Towards a Framework for Multidirectional Model Transformations

Nuno Macedo
HASLab
INESC TEC & Universidade do Minho, Braga, Portugal
nfmmacedo@di.uminho.pt

Alcino Cunha
HASLab
INESC TEC & Universidade do Minho, Braga, Portugal
alcino@di.uminho.pt

Hugo Pacheco
National Institute of Informatics
Tokyo, Japan
hpacheco@nii.ac.jp

ABSTRACT

The *Query/View/Transformation Relations* (QVT-R) standard for bidirectional model transformation is notorious for its underspecified semantics. When restricted to transformations between pairs of models, most of the ambiguities and omissions have been addressed in recent work. Nevertheless, the application of the QVT-R language is not restricted to that scenario, and similar issues remain unexplored for the multidirectional case (maintaining consistency between more than two models), that has been overlooked so far.

In this paper we first discuss ambiguities and omissions in the QVT-R standard concerning the mutidirectional transformation scenario, and then propose a simple extension and formalization of the checking and enforcement semantics that clarifies some of them. We also discuss how such proposal could be implemented in our `Echo` bidirectional model transformation tool. Ours is just a small step towards making QVT-R a viable language for bidirectional transformation in realistic applications, and a considerable amount of basic research is still needed to fully accomplish that goal.

Categories and Subject Descriptors

D.2.2 [**Software Engineering**]: Design Tools and Techniques

1. INTRODUCTION

In model-driven engineering (MDE), models are the main development artifact. Typically, multiple models may coexist in the same environment, to represent different views of the overall system or similar components at different levels of abstraction, and all these models must ideally be kept consistent with each other. In the past years, extensive work on *bidirectional model transformations* [2] has been devoted to the particular purpose of maintaining the consistency of two models. One of the most popular approaches in MDE is the OMG's QVT standard [8], and in particular the QVT-R language, that proposes describing a bidirectional transformation as a declarative relation between two meta-models.

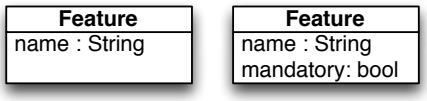

Figure 1: Configuration (`CF`) and feature model (`FM`).

The standard prescribes two modes to interpret a QVT-R specification: *checkonly mode* tests the consistency between particular models and *enforce mode* runs a transformation in a particular direction to repair inconsistent models. Thus, a QVT-R transformation between two metamodels can be understood in the abstract framework of *constraint maintainers* [7]: a specification denotes a consistency relation $R \subseteq A \times B$, from which a forward transformation $\overrightarrow{R} : A \times B \to B$ and a backward transformation $\overleftarrow{R} : A \times B \to A$, that modify one of the elements to restore consistency, are inferred. In this paper we will be dealing with multiple target models, so these will instead be denoted by $\overrightarrow{R}_A : B \to A$ and $\overrightarrow{R}_B : A \to B$, the subscript identifier denoting the direction of the transformation and the fact that it retrieves information from the original target models.

Acceptance and development of effective tool support for the QVT-R standard has been slow, possibly due to ambiguities in its checking and enforcement semantics. For the bidirectional scenario (maintaing consistency between two models), most of the issues have been clarified in recent work [9, 1] and implementations with precise semantics already exist, namely our own `Echo` tool [5]. Nonetheless, the bidirectional scenario is not sufficient to tackle some applications. An arbitrary number of models may coexist in the same model-driven environment, and their complex interrelationship may not be decomposable into a set of bidirectional relationships to be maintained separately.

As an example, consider the problem of keeping the consistency between a *feature model* and a set of valid *configurations*. For the sake of simplicity, assume that feature models `FM` consist of named features, that may or not be mandatory, and configurations `CF` are simply a set of selected features; the respective meta-models are depicted in Figure 1.

