Towards Co-evolution in Model-Driven Development Via Bidirectional Higher-Order Transformation

Bernhard Hoisl[1,2], Zhenjiang Hu[3] and Soichiro Hidaka[3]
[1]*Institute for Information Systems and New Media, WU Vienna, Vienna, Austria*
[2]*Secure Business Austria Research (SBA Research), Vienna, Austria*
[3] *National Institute of Informatics, Tokyo, Japan*

Keywords: Model-Driven Development, Model Co-evolution, Bidirectional Transformation, Higher-Order Transformation.

Abstract: In model-Driven development (MDD), metamodels, models, and model transformations are interdependent. A change in one artifact must be reflected in all other related artifacts. Regardless of their dependencies, (meta)models and transformations can evolve autonomously rendering referenced artifacts invalid. Coupling the evolution of models to their corresponding metamodels tries to prevent such mismatches, but is currently limited to one-way adaptations and does not take model transformations into account. To eliminate these shortcomings, we combine first-class transformation models with bidirectional transformations (BX). Our generic approach integrates BX into well-established Eclipse-based MDD tools, thereby neither being restricted to a specific modeling nor model transformation language.

1 INTRODUCTION

In model-driven development (MDD), numerous models and transformations on different abstraction levels need to be taken into account. The high number of models involved originate from a layered modeling architecture (i.e. metamodels, MMs) as well as from refinements (i.e. transformations) from generic to implementation-centric model representations (Sendall and Kozaczynski, 2003). On the one hand, the need for model transformations is inherent to the abstraction mechanism in MDD to represent platform-specific concepts (e.g., statements in a programming language) as platform-independent models (Schmidt, 2006). On the other hand, model transformation necessities stem also from, for instance, changes in MMs (model-to-model transformations, M2M) or the support for multiple platforms (model-to-text transformations, M2T).

MDD-based artifacts are frequently subject to change and evolve over time (Di Ruscio et al., 2012). In most cases, the evolution of (meta)models and model transformations is a manual process (Meyers and Vangheluwe, 2011). Individually maintain and manually evolve MDD specifications is a tedious and error-prone task (Stevens, 2010; Di Ruscio et al., 2011). For instance, consider an evolution of a MM and accompanying constraints. First, all instance models need to be migrated in order to conform to the new MM definition. Furthermore, all model transformations need to be adapted (e.g., due to model type changes). Moreover, tests need to be rewritten to check that the generated source code fulfills the specified constraints.

The artifacts which make up a MDD process (models, M2M/M2T transformations, model and transformation constraints etc.)—although highly interdependent—are autonomously maintained. Changes in one artifact (e.g., in a model) are not automatically reflected in other dependent artifacts (e.g, in a M2T transformation). The barrier for a tight integration of MDD-based artifacts stems from two limitations of current approaches: 1) Model transformations are unidirectional and changes can be propagated in one direction only (e.g., a model change is reflected in generated code); 2) changes can only be propagated into output artifacts of transformations (e.g., models), not into transformation definitions themselves.

In order to overcome these co-evolution problems, our approach, on the one hand, is based on 1) establishing bidirectional transformations (BX) between modeling artifacts (Stevens, 2010). *BX* is a mechanism for maintaining the consistency of two (or more)

Hoisl B., Hu Z. and Hidaka S. (2014).
Towards Co-evolution in Model-Driven Development Via Bidirectional Higher-Order Transformation.
In *Proceedings of the 2nd International Conference on Model-Driven Engineering and Software Development*, pages 466-471
DOI: 10.5220/0004809004660471

related sources of information. A BX between two sources of information `A` and `B` (e.g., two different models) comprises a pair of unidirectional transformations: one from `A` to `B` (*forward transformation*) and another from `B` to `A` (*backward transformation*) (Czarnecki et al., 2009).

On the other hand, we apply 2) higher-order transformations (HOTs) on first-class model representations of transformation specifications (Tisi et al., 2009). A *HOT* "is a model transformation such that its input and/or output models are themselves transformation models. [...] This demands the representation of the transformation as a model conforming to a transformation MM" (Tisi et al., 2009).

In this way, we are able to propagate changes in two directions (1): From a source model to a target model and vice versa. These changes can be propagated into models on the same or on different abstraction levels. Furthermore, we ensure not only the co-evolution of models, but (2) model transformations, as well. We represent transformation definitions as models and are able to propagate changes into horizontal and vertical model transformations (i.e. transformations between models on the same and on different abstraction levels).

