On the Evolution of Modeling Ecosystems: An Evaluation of Co-Evolution Approaches

Juergen Etzlstorfer, Elisabeth Kapsammer and Wieland Schwinger
Johannes Kepler University, Linz, Austria
firstname.lastname@cis.jku.at

Keywords: Model-Driven Engineering, Evolution, Co-Evolution, Modeling Ecosystems.

Abstract: In Model-Driven Engineering, several artifacts together form a so-called modeling ecosystem, comprising metamodels defining prevailing concepts of a domain and depending artifacts using these concepts. However, evolutionary pressure causes the need for changes in the metamodel, necessitating all artifacts in the modeling ecosystem to migrate to again conform to the evolved version of the metamodel, i.e., they have to co-evolve accordingly. Several approaches for the co-evolution of artifacts have been proposed, however, they differ substantially from each other and, thus, an in-depth investigation of these approaches is needed to allow for a systematic comparison. Therefore, the contribution of this paper is a dedicated evaluation framework for co-evolution approaches focusing on aspects relevant in the context of modeling ecosystems, and its application to a representative set of recent approaches. Based on this evaluation lessons learned as well as future research lines are presented.

1 INTRODUCTION

In Model-Driven Engineering (MDE) (Bézivin, 2005), the usage of models is promoted to perform different phases of the software development life-cycle on a higher-level of abstraction. Thereby, metamodels are the central artifacts defining domain concepts, their relationships as well as constraints among each other, constituting the basis other modeling artifacts rely on. The most prominent artifacts depending on the metamodel are models instantiating concepts from the metamodel, and transformations using the concepts described in the metamodel to generate or manipulate models (Brambilla et al., 2012; Sendall and Kozaczynski, 2003). Besides these prominent representatives, other depending artifacts like tools and concrete syntaxes might also be present. The metamodel together with its depending artifacts forms a so-called *modeling ecosystem* (Di Ruscio et al., 2012), in which different artifacts have *different kinds of relationships* and *dependencies* to the metamodel and among each others. For example, a model is *instantiating* concepts defined in the metamodel, while transformations *refer* to the concepts in their transformation definitions. In this context, it is of utmost importance that all artifacts are in a valid state, i.e., they have to satisfy the given relationships, both with respect to the metamodel, as well as to each other, in order to preserve the operability of the whole modeling ecosystem.

As a matter of course, ecosystems are not static. Due to evolution, e.g., caused by changing requirements, *metamodels* might change, necessitating all depending artifacts in the ecosystem to be *migrated* to be again valid with respect to the evolved version of the metamodel, i.e., they have to *co-evolve*. Since the distinct kinds of artifacts existing in an ecosystem may hold different kinds of relationships, each kind of artifact has to be treated specifically. Thus, migration is not limited to one kind of artifact, only, but in fact has to be performed for all kinds of depending artifacts, entailing the risk of introducing divergence between the various migrations leading to *inconsistencies* (Kusel et al., 2015). Discovering and eliminating already introduced inconsistencies is a tedious task, since they might *spread* across all artifacts in the ecosystem.

Although several approaches for the co-evolution of modeling artifacts exist, e.g., (Rose et al., 2010b; Herrmannsdoerfer et al., 2009) for models, and (Levendovszky et al., 2010; Méndez et al., 2010) for transformations, they differ substantially from each other regarding their capabilities with respect to modeling ecosystems, since in a more ecosystem-wide perspective the artifacts in the ecosystem are multiple. Furthermore, there are several aspects in co-evolution

Etzlstorfer, J., Kapsammer, E. and Schwinger, W.
On the Evolution of Modeling Ecosystems: An Evaluation of Co-Evolution Approaches.
DOI: 10.5220/0006167900900099
In *Proceedings of the 5th International Conference on Model-Driven Engineering and Software Development (MODELSWARD 2017)*, pages 90-99
ISBN: 978-989-758-210-3

that are essential in the context of an ecosystem-wide perspective, in contrast to an isolated view on one kind of artifact, only. Thus, an in-depth investigation of existing co-evolution approaches is needed to allow for a systematic comparison.

Therefore, in this paper our contribution is threefold: We (i) elaborate the relationships between artifacts in modeling ecosystems, which further build the basis for (ii) an evaluation framework for the comparison of co-evolution approaches with special respect to modeling ecosystems. Furthermore, we (iii) apply the evaluation framework to state-of-the-art co-evolution approaches and draw lessons learned from the evaluation which outline current drawbacks and possible future research lines. With respect to evolving and depending artifacts, this paper focuses on metamodels as evolving artifacts and models and transformations as depending artifact, respectively. This is conform with related research and existing approaches. Nevertheless, the findings presented in this paper as well as the general parts of the evaluation framework are applicable to arbitrary other modeling artifacts, too.

The paper is organized as follows: The next section explores modeling ecosystems and the relationships between artifacts in these ecosystems, while in Section 3 the evaluation framework is presented and applied on current approaches. In Section 4 we discuss lessons learned, while Section 5 presents related work. Finally, Section 6 concludes the paper.