The relationship $F \subseteq \mathsf{FM} \times \mathsf{CF}^k$ between a feature model and k configurations can be decomposed into two parts $F = MF \cap OF$: relation $MF \subseteq \mathsf{FM} \times \mathsf{CF}^k$ expresses that mandatory features match exactly the set of features appearing in every `CF`; and relation $OF \subseteq \mathsf{FM} \times \mathsf{CF}^k$ expresses that the `FM` contains at least the union of all selected features. Note that due to the intention of having features present in all `CF`s set

as mandatory in the FM, relation *MF* cannot be decomposed into k bidirectional relations between the FM and each CF.

This kind of *multidirectional* scenario is already informally foreseen in the QVT-R standard, that admits an arbitrary number of domains in a QVT-R relation. However, the proposed checking semantics are too inflexible and only able to represent a restricted subset of consistency relations. For instance, none of the above relations can be specified using the standard checking semantics. Moreover, the standard hints at an enforcement semantics with very limited applicability. Namely, from a consistency relation such as F, the standard only prescribes the derivation of two transformations:

- $\overrightarrow{F}_{\mathsf{FM}} : \mathsf{CF}^k \rightarrow \mathsf{FM}$, for propagating updates to the configurations back to the feature model;
- $\overrightarrow{F^i}_{\mathsf{CF}} : \mathsf{FM} \times \mathsf{CF}^{k-1} \rightarrow \mathsf{CF}$ for any $i \in 0..k$, for propagating updates to the feature model and the remaining configurations into a specific target configuration.

In the multidirectional scenario this is a very restrictive view of the enforcement semantics, as the user may wish to restore consistency in many ways depending on the context, leading to different update propagation transformations. Consider, e.g., the following interesting and alternative instantiations:

- $\overrightarrow{F}_{\mathsf{CF}^k} : \mathsf{FM} \rightarrow \mathsf{CF}^k$. This would allow updates to the feature model to be propagated to more than one configuration. For example, if a feature is changed to mandatory it must be selected in all configurations; this simple update could not be handled by the standard transformations, since full consistency could not be restored by a single update translation.
- $\overrightarrow{F^i}_{\mathsf{FM} \times \mathsf{CF}^{k-1}} : \mathsf{CF} \rightarrow \mathsf{FM} \times \mathsf{CF}^{k-1}$ for any $i \in 0..k$. This would provide more flexibility in the propagation of updates to a configuration, as all the remaining artifacts are allowed to change. For example, if name of a feature is changed, the natural way to recover consistency is to change the name of that feature in all the remaining configurations and in the feature model.

The main goal of this paper is to start shedding some light on this currently unexplored multidirectional scenario. In particular, we explore the applicability of QVT-R for specifying multidirectional model transformations and discuss some semantic issues that arise in this setting. To overcome its current limitations, we propose a simple extension that enables the specification of interesting multidirectional transformations, and discuss how to infer different kinds of consistency-restoring transformations from the *same* multidirectional specification. Finally, we show how Echo [6], a tool supporting QVT-R bidirectional transformations, could be easily adapted to accommodate such extensions.

2. QVT-R CHECKING SEMANTICS

Typical model transformation languages (like QVT-R [8] and ATL [4]) provide mechanisms that allow reasoning about transformations in a structured way, rather than simply specifying arbitrary constraints over the models in some general constraint language (like OCL). As evidence, both these languages rely on domain patterns, controlling the elements over which a transformation is applied, while QVT-R supports additional pre- and post-conditions. A QVT-R program is defined as a set of *relations* in the following syntax.

```
[top] relation R {
    [variable declarations]
    domain m1 a1 : A1 { π1 }
    ...
    domain mn an : An { πn }
    [when { ψ }] [where { φ }] }
```

In this notation, π_i denotes a domain pattern over an element a_i of model m_i (for $i \in 1..n$), and ψ and ϕ are arbitrary pre- and post-conditions. According to the standard, testing the consistency specified by top relations consists of running n *directional* tests (denoted by the subscript metamodel identifier), each validating one of the models, as:

$$R\ (m_1 : M_1, ..., m_n : M_n) \equiv \wedge_{i \in 1..n}\ R_{M_i}\ (m_1, ..., m_n)$$

