Approaches to Co-Evolution of Metamodels and Models: A Survey

Regina Hebig, Djamel Eddine Khelladi, Reda Bendraou

Approaches to Co-Evolution of Metamodels and Models: A Survey

Regina Hebig, Djamel Eddine Khelladi, and Reda Bendraou,

Abstract—Modeling languages, just as all software artifacts, evolve. This poses the risk that legacy models of a company get lost, when they become incompatible with the new language version. To address this risk, a multitude of approaches for metamodel-model co-evolution were proposed in the last 10 years. However, the high number of solutions makes it difficult for practitioners to choose an appropriate approach.
In this paper, we present a survey on 31 approaches to support metamodel-model co-evolution. We introduce a taxonomy of solution techniques and classify the existing approaches. To support researchers, we discuss the state of the art, in order to better identify open issues. Furthermore, we use the results to provide a decision support for practitioners, who aim to adopt solutions from research.

Index Terms—Survey, software engineering, metamodels, models, design notations and documentation

1 Introduction

In the last decade, Model-Driven Engineering has proven to be effective in the development and maintenance of large scale and embedded systems [1], [2]. At the heart of this vision is the notion of Metamodel, a formal definition of the language that will be used to represent models of the application's domain. Raising the level of abstraction from code to models combined with the promise of better productivity and a shorter product development life-cycle, have led to the emergence of an impressive number of DSMLs (Domain Specific Modeling Languages). The use of DSMLs would increase productivity between 200% (Adaptive Cruise Control) to 750% (Home automation, phone switch features), according to [3]. Today, almost every business and industrial domain has a dedicated DSML [4] and plenty of tools are provided to help you design your own modeling language, e.g. EMF[1], OBEO Designer[2], or Metaedit[3].

However, DSMLs do not come alone. Their productivity is mainly ensured by the ecosystem that surrounds them. This includes code generators, text and graphical editors, e.g. Xtext [5] or GMF[4], rule and constraints consistency checker engines [6] [7], and many other artifacts that constitute the development chain, spanning from modeling the problem domain to the code of the future system. While this development chain is effective and efficient, its Achilles heel remains the metamodel. One can see it as the central point of a dependency graph of artifacts and tools since any modification in the metamodel may impact all the other artifacts i.e. editors, code generators, transformation rules, consistency constraints, etc. [8]. In [9] authors reported how difficult it can be to update a GMF-based graphical editor after a metamodel evolution. Our industrial partner KI[5] reported that in the automotive domain, companies, such as Renault, use DSMLs for modeling their products. However, due to the competitive nature of this industrial field, metamodels have to be extended and evolved almost every two years to include new concepts from the problem domain. This has an effect on the tedious task of not only evolving the tooling chain but also model instances. Airbus[6] and Thales[7], which were involved with us in the research projects MOVIDA[8], ANR Galaxy[9] and MeRGE[10], mentioned model instances with up to 500 000 model elements to be migrated in some cases. Another example of artifacts that may be impacted by metamodel evolution are OCL constraints. The UML metamodel[11] comes with more than 700 OCL constraints which may be impacted in case of an evolution. However, the UML metamodel evolved quite often in the last decade[12]. BPMN evolved every three to four years since it has been specified in 2004[13]. Transformation rules are also highly related to metamodel elements. Especially if you are dealing with exogenous model transformation (source and target metamodels are different). Every modification of either source or target metamodels will impact the transformation rules, sometimes hundreds of them, and requires to manually or semi-automatically migrate them [10], [11], [12].

• *R. Hebig is with the Chalmers | University of Gothenburg, Computer Science and Engineering Göteborg, Västra Götaland County, Sweden. D. Khelladi and R. Bendraou are with the Sorbonne Universités, UPMC Univ Paris 06, UMR 7606, LIP6, F-75005, Paris, France. E-mail: regina.hebig@lip6.fr, djamel.e.khelladi@gmail.com, reda.bendraou@gmail.com*

Manuscript received April 19, 2015;