2 EVOLVING MODELING ECOSYSTEM

In practicing MDE, a model does not come alone, but in fact it is interwoven in a diversity of different artifacts that may depend on each other, thereby building a modeling ecosystem (Di Ruscio et al., 2012). More precisely and with respect to evolution of modeling ecosystems, we differentiate the artifacts in metamodels and their depending artifacts, being again specialized into different kinds of artifacts, like models, transformations, and tools, as shown in Figure 1. Thus, the metamodel can be seen as a central artifact, defining the prevailing concepts of a domain, which are utilized in other artifacts, that therefore *depend on* the metamodel. This general dependency may be refined for each kind of artifact living in the modeling ecosystem. For example, in case of models that *conform to* their respective metamodels this conformance relationship is a refinement of the more general depends on relationship. Regarding transformations, they *conform to* their transformation metamodel defining the syntactic constraints of transformation definitions, but additionally, have dependencies on the *source domain* and *target domain* employed in the transformation definition. Furthermore, other depending artifacts like concrete syntax specifications and tools assisting the modeler in diverse tasks, might live in the ecosystem having their specific dependencies on the metamodel.

However, artifacts in the modeling ecosystem are not static entities but in fact are subject to *evolution*, e.g., due to changing requirements. Each artifact might undergo changes (denoted as curved yellow arrows in Figure 1) which might impact depending artifacts (denoted as angled yellow arrows). Please note that the color-intensity indicates the concreteness of the impact, i.e., a more transparent arrow indicates a more abstract impact. As one might see, an evolution of the *ModelMM* impacts all conforming models as well as all transformations that are either source- or target-domain-conform to it. In addition, all other artifacts depending on the *ModelMM* might be impacted as well. Analogously, changes on the *TrafoMM* impact the conformance of all transformations depending on this metamodel. In this context, we do not explicitly differ between graph transformations, bi-directional transformations, or plain text-based transformations, since the conceptual work presented in this paper is applicable to both of them. In general, the kind of dependency determines the impact of a metamodel change on the depending artifact, thus, the impacts are specific for each kind of dependent artifact.

In order to be operable, the modeling ecosystem has to be in a consistent state, i.e., all artifacts have to hold valid relationships to the metamodel and across each other. However, due to their dependencies the artifacts may influence each other, e.g., in case of evolution when changes are applied on one artifact in the ecosystem and might render the ecosystem inconsistent, necessitating a corresponding migration of the affected artifacts. The inter-dependencies between the artifacts demand for an ecosystem-wide perspective and, thus, an ecosystem-wide migration across all artifacts to re-establish the consistency and preserve the operability of the modeling ecosystem is needed.

In the following, the relationships in a modeling ecosystem are discussed in more detail, to highlight the impacts of evolution.

2.1 Relationships in a Modeling Ecosystem

In order to understand the inter-dependencies between the different kinds of artifacts, in the following, the relationships in a modeling ecosystem are identified

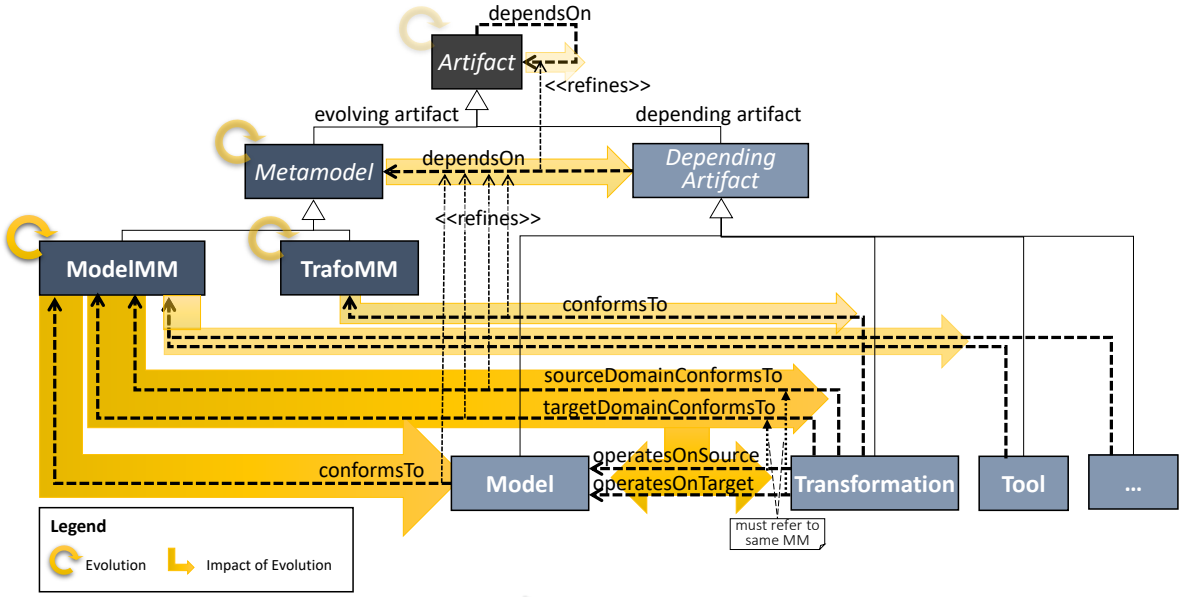

Figure 1: Relationships in a Modeling Ecosystem.

and discussed, based on studied literature (e.g., (Rose et al., 2010a; Di Ruscio et al., 2011; Bézivin, 2005; Méndez et al., 2010)) as well as our own experiences and findings. Therefore, the most generic relationship is presented first, while refinements for models and transformations are subsequently discussed.