For each of these R_{M_i} relations, the idea is that if ψ holds, then for all a_j elements such that π_j holds, for $j \neq i$, there must exist an element a_i such that π_i and ϕ hold. For a non-top relation, its constraints must hold only when called by other relations. In the bidirectional case, we end up with:

$$\begin{array}{l} R_{M_1}\ (m_1 : M_1, m_2 : M_2) \equiv \\ \quad \forall\ xs \mid \psi_{M_1} \wedge \pi_{M_2} \Rightarrow (\exists\ ys \mid \pi_{M_1} \wedge \phi_{M_1}) \\ \quad \textbf{where}\ xs = \mathsf{fv}(\psi \wedge \pi_{M_2}), ys = (\mathsf{fv}(\pi_{M_1} \wedge \phi)) - xs \\ R_{M_2}\ (m_1 : M_1, m_2 : M_2) \equiv \\ \quad \forall\ xs \mid \psi_{M_2} \wedge \pi_{M_1} \Rightarrow (\exists\ ys \mid \pi_{M_2} \wedge \phi_{M_2}) \\ \quad \textbf{where}\ xs = \mathsf{fv}(\psi \wedge \pi_{M_1}), ys = (\mathsf{fv}(\pi_{M_2} \wedge \phi)) - xs \end{array}$$

This checking semantics has a close correspondence to the enforcement semantics: roughly, we just need to replace existential quantifiers for generation procedures [8]. Relations may call other relations in their pre- and post-conditions, which are also run in the appropriate direction (hence the identifier on ψ and ϕ denoting the required direction).

2.1 Issues with the Multidirectional Scenario

Back to our running example from Section 1, consider that we have a pair of configurations ($k = 2$). How can the *MF* consistency relation be specified in QVT-R? As a first attempt, let us consider the following specification.

```
top relation MF { n : String;
      domain cf1 s1 : Feature { name = n }
      domain cf2 s2 : Feature { name = n }
      domain fm  f  : Feature { name = n,
                               mandatory = true } }
```

The free variable n relates selected features in each configuration with mandatory features in the feature model, resulting the consistency relation:

$$\begin{array}{l} MF\ (cf_1 : \mathsf{CF}_1, cf_2 : \mathsf{CF}_2, fm : \mathsf{FM}) \equiv \\ \quad MF_{\mathsf{FM}}\ (cf_1, cf_2, fm) \wedge \\ \quad MF_{\mathsf{CF}_1}\ (cf_1, cf_2, fm) \wedge MF_{\mathsf{CF}_2}\ (cf_1, cf_2, fm) \end{array}$$

Each of these directional tests is then concretized as:

$$\begin{array}{l} MF_{\mathsf{FM}}\ (cf_1 : \mathsf{CF}, cf_2 : \mathsf{CF}, fm : \mathsf{FM}) \equiv \\ \quad \forall\ n : \mathsf{String}, s_1 : \mathsf{Feature}_{cf_1}, s_2 : \mathsf{Feature}_{cf_2} \mid \\ \qquad n = s_1.\mathsf{name} \wedge n = s_2.\mathsf{name} \Rightarrow \\ \qquad\quad (\exists\ f : \mathsf{Feature}_{fm} \mid n = f.\mathsf{name} \wedge f \in \mathsf{mandatory}) \\ MF_{\mathsf{CF}_1}\ (cf_1 : \mathsf{CF}, cf_2 : \mathsf{CF}, fm : \mathsf{FM}) \equiv \\ \quad \forall\ n : \mathsf{String}, f : \mathsf{Feature}_{fm}, s_2 : \mathsf{Feature}_{cf_2} \mid \\ \qquad n = s_1.\mathsf{name} \wedge n = f.\mathsf{name} \wedge f \in \mathsf{mandatory} \Rightarrow \\ \qquad\quad (\exists\ s_1 : \mathsf{Feature}_{cf_1} \mid n = s_1.\mathsf{name}) \\ MF_{\mathsf{CF}_2}\ (cf_1 : \mathsf{CF}, cf_2 : \mathsf{CF}, fm : \mathsf{FM}) \equiv ... \end{array}$$