Our contributions are as follows:

- *A Method for MDD-based Co-evolution:* Our approach of bidirectional higher-order transformation (B-HOT) for the co-evolution of model artifacts builds on former work (Hidaka et al., 2009; Hidaka et al., 2011; Sasano et al., 2011; Hoisl et al., 2013). This paper presents first enhancement steps which will allow for coupling, synchronization, and tracing of all model artifacts involved in a MDD process.
- *Integrated Tool Support:* We provide initial prototypes for an integrated MDD-based tool support for B-HOTs via the Eclipse IDE. Our implementations build on well-established MDD tools (e.g., Eclipse EMF, ATL, Epsilon)[1].
- *Conformance between BX and MDD:* To facilitate reproduction and transferability, we present an approach independent of any transformation language and we prepare for generalizations according to OMG specifications, for example, MOF, QVT, MOFM2T. Besides integrated BX and MDD tooling, we also want to contribute to establish a common terminology to bridge the gap between the BX and MDD communities (Czarnecki et al., 2009; Hu et al., 2011).

[1]All software artifacts are publicly available at http://www.biglab.org and http://nm.wu.ac.at/modsec.

2 CURRENT APPROACHES

A traditional model-driven architecture (MDA), as proposed by the OMG (Bézivin and Gerbé, 2001) and as supported by a variety of tools, is sketched in Figure 1. MMs provide the reference frame to which instance models must conform to, for example, a UML class model conforms to its MM defined in the UML specification. M2M transformations are applied over one or more input models with the purpose of generating one or more output models conforming to the same or different MMs. A typical M2M transformation example is the generation of platform-specific models (PSMs) from platform-independent models (PIMs). As models are a means for abstraction, they mostly do not capture enough implementation details to be directly executable. Hence, M2T transformations generate textual artifacts (e.g., source code, configuration documents) which can be deployed on a specific platform.

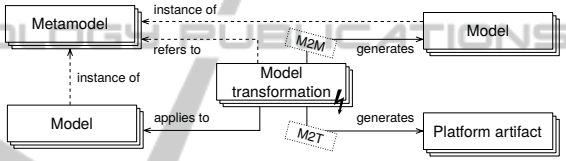

Figure 1: Traditional model-driven architecture.

When a MDD-based artifact evolves, changes must be reflected in all dependent (meta)models, transformations, and platform artifacts. The complexity of change operations increases with the number of different modeling languages (MMs, models), intermediary model representations (M2M transformations), and supported platforms (M2T transformations) involved. Current approaches cannot sufficiently cope with the co-evolution of multiple MDD-based artifacts because of restrictions to express and propagate changes which manifest in 1) unidirectional model transformations and 2) disregarding transformation definitions as first-class models (see Figure 1). An example for evolution mismatches is the inability to reflect changes in generated platform artifacts back to their corresponding instance models. For example, in Eclipse EMF, changes to the generated Java source code may be lost when executing the unidirectional M2T transformation once again. Furthermore, by default, the generation templates (i.e. JET) cannot be adapted, excluding the possibility to reflect changes in the code generator logic (i.e. transformation definitions are not treated as first-class artifacts).

In (Yu et al., 2012), a platform-specific (i.e. Java-bound) solution for the co-evolution problem stated above is provided. In the approach, BX is used to

synchronize models with generated and user-modified code. Prerequisites are that the platform-specific language encodes a textual duplicate of the PIM (i.e. `@model` annotations) and that a MM representation exists for the platform-specific language (i.e. a Java Ecore MM).

To establish BX, triple graph grammars (Giese and Wagner, 2009) are commonly employed in MDD for keeping related models consistent. Triple graph transformations relate a source and a target graph (i.e. a model) by some correspondence graph. In this way, source and target graphs are coupled which provides a basic structure for their co-evolution.

In (Wachsmuth, 2007), MM/model co-evolution is considered as a step-wise adaptation of MMs (via transformation relations) and instance-preservation of models. Instead of describing the co-evolution of models as a transformation between two MMs, (Wimmer et al., 2010) employs in-place transformations. In (Herrmannsdoerfer et al., 2009), a framework is presented to model the co-evolution of MMs/models via the composition of coupled transactions to adapt the MM and specify the corresponding model migrations.

