A Systematic Taxonomy of Metamodel Evolution Impacts on OCL Expressions[*]

Angelika Kusel[1], Juergen Etzlstorfer[2], Elisabeth Kapsammer[1], Werner Retschitzegger[1], Johannes Schoenboeck[3], Wieland Schwinger[1], and Manuel Wimmer[2]

[1] Johannes Kepler University Linz, Austria
`[firstname.lastname]@jku.at`
[2] Vienna University of Technology, Austria
`[lastname]@big.tuwien.ac.at`
[3] University of Applied Sciences Upper Austria, Campus Hagenberg, Austria
`[firstname.lastname]@fh-hagenberg.at`

Abstract. Metamodel evolution is prevalent in Model-Driven Engineering, necessitating the co-evolution of dependent artifacts like models and transformations. Whereas model co-evolution has been extensively studied, the co-evolution of transformations and especially its substantial ingredient in terms of OCL expressions has received little attention up to now. Thus, the goal of this paper is a systematic analysis of potential impacts of metamodel evolution on OCL expressions in model transformations. For this, a *complete and minimal* set of atomic metamodel changes has been derived from Ecore, which is analyzed with respect to its effects on structural complexity and information capacity. This analysis builds the basis for investigating the *impacts* concerning *syntactical conformance* and *scope* of affected OCL expressions. Finally, we report on lessons learned gained from establishing the set of changes and examining the impacts thereof.

1 Introduction

Model-Driven Engineering (MDE) proposes the use of models to conduct software development on a higher level of abstraction [1]. Thereby, model transformations play a vital role for systematic transformations of models conforming to different metamodels (MMs). Just like any other software artifact, MMs evolve, necessitating the co-evolution of dependent artifacts like models and transformations [10].

While the automated co-evolution of models has been subject to extensive research in the past (cf., [9] for a survey), the automated co-evolution of transformations has been less examined so far (cf., e.g., [4–6, 13]). Especially the co-evolution of Object Constraint Language (OCL) [18] expressions has not been a major focus up to now, despite the fact that OCL expressions are used to perform complex queries on the input models [2, 22]. Therefore, they represent a substantial ingredient in rule-based model transformation languages, such as ATL [11] or QVT [17].

[*] This work has been funded by the Austrian Federal Ministry for Transport, Innovation and Technology (BMVIT) grant FFG BRIDGE 832160 and FFG FIT-IT 825070 and 829598, FFG Basisprogramm 838181, and by ÖAD grant AR18/2013 and UA07/2013.

To tackle this limitation, this paper focuses on the co-evolution of OCL expressions in model transformations by first proposing a *complete and minimal set of atomic changes* focusing on structure, which has been systematically derived from the Ecore[4] meta-MM, enabling the definition of arbitrary evolutions of Ecore-based MMs. All changes of this set are subsequently analyzed concerning their effects on the MM with respect to *structural complexity*, i.e., the number of instantiable MM elements, and *information capacity*, i.e., the potential number of instances of the MM, since these two criteria are significant for the impacts on OCL expressions which will be revealed in the remainder of this paper. Second, the *potential impacts* of these changes on OCL expressions are investigated by systematically analyzing the impacts of each of these changes concerning affected OCL expressions, revealing *non-breaking* and *breaking* impacts [7] with respect to *syntax* and scattering of impacts considering their *scope*, being *local*, in case that OCL expressions use the changed MM element itself, or *global*, if they use inherited versions thereof. Thus, this investigation builds the foundation for identifying resolution actions to co-evolve syntactically broken OCL expressions and serves as basis for implementing an impact analysis tool, constituting the first and fundamental step towards the automated co-evolution of OCL expressions in model transformations.

Outline: Section 2 systematically analyzes the impacts of MM evolution on OCL expressions. While lessons learned are presented in Section 3, related work is surveyed in Section 4. Finally, Section 5 concludes the paper with an outlook to future work.

