to its corresponding element, a reference addition of the same type appears. It is likely these differences are trivial and these references are to the same object: *EMF Compare* generates false positives, that is, elements that are identified as different but should not be. In addition, it is very difficult to pinpoint the differences.

4.1.2 ECL

While *ECL* requires more work to define the rules for matching models, it excels at matching corner cases based on domain knowledge and user input (Kolovos, 2009). So, if we are able to ascertain the lower level problems, we can reduce the number of false positives using an appropriate *ECL* rule.

We find the lower level problem from our example in Figure 1: The elements are not matched because the *MOLA* transformation sets the name and visibility of these elements to **null** and **public**, respectively, rather than to **unset**, as in our expected output. The issue is not that the *MOLA* transformation is wrong in doing this, it is that our comparison method should interpret them as equal. We specify this using the *ECL* as demonstrated in the rule definitions in Figure 2. The top **rule** block accounts for the **ObjectFlow** false positives while the bottom **rule** block accounts for **ControlFlow** false positives. Implementing this rule removes 34 of 95 listed differences outlined in our transformation test. Many remaining false positives can be rectified with analogous rule definitions to those in Figure 2.

4.2 Heterogeneous Comparison

An interesting application of MC in MTT that has yet to be investigated is comparing metamodels in a heterogeneous transformation and using that to guide testing input, that is, allow test generation from metamodel MC. This differs from existing approaches (Sen et al., 2009). We provide a brief illustration using *EMF Compare* in Figure 3 showing MC of the different metamodels from the provided case study, with the evolved model on the left. This list of differences might be a good starting place for test-case generation. For example, to test a model

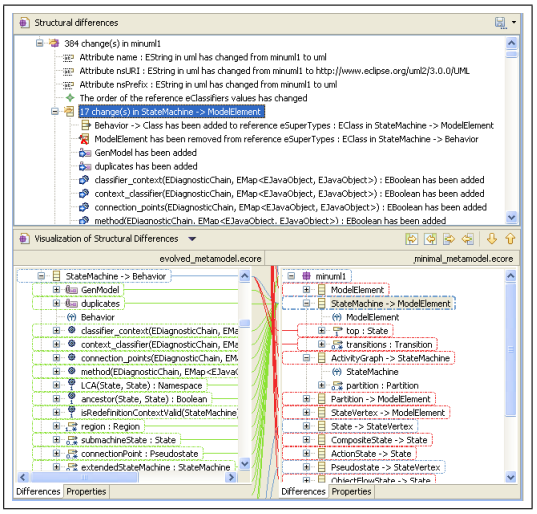

Figure 3: Comparison of metamodels.

transformation with respect to the **StateMachine** element in isolation, we could write assertions that ensure those 17 or so changes have been represented accordingly.

It is clear that *EMF Compare* is not well suited for heterogeneous comparisons. The *EMF Compare* matching algorithm produces the same, relatively unhelpful information in that they fail to match what, semantically, we know should be a match near the top of the model hierarchy. Only straight-forward matches are discovered, such as those that have the same or similar names. For example, **StateMachine**, **State**, and **PseudoState** are present in both metamodels, but it is difficult to identify the differences of their children and other should-be matches at the same level are missed.

Thus, we are left with *ECL* or, possibly, *SmoVer*, if it were to be extended appropriately. We would write rules, like the ones in Figure 2, that match UML 1.4 components to their corresponding 2.2 components. We can indicate matching metamodel elements at all levels and will, consequently, be left with more meaningful comparisons. While this is somewhat equivalent to writing the actually transformation itself, it is done from a comparison and declarative perspective, allowing for an extra level of verification.

5 SUMMARY

Our evaluation is summarized in Table 1. The first column lists desirable criteria: For boolean criteria, a check mark is better than an 'X'. For non-boolean criteria, the more pluses, the better. In terms of being widely-applicable, *C-Saw* limits itself to models with static identifiers. *EMF Compare*, *ECL*, and *SmoVer* work with EMF only, but the first two can be extended to work with other formats. *DSMDiff* is the most gen-

Table 1: Evaluation of MC approaches using the criteria from Mazanek and Rutetzki (2011).

Criteria	Tool				
	C-Saw	DSMDiff	EMF Compare	ECL	SmoVer
Widely applicable	+	+++	++-	++-	++
No case-specific configuration	√	X	√	X	X
Not rely on unique identifiers	X	√	√	√	√
Output is human-readable and precise	+	+	++	+++	X
Extent of Match Information	+	+	++	++	+
Command Line Execution	X	X	√	√	X

Legend
√ = yes , X = no ; - = 0.5, + = 1 , ++ = 2 , +++ = 3

eral. *C-Saw* and *EMF Compare* are the only ones that do not require case-specific configuration. While the criteria authors state this is positive, we disagree and believe case-specific configuration is important for satisfying the other criteria, notably precision of matches and differences. For readability and precision, all of the comparison tools seem to have room for improvement. We give *EMF Compare* and *ECL* the top scores, giving *ECL* the advantage because the lower-level match rules increase precision. None of the tools provide as much matching data as they could and should, but *EMF Compare* and *ECL* provide the most. The case study and other examples revealed no significant differences in performance.