- **dependsOn:** All artifacts in the modeling ecosystem *depend on* a metamodel in general, whereas the kind of dependency has to be specialized for each kind of artifact, since the *dependsOn* relationship does not impose any constraints on the validity of this relationship between the metamodel and other modeling artifacts. This kind of relationship might not have a formalized representation, but can relate arbitrary artifacts to the metamodel in general, as for example GMF models[1] (Di Ruscio et al., 2011). Consequently, the *dependsOn* relationship is the most generic dependency relationship between artifacts in the ecosystem and is *refined* by other dependency relationships.
- **conformsTo:** The most prominent refinement of the *dependsOn* relationship in a modeling ecosystem holds between a model and its according metamodel. More strictly, a model *conforms to* a metamodel if only concepts that are defined in the metamodel are used, according to the rules and constraints specified in the metamodel (Schönböck et al., 2014), e.g., multiplicity constraints. Furthermore, metamodels themselves have to *conform to* their meta-metamodels, e.g., MOF (Object Management Group, 2011) or its open-source implementation Ecore as basis of the Eclipse Modeling Framework[2]. Moreover and as already highlighted in (Bézivin, 2005), model transformations themselves can be seen as models that *conform to* their respective transformation metamodels, e.g., the ATL metamodel (Jouault et al., 2008). Thus, an evolution of the metamodel has impact on the *conformsTo* relationship of all depending artifacts, which in response have to co-evolve to re-establish a broken *conformsTo* relationship.
- **sourceDomainConformsTo** and **targetDomainConformsTo:** Besides models, transformations play a vital role in MDE, since they are comparable in role and importance to compilers in high-level programming languages (Kappel et al., 2012). Model transformations take input models conforming to a source metamodel and generate output models conforming to a target metamodel (cf. Figure 2). The transformation itself has two different kinds of relationships to the source and target metamodels, respectively. First, the *sourceDomainConformsTo* relationship states that in the source domain of the transformation definition only concepts from the source metamodel are permitted. In contrast, the *targetDomainConformsTo* relationships only holds, if in the target domain of the transformation definition concepts from the target metamodel are used, only. In this context,

[1] https://eclipse.org/modeling/gmp/

[2] https://eclipse.org/modeling/emf/

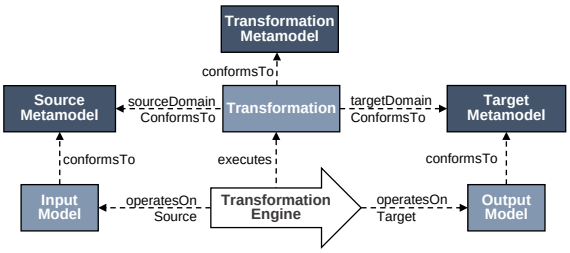

Figure 2: Conceptual View on Model Transformations based on (Czarnecki and Helsen, 2006).

the domain conformances can be seen as specifications the input/output has to conform to.

Although in (Méndez et al., 2010) the term *domainConformsTo* for the relation between the source metamodel and the source domain of the transformation has been introduced (and analogously the term *coDomainConformsTo* for the target representatives), we prefer the terms proposed in this paper – *sourceDomainConformsTo* and *targetDomainConformsTo* – being more intuitive and precise, since they clearly explicate the respective source or target roles of the involved metamodels.

Regarding an evolution of a metamodel involved in a transformation specification, i.e., the source or target domain, its impact is determined by both the role of the metamodel, i.e., the source or target metamodel, and the models it operates on, since impacts might not only reveal at specification time, but also at run-time.

- **operatesOn:** When having model transformation as part of the modeling ecosystem, they might be executed on models, thus, they *operate on* them. An important aspect for this relation is that it only holds if the metamodel of the source model is the same metamodel as for the sourceDomainConformsTo relationship of the transformation, i.e., the transformation operates on a model which is conform to the source domain of the transformation. An evolution of a metamodel affects the *operatesOn* relationship, since the models as well as the transformations depending on the metamodel are affected, having a transitive effect on the *operatesOn* relationship. Thus, by co-evolving the depending artifacts, the *operatesOn* relationship is re-established as a result thereof. However, it is of utmost importance that the co-evolution of models and transformations is performed consistently, i.e., following the same strategy, to successfully re-establish this relationship.

In summary, one might see that modeling ecosystems not only comprise different kinds of artifacts, but also different kinds of relationships between the artifacts and the metamodel as well as between the artifacts themselves, which are affected by an evolution of the metamodel. Consequently, an ecosystem-wide perspective on evolution and co-evolution is necessary to maintain the operability of a modeling ecosystem. Therefore, in the next section, different aspects crucial to evolving ecosystems are discussed.

3 ASPECTS OF CO-EVOLUTION

After having discussed the prevailing relationships in a modeling ecosystem and the impacts of evolution on them, in this section aspects of co-evolution with respect to modeling ecosystems are discussed and proposed as criteria for a corresponding evaluation framework (cf. Section 3.1), which serves for a subsequent evaluation of existing co-evolution approaches (cf. Section 3.2).