But let us concretely analyze the meaning of these predicates. $MF_{\mathtt{FM}}$ expresses part of the intended behavior — if the two configurations have the same selected feature then such feature is mandatory. It can be rephrased as:

$$MF_{\mathtt{FM}}\ (cf_1 : \mathtt{CF}, cf_2 : \mathtt{CF}, fm : \mathtt{FM}) \equiv \\ \mathtt{Feature}_{cf_1}.\mathtt{name} \cap \mathtt{Feature}_{cf_2}.\mathtt{name} \subseteq \\ (\mathtt{Feature}_{fm} \cap \mathtt{mandatory}).\mathtt{name}$$

However, the reverse implication

$$MF_{\mathtt{CF}_1 \times \mathtt{CF}_2}\ (cf_1 : \mathtt{CF}, cf_2 : \mathtt{CF}, fm : \mathtt{FM}) \equiv \\ (\mathtt{Feature}_{fm} \cap \mathtt{mandatory}).\mathtt{name} \subseteq \\ \mathtt{Feature}_{cf_1}.\mathtt{name} \cap \mathtt{Feature}_{cf_2}.\mathtt{name}$$

is not entailed by $MF_{\mathtt{CF}_1}$ and $MF_{\mathtt{CF}_2}$. Looking at $MF_{\mathtt{CF}_1}$, the selection of the s_1 feature depends both on f and s_2, thus, if there are *no* selected features in cf_2, $MF_{\mathtt{CF}_1}$ will be trivially true due to the empty range in the universal quantification. This problem persists whatever the domain patterns (and pre- or post-conditions), and the intended specification for MF cannot be realized by any QVT-R relation (with the standard semantics) between features in the three models.

This problem could be easily solved if we could control the extent of the universal quantifications in the semantics of $MF_{\mathtt{CF}_1}$ and $MF_{\mathtt{CF}_2}$, namely to range only over the feature model and ignore the remaining configurations.

$$MF_{\mathtt{CF}_1}\ (cf_1 : \mathtt{CF}, cf_2 : \mathtt{CF}, fm : \mathtt{FM}) \equiv \\ \forall\ n : \mathtt{String}, f : \mathtt{Feature}_{fm} \mid \\ n = f.\mathtt{name} \wedge f \in \mathtt{mandatory} \Rightarrow \\ (\exists\ s_1 : \mathtt{Feature}_{cf_1} \mid n = s_1.\mathtt{name})$$

$$MF_{\mathtt{CF}_2}\ (cf_1 : \mathtt{CF}, cf_2 : \mathtt{CF}, fm : \mathtt{FM}) \equiv \\ \forall\ n : \mathtt{String}, f : \mathtt{Feature}_{fm} \mid \\ n = f.\mathtt{name} \wedge f \in \mathtt{mandatory} \Rightarrow \\ (\exists\ s_2 : \mathtt{Feature}_{cf_2} \mid n = s_2.\mathtt{name})$$

The conjunction of these predicates entails the missing part of the desired MF specification, and hints at a possible extension to the standard checking semantics that largely improves its expressiveness, as described in the next section.

Although our example could alternatively be interpreted as a bidirectional transformation between a feature model $\mathtt{FM}$ and a tuple of configurations $\mathtt{CF}^k$, in general the n models may be of different nature. Moreover, the standard checking semantics could not be reproduced under such view.