Furthermore, state-based MM/model co-evolution approaches, for instance, adopt HOTs which take a difference model obtained by comparing two MMs and generate a model transformation able to produce the co-evolution of involved models (Di Ruscio et al., 2011).

Although all of these co-evolution methods cope with model transformation restrictions, a combined and uniform solution is missing, so far. Either the approaches provide only one-way co-evolution possibilities (i.e. unidirectional) or only for a subset of MDD artifacts (e.g., only MM/model co-evolution). Therefore, in the next section, we propose a generic approach for co-evolving MDD-related artifacts.

3 MODEL CO-EVOLUTION VIA B-HOT

With our approach (Figure 2 provides an overview) we want to overcome shortcomings of current methods and offer a generic solution for co-evolving MDD artifacts. The upper part of Figure 2 reflects a traditional MDA. Model co-evolution is achieved by integrating 1) BX capabilities (lower part of Figure 2) and 2) support for HOT into the MDA.

As an example, consider a model transformation from an object-oriented representation (e.g., a class diagram) to a relational database model. For instance, both MMs are defined in a MOF-based language (see upper part of Figure 2). Hence, their instance models (e.g., using Ecore as technological projection) conform to the EMOF MM. A transformation (e.g., specified via ATL or ETL) is applied to class models and generates database models. This forward transformation proves useful in one case only: Changes in evolving class models should be reflected in database models, as well. Updates in database models cannot be propagated back to their source class models. A coupling of both representations limits the target model to be read-only (otherwise changes get lost when re-executing the transformation).

In our approach, we integrate BX capabilities by reusing native MDD concepts. *Every* transformation is represented as a first-class model conforming to a transformation MM. B-HOTs (see Figure 2) provide for the mapping of unidirectional MDD-based transformation models (e.g. defined via ATL or ETL) into a bidirectional graph transformation model (and vice versa). Reconsidering the BX of the class-to-database model transformation example, in a next step, both source and target models are mapped to a graph structure (defined via UnCAL, a graph algebra). Source and target graph schemas are represented as MOF-based MMs. The BX provides both, a forward transformation from class to database graphs as well as a corresponding backward transformation (both defined via UnQL$^+$, a SQL-like graph query/transformation language). Thus, changes in the database graph can be propagated back to the class graph. As the transformation of models to graphs is also bidirectional, updated class and database graphs can be represented in their initial model-based forms. Therefore, a BX of source and target models (class and database models in our example) is established. The backward database-to-class transformation is distinct to the BX and no corresponding MDD-based transformation equivalent exists. Therefore, as a last step, the backward transformation (in UnQL$^+$) can be represented in its original MDD-based form (in ATL, ETL) via a B-HOT mapping (see Figure 2).

We discuss co-evolution properties of our approach according to the following four categories.

Model Relations: Our approach establishes BX-based relations between models, graphs, and model and graph representations. The mapping relation between traditional MDD-based transformations and BX representations (B-HOT) allows to add BX support in traditional MDAs. Furthermore, relations are not restricted to one source and one target artifact only, but can be used for the transformation of multiple dependent models/graphs, as well (see also compositional BX down below). The coupling of models via BX allows, on the one hand, to establish synchronization definitions and, on the other hand, to collect