As shown in Figure 3, the process of co-evolution in the context of modeling ecosystems spans over four phases. Starting with a consistent modeling ecosystem in its original version V0, the executed phases are as follows: (i) *change detection*, i.e., the identification of all applied changes on the metamodel, (ii) *impact analysis*, i.e., the determination of impacts of the applied changes on the diverse kinds of artifacts, (iii) *change propagation*, i.e., the actual propagation of changes to ultimately migrate the artifacts, and finally (iv) an optional *validation* of the migrated artifacts. The result of this co-evolution process is a consistent ecosystem in its migrated version V1. While the first phase operates on the changed metamodel, only, the subsequent phases have to be performed on each kind of artifact (cf. KA_1 - KA_n in Figure 3). Please note that in order to allow for a comprehensive co-evolution of the diverse artifacts, the change propagation phase has to be in accordance across all different kinds of artifacts, but is performed on each artifact individually. The last but optional phase of validation has to be performed across all depending artifacts in the ecosystem to ensure both, the conformance of the artifacts to the metamodel and the re-establishment of inter-dependencies between the artifacts, i.e., inter-artifact consistency.

3.1 Evaluation Framework

In the following, aspects of co-evolution in modeling ecosystems are discussed, which will further serve as criteria of an evaluation framework for investigating existing co-evolution approaches (cf. Section 3.2 for the evaluation). The aspects have been assembled top-down by deriving criteria from the modeling ecosys-

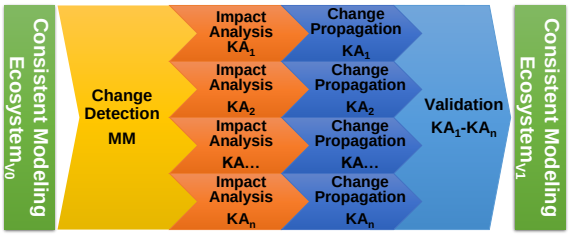

Figure 3: Co-Evolution Process of a Modeling Ecosystem.

tem co-evolution process (cf. Figure 3) as well as in bottom-up manner by considering related criteria discussed in literature (e.g., (Paige et al., 2016; Herrmannsdoerfer and Wachsmuth, 2014)). The evaluation framework can be easily extended for other modeling artifacts by adding corresponding criteria, while in the following it is elaborated in detail for models and transformations.

3.1.1 General Criteria

The first set of criteria is used to determine general characteristics of co-evolution approaches.

- **Evolving Artifact:** Changes affecting the ecosystem can be applied on a metamodel (cf. Figure 1), which might act as the *ModelMM*, or the *TrafoMM*. Regarding the *ModelMM*, it might further act as *source* or *target metamodel* of a transformation. Finally, also the evolution of *other* artifacts may be dealt with.
- **Depending Artifact:** As discussed in Section 2.1 the kind of dependency is specific for each kind of artifact. Depending artifacts mainly comprise *models* and *transformations* but can also include *other* artifacts, which are affected by changes to the evolving artifact.
- **Monitoring of Artifacts in the Ecosystem:** The observation of artifacts in the ecosystem might be advantageous to automatically detect changes in the ecosystem. Thus, this criterion determines if an approach monitors any artifact in the ecosystem to alert the user in case of changes.
- **Transactions:** Transaction support ensures that artifacts are either completely migrated to their new version or set back to their initial state. Transactions may span over different *ranges* and *phases*. Regarding the range, a transaction may span over *one artifact*, *all instances of one kind of artifact*, or *all instances of all kinds of artifacts*. Considering the phases, they may include *change application, change detection, impact analysis*, and *change propagation*.
- **Version Management:** Approaches may support versioning of artifacts, e.g., to determine which artifacts are conforming to which versions of the metamodel(s).

3.1.2 Change Detection

The second set of criteria is related to the change detection phase of the co-evolution process (cf. Figure 3) and is employed to determine how and which changes can be detected.

- **Kind of Detection:** Changes may be detected *state-based*, i.e., matching the original version and the evolved one in order to derive the applied changes, *operation-based*, i.e., recording the actually applied changes, *manual*, i.e., defining changes by hand, or they may be detected by a *hybrid* form of these kinds.
- **Granularity of Change:** Changes may be detected as *atomic* units, i.e., single changes that are not interconnected, or as *composite* units, i.e., semantically connected sequences of changes.
- **Target of Change:** The target of the change may either be the *syntax*, i.e., a syntactical change on the evolving artifact, or *semantics*, i.e., a change in the meaning of a metamodel element, e.g., interpreting the unit of the values of an attribute in centimeter instead of millimeter, which won't have an effect on the syntax, but in fact the values in the instances have to be adapted, e.g., in this case multiplied or divided by 10.

3.1.3 Impact Analysis

Since changes might have impacts on depending artifacts and their relationships to the evolved artifact, this set of criteria is used to determine if and at which detail impacts are identified.

- **Explicated Analysis:** Impacts of changes may be explicitly analyzed on the depending artifacts, revealing potential affects on the artifacts. In this context, the impacts may be detected on the *conforms-to*, *source-domain-conforms-to*, *target-domain-conforms-to* or *other* relationships in the ecosystem.
- **Level:** The impact analysis can be either performed on *type level*, i.e., an analysis of potential impacts on models or transformations of metamodel changes, or on *instance level*, i.e., revealing the impacts on the actual instances that are affected in the ecosystem.
- **Result Usable for Intervention:** Results of the impact analysis should ideally be usable for user

intervention, e.g., a score determining a high extent of impacts may indicate to undo the changes and re-evaluate the applied changes or find a more suitable set of changes.