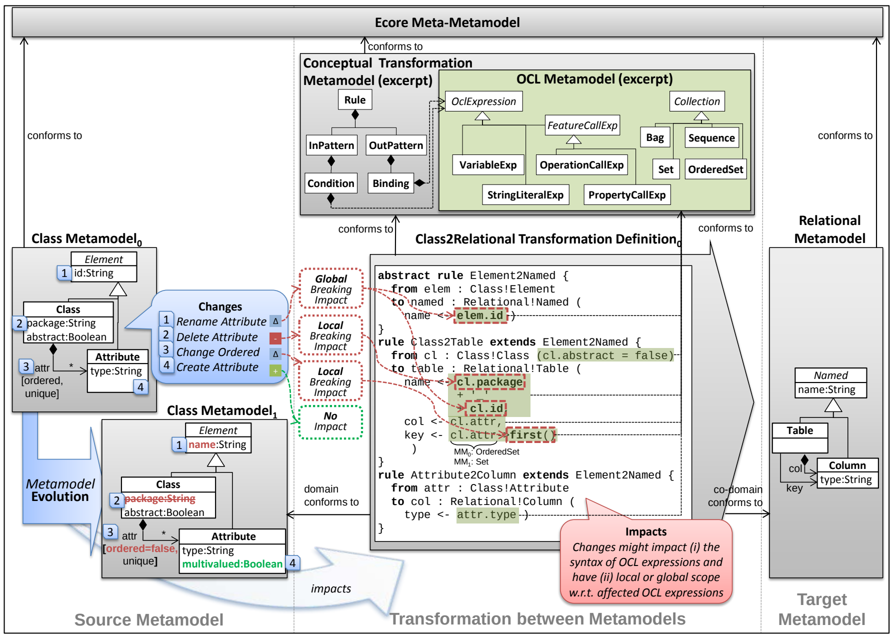
2 Systematic Impact Analysis

In this section, role and importance of OCL in model transformations are highlighted, before the complete and minimal set of changes as well as the investigation of impacts of each change on OCL expressions are presented. Although OCL might also be used in other contexts, e.g., to specify MM constraints restricting the instantiability of the MM, we focus on the co-evolution of OCL in model transformations. Nevertheless, this work might also be applied to other application contexts. A detailed investigation of impacts on OCL constraints in MMs is, however, left to future work.

2.1 Role and Importance of OCL in Model Transformations

In order to illustrate the role and importance of OCL, Figure 1 shows an excerpt of the well-known Class2Relational transformation[5], serving as a running example throughout the paper. From the example one might see that OCL expressions are used in two indispensable roles [13]. First, OCL is used in *bindings* to query elements of the source model, which are used to produce the target model (cf., e.g., “cl.package+' '+cl.id” calculating the values for the target attribute Table.name). Second, OCL is utilized in *conditions* to steer the control flow (cf., e.g., “cl.abstract=false” to transform non-abstract classes, only). Through these two essential roles, OCL expressions constitute large parts of transformation definitions [2, 22], and thus, it is of utmost importance to consider OCL in detail in the context of transformation co-evolution.

[4] http://eclipse.org/modeling/
[5] For a complete example see: http://www.eclipse.org/atl/atlTransformations/



```
abstract rule Element2Named {
  from elem : Class!Element
  to named : Relational!Named (
    name <- elem.id )
}
rule Class2Table extends Element2Named {
  from cl : Class!Class (cl.abstract = false)
  to table : Relational!Table (
    name <- cl.package
            + '_'
            + cl.id
    col <- cl.attr,
    key <- cl.attr->first()
    )
}
rule Attribute2Column extends Element2Named {
  from attr : Class!Attribute
  to col : Relational!Column (
    type <- attr.type )
}
```

Fig. 1. Running Example: Class2Relational

In general, model transformations depend on three distinct MMs, being (i) the source MM, (ii) the target MM, and (iii) the transformation MM (cf. Fig. 1). Thereby, OCL expressions depend on the source MM by means of a so-called "domain conforms to"-relationship [15] and the OCL MM as part of the transformation MM by means of a "conforms to"-relationship [11]. This is, since OCL expressions are used to query source models and do not refer to concepts of the target MM. Therefore, this paper focuses on the evolution of the source MM and its impact on OCL expressions. For this, a systematic set of changes is needed, which will be the focus of the next two subsections.