1. Eclipse Modeling Framework http://www.eclipse.org/modeling/emf/
2. OBEO Designer http://www.obeodesigner.com/
3. Metaedit http://www.metacase.com/products.html
4. GMF http://eclipse.org/modeling/gmf
5. KnowledgeInside http://www.k-inside.com/web/fr
6. Airbus http://www.airbus.com/
7. Thales https://www.thalesgroup.com
8. Movida http://www.agence-nationale-recherche.fr/?Projet=ANR-08-SEGI-0011
9. ANR Galaxy http://www.galaxy.lip6.fr
10. MeRGE http://www.merge-project.eu/
11. UML standard http://www.omg.org/spec/UML/
12. For an overview of different versions of the UML see http://www.omg.org/spec/UML/
13. BPMN http://www.omg.org/spec/BPMN/

The examples given above highlight the importance of dealing with metamodels evolution especially with the wide spread of DSMLs. Some companies are already facing this problem. Through KI we know that Renault has invested money and time to build DSMLs while underestimating the cost of co-evolving the related tools and model instances every time the metamodel is impacted. We believe that in the near future, this problem, if not handled, can be an obstacle to the adoption and use of DSMLs.

Metamodels co-evolution has already been identified as a main issue in the literature [13], [14], [15]. The evolution between two versions of the same metamodel may consist of hundreds of additions, deletions, and changes of elements. For example, during the evolution of UML Class Diagrams from UML version 1.5 to 2.0 a total of 238 language elements where added, deleted, or changed [16]. For the evolution from GMF version 1.0 to 2.0, 136 changes have been identified by Herrmannsdoerfer et al. [13].

In the last decade, the literature introduced 31 approaches that focus exclusively on the co-evolution of models when the metamodel is evolved. These 31 approaches differ in the way they solve the involved tasks which are a) the identification of metamodel changes and b) the co-evolution of model instances. Thus, it is a challenge for the users to identify which approach can best fit their needs. Indeed, from the user's perspective these approaches may differ significantly. The trade-off between correctness and automation could be one way to compare them. Fully automated approaches make it easier to handle huge sets of models while it might not be possible to guarantee that the resulting models conform to the required outcome. Approaches that provide more control for the user while performing the co-evolution might be preferable when some adaptation and customizations are required. Besides, some implicit prerequisites, such as required skills or language's ownership, need to be identified and considered before deciding which would be the appropriate co-evolution approach to be used. For example, some co-evolution approaches require that specific editors are used for the evolution of the metamodel. Wrong usage of these editors, even if not visible in the resulting metamodel, can impact the correctness of the result of the co-evolution (as explained further in Section 3.2.2).

This paper aims at helping companies and users in choosing and comparing this jungle of proposed approaches for metamodel-model co-evolution (MM-M co-evolution) according to their needs and constraints. It does so by providing a survey and a detailed catalog of existing co-evolution approaches and their technical properties. Furthermore, we want to provide researchers with a framework for comparing their works and enable them to identify open research needs. Therefore, we provide a taxonomy of metamodel-model co-evolution. We present the taxonomy in Section 3. The method used to select and analyze the existing approaches for this survey is then presented in Section 2. Further, we classify existing approaches regarding this taxonomy, to compare their potential and current limitations (Section 4). In Section 5, we discuss the results and provide decision support for practitioners. In Section 6 we discuss related surveys on approaches and tools for MM-M co-evolution and conclude in Section 7.

2 Method

In this section, we describe how we identified and analyzed the approaches considered within this survey.

2.1 Exclusion Criteria for Supporting Approaches

This survey includes approaches for co-evolution that either support the identification of metamodel changes as an input for model resolution or propose model resolutions in form of more generic strategies or model specific resolutions. To make this survey feasible and keep the comparison meaningful, we excluded supporting approaches that might be used to perform co-evolution manually or that are a standard inbuilt part of co-evolution approaches today.

There are approaches for conformance checking, such as discussed by Paige et al. [17] or the approach of Iovino et al. [18]. Similarly, there are a multitude of approaches that allow to check for constraints on models, e.g. using Answer Set Programming (ASP [19]) or the Epsilon Validation Language[14]. All of these approaches can help a user to identify what elements of a model are broken with regard to a new metamodel version. However, we do not include these approaches in this survey, as they do not lead to an automated resolution support.

Naturally, all approaches that consume metamodel changes for the co-evolution rely on tools to identify these changes. These can be tools used to retrieve model differences, such as EMF compare[15], or tools to record changes online, such as the seminal work of Alan and Porres [20] or Praxis [6]. Thus, the identification of atomic changes, i.e. the creation, deletion or modification of single metamodel elements, can be considered as being well established and is included in most approaches today. In the remainder of this survey we therefore do not address these works as standalone approaches. However, we explicitly address approaches for the detection of complex changes, which consist of multiple changes, since this is still a challenge and not included in all co-evolution works.