- **Granularity:** Impacts on the artifacts may be detected on different level of granularity. In case of models, granularity spans from fine-grained, i.e., *feature level*, over *class level* to *package level*, i.e., the more coarse-grained level. In case of transformations, impacts can be detected analogously on the level of *bindings*, *rules*, and *modules*. Furthermore, impacts may be detected on *OCL level* if the transformation language employs OCL for querying model elements.
- **Run-time Effects Detection:** Effects on executable artifacts, e.g., transformations, may only be detectable at run-time. For example, when changing a feature from mandatory to optional, transformation definitions that assume the existence of this feature might break at run-time, if a value for this feature is not set, similarly to null-pointer exceptions in programming languages.

3.1.4 Change Propagation

The fourth set of criteria is used to determine how the identified changes from the change detection phase (cf. Figure 3) are actually propagated to the depending artifacts, and if consistence of the modeling ecosystem might be ensured.

- **Strategies:** For the actual propagation of changes, approaches may offer built-in predefined strategies, that may vary in *number* and might be *customizable* in the sense that they might be, e.g., adapted, parametrized or overwritten to migrate the artifact in a customized way. Additionally, a *user-definable* propagation may be possible by providing new migration strategies.
- **Automation:** Propagation can be either done *semi-automatic*, i.e., with user intervention, or *automatic*, i.e., without user intervention.
- **Consistence:** Approaches may ensure *intra-artifact* consistency, i.e., a specific change is propagated consistently regardless of its usage as atomic or part of an composite change, and *inter-artifact* consistency, i.e., a change is propagated with the same intention across all artifacts in the ecosystem.

3.1.5 Validation

Related to the final and optional phase of validation, this set of criteria is employed to determine to which extent the performed change propagation is validated to ensure a consistent ecosystem.

- **Type of Validation:** In the optional phase of validation, the performed propagation may be validated *syntactically*, e.g., for models the conformance may be validated with EMF, while for transformations the syntactical correctness may be validated with an according compiler. Additionally, the *semantical* correctness may be validated, e.g., with regression testing (Fowler et al., 1999).
- **Range of Validation:** The validation may range over *one instance of an artifact*, *all instances of one kind of artifact* or *all instances of all kinds of artifact*.

After having presented the criteria of the evaluation framework, in the next section current co-evolution approaches will be evaluated.

3.2 Evaluation

In this section, approaches for the co-evolution of artifacts in a modeling ecosystem are evaluated, applying the evaluation framework discussed above. The selection of approaches supporting co-evolution comprises approaches that support the co-evolution of any artifact in the modeling ecosystem, or provide a language or framework for defining and performing the co-evolution of any artifact. Eleven approaches that meet this criteria have been identified in the literature by surveying the proceedings of conferences, workshops as well as journals dealing with model-driven engineering topics, and, thus, participate in the evaluation. Literature that presents the actual application of an already proposed approach has not been included in this selection, since it would lead to duplicate results. The results of the evaluation are presented in the following and summarized in Table 1.

Six approaches of the selected eleven approaches target model co-evolution (Herrmannsdoerfer et al., 2009; Rose et al., 2010b; Gruschko et al., 2007; Cicchetti et al., 2008; Garcés et al., 2009; Wachsmuth, 2007), three perform transformation co-evolution (Garcés et al., 2013; García et al., 2013; Kruse, 2011), and two approaches consider a more ecosystem-wide perspective by supporting models and transformations (Di Ruscio et al., 2012; Kusel et al., 2015), whereas one of these approach serves as a framework for co-evolution of a more wide-range set of modeling artifacts, e.g., GMF models (Di Ruscio et al., 2012). Three of the five transformation co-evolution approaches support the evolution for both the source and target metamodels (Garcés et al., 2013; García

Table 1: Evaluation of Co-Evolution Approaches.