2.2 Complete and Minimal Set of Changes

A *systematic* set of changes, as a prerequisite for investigating impacts, has to fulfill two criteria – *completeness* to allow for any possible change and *minimality* to avoid the analysis of redundant changes. To fulfill both, we focus on *atomic changes*, transferring the MM from one consistent state, i.e., conforming to Ecore [20], to another one.

Example. Before proposing the systematic set of atomic changes, four exemplary atomic changes (cf. Fig. 1) with their effects on structural complexity and information capacity of the MM are discussed. First, the attribute Element.id has been renamed to name (cf. ① in Fig. 1), being updative, i.e., a state change, by nature. Although this causes neither a change in structure nor a change in information capacity, OCL expressions are affected. Second, the attribute Class.package has been deleted (cf. ② in Fig. 1), being destructive by nature, thus, decreasing structural complexity and information capacity, which impacts OCL expressions significantly, since the deleted element

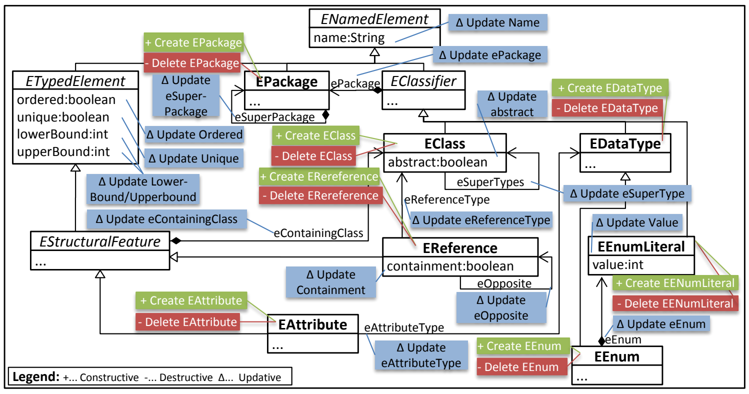


Fig. 2. Derived Set of Atomic Changes for Ecore-based MMs

must not be accessed anymore. Third, the reference Class.attr has been changed from ordered to unordered (cf. ③ in Fig. 1), being again updative by nature, leaving structural complexity and information unaffected, but, however, affecting OCL expressions. Finally, the attribute Attribute.multivalued of type Boolean has been created (cf. ④ in Fig. 1), being constructive by nature, therefore increasing both, structural complexity and information capacity, since this attribute might now be instantiated with true or false, without, however, affecting OCL expressions.

Systematic Set of Changes. Going beyond these four exemplary changes, Figure 2 shows the relevant excerpt of Ecore, including all elements for defining *structure*, while disregarding (i) properties for *code generation* (e.g., volatile), (ii) *derived properties* (e.g., required), since they may be led back from other properties (e.g., lowerBound), and (iii) *operations* (e.g., EOperation), since the focus is on MMs defining structure and not behavior. For deriving all *constructive* and *destructive* changes, one has to resort to all concrete meta-classes, e.g., EClass. For receiving all *updative* changes, i.e., state changes of features, one has to refer to all meta-features, e.g., EClass.abstract. The resulting set of atomic changes is shown in Figure 2 as well as in Table 2.

Criteria. Before analyzing the impacts of the changes on OCL expressions, their effects with respect to (i) *structural complexity* and (ii) *information capacity* are analyzed, being *increasing*, *neutral*, or *decreasing*. Changes affecting *structural complexity* indicate impacts in accessing MM elements in OCL expressions and might be evaluated by counting the number of instantiable MM elements [19]. In contrast, changes concerning *information capacity* indicate impacts on the results of OCL expressions and might be evaluated by counting the potential number of all valid instances of a MM [16].

Evaluation. In the following, the set of changes is evaluated (cf. Table 2).

Constructive/Destructive Changes: All constructive changes have an increasing effect on both, structural complexity and information capacity, since they increase the number of instantiable MM elements and by this also the potential number of valid instances. In contrast, all destructive changes have the exact opposite effect.