2.2 Extending the Standard Semantics

As the previous section makes clear, standard QVT-R relations are not suitable for expressing many transformations of interest, namely those where relationships are not symmetric. In fact, this is already a problem in the bidirectional setting (for example, how to express a plain subset relationship?), but is aggravated in the multidirectional setting due to the explosion of possible dependencies between domains. In this paper, we propose precisely to extend QVT-R with a language of dependencies between domains in order to express the desired directionality of the checking semantics.

Let $dom\ R$ denote the set of meta-model identifiers M_1, ..., M_n in a relation $R \subseteq M_1 \times ... \times M_n$. A *checking dependency* $S \rightarrow T$ for R, where $S \subseteq dom\ R$ is a set of identifiers and $T \in dom\ R$ a single identifier (with $T \notin S$), states that the model conforming to T depends on all the models conforming to the meta-models in S. Formally, the semantics of a rule R according to a dependency $S \rightarrow T$, denoted by $R_{S \rightarrow T}$, prescribes that R should be checked by quantifying universally over all the domains in S and, when the respective domain patterns and pre-condition hold, demanding an element satisfying the respective domain pattern and post-condition to exist in the T domain. The set of checking dependencies attached to a relation R will be denoted by $\underline{R}$. The semantics of a top relation R is now the conjunction of all directional checks $\wedge_{d \in \underline{R}}\ R_d\ (m_1, ..., m_n)$.

For example, to obtain the desired specification of the MF relation, it suffices to attach to the above QVT-R specification the dependencies $\underline{MF} \equiv \{\mathtt{CF}_1\ \mathtt{CF}_2 \rightarrow \mathtt{FM}, \mathtt{FM} \rightarrow \mathtt{CF}_1, \mathtt{FM} \rightarrow \mathtt{CF}_2\}$. This extension is conservative, in the sense that the standard semantics can still be specified by setting:

$$\underline{R} \equiv \bigcup_{i \in \{0..n\}} (dom\ R \setminus M_i \rightarrow M_i)$$

The OF relation, stating that the union of selected features should be included in the set of all available features

$$OF_{\mathtt{FM}}\ (cf_1 : \mathtt{CF}, cf_2 : \mathtt{CF}, fm : \mathtt{FM}) \equiv \\ \mathtt{Feature}_{cf_1}.\mathtt{name} \cup \mathtt{Feature}_{cf_2}.\mathtt{name} \subseteq \mathtt{Feature}_{fm}.\mathtt{name}$$

can now be represented by the QVT-R relation

```
top relation OF { n : String;
      domain cf1 s1 : Feature { name = n }
      domain cf2 s2 : Feature { name = n }
      domain fm  f  : Feature { name = n } }
```

associated with the checking dependencies $\underline{OF} \equiv \{\mathtt{CF}_1 \rightarrow \mathtt{FM}, \mathtt{CF}_2 \rightarrow \mathtt{FM}\}$. Of course, this extension also improves the expressiveness in the bidirectional setting, allowing for example to specify a subset constraint between two domains by just attaching one dependency between them. Note that, at this point, we are just disregarding the dependencies implied by the QVT-R standard. Expressing our dependency would require some sort of extended QVT-R syntax.

Although at first sight this extension may seem too conservative, the fact is that from these simple dependencies more complex ones can be built. In particular, multiple model dependencies can be attained through the entailment $\{M_1 \rightarrow M_2, M_1 \rightarrow M_3\} \vdash \{M_1 \rightarrow M_2\ M_3\}$ (thus resulting in the expected $MF_{\mathtt{CF}_1 \times \mathtt{CF}_2}$) while dependencies over unions of models can be attained through $\{M_1 \rightarrow M_3, M_2 \rightarrow M_3\} \vdash \{M_1 \mid M_2 \rightarrow M_3\}$ (from which $OF_{\mathtt{FM}}$ arises).