				Hermannsdoerfer et al., 2009	Rose et al., 2010a	Gruschko, 2007	Cicchetti et al., 2008	Garcés et al., 2009	Wachsmuth, 2007	Garcés et., 2013	García et al., 2013	Kruse, 2011	Di Ruscio et al., 2012	Kusel et al., 2015
General Criteria	Evolving Artifact	Source Metamodel		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
		Target Metamodel		n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	✓	✓	~	✓	×
		Transformation Metamodel		×	×	×	×	×	×	×	×	×	~	×
		Other		×	×	×	×	×	×	×	×	×	~	×
	Depending Artifact	Models		✓	✓	✓	✓	✓	✓	×	×	×	✓	✓
		Transformations		×	×	×	×	×	×	✓	✓	✓	✓	✓
		Other		×	×	×	×	×	×	×	×	×	✓	×
	Monitoring of Artifacts in the Ecosystem			×	×	×	×	×	×	×	×	×	×	×
	Transactions	Range	One Artifact	×	×	×	×	×	×	×	×	×	×	×
			One kind of Artifact	×	×	×	×	×	×	×	×	×	×	×
			All kinds of Artifacts	×	×	×	×	×	×	×	×	×	×	×
		Phases	Change Application	×	×	×	×	×	×	×	×	×	×	×
			Change Detection	×	×	×	×	×	×	×	×	×	×	×
			Impact Analysis	×	×	×	×	×	×	×	×	×	×	×
			Change Propagation	×	×	×	×	×	×	×	×	×	×	×
	Version management			×	×	×	×	×	×	×	×	×	×	×
Change Detection	Kind of detection	State-based		-	-	✓	✓	✓	-	✓	✓	-	-	-
		Operation-based		✓	-	-	-	-	✓	-	-	✓	-	✓
		Manual		-	✓	-	-	-	-	-	-	-	✓	
		Hybrid		-	-	-	-	-	-	-	-	-	-	-
	Granularity of change	Atomic		✓	n.a.	✓	✓	✓	✓	✓	✓	✓	n.a.	✓
		Composite		✓	n.a.	×	✓	✓	✓	✓	✓	✓	n.a.	✓
	Target of change	Syntax		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
		Semantics		×	×	×	×	×	×	×	×	×	×	×
Impact Analysis	Explicated analysis	ConformsTo		×	×	✓	×	×	×	×	×	×	×	✓
		SourceDomainConformsTo		×	×	×	×	×	×	×	✓	×	×	✓
		TargetDomainConformsTo		×	×	×	×	×	×	×	✓	×	×	×
		Other		×	×	×	×	×	×	×	×	×	×	×
	Level	Type-level		✓	×	✓	✓	×	×	✓	✓	✓	×	✓
		Instance-level		×	×	×	×	×	×	×	×	×	×	×
	Results Usable for Intervention			×	×	×	×	×	×	×	×	×	×	×
	Granularity	Models	Feature-level	✓	×	✓	✓	×	×	n.a.	n.a.	n.a.	×	✓
			Class-level	✓	×	✓	✓	×	×	n.a.	n.a.	n.a.	×	✓
			Package-level	✓	×	✓	✓	×	×	n.a.	n.a.	n.a.	×	✓
		Trans-formations	OCL-level	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	×	✓	×	×	✓
			Bindings-level	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	×	✓	✓	×	✓
			Rule-level	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	×	✓	✓	×	✓
			Module-level	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	×	✓	✓	×	✓
		Other		n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	×	×
	Run-time Effects Detection			×	×	×	×	×	×	×	×	×	×	×
Propagation	Strategies	Predefined	Number	1	0	1	1	1	1	1	1	1	0	1
			Customizable	✓	n.a.	×	✓	✓	✓	✓	✓	✓	n.a.	✓
		User-definable		✓	✓	×	×	×	×	×	✓	×	✓	✓
	Automation	Automatic		✓	✓	✓	✓	✓	✓	-	-	-	✓	✓
		Semi-automatic		-	-	-	-	-	-	✓	✓	✓	-	-
	Consistence	Intra-artifact		×	×	×	×	×	×	✓	×	×	×	✓
		Inter-artifact		×	×	×	×	×	×	×	×	×	×	✓
Validation	Type of Validation	Syntactic		×	×	×	×	×	×	×	×	×	×	✓
		Semantic		×	×	×	×	×	×	×	×	×	×	✓
	Range of Validation	One Artifact		×	×	×	×	×	×	×	×	×	×	✓
		One kind of Artifact		×	×	×	×	×	×	×	×	×	×	✓
		All kinds of Artifacts		×	×	×	×	×	×	×	×	×	×	✓

Legend: ✓ yes × no ~ partially n.a. not applicable

et al., 2013; Di Ruscio et al., 2012), while one approach supports them only in case of copy transformations, a specific form of a model transformation (Kruse, 2011). Evolution of the transformation metamodel or other modeling artifacts is not considered in the surveyed approaches, with a sole exception of (Di Ruscio et al., 2012) by providing a framework instead of a concrete approach. The monitoring of artifacts is not supported by any of the approaches, just as transactions and version management.

Regarding change detection, five approaches detect changes in a state-based manner (Gruschko et al., 2007; Cicchetti et al., 2008; Garcés et al., 2009; Garcés et al., 2013; García et al., 2013), four approaches employ operation-based change detection (Herrmannsdoerfer et al., 2009; Wachsmuth, 2007; Kruse, 2011; Kusel et al., 2015), and two rely on manual change specifications (Rose et al., 2010b; Di Ruscio et al., 2012). Almost all approaches support both atomic and composite changes, a single approach only does not support composite changes (Gruschko et al., 2007), while another approach considers composite changes as the pure composition of atomic changes, instead of its own set, thus allowing for new, self-defined composites (Kusel et al., 2015). All of the examined approaches consider syntax as the target of changes, while no approach is capable of targeting semantic changes.

Considering impact analysis, two approaches provide an explicated analysis of changes on the conforms-to relationship between models and metamodels (Gruschko et al., 2007; Kusel et al., 2015), two approaches the source-domain-conforms-to relationship (García et al., 2013; Kusel et al., 2015), and a sole approach for the target-domain-conforms-to relationship (García et al., 2013). Impact analysis is performed on type level by seven approaches (Herrmannsdoerfer et al., 2009; Gruschko et al., 2007; Cicchetti et al., 2009; Garcés et al., 2013; García et al., 2013; Kruse, 2011; Kusel et al., 2015), while impact analysis on instance level is supported by none of the examined approaches. Furthermore, results can not be used for intervention in any of the examined approaches. Approaches that support impact analysis on models provide analysis on feature, class and package level (Herrmannsdoerfer et al., 2009; Gruschko et al., 2007; Cicchetti et al., 2008; Kusel et al., 2015), while in case of transactions, the level of bindings, rules, and modules is supported by three approaches (García et al., 2013; Kruse, 2011; Kusel et al., 2015), whereas two approach additionally detect impacts on OCL level (García et al., 2013; Kusel et al., 2015). None of the examined approaches is capable of detect run-time effects before the actual execution of artifacts, e.g., running a transformation.