Updative Changes: Whereas all constructive as well as destructive changes behave equally with respect to our criteria, updative changes do not and might be further subdivided into four groups according to their behavior.

Group ① Renaming Updates: The first group includes updates on ENamedElement.name and EEnumLiteral.value, i.e., renames, being neutral with respect to both, structural complexity and information capacity.

Group ② Moving Updates: This group regards updates on containment references, i.e., EPackage.eSuperPackage, EClassifier.ePackage, EStructuralFeature.eContainingClass, as well as EEnumLiteral.eEnum, which enable the movement of a feature from one container to another one. Such updates increase structural capacity in the target container, but decrease structural complexity in the source container. Since the features are still available in the MM, yet at another position, the effect on the information capacity is neutral, i.e., not affecting the number of valid instances.

Group ③ Restricting/Relaxing Updates: The third group considers updates on restricting or relaxing the instantiability of MM elements, comprising the features abstract of EClass, upperBound, lowerBound, and unique of ETypedElement as well as all features of EAttribute and EReference. Their effect on structural complexity is neutral, but their effect on information capacity is either increasing or decreasing, depending on the concrete state change. For instance, in case of an increase of feature lowerBound, the number of valid instances decreases, since more values are required. In contrast, a decrease of lowerBound has the opposite effect. Furthermore, type specialization has decreasing effect on information capacity, since the set of valid instances decreases. In contrast, type generalization has increasing effect on information capacity. Please note that feature ETypedElement.ordered has neutral effect on both, structural complexity and information capacity, but impacts the underlying OCL datatype (cf. Sect. 2.3).

Group ④ Constructive/Destructive Updates: Finally, this group considers updates on types, i.e., EClasses themselves, or the datatypes of EStructuralFeatures. This group may be further subdivided into two categories according to their effects. First, the addition of eSuperTypes and pulling up of EStructuralFeatures have increasing effect on information capacity, while their effect on structural complexity is increasing for the addition of eSuperTypes and both, increasing and decreasing for EStructuralFeatures, analogously to moving updates. Second, the deletion of eSuperTypes and pushing down of EStructuralFeatures has decreasing effect on information capacity, while their effect on structural complexity is decreasing for the removal of eSuperTypes and again both, increasing and decreasing for EStructuralFeatures.

2.3 Impact Analysis

In the following, impacts of MM evolution on OCL expressions are exemplified and on basis of this, dedicated criteria are derived, which are finally evaluated with respect to the complete and minimal set of changes.

Example. To reveal impacts of MM evolution on OCL expressions, the running example is utilized again: first, the renaming of the attribute Element.id (cf. ① in Fig. 1) has no effect with respect to structural complexity and information capacity, but breaking impact on the syntax of all OCL expressions accessing the element either directly or indirectly via inherited versions thereof, i.e., the impact scatters. Second, the deletion of the attribute Class.package (cf. ② in Fig. 1) naturally has breaking impact on all OCL expressions accessing this element, since the structure has been changed in a destructive way, and since belonging to a leaf class, the impact does not scatter. Third, the

Ecore Meta-Feature		OCL Type				
		Scalar Type	Collection			
			Bag	Sequence	Set	OrderedSet
lowerBound		no impact on OCL type				
upperBound = 1	unique/ordered not applicable	✓				
upperBound > 1	unique = true and ordered = true					✓
	unique = true and ordered = false				✓	
	unique = false and ordered = true			✓		
	unique = false and ordered = false		✓			

Table 1. Resulting OCL Types out of Ecore Settings

change of the reference Class.attr from ordered to unordered, i.e., ordered=false (cf. ③ in Fig. 1), has breaking impact, although the structural complexity is unaffected, i.e., the feature is still accessible. However, it causes a change of the internally employed OCL collection type from OrderedSet to Set and by this, invalidates the usage of now undefined operations such as first(). In this context, Table 1 shows the possible Ecore settings related to collections and the resulting OCL collection type. Finally, the creation of the attribute Attribute.multivalued (cf. ④ in Fig. 1) naturally has no impact.