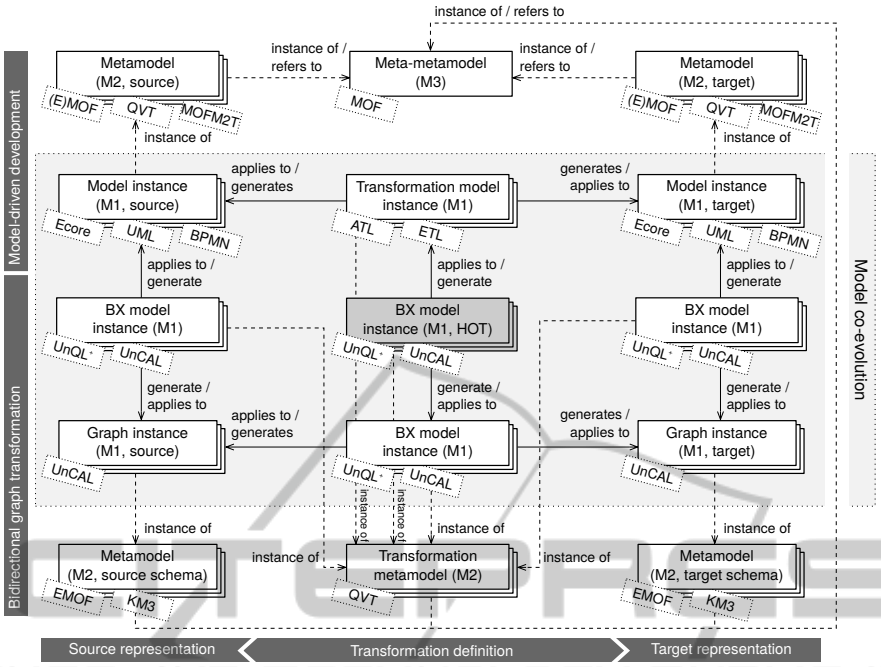

Figure 2: Overview of our model co-evolution approach.

transformation traces. As many modeling artifacts make up a MDD process, keeping models consistent is of special importance. Moreover, trace information are a relevant source for documentation and debugging purposes.

Model Co-evolution Scenarios: Our approach supports any M2M relation and any number of MM-layers. Horizontal co-evolution examples are, for instance, the synchronization of different MMs or different instance models. Vertical co-evolution examples are, for instance, keeping PIMs and PSMs or instance models and corresponding MMs consistent. Transformation models permit to propagate changes also in horizontal and vertical M2M and M2T transformation definitions, for instance, for the co-evolution of MMs and transformation models or different transformation models.

Language-independent Integration: Our approach is not dependent on a specific model transformation language, i.e. it does not matter if the model transformation is defined via ATL, ETL, or any other language. This is because we do not integrate bidirectionality into a model transformation language directly. The B-HOT definition serves as a language-specific binding between the concepts of the unidirectional MDD-based transformation and the bidirectional graph transformation. These bindings must be specified only once for each MDD-based transformation language (e.g., ATL, ETL) and facilitate reuse of our approach.

BX Properties: We develop B-HOTs via a functional bidirectional graph transformation language (Hidaka et al., 2011). The BX ensures the *well-behavedness* of forward and backward transformations (i.e. that they are consistent with each other) and satisfy the round-trip property (Czarnecki et al., 2009). As the BX does not restrict forward transformations to be information preserving, a backward transformation requires not only the modified target graph/model, but also the original source graph/-model. Large BX can be developed in a *compositional* way of reusing existing information (e.g., via intermediate models). Compositional BX can be employed, on the one hand, for a pair of consecutive transformations, where the output of transformation `A` is the input of transformation `B`; for example, the output of the source model-to-graph transformation is fed into the forward source-to-target graph transformation (see Figure 2). On the other hand, compositional BX can be used for a pair of transformations that share an identical input model, for instance, transformations from one PIM to multiple PSMs (Hidaka et al., 2009).

4 B-HOT REQUIREMENTS

This paper provides a first step to make co-evolution in MDD via B-HOTs possible. Initial work regarding the methodology and accompanying tool support has

been performed, but is far from being finalized. In this section, we list challenging requirements for the implementation of our approach. We present completed work and discuss prerequisites for future developments.

Transformation MMs: Our B-HOT approach relies on transformation (meta)models. MDD-based M2M transformation MMs exist, for instance, for ATL (Tisi et al., 2009) and for a subset of the Epsilon-language family (Wei, 2012). Regarding M2T transformations, (Hoisl et al., 2013) extended the Epsilon model representations of (Wei, 2012). The BX framework (Hidaka et al., 2011) does not need MM representations for the UnQL$^+$ and UnCAL languages. Syntax definitions in BNF exist for both languages and need to be mapped to EMOF-compliant (i.e. Ecore-based) MM representations (ongoing work; see also Section 5).

MM-specific Bindings: For a B-HOT, language-specific bindings need to be established between uni- and bidirectional transformation MMs. Initial work provides an unidirectional transformation from ATL to UnQL$^+$, whereby the transformation does not take the model representation of ATL into account (Sasano et al., 2011)[2]. Thus, language-specific bindings for, for instance, ATL and/or ETL to UnQL$^+$ and/or UnCAL via B-HOT is future work. Furthermore, a first prototype exists for the BX of model-to-graph representations (Ecore-based models to UnCAL and vice versa), but needs improvements (future work).