2.2 Identification Method

The first goal was to capture all relevant approaches that support the different co-evolution tasks. Therefore, we performed a systematic literature search as described by Kitchenham [21]. We defined a set of keywords that we used for a systematic search, starting independently from three different types of entry points. The keywords that we used were: "metamodel", "evolution" or "co-evolution", "model" (optional), "adaptation" (optional) and "migration" (optional).

As a primary type of entry points we used the big online libraries: IEEE Xplore digital library, ACM digital library, Wiley Online Library, Springer Link, and ScienceDirect. As a second type of entry points we used proceedings of conferences and workshops, as well as journals that address topics of modeling or language evolution, e.g. the MODELS, the

14. EVL http://www.eclipse.org/epsilon/doc/evl/
15. EMF Compare https://www.eclipse.org/emf/compare/

ECMFA, the ME workshop, and the SoSyM journal[16]. As a third entry point we used Google Scholar, which we used to identify papers that included our keywords within the title or abstract. Further, we combined this literature search with the strategy to follow references found in already identified papers, as it was also done by Cornelissen et al. [22].

This process led to the identification of more than 130 papers. In a next step we reexamined these papers to exclude irrelevant candidates. This was done independently by two authors in order to reduce mistakes. We excluded, e.g., papers that focused on the co-evolution of metamodels with artifacts other than models. Note that papers addressing co-evolution in a more generalized form were not excluded, if they also covered models. We did not exclude such papers that introduced approaches for complex change detection in metamodels, even if these had not been motivated by co-evolution of models. Further, we did exclude papers that did not present approaches that support MM-M co-evolution. This includes overview papers, which serve as related work for this survey.

We agreed to include 56 papers, that belong to 31 different approaches, into our survey.

2.3 Identified Approaches

Table 1 summarizes the 31 identified approaches and further indicates if implementations exist for the approaches and if they are publicly available. These approaches can be divided into six groups:

Resolution Strategy Languages The first group contains transformation languages that are specialized in the specification of strategies to co-evolve (resolve) models for given metamodel changes. We identified 7 such languages, that have been introduced by Sprinkle et al. [24], Wimmer et al. [25], Di Ruscio et al. (EMFMigrate) [8], [26], [27], [28], Krause et al. [29], Levendovszky et al. (Model Change Language (MCL)) [30], [31], [32], and Rose et al. (Epsilon Flock) [33], [34], [35]. A special case was proposed by Vermolen & Visser in 2008: an approach for the automated generation of domain specific transformation languages that allow to specify metamodel changes and resolution strategies for a given metamodel [23].

Resolution Strategy Generation The second group contains 6 approaches that allow the (partial) generation of resolution strategies for a given metamodel change. These approaches have been presented by Didonet et al. [36], de Geest et al. [37], Garcés et al. [38], [39], Meyers et al. [40], Mantz & Taentzer et al. [41], [42], [43], [44], [45], [46], and Anguel et al. [47]. Note that the approach of Didonet et al. is not dedicated to metamodel-model co-evolution, but to the generation of model transformations in general [36].

Predefined Resolution Strategies The third group contains approaches that provide automation by applying predefined resolution strategies, where possible. We identified 8 of these approaches and proposals presented by Hößler et al. [48], Florez et al. (ASIMOV) [49], [50], Wachsmuth [51], Cicchetti et al. [9], [15], [52], Brand et al. [53], Gruschko et al. [14], [54], Wittern [61], and Herrmannsdoerfer et al. (COPE/Edapt) [13], [55], [56], [57], [58], [60], [73].

Resolution Strategy Learning A paper that forms its own group proposes to learn resolution strategies that are specified by users. These learned strategies should later be applied automatically, if possible, so that the need to specify new resolution strategies is reduced over time. This proposal is called CBRMig and was published in 2013 by Anguel et al. [62].

Constrained Model Search We identified 4 approaches that do not use the metamodel change, but apply a constrained-based search of valid model variants considering the original model and the new metamodel. In consequence, these approaches support and partially automate the resolution per model (and not per metamodel change). These approaches are the Cross-Layer Modeler of Demuth et al. [63], [64], GraCoT of Gomez et al. [65] , CARE of Schoenboeck et al. [68] and the approaches of Kessentini et al. [66], [67].