2.3 Relation Invocations

As in the bidirectional scenario, multidirectional relation calls must preserve the direction of the caller. However, the QVT-R syntax does not guarantee that every relation in a specification can be run in the same direction, e.g., nothing prevents a relation $R \subseteq \mathtt{CF}^k \times \mathtt{FM}$ running in the $\mathtt{FM}$ direction from calling another relation $S \subseteq \mathtt{CF}^k$, which has no $\mathtt{FM}$ direction. The standard is omissive about these situations. The newly introduced checking dependencies must also be taken into consideration, e.g., should a relation $\underline{R} \equiv \{M_1 \rightarrow M_2\}$ be allowed to call another relation $\underline{S} \equiv \{M_2 \rightarrow M_1\}$? We think the answer should be no, and this situation should be flagged as a typing error at static time.

Notwithstanding, it is worth noting that the dependencies of R and S need not be perfect matches. In fact, a relation $\underline{R} \equiv D$ may be called in the direction R_d by another relation $\underline{S} \equiv \{...d...\}$ if $D \vdash d$, i.e., D entails d. In our restricted language this will allow, for instance, the call $R_{M_1 \rightarrow M_3}$ when $\underline{R} \equiv \{M_1 \rightarrow M_2, M_2 \rightarrow M_3\}$, since $\{M_1 \rightarrow M_2, M_2 \rightarrow M_3\} \vdash M_1 \rightarrow M_3$. Since our dependencies are equivalent to *Horn clauses* (disjunctions with a single positive literal) this "type checking" can be done in linear time.

3. QVT-R ENFORCEMENT SEMANTICS

In [5], we proposed a technique for bidirectional QVT-R model transformation following the least-change principle [7], which was implemented in the bidirectional transformation tool `Echo` [6]. Given a binary consistency relation $R \subseteq M_1 \times M_2$ and a model distance metric $\Delta_{M_1} : M_1 \times M_1 \rightarrow \mathbb{N}$, if m_1 and m_2 are two inconsistent models, the new model m_1' produced by transformation $\overrightarrow{R}_{M_1}$ is a consistent model that is as close as possible to the original one, according to the given metric. This results in a clear and predictable enforcement semantics. Note that although $\overrightarrow{R}_{M_1}$ is a transformation from M_2 to M_1, the original model m_1 is also taken into consideration in order to achieve minimality. The technique embeds the QVT-R checking semantics in `Alloy` specifications [3], then calling its model finder in an iterative process of searching for all consistent models at increasing distance from the original m_1 (or alternatively, using optimizing solvers, such as PMax-Sat, as proposed in a recent extension [**?**]). The concretization of the Δ metric is outside the scope of this paper, and the reader is redirected to the original paper. This transformation technique requires only the definition of a consistency relation and a suitable distance metric for the target domain, thus extending it for the multidirectional scenario requires only the definition of suitable distances for the selected output.

Let us use the sample transformations from Section 1 to explore the transformation space. Those with a single output model can be trivially applied. For instance, assuming a model distance Δ_{FM}, $\overrightarrow{F}_{\mathsf{FM}} : \mathsf{CF}^k \rightarrow \mathsf{FM}$ would produce a feature model fm' consistent with the input configurations $(MF\ (cf_1, ..., cf_k, fm'))$ and closest to the original feature model (minimizing $\Delta_{\mathsf{FM}}\ (fm, fm')$). Similarly for $\overrightarrow{F^i}_{\mathsf{CF}}$. As for the tuple returning transformations, a naive way to achieve the combined distance of the target models is to add up the distance between every model, e.g.,

$$\Delta_{\mathsf{CF}^k}\ ((cf_1, ..., cf_k), (cf'_1, ..., cf'_k)) = \\ \Delta_{\mathsf{CF}}\ (cf_1, cf'_1) + ... + \Delta_{\mathsf{CF}}\ (cf_k, cf'_k)$$

for the transformation $\overrightarrow{F}_{\mathsf{CF}^k} : \mathsf{FM} \rightarrow \mathsf{CF}^k$ that updates all configurations. Of course, this means that all changes in all the models have the same weight, what may not be desirable (e.g., in $\overrightarrow{F^i}_{\mathsf{FM}\times\mathsf{CF}} : \mathsf{CF} \rightarrow \mathsf{FM} \times \mathsf{CF}^{k-1}$ changes to configurations could be prioritized over those to feature models). We leave that customization for future work.