Round-tripping of transformation definitions: Our B-HOT approach demands transformation models as input. In contrast, most model transformation engines cannot execute model representations of transformation definitions. Therefore, the round-tripping of executable (i.e. text-based) transformation specifications and their model representations must be provided. For M2M transformations, "an ATL transformation is itself a model, conforming to the ATL MM" (Tisi et al., 2009). Furthermore, (Wei, 2012) developed initial round-tripping support for an Epsilon subset which was extended for M2T transformations (i.e. EGL) by (Hoisl et al., 2013). Currently, the automatically derived backward transformation of a BX can neither be expressed as UnQL$^+$ or UnCAL textual statements nor via corresponding model representations (future work).

Generic Mappings: Prototype developments define transformations in a specific language as implementation vehicle (e.g, ATL, ETL). To support uptake and transferability of our approach we need to establish mappings to OMG specifications (see also Figure 2). (Hoisl et al., 2013) provides mappings between EGL-based M2T transformation concepts used for the prototype and the MOFM2T specification. As future work, uni-/bidirectional M2M transformation concepts (e.g., ATL, ETL, UnQL$^+$) will be mapped to the QVT specification.

Development Support: Initial support for the requirements-driven testing of (meta)models and model transformations via scenarios is provided by (Sobernig et al., 2013). Furthermore, validation for source and target models as well as for BX is presented in (Hidaka et al., 2011). As future work we will implement an IDE (e.g., a text editor) to support the development of UnQL$^+$ BX and UnCAL-based graphs. For this task, Eclipse Xtext is a candidate framework as it combines the grammar specification for a textual syntax with an Ecore-based model representation and provides for an Eclipse-based IDE.

Combine BX and MDD: Model (i.e. graph) transformations are important to both BX and MDD (Czarnecki et al., 2009; Hu et al., 2011). Our approach allows to integrate BX into MDD, thereby reusing native methods and tools for both. We want to support the creation of a shared terminology (Czarnecki et al., 2009) via the generalization and mapping of language-specific uni-/bidirectional transformation concepts to OMG specifications. After our developments have matured, as future work, we need to provide for a larger case study to show that our approach works in practice.

5 CONCLUDING REMARKS

In this paper, we presented an approach for model co-evolution by combining BX and HOT for MDD. The developed method (B-HOT) intends to overcome current limitations for model co-evolution as transformations are represented as models and model transformations are bidirectionalized. In our approach, models are coupled via BX providing the benefit that synchronization of models is ensured via forward/backward transformations. Another advantage is that changes can be propagated into model transformations keeping them consistent with their evolving dependent artifacts (MMs, model instances). Our approach of integrating BX into MDD is generic and can be applied to any model transformation language via binding specifications.

A drawback of our proposal is that the efforts of

[2]This separate work of integrating ATL and BX is performed in collaboration with the AtlanMod team, uses the same BX framework (*GRoundTram*), but in contrast focuses on unidirectional transformations from ATL to UnQL$^+$ with a concrete semantic alignment between these two technical spaces.

creating initial modeling and transformation artifacts can be high. Transformation MMs may have to be defined for the intended target language. Currently, no bindings for transformation languages exist. Although these have to be defined only once for each language, this is a barrier for uptake. Transformation engines might not execute models directly making text/model round-tripping functions necessary (but again these can be reused per language). Adequate tool support must be provided to facilitate the development of models and transformations.

Currently, we are developing an EMOF-based MM for the UnQL$^+$ BX language (in Ecore). In parallel, we transfer the BNF-based grammar definition to Xtext. This will ensure the consistent mapping of transformations written in UnQL$^+$ to their modeling equivalents. An editor to support the definition of UnQL$^+$ statements will be provided, as well. Initial developments are available at the URLs mentioned in the footnote of Section 1 and are continuously updated. UnQL$^+$ concepts will be mapped to the QVT relations language in the near future.

ACKNOWLEDGEMENTS

This work has partly been funded by the Austrian Research Promotion Agency (FFG) of the Austrian Federal Ministry for Transport, Innovation and Technology (BMVIT) through the Competence Centers for Excellent Technologies (COMET K1) initiative and the FIT-IT program.