Regarding the actual propagation of changes, all but two approaches (Rose et al., 2010b; Di Ruscio et al., 2012) support a predefined migration strategy, which is customizable in all but one approach (Gruschko et al., 2007). Eight approaches allow for an automatic migration process (Herrmannsdoerfer et al., 2009; Rose et al., 2010b; Gruschko et al., 2007; Cicchetti et al., 2008; Garcés et al., 2009; Wachsmuth, 2007; Di Ruscio et al., 2012; Kusel et al., 2015), while three approaches targeting transformation co-evolution are semi-automatic (Garcés et al., 2013; García et al., 2013; Kruse, 2011), meaning that the user is needed in the co-evolution process. Intra-artifact consistence is ensured in two approaches (Garcés et al., 2013; Kusel et al., 2015), while a sole approach also provides inter-artifact consistency (Kusel et al., 2015).

A single approach supports the validation of migrated artifacts, both syntactically and semantically (Kusel et al., 2015). In this context, the approach is able to validate all instances of all supported kinds of artifacts, i.e., models and transformations (Kusel et al., 2015).

In the next section, lessons learned from the evaluation of approaches are presented.

4 LESSONS LEARNED

After having evaluated existing co-evolution approaches with the presented evaluation framework, in the following paragraphs, lessons learned drawn from the evaluation are discussed.

Need for Ecosystem-wide Perspective. As one might see from the evaluation in Section 3.2, most current co-evolution approaches tackle either models or transformation as dependent artifacts, but miss an ecosystem-wide perspective spanning over several modeling artifacts. Consequently, a systematic and efficient co-evolution is hindered due to the fact that diverse tools or approaches have to be employed to co-evolve a complete modeling ecosystem. This may lead to an inter-artifact inconsistency, i.e., different kinds of artifacts are co-evolved differently, thus, the operability of the modeling ecosystem is broken.

Incorporation of Semantical Changes Needed. As discussed earlier, changes in a modeling ecosystem might be of semantical nature, thus, not affecting the syntax of a metamodel. As an example, a modeler might intrinsically assign the unit centimeter to a "height" attribute of a Person. Thus, a change of centimeter to millimeter affects all depending artifacts, since the actual values in the models or potential cal-

culations in transformations have to be adapted. However, a change of a unit can not be explicitly expressed in current co-evolution approaches, thus, hindering an automated co-evolution of depending artifacts. Consequently, means to express semantical changes are needed.

Transactional Execution Beneficial for Consistency. Since transactions are currently not supported in any of the surveyed approaches, errors that occur during co-evolution have to be identified and corrected manually. In order to mitigate this burden, transactions spanning over all phases of the co-evolution process including the actual propagation of changes enable consistency of the ecosystem, since either all artifacts are co-evolved or the whole ecosystem is set back to its initial state.

Enabling Run-time Effects Detection by Employing Code Analysis Techniques. Changes on the metamodels might impact the ecosystem not only at design-time but also at run-time, e.g., when a transformation queries the value of an element which is no longer set. However, by employing static code analysis techniques (Louridas, 2006) to detect errors, e.g., potential null-pointer references, before executing the transformation, run-time effects can be detected and corrected in an earlier stage, i.e., before execution. Consequently, a more sophisticated impact analysis that reveals potential run-time effects helps the evolution designer in deciding if one or more changes should be applied on the metamodel.

5 RELATED WORK

In this section, we will discuss related work focusing on comparison and evaluation of co-evolution approaches in MDE.

In a recent survey, the authors compare several approaches for the co-evolution of models and their metamodels (Herrmannsdoerfer and Wachsmuth, 2014) on the basis of a comprehensive criteria catalog. In contrast to our contribution, the authors consider models as depending artifacts, only. Another recent work discusses state-of-the-art techniques and approaches for co-evolution in MDE, and outlines future research challenges (Paige et al., 2016). However, in contrast to the contribution of this paper, there is no concrete focus on modeling ecosystems or a comparison of concrete co-evolution approaches. A comparison between co-evolution of models and transformations in response to metamodel evolution has been presented in (Rose et al., 2010b). The authors discuss the differences between the migration of models and transformation and highlight the need for a more uniform, i.e., consistent, co-evolution of both artifacts. However, concrete approaches are not compared in contrast to our contribution. In (Meyers and Vangheluwe, 2011) the authors propose a framework for the evolution of metamodels and co-evolution of various kinds of depending artifacts, such as models and transformations. Thus, they propose a taxonomy for metamodel evolution, which might be applied to existing approaches. However, an evaluation of co-evolution approaches has not been part of their work.

In summary, one might see that the work presented in this paper is unique with respect to a modeling ecosystem-wide perspective regarding co-evolution and the consideration of diverse depending artifacts, in particular models and transformations.

6 CONCLUSION & FUTURE WORK

In this paper, we presented a comparison of co-evolution approaches in MDE, with a specific focus on modeling ecosystems and metamodels as evolving artifacts and models and transformations as depending artifacts. Therefore, criteria for evaluating co-evolution approaches have been proposed, and eleven approaches have been evaluated by applying the evaluation framework, which was built upon the identified criteria. Finally, lessons learned drawn from the evaluation have been presented.

Summing up, although several co-evolution approaches in MDE exist, there is still space for improvements, including (i) the dedicated co-evolution support of more than one depending artifact, (ii) the detection and incorporation of semantic changes on the metamodel, (iii) the implementation of transaction mechanisms to cope with unforeseen errors in the co-evolution process, and (iv) strengthening impact analysis by employing static code analysis techniques.

REFERENCES

Bézivin, J. (2005). On the Unification Power of Models. *SoSym*, 4(2):171–188.