Identify Complex Changes 5 approaches do not focus on the resolution of changes, but are specialized on identifying complex metamodel changes: the approaches of Williams et al. [69], Vermolen et al. [70], and Langer et al. [71]. Further, we identified one paper on metamodel-transformation co-evolution that provides an approach for the identification of complex changes: CO-URE from García et al. [10]. Finally, Müller et al. present an approach that allows users to combine an automated detection of atomic changes with a manual pre-definition of complex changes [72].

2.4 Analysis Method

We analyzed the identified papers with 2 goals: providing a consistent terminology about co-evolution techniques and classifying existing approaches accordingly.

Existing terminology is inconsistent between approaches and partially has flaws. For example, Hermannsdoerfer et al. differentiate between state-based and operator-based approaches for co-evolution [74]. The two terms distinguish approaches purely on the basis of the identification of the metamodel change and do not allow for many conclusions about the applied model resolution. Furthermore, there are approaches that already use strategies for the change identification, that are neither state-based nor operator-based. Finally, the term state-based is typically associated with the need for additional analysis techniques in order to identify metamodel changes. It is thereby often overlooked that also operator-based approaches can fail in recognizing changes,

16. Used conference-, workshop-, and journal entry points: International Conference on Software Engineering (ICSE), International Conference on Model Driven Engineering Languages and Systems (MODELS), Workshop on Models and Evolution (ME), Workshop on Model-Driven Software Evolution (MoDSE), European Conference on Modelling Foundations and Applications (ECMFA), International Conference on Automated Software Engineering (ASE), Symposium on the Foundations of Software Engineering (FSE), International Conference on Software Language Engineering (SLE), International Conference on Fundamental Approaches to Software Engineering (FASE), Workshop on the Analysis of Model Transformations (AMT), International Workshop on Model Comparison in Practice (IWMCP), Workshop on Comparison and Versioning of Software Models, European Conference on Object-Oriented Programming (ECOOP), International Conference on Model Transformations (ICMT), Software and Systems Modeling Journal (SoSyM), Journal of Software: Evolution and Process (JSEP), Journal of Software Maintenance and Evolution: Research and Practice (JSME), Journal of Systems and Software (JSS), & Journal on Object Technology.

TABLE 1
Summary of identified approaches that support co-evolution of models with evolving metamodels

Group	Approach	Years	Implementation
Resolution Strategy Languages	Vermolen & Visser 2008 [23]	2008	available
	Sprinkle et al. [24]	2004	exist
	Wimmer et al. [25]	2010	available
	EMFMigrate (Di Ruscio et al.) [8], [26], [27], [28]	2011-2012	available
	Krause et al. [29]	2013	exist
	MCL (Levendovszky et al.) [30], [31], [32]	2009-2014	exist
	Epsilon Flock (Rose et al.) [33], [34], [35]	2009-2014	available
Resolution Strategy Generation	Didonet et al. [36]	2007	exist
	de Geest et al. [37]	2008	exist
	Garcés et al. [38], [39]	2008-2009	exist
	Meyers et al. [40]	2012	no
	Mantz & Taentzer et al. [41], [42], [43], [44], [45], [46]	2012-2013	no
	Anguel et al. 2014 [47]	2014	no
Predefined Resolution Strategies	Hößler et al. [48]	2005	no
	ASIMOV (Florez et al.) [49], [50]	2012-2013	exist
	Wachsmuth [51]	2007	exist
	Cicchetti et al. [9], [15], [52]	2008-2009	available
	Brand et al. [53]	2011	available
	Gruschko et al. [14], [54]	2007	exist
	COPE/Edapt (Herrmannsdoerfer et al.) [13], [55], [56], [57], [58], [59], [60]	2008-2011	available
	Wittern [61]	2013	exist
Resolution Strategy Learning	CBRMig (Anguel et al. 2013) [62]	2013	no
Constrained Model Search	Cross-Layer Modeler (Demuth et al.) [63], [64]	2013-2015	exist
	GraCoT (Gomez et al.) [65]	2014	available
	Kessentini et al. [66], [67]	2015-2016	exist
	CARE (Schoenboeck et al.) [68]	2014	exist
Identify Complex Changes	Williams et al. [69]	2012	exist
	Vermolen et al. 2012 [70]	2012	available
	Langer et al. [71]	2013	available
	CO-URE (García et al.) [10]	2013	available
	Müller et al. [72]	2014	exist

when users are not using the tool correctly. Thus, additional analysis techniques might be combined with both forms of approaches.