Criteria. As one might see from the exemplary discussion above, changes may have potential impact on the *syntax* of OCL expressions being either *non-breaking* or *breaking*. Moreover, a change exhibits a certain *scope*, i.e., the scattering of the impact, being *local*, i.e., OCL expressions using the MM element itself, or *global*, i.e., on OCL expressions using inherited versions thereof.

Evaluation. In the following, all changes are systematically evaluated with respect to these criteria. Please note that the evaluation assumes that changed MM elements have been used by at least one OCL expression and the worst case scenario is considered, i.e., changes are evaluated as breaking, if there exists at least one case that breaks the OCL expression. Since the vast majority of changes have local impact regarding the scope as long as they concern elements in leaf classes, this criterion is discussed for exceptional cases, only. The detailed results of the evaluation may be found in Table 2.

Constructive/Destructive Changes: Constructive changes do not have any impact on OCL expressions, since newly created elements can not have been referred to. In contrast, destructive changes always have breaking impact on OCL expressions, since having a destructive effect on the structure.

Updative Changes: Updative changes are evaluated on basis of the groups introduced in Section 2.2, in the following.

Group ① Renaming Updates: Although renames do neither affect structural complexity nor information capacity, their impact is nevertheless breaking, since renamed elements are no longer accessible under their original name.

Group ② Moving Updates: Since moves change the structure of instances by changing the position of elements, the impact on OCL is always breaking.

Group ③ Restricting/Relaxing Updates: Although these updates leave the structural complexity unaffected, they impact information capacity, which may also break the syntax of OCL expressions. This is since different settings of features such as ETypedElement.ordered result in different OCL datatypes (cf. Table 1). For example, changing the feature ordered from true to false implies a change from the OCL collection type OrderedSet to Set, thereby invalidating, e.g., the usage of the operation first().

Group ④ Constructive/Destructive Updates: This group is divided into two categories concerning their effect with respect to information capacity as already mentioned

	Change on Ecore-based Metamodel					Effect on						Impact on OCL			
						Structural Complex-ity			Informa-tion Capacity			Syntax		Scope	
	Ecore Meta-Class	Name of Atomic Change	Meta-Feature	State Change of Meta-Feature	Group	Increase	Neutral	Decrease	Increase	Neutral	Decrease	Non-breaking	Breaking	Local	Global[1]
Constructive	`EPackage`	Create EPackage	*n.a.*	n.a.	n. a.	+			+			no impact possible			
	`EClass`	Create EClass	*n.a.*	n.a.		+			+						
	`EAttribute`	Create EAttribute	*n.a.*	n.a.		+			+						
	`EReference`	Create EReference	*n.a.*	n.a.		+			+						
	`EDataType`	Create EDataType	*n.a.*	n.a.		+			+						
	`EEnum`	Create EEnum	*n.a.*	n.a.		+			+						
	`EEnumLiteral`	Create EEnumLiteral	*n.a.*	n.a.		+			+						
Destructive	`EPackage`	Delete EPackage	*n.a.*	n.a.	n. a.			-			-		x	x	
	`EClass`	Delete EClass	*n.a.*	n.a.				-			-		x	x	
	`EAttribute`	Delete EAttribute	*n.a.*	n.a.				-			-		x	x	
	`EReference`	Delete EReference	*n.a.*	n.a.				-			-		x	x	
	`EDataType`	Delete EDataType	*n.a.*	n.a.				-			-		x	x	
	`EEnum`	Delete EEnum	*n.a.*	n.a.				-			-		x	x	
	`EEnumLiteral`	Delete EEnumLiteral	*n.a.*	n.a.				-			-		x	x	
Updative	*`ENamedElement`* `(inherited by: EClass, EPackage, EAttribute, EReference, EDataType, EEnum EEnumLiteral)`	Update Name	*name*	oldName → newName and oldName <> newName	①		o			o			x	x	
	`EPackage`	Update ESuperPackage	*eSuperPackage*	oldeSuperPackage → neweSuperPackage	②	+		-		o			x	x	
	`EClassifier` `(inherited by: EClass, EDataType, EEnum)`	Update EPackage	*ePackage*	oldePackage → newePackage		+		-		o			x	x	
	`EClass`	Update Abstract	*abstract*	true → false	③		o		+			x		x	
				false → true			o				-	x		x	
		Update ESuperType	*eSuperType*	add	④	+			+			x		x	
				remove				-			-		x	x	x
	`ETypedElement` `(inherited by: EAttribute, EReference)`	Update UpperBound	*upperBound*	x → y, x>1, y>x	③		o		+			x		x	
				1 → >1			o		+				x	x	
				x → y, y>1, y<x			o				-		x	x	
				>1 → 1			o				-		x	x	
		Update LowerBound	*lowerBound*	x → y, x>0, y>x			o				-	x		x	
				0 → >0			o				-	x		x	
				x → y, y>0, y<x			o		+			x		x	
				>0 → 0			o		+			x		x	
		Update Ordered	*ordered*	false → true			o			o			x	x	
				true → false			o			o			x	x	
		Update Unique	*unique*	false → true			o				-		x	x	
				true → false			o		+				x	x	
	`EStructuralFeature` `(inherited by: EAttribute, EReference)`	Update EContainingClass	*eContainingClass*	push down	④	+		-			-		x	x	
				pull up		+		-	+			x		x	
				along reference with upperBound = 1	②	+		-		o			x	x	
				other	④	+		-	+		-		x	x	
	`EAttribute`	Update EAttributeType	*eAttributeType*	specialize	③		o				-	x		x	
				generalize			o		+				x	x	
				other			o		+		-		x	x	
	`EReference`	Update Containment	*containment*	false → true			o				-	x		x	
				true → false			o		+			x		x	
		Update EReferenceType	*eReferenceType*	specialize			o				-	x		x	
				generalize			o		+				x	x	
				other			o		+		-		x	x	
		Update EOpposite	*eOpposite*	add			o				-	x		x	
				remove			o		+			x		x	
	`EEnumLiteral`	Update Value	*value*	oldValue <> newValue	①		o			o		x		x	
		Update EEnum	*eEnum*	oldeEnum <> neweEnum	②	+		-		o			x	x	