Brambilla, M., Cabot, J., and Wimmer, M. (2012). *Model-Driven Software Engineering in Practice*. Morgan & Claypool Publishers.

Cicchetti, A., Di Ruscio, D., Eramo, R., and Pierantonio, A. (2008). Automating Co-evolution in Model-Driven Engineering. In *EDOC '08*, pages 222 –231.

Cicchetti, A., Di Ruscio, D., and Pierantonio, A. (2009). Managing Dependent Changes in Coupled Evolution. In *ICMT*, volume 5563 of *LNCS*, pages 35–51. Springer.

Czarnecki, K. and Helsen, S. (2006). Feature-Based Survey of Model Transformation Approaches. *IBM Systems Journal*, 45(3):621–645.

Di Ruscio, D., Iovino, L., and Pierantonio, A. (2011). What is Needed for Managing Co-Evolution in MDE? In *Proc. of the Int. Workshop on Model Comparison in Practice*, pages 30–38. ACM.

Di Ruscio, D., Iovino, L., and Pierantonio, A. (2012). Evolutionary Togetherness: How to Manage Coupled Evolution in Metamodeling Ecosystems. In *ICGT*, pages 20–37. Springer.

Fowler, M., Beck, K., Brant, J., Opdyke, W., and Roberts, D. (1999). *Refactoring: improving the design of existing code*. Addison-Wesley.

Garcés, K., Jouault, F., Cointe, P., and Bézivin, J. (2009). Managing Model Adaptation by precise Detection of Metamodel Changes. In *Model Driven Architecture-Foundations and Applications*, pages 34–49. Springer.

Garcés, K., Vara, J. M., Jouault, F., and Marcos, E. (2013). Adapting transformations to metamodel changes via external transformation composition. *SoSym*, pages 1–18.

García, J., Diaz, O., and Azanza, M. (2013). Model Transformation Co-evolution: A Semi-automatic Approach. In *SLE*, pages 144–163. Springer.

Gruschko, B., Kolovos, D., and Paige, R. (2007). Towards Synchronizing Models with Evolving Metamodels. In *Proc. of the Int. Workshop on Model-Driven Software Evolution*.

Herrmannsdoerfer, M., Benz, S., and Juergens, E. (2009). COPE - Automating Coupled Evolution of Metamodels and Models. In *ECOOP*, pages 52–76. Springer.

Herrmannsdoerfer, M. and Wachsmuth, G. (2014). Coupled Evolution of Software Metamodels and Models. In *Evolving Software Systems*, pages 33–63. Springer.

Jouault, F., Allilaire, F., Bézivin, J., and Kurtev, I. (2008). ATL: A model transformation tool. *Science of Computer Programming*, 72(12):31–39.

Kappel, G., Langer, P., Retschitzegger, W., Schwinger, W., and Wimmer, M. (2012). Model Transformation By-Example: A Survey of the First Wave. In *Conceptual Modelling and Its Theoretical Foundations*, volume 7260 of *LNCS*, pages 197–215. Springer.

Kruse, S. (2011). On the Use of Operators for the Co-Evolution of Metamodels and Transformations. In *Int. Workshop on Models and Evolution @ MODELS*.

Kusel, A., Etzlstorfer, J., Kapsammer, E., Retschitzegger, W., Schwinger, W., and Schönböck, J. (2015). Consistent Co-Evolution of Models and Transformations. In *Proc. of MODELS*, pages 116–125.

Levendovszky, T., Balasubramanian, D., Narayanan, A., and Karsai, G. (2010). A Novel Approach to Semi-automated Evolution of DSML Model Transformation. In *SLE*, volume 5969 of *LNCS*, pages 23–41. Springer.

Louridas, P. (2006). Static code analysis. *IEEE Software*, 23(4):58–61.

Méndez, D., Etien, A., Muller, A., and Casallas, R. (2010). Towards Transformation Migration After Metamodel Evolution. In *Proc. of Int. Workshop on Models and Evolution @ MODELS*.

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

Object Management Group (2011). Meta Object Facility (MOF) 2 Core Specification. www.omg.org/spec/MOF/2.4.1.

Paige, R., Matragkas, N., and Rose, L. (2016). Evolving models in Model-Driven Engineering: State-of-the-art and future challenges. *Journal of Systems and Software*, 111:272 – 280.

Rose, L., Etien, A., Méndez, D., Kolovos, D., Paige, R., and Polack, F. (2010a). Comparing Model-Metamodel and Transformation-Metamodel Co-evolution. In *Proc. of Models and Evolution Workshop*.

Rose, L., Kolovos, D., Paige, R., and Polack, F. (2010b). Model Migration with Epsilon Flock. *Proc. of ICMT*, pages 184–198.

Schönböck, J., Kusel, A., Etzlstorfer, J., Kapsammer, E., Schwinger, W., Wimmer, M., and Wischenbart, M. (2014). CARE – A Constraint-Based Approach for Re-Establishing Conformance-Relationships. In *Proc. of APCCM*.

Sendall, S. and Kozaczynski, W. (2003). Model Transformation: The Heart and Soul of Model-Driven Software Development. *Software, IEEE*, 20(5):42–45.

Wachsmuth, G. (2007). Metamodel Adaptation and Model Co-adaptation. In *Proc. of European Conf. on Object-Oriented Programming*, volume 4609 of *LNCS*, pages 600–624. Springer.