While this is just one example, it already shows that the current terminology is not sufficient to cover the state of the art and to enable the identification of future potentials. Therefore, our target is to unify and expand the existing terminology, so that it enables a comparison of existing approaches.

The second part of the analysis focusses on classifying existing approaches accordingly.

2.4.1 *Creating a Taxonomy*

For the creation of the taxonomy we considered the 56 papers that belong to the identified approaches. In a first step, we systematically read the papers and collected information about proposed and applied techniques. By doing so, we focused on the different tasks covered and the strategies and features provided. In a next step, we discussed and reflected upon the collected tasks, features, and strategies, focussing on the following questions:

- Do tasks or parts of those tasks occur in some form in all approaches or just in specific ones? If the latter: can we characterize the circumstances for needing these tasks using identified features and strategies?
- Can features or strategies be generalized?
- Do different identified features/strategies actually reflect variants of the same feature or strategy? What are differentiating sub-features? Do existing sub-features cover the frame of possibilities or can they be enhanced?
- Do features and strategies that were found together need to occur together necessarily or could they be reassembled with alternative features and strategies?

As a result, we gained a unified and overhauled taxonomy for the field, which is strong enough to cover tasks, features, and strategies of existing approaches as well as potential variations or approaches that provide alternative strategies.

2.4.2 *Classification*

We analyzed and compared the approaches regarding the question, which of the co-evolution tasks and features are supported. Therefore, we thoroughly studied the identified papers of each approach to systematically evaluate how change collection, change identification, and resolution on the model are addressed and which features described in the taxonomy in Section 3 are manifested or supported. For each approach we also carefully investigated the papers found and the websites referenced for hints on whether an implementation of the approach has been created. Furthermore, we tried to find if implementations are accessible. When no statement about existing prototypes, evaluations of the running system or implementations was made, we classified this as "no implementation". If a link for downloading or accessing the implementation existed we classified this as available. Please not that the maturity, e.g. fault tolerance or platform independence, of the implementation was not assessed. To ensure quality, the results were double checked by two of the authors.

2.5 Validity Threats

The first threat to the validity of a literature survey is the completeness of the search results. To reduce the risk of missing relevant approaches we combined three different types of media search. During the selection of papers, we needed to address the risk that relevant papers are excluded. Therefore, this selection was performed by two of the authors independently. Finally, there is always the threat of making mistakes when analyzing and classifying the approaches. Similarly, as for the paper selection, we addressed this threat by ensuring that each paper was read by two of the authors, who agreed on the final classification shown in this paper. We consider the remaining risk of missing or misclassifying approaches minimal.

3 A Taxonomy

In this section, we introduce a new taxonomy of metamodel changes and the three steps of metamodel-model co-evolution. As described in Section 2.4.1 the taxonomy is the result of our survey and our efforts to provide a consistent terminology that captures existing concepts. These steps are (1) *collecting* and (2) *identifying metamodel changes*, as well as (3) *resolving the models*. We consider the techniques and solutions involved in these tasks and start with providing our definitions for the basic terminology on metamodel changes.

3.1 Metamodel Changes

The evolution of two versions of a metamodel consists of multiple *metamodel changes*. Basically, all elements of a metamodel might be modified or deleted. Similarly, all possible types of metamodel elements might be added.

3.1.1 Introduction to atomic and complex changes

The term atomic change is used to refer to a change, i.e. addition, deletion, or modification, of a single metamodel element. Examples are the addition of a new empty class, the deletion of an attribute from a class, the change of the name of a class, or the change of a property of a class from abstract to non-abstract. In contrast, complex changes consist of multiple atomic changes, including the additional knowledge about the interrelation between these atomic changes.

An example for a complex change is *move property*, where a property is deleted from one and added to another (related) class. The knowledge about that relation between both atomic changes, *delete property* and *add property*, can be used during the automated adaptation of models: by moving the properties' values from the deleted to the added instance. Without knowledge about the complex change, the *delete property* could be solved by removing the instance values, while no knowledge on how to instantiate the newly *added property* would be available. Thus, instance values would get lost. The example shows that complex changes can enable automated co-evolution.