Legend: x ... True n.a. ... Not applicable [1] ... Always true when changed in superclass +... Increase o... Neutral -... Decrease
Group: ①... Renaming updates ②... Moving updates ③... Restricting/relaxing updates ④... Constructive/destructive updates

Table 2. Set of Atomic Changes and their Impacts on OCL Expressions

above. First, updates increasing information capacity are comparable to constructive changes and thus, non-breaking. Second, updates decreasing information capacity are comparable to destructive changes and are thus, breaking. The scope of these updates is local, unless eSuperTypes are removed, affecting inheriting elements and therefore, having global impact.

3 Lessons Learned

This section discusses lessons learned gained from (i) establishing the complete and minimal set of changes as well as from (ii) investigating impacts.

Universal Applicability of Change Set Derivation Procedure. Although we focused on one specific meta-MM, i.e., Ecore, the approach of deriving constructive and destructive changes from concrete meta-classes as well as updative changes from all meta-features is universally applicable since it might be applied to any meta-metamodel.

Atomic Changes Allow for Non-redundant Impact Analysis. Since the employed change set is minimal comprising atomic changes, only, it allows for non-redundant impact analysis. In contrast, a set of composite changes might include overlaps, e.g., "Extract Class" and "Extract Superclass" both include the change "Create EClass". Composite changes, however, might held more information to be exploited for co-evolution.

State-Changes of Meta-Features are Pivotal. As might be seen in Table 2, all changes of meta-features have been broken down into several cases, explicating different state changes. This has been necessary, since different state changes entail different effects on structural complexity and information capacity and, consequently, impact OCL expressions differently, e.g., the state change of upperBound from 1 to > 1 has breaking impact, whereas the state change from > 1 to another number > 1 has not.

Increase of Structural Complexity Breaks Models, but not OCL. All constructive changes as well as updates with constructive effects (cf. part of group ④) that increase structural complexity never break OCL expressions. This is in contrast to model co-evolution where the introduction of required elements has breaking impact on models, since models rely on a different kind of relationship to their MM, i.e., "conforms to", while transformations "domain conform to" their source and target MMs.