MC is key for achieving acceptable MTT. We presented a subset of MC techniques that are both well-suited to MTT and representative of the available approaches. It is clear that *C-Saw* and *SmoVer* are currently inadequate because of their reliance on unique identifiers and lack of implementation, respectively.

Using a UML activity diagram case study, we discussed *EMF Compare*'s and *ECL*'s ability to perform heterogeneous and homogeneous comparisons for MTT. *ECL* scores the highest out of the approaches, however, some may be discouraged by its need for case-specific configurations. We argue that it is a good trade off, especially in the context of MTT.

A good visualization of both matches and differences is crucial in the context of MTT. Existing visualization techniques can be improved and new ones should be investigated. We also presented the idea of performing metamodel comparison to achieve test generation from models. Lastly, a hybrid approach to MTT by reverse engineering the metamodels and is built on top of *ECL* may be very useful. While some advancement has been made in MTT, this paper shows that there is still some work to be done and improvements to be made.

REFERENCES

Baudry, B., Dinh-trong, T., Mottu, J.-M., Simmonds, D., France, R., Ghosh, S., Fleurey, F., and Traon, Y. L. (2006). Model transformation testing challenges. In *ECMDA*.

Baudry, B., Ghosh, S., Fleurey, F., France, R., Le Traon, Y., and Mottu, J. (2010). Barriers to systematic model transformation testing. *Comm. ACM*, 53(6):139–143.

Bézivin, J., Brunette, C., Chevrel, R., Jouault, F., and Kurtev, I. (2005). Bridging the generic modeling environment (GME) and the Eclipse modeling framework (EMF). In *OOPSLA*.

Brun, C. and Pierantonio, A. (2008). Model differences in the Eclipse modelling framework. *EJIP*, pages 29–34.

Kalnins, A., Barzdins, J., and Celms, E. (2005). Model transformation language MOLA. *Model Driven Architecture*, pages 900–915.

Kolovos, D. (2009). Establishing correspondences between models with the Epsilon Comparison Language. In *MDAFA*, pages 146–157. Springer.

Kolovos, D., Paige, R., and Polack, F. (2006). Model comparison: a foundation for model composition and model transformation testing. In *IWGIMM*, pages 13–20. ACM.

Lin, Y., Gray, J., and Jouault, F. (2007). DSMDiff: a differentiation tool for domain-specific models. *EJIS*, 16(4):349–361.

Lin, Y., Zhang, J., and Gray, J. (2005). A testing framework for model transformations. *MDSD*, pages 219–236.

Mazanek, S. and Rutetzki, C. (2011). On the importance of model comparison tools for the automatic evaluation of the correctness of model transformations. In *IWMCP*, pages 12–15. ACM.

Mottu, J., Baudry, B., and Traon, Y. (2008). Model transformation testing: oracle issue. In *IWSTVV*, pages 105–112.

Reiter, T., Altmanninger, K., Bergmayr, A., Schwinger, W., and Kotsis, G. (2007). Models in conflict-detection of semantic conflicts in model-based development. In *WMDEIS*, pages 29–40.

Rose, L., Kolovos, D., Paige, R., and Polack, F. (2010a). Model migration with epsilon flock. *TPMT*, pages 184–198.

Rose, L. M., Kolovos, D. S., Paige, R. F., and Polack, F. A. (2010b). Model migration case for TTC 2010. In *Transformation Tool Contest 2010*, pages 1–6.

Selim, G., Cordy, J. R., and Dingel, J. (2012). Model transformation testing: The state of the art. In *AMT*, page 6 pp.

Sen, S., Baudry, B., and Mottu, J. (2009). Automatic model generation strategies for model transformation testing. *Theory and Practice of Model Transformations*, pages 148–164.

Sendall, S. and Kozaczynski, W. (2003). Model transformation: The heart and soul of model-driven software development. *IEEE Software*, 20(5):42–45.

Stephan, M. and Cordy, J. R. (2012). A survey of methods and applications of model comparison. Technical report, Queen's University. TR. 2011-582 Rev. 2.

Stephan, M. and Cordy, J. R. (2013). A survey of model comparison approaches and applications. In *Modelsward*. to appear.