3.1.2 Definitions

So far we introduced a rough picture of the ideas behind metamodel changes. In the following, we present a more specific terminology by providing our definitions of the involved concepts:

3.1.2.1 ***Atomic change type***: An *atomic change type* denotes a set of changes that can happen to a metamodel, defined by

- An editing action, i.e. delete, add, or modify.
- A meta-metamodel element, i.e. a type of elements that might occur in a metamodel, such as meta-property or meta-class (for simplicity referred to as property and class in the remainder of this paper). For example, the atomic change type "add property to an abstract class" includes the editing operation "add" and the meta-metamodel element "property".
- Conditions defined on the metamodel, such as constraints on the atomic change's metamodel element (e.g. is a class abstract or not, is an attribute mandatory or not), relations to other elements (e.g. is a class referenced by another class) and properties of directly or indirectly related elements. E.g., change type "add property to an abstract class" includes the condition that the class to which the "property" is added is abstract. It is possible that an atomic change type defines no conditions.

3.1.2.2 ***Atomic change***: An *atomic change* is the application of the *atomic change type*'s *editing operation* on a metamodel element that conforms to the *atomic change type*'s *meta-metamodel element*, where the *atomic change type*'s *conditions* are met.

3.1.2.3 ***Complex change type***: A *complex change type* denotes a set of changes that can occur to a metamodel, defined by a set of atomic change types and constraints on the relations between the respective atomic changes. Included atomic change types can have a fixed or flexible multiplicity. An example for a *complex change type* is "pull property to an abstract superclass". Included atomic change types are "add property to an abstract class" (occurring once) and "delete property" (which occurs one or more times according to the number of subclasses). The relations between the atomic changes are as follows: Each property in the involved atomic changes "delete property" is deleted from a class that is a subclass of the class where the property from the atomic change "add property to an abstract class" is added to. All involved properties have the same name and type.

Note that complex change types can also include other complex change types. We say that a complex change type $icct$ is included in a complex change type cct, when the set of atomic changes and relations between atomic changes of $icct$ is a subset of the set of atomic changes and relations between atomic changes of cct.

3.1.2.4 ***Complex change***: A *complex change* consists of a set of *atomic changes* that conform in number and relation to the specification of the *complex change type*.

Examples for changes and change types:

Atomic change type: add property to an abstract class
Atomic change: add property "id:String" to abstract class "user"

Complex change type: pull property to an abstract superclass
Complex change: pull property "id:String" to abstract superclass "user" from subclasses "basic-user" and "premium-user"

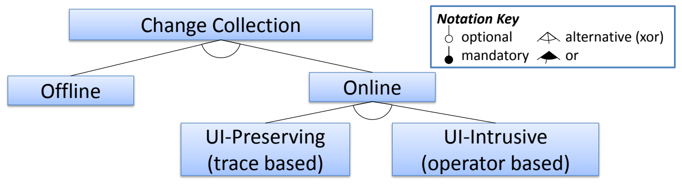

Fig. 1. Feature model for collecting metamodel changes

3.2 Collecting Metamodel Changes

Collecting metamodel changes can happen offline or online (see Figure 1).

3.2.1 Collecting offline changes

Offline change collection happens after the metamodel change is performed. The collected change information consists of the two metamodel versions only. Offline change collection can happen independently of the modeling tool that is used to edit the metamodel. However, changes are not directly collected, but need to be identified in the next step. In reference to the fact that this identification is often done based on a comparison of the before and after state of a metamodel (see below in Section 3.3), approaches that are based on offline changes are sometimes referred to as "state-based approaches" [74].

3.2.2 Collecting online changes

Online change collection happens while the changes on the metamodel are performed. Here, the collected information is a list of occurred changes. In general, online change collection has the advantage that the captured changes are chronologically ordered. However, online change collection is only possible, if the organization that changes the metamodel 1) applies tools that allow for online change collection and 2) provides the outcome of the change collection to the companies that are responsible for co-evolving existing models.

Furthermore, we differentiated between *UI-preserving* and *UI-intrusive* approaches.

UI-preserving approaches do not affect the front-end of the editor that is used for changing the metamodel. Thus, they do not affect the way the modeler changes the metamodel. Collecting changes is done by observing/tracing the actions performed in the editor, which leads to a list of ordered atomic changes.