Changes not Affecting Structural Complexity may Break OCL. Impact analysis revealed that changes not affecting structural complexity, e.g., updates of group ③, may nevertheless induce a syntactical breakage of OCL expressions in certain cases as explicated above. This is, since changes of group ③ may induce implicit type changes of the underlying OCL datatypes and by this, change the set of valid operations.

4 Related Work

Subsequently, related work is evaluated with respect to its focus, supported changes, impact analysis on OCL, and support by a prototypical implementation (cf. Table 3).

Regarding the *focus of co-evolution* in a specific *technical space*, two groups of approaches exist. Most closely related, the first group of approaches targets the co-evolution of transformations employing OCL expressions [5, 6, 13] in the *technical space* of Ecore, whereby the co-evolution of the OCL-part is considered particularly

Approach	Focus of Work					Supported Changes					Impact on OCL		Imple-menta-tion
	Co-Evolution of		Technical Space			Classes of Supported Changes			Complete Set	Minimal Set			
	OCL in Model Transformations	OCL Constraints in Metamodels	Ecore	UML	MOF	Constructive	Destructive	Updative			Syntax	Scope	
García et al. [6]	~ (ATL)		✓			~	~	~	× (23 atomic & composite changes)	×	~	×	✓
Garcés et al. [5]	× (ATL)		✓			~	~	~	× (number of atomic & composite changes unkown)	×	×	×	✓
Kruse [13]	× (ATL)		✓			~	~	~	× (16 atomic & composite changes)	×	✓	×	✓
Hassam et al. [8]		✓			✓	~	~	~	× (17 atomic & composite changes)	×	✓	×	✓
Markovic et al. [14]		✓		✓		×	×	~	× (6 atomic & composite changes)	×	✓	×	✓
Kosiuczenko [12]		✓		✓		~	~	~	× (number of atomic & composite changes unkown)	×	×	×	×
Correa et al. [3]		✓		✓		~	×	~	× (number of atomic & composite changes unkown)	×	×	×	✓
Own work	✓ (ATL)		✓			✓	✓	✓	✓ (30 atomic changes)	✓	✓	✓	Future work

Legend: ✓ ... true × ... false ~ ... partially true

Table 3. Comparison of Related Approaches

by one of them [6], only. More widely related since not basing on Ecore and by this entailing a different set of changes, but nevertheless facing similar challenges, the second group concentrates on the co-evolution of OCL constraints as parts of UML class diagrams [3, 12, 14], with one exception basing on MOF [8].

Considering the *supported changes*, six approaches [3, 5, 6, 8, 12, 13] partially allow for constructive changes, five of those [5, 6, 8, 12, 13] partially consider destructive changes, and updative changes are partially supported by all approaches. Thus, no approach covers a *complete* change set. However, the surveyed approaches additionally consider composite changes, which will be one line of future work as detailed below. By concentrating on composite changes, no approach presents a minimal change set, which is different to our work providing a systematically derived, minimal set of changes.

Regarding the *impact on OCL*, four approaches [6, 8, 13, 14] consider breaking and non-breaking impacts on the *syntax*, whereby one of them [6] considers impacts partially, only. Considering the *scope* of impact on OCL, no approach regards this. Finally, six approaches [3, 5, 6, 8, 13, 14] provide an *implementation*, while a sole approach is conceptual, only, like the work presented in this paper.

In summary, one might see that the work presented in this paper is unique with respect to the complete and minimal set of changes and a systematic in-depth investigation of impacts. This is in contrast to related approaches, which rather concentrate on fully supporting co-evolution for smaller sets of selected composite changes.

5 Conclusion & Future Work

This paper provided a systematic investigation of impacts of MM evolution on OCL expressions in model transformations. Basing thereupon, several lines of future work remain open. First, resolution actions to resolve violations caused by MM evolution have to be identified. Their goal will be to perform local repairing by establishing a view simulating the old MM version, e.g., in case of decreasing the upperBound from > 1 to 1, the now single-valued feature will be wrapped into a collection. In case that multiple changes have been performed on a single MM-element, the resolution actions should be chained analogous to the idea presented in [21]. This chaining will represent a first step towards the support for composite changes out of atomic changes, which will be the next step in our research agenda. Moreover, we plan to investigate impacts and resolution actions for complete transformation definitions, i.e., not only parts written in OCL, thereby also focusing on impacts caused by an evolution of the target MM. Finally, we will implement the conceptual approach presented in this paper.