When applying a transformation, the user must be aware that not all update directions are able to restore the consistency of the system. Consider, for instance, that a new mandatory feature is introduced in the feature model. Then $\overrightarrow{F^i}_{\mathsf{CF}}$, which updates a single model, will clearly not be able to restore consistency of the model-driven environment. Instead, the user should apply $\overrightarrow{F}_{\mathsf{CF}^k}$ and update all `CF`s.

4. FUTURE WORK

To the best of our knowledge, there exists no work dedicated to multidirectional transformations in QVT-R (or in any other model transformation language). Therefore, the natural direction for this work is to collect reasonable case studies and to study syntactic means to describe multidirectional transformations in order to validate our approach.

We have shown that the checking semantics proposed in the QVT-R standard is not suitable for specifying even simple examples of multidirectional transformations. We intend to explore the expressive power of our multidirectional semantics for writing more realistic examples of feature model synchronization and co-evolution.

In the present paper, we have purposely left out subjective considerations about the most adequate syntactic extensions to the QVT-R language for expressing our proposed checking dependencies, and have focused primarily on the multidirectional semantics. We are currently considering several syntactic extensions to allow the specification of the checking dependencies in QVT-R.

Our `Echo` [6] model repair tool is deployed as an Eclipse plug-in that implements the bidirectional least-change technique from [5]. We plan to release a multidirectional version that naturally extends the existing one: users write multidirectional relations between models and, when inconsistencies are found, select which models are to be updated, establishing the shape of the consistency-repairing transformation.

Acknowledgments

This work is funded by ERDF - European Regional Development Fund through the COMPETE Programme (operational programme for competitiveness) and by national funds through the FCT - Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) within project FATBIT, reference FCOMP-01-0124-FEDER-020532. The first author is also sponsored by FCT grant SFRH/BD/69585/2010.

5. REFERENCES

[1] J. Bradfield and P. Stevens. Enforcing QVT-R with mu-calculus and games. In *FASE'13*, volume 7793 of *LNCS*, pages 282–296. Springer, 2013.

[2] A. Cunha, N. Macedo, and T. Guimarães. Target oriented relational model finding. In *FASE'14*, LNCS. Springer, 2014. To appear.

[3] K. Czarnecki, J. Foster, Z. Hu, R. Lämmel, A. Schürr, and J. Terwilliger. Bidirectional transformations: A cross-discipline perspective. In *ICMT'09*, volume 5563 of *LNCS*, pages 260–283. Springer, 2009.

[4] D. Jackson. *Software Abstractions: Logic, Language, and Analysis*. MIT Press, revised edition, 2012.

[5] F. Jouault and I. Kurtev. Transforming models with ATL. In *MoDELS'05 Satellite Events*, volume 3844 of *LNCS*, pages 128–138. Springer, 2005.

[6] N. Macedo and A. Cunha. Implementing QVT-R bidirectional model transformations using Alloy. In *FASE'13*, volume 7793 of *LNCS*, pages 297 – 311. Springer, 2013.

[7] N. Macedo, T. Guimarães, and A. Cunha. Model repair and transformation with Echo. In *ASE'13*, pages 694–697. IEEE, 2013.

[8] L. Meertens. Designing constraint maintainers for user interaction. In *Third Workshop on Programmable Structured Documents*. Tokyo University, 2005.

[9] OMG. MOF 2.0 Query/View/Transformation specification (QVT), version 1.1. `http://www.omg.org/spec/QVT/1.1/`, January 2011.

[10] P. Stevens. A simple game-theoretic approach to checkonly QVT relations. *Software and System Modeling*, 12(1):175–199, 2013.