UI-intrusive approaches affect the editors front-end. In all corresponding approaches that can be found in the literature so far, this change of the editor happens by adding new editing actions in form of operators, which are defined by complex or atomic metamodel change. That is why these approaches are often referred to as "operator-based approaches" [74]. As a result, UI-intrusive change collection approaches can record an ordered list of atomic and complex changes as well.

However, the user who applies changes to the metamodel has to use and be familiar with the operators, which can be difficult since approaches can provide up to 60 or more different operators. Furthermore, since atomic operations will not be disabled, users might use workarounds and apply complex changes by using multiple atomic operations instead of less familiar complex operators. Thus, there is no guarantee that the additional operators are always used and, with it, that the set of collected complex changes is complete.

A special detail about operators is that they enforce to make decisions that concern the resolution of the models at the moment of the metamodel change. With it, resolution decisions are moved explicitly from the person responsible for the models to the person who changes the metamodel. These decisions are made via two mechanisms. First, this is the choice of the operator. The strategy for the resolution of models is coupled to the operator and there can be alternative operators for the same metamodel change. Second, some operators consume additional information as input. We call them *outreaching operators*. There, *non-changed metamodel elements* might be consumed as additional input to specify model elements that play a role during change resolution. Furthermore, additional *mapping information*, e.g. between different parameters, might be consumed to configure the change resolution strategy. An example is the operator "Replace Class", provided by the Edapt Eclipse Project[17]. It is an implementation for the metamodel change of deleting a class and its properties. During model resolution of "Replace Class", instances of the deleted class are not deleted, but preserved by transforming them to instances of another class. For that model resolution, the operator consumes additional pieces of information, which are 1) the class of the metamodel that will be used as new meta-class for the instances of the deleted class (*non-changed metamodel element*) and 2) a mapping of properties from the deleted class to properties of the substituting meta-class, to indicate how the class instances are migrated (*mapping information*).

3.3 Identifying Metamodel Changes

The goal of the change identification is to gain a precise list of atomic and complex changes and their types that had been applied to the metamodel. The more precise the detection of complex changes is, the better the automation and correctness of the co-evolution will be. The tasks involved in change identification depend on the kind and precision of the information provided by collecting changes. Figure 2 summarizes the different variations of change identification.

3.3.1 Atomic change identification

Atomic changes need to be identified in case of *offline change collection*, where change information is only provided in the form of a pair of metamodel versions. The identification of atomic changes happens on the basis of a diff of the two metamodels. However, in the general case, the precision of the results is not necessarily optimal for two reasons. First, no information on the chronological order of the application of the atomic changes is available. Second, it is possible that effects are hidden, when multiple changes are applied. For example, a change "*move property* y *from class* a *to class* b" that is followed by a change "*change property name* y to x" will occur as a group of two atomic changes only: "deletion of y from a" and "addition of x to b". The atomic changes "add y to b" and "rename y to x" are hidden. Note, that in

17. see http://www.eclipse.org/edapt/operations.php

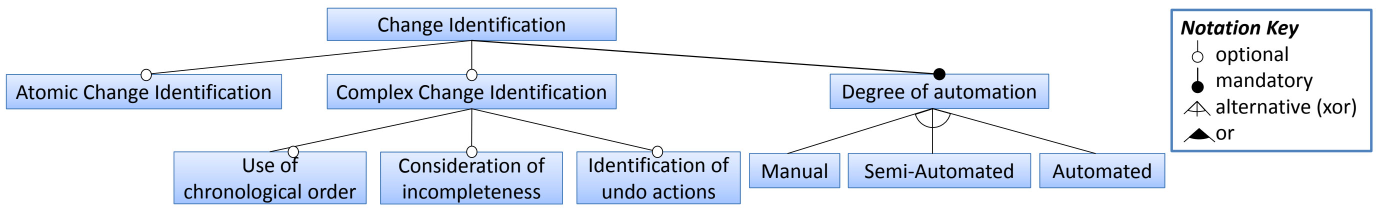
Change Identification
Atomic Change Identification
Complex Change Identification
Degree of automation
Use of chronological order
Consideration of incompleteness
Identification of undo actions
Manual
Semi-Automated
Automated
Notation Key
optional
mandatory
alternative (xor)
or