REFERENCES

Bézivin, J. and Gerbé, O. (2001). Towards a precise definition of the OMG/MDA framework. In *Proc. 16th Int. Conf. Automated Softw. Eng.*, pages 273–280. IEEE.

Czarnecki, K., Foster, J. N., Hu, Z., Lämmel, R., Schürr, A., and Terwilliger, J. F. (2009). Bidirectional transformations: A cross-discipline perspective. In *Theory and Practice of Model Transformations*, volume 5563 of *LNCS*, pages 260–283. Springer.

Di Ruscio, D., Iovino, L., and Pierantonio, A. (2011). What is needed for managing co-evolution in MDE? In *Proc. 2nd Int. Workshop Model Comparison in Practice*, pages 30–38. ACM.

Di Ruscio, D., Iovino, L., and Pierantonio, A. (2012). Coupled evolution in model-driven engineering. *IEEE Softw.*, 29(6):78–84.

Giese, H. and Wagner, R. (2009). From model transformation to incremental bidirectional model synchronization. *SoSyM*, 8(1):21–43.

Herrmannsdoerfer, M., Benz, S., and Juergens, E. (2009). COPE – automating coupled evolution of metamodels and models. In *Proc. 23rd European Conf. Object-Oriented Programming*, pages 52–76. Springer.

Hidaka, S., Hu, Z., Inaba, K., Kato, H., and Nakano, K. (2011). GRoundTram: An integrated framework for developing well-behaved bidirectional model transformations. In *Proc. 26th Int. Conf. on Automated Softw. Eng.*, pages 480–483. IEEE.

Hidaka, S., Hu, Z., Kato, H., and Nakano, K. (2009). Towards a compositional approach to model transformation for software development. In *Proc. 24th Symposium on Applied Computing*, pages 468–475. ACM.

Hoisl, B., Sobernig, S., and Strembeck, M. (2013). Higher-order rewriting of model-to-text templates for integrating domain-specific modeling languages. In *Proc. 1st Int. Conf. Model-Driven Eng. and Softw. Development*, pages 49–61. SciTePress.

Hu, Z., Schurr, A., Stevens, P., and Terwilliger, J. F. (2011). Dagstuhl seminar on bidirectional transformations (BX). *SIGMOD Rec.*, 40(1):35–39.

Meyers, B. and Vangheluwe, H. (2011). A framework for evolution of modelling languages. *Sci. Comput. Program.*, 76(12):1223–1246.

Sasano, I., Hu, Z., Hidaka, S., Inaba, K., Kato, H., and Nakano, K. (2011). Toward bidirectionalization of ATL with GRoundTram. In *Proc. 4th Int. Conf. Theory and Practice of Model Transformations*, pages 138–151. Springer.

Schmidt, D. C. (2006). Guest editor's introduction: Model-driven engineering. *Computer*, 39(2):25–31.

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

Sobernig, S., Hoisl, B., and Strembeck, M. (2013). Requirements-driven testing of domain-specific core language models using scenarios. In *Proc. 13th Int. Conf. Quality Softw.*, pages 163–172. IEEE.

Stevens, P. (2010). Bidirectional model transformations in QVT: Semantic issues and open questions. *SoSyM*, 9(1):7–20.

Tisi, M., Jouault, F., Fraternali, P., Ceri, S., and Bézivin, J. (2009). On the use of higher-order model transformations. In *Model Driven Architecture – Foundations and Applications*, volume 5562 of *LNCS*, pages 18–33. Springer.

Wachsmuth, G. (2007). Metamodel adaptation and model co-adaptation. In *Proc. 21st European Conf. Object-Oriented Programming*, pages 600–624. Springer.

Wei, W. (2012). EpsilonLabs: Epsilon static analysis. Available at: `http://code.google.com/p/epsilonlabs/wiki/EpsilonStaticAnalysis` .

Wimmer, M., Kusel, A., Schönböck, J., Retschitzegger, W., Schwinger, W., and Kappel, G. (2010). On using inplace transformations for model co-evolution. In *Proc. 2nd Int. Workshop Model Transformation with ATL*. INRIA & Ecole des Mines de Nantes.

Yu, Y., Lin, Y., Hu, Z., Hidaka, S., Kato, H., and Montrieux, L. (2012). Maintaining invariant traceability through bidirectional transformations. In *Proc. 34th Int. Conf. Softw. Eng.*, pages 540–550. IEEE.