References

1. Bézivin, J.: On the Unification Power of Models. SoSym 4(2) (2005)
2. Cabot, J., Gogolla, M.: Object Constraint Language (OCL): A Definitive Guide. In: Formal Methods for Model-Driven Engineering. Springer (2012)
3. Correa, A., Werner, C.: Applying Refactoring Techniques to UML/OCL Models. In: UML 2004. Springer (2004)
4. Di Ruscio, D., Iovino, L., Pierantonio, A.: Evolutionary Togetherness: How to Manage Coupled Evolution in Metamodeling Ecosystems. In: ICGT. Springer (2012)
5. Garcés, K., Vara, J., Jouault, F., Marcos, E.: Adapting transformations to metamodel changes via external transformation composition. SoSym (2013)
6. García, J., Diaz, O., Azanza, M.: Model Transformation Co-evolution: A Semi-automatic Approach. In: Software Language Engineering. Springer (2013)
7. Gruschko, B., Kolovos, D., Paige, R.: Towards Synchronizing Models with Evolving Metamodels. In: Int. Workshop on Model-Driven Software Evolution (2007)
8. Hassam, K., Sadou, S., Gloahec, V.L., Fleurquin, R.: Assistance System for OCL Constraints Adaptation during Metamodel Evolution. In: SMR. IEEE (2011)
9. Herrmannsdörfer, M., Wachsmuth, G.: Coupled Evolution of Software Metamodels and Models. In: Evolving Software Systems. Springer (2014)
10. Iovino, L., Pierantonio, A., Malavolta, I.: On the Impact Significance of Metamodel Evolution in MDE. JoT 11(3) (2012)
11. Jouault, F., Allilaire, F., Bézivin, J., Kurtev, I.: ATL: A model transformation tool. Science of Computer Programming 72(1–2) (2008)
12. Kosiuczenko, P.: Redesign of UML class diagrams: a formal approach. SoSym 8(2) (2009)
13. Kruse, S.: On the Use of Operators for the Co-Evolution of Metamodels and Transformations. In: Int. Workshop on Models and Evolution (2011)
14. Markovic, S., Baar, T.: Refactoring OCL annotated UML class diagrams. SoSym 7(1) (2008)
15. Méndez, D., Etien, A., Muller, A., Casallas, R.: Towards Transformation Migration After Metamodel Evolution. In: Int. Workshop on Models and Evolution (2010)
16. Miller, R.J., Ioannidis, Y.E., Ramakrishnan, R.: The use of information capacity in schema integration and translation. In: VLDB. vol. 93. Morgan Kaufmann (1993)
17. Object Management Group: Meta Object Facility (MOF) 2.0 Query/View/Transformation (QVT). http://www.omg.org/spec/QVT/1.1 (2011)
18. Object Management Group: OMG Object Contraint Language (OCL). http://www.omg.org/spec/OCL/2.4 (2014)
19. Rossi, M., Brinkkemper, S.: Complexity Metrics for Systems Development Methods and Techniques. Information Systems 21(2) (1996)
20. Schönböck, J., Kusel, A., Etzlstorfer, J., Kapsammer, E., Schwinger, W., Wimmer, M., Wischenbart, M.: CARE - A Constraint-Based Approach for Re-Establishing Conformance-Relationships. In: APCCM (2014)
21. Wimmer, M., Kusel, A., Retschitzegger, W., Schönböck, J., Schwinger, W., Sánchez Cuadrado, J., Guerra, E., De Lara, J.: Reusing Model Transformations across Heterogeneous Metamodels. In: Int. Workshop on Multi-Paradigm Modeling (2011)
22. Wimmer, M., Martínez, S., Jouault, F., Cabot, J.: A Catalog of Refactoring for Model-to-Model Transformations. JOT 11(2) (2011)