Fig. 2. Feature model for identifying metamodel changes

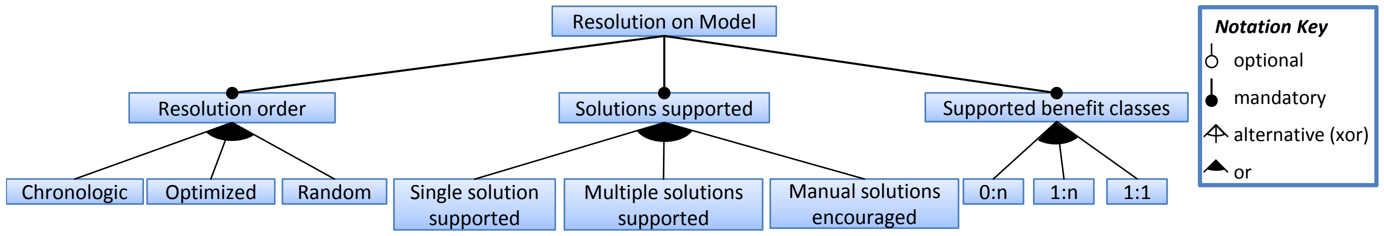

Fig. 3. Feature model for resolving models

special cases the used tool might capture enough metadata to compensate these flaws.

3.3.2 Complex change identification

Complex changes help to increase the amount of automation during model co-evolution, as explained above in Section 3.1. The identification of complex changes happens on the basis of a list of atomic changes and optionally a list of collected complex changes. The goal is to identify sets of changes in these lists that together form a complex change.

Challengingly, the same list of atomic changes might allow for alternative lists of complex changes. For example, two changes “add property” and “delete property” might be interpreted as “move property” or “pull property” (from subclass to metaclass). Thus, during complex change identification it is not sufficient to only search for combinations of changes that might form a complex change. Rather, it needs to be estimated how probable the correctness of the alternative identified complex changes is.

Use of chronological order: The *chronological distance* of atomic changes might be considered as a factor when estimating the probability if changes belong to the same complex change or not. Note that information about the chronological order is only available when *online change collection* was used.

Consideration of incompleteness: Further, some algorithms that require an *atomic change identification* are able to take into account that the identified atomic changes can be incomplete due to hidden effects, as explained in Section 3.3.1.

Identification of undo actions: Approaches that are based on a complete list of atomic changes need to deal with a different problem: change traces also include actions that have been canceled. For example, when the modeler accidentally deletes a property and adds it again to correct this action, this might be counted as two changes, namely “delete property” and “add property”. However, just applying co-evolution to both changes can be harmful, since co-evolution of the deletion leads to loss of information in the models. For example, when the accidental deletion and re-addition concern an optional property x, the application of the co-evolution on models would mean to first delete instances of that property, including their values. The re-addition of the property to the metamodel would not lead to any automated action on the models, since the property is optional. Thus, while the user who changed the metamodel performed an undo, property values get lost within the models. Therefore, approaches that are based on a complete list of atomic changes need to identify and filter changes that belong to canceled/undone actions.

3.3.3 Degree of automation

Change identification can be done manually, semi-automated, or fully automated. While a full automation would be ideal, it is hardly possible to exclude imprecision. The following example illustrates the basic problem of change identification: Within the metamodel a composite pattern [75] might be used. During the change, it might be decided to pull a property from the composite to the component (so that it is available in instances of both, composite and leaf). In terms of atomic changes, this *pull property* occurs as *delete property* and *add property*. However, during complex change identification the algorithm might not detect a *pull property*, but instead a *move property*, since the property might as well be moved via the children relation of the component and the composite. To cope with such problems, semi-automated approaches rely on final user decisions.

3.4 Resolving Models

Based on the identified changes, the actual task of model co-evolution can be performed. Two central concepts for this are *“resolution technique”* and *“resolution strategy”*.

A *resolution strategy* specifies how a model should be resolved for a given change.

In contrast, a *resolution technique* is an overall approach to create and apply resolution strategies. *Resolution techniques* differ in the order of change resolution, the way users can control the enactment of resolution strategies, and the ratio between manual effort and automation (see Figure 3).

3.4.1 Resolution order

Metamodel evolution often includes changes that can be more or less independent of each other. The resolution strategies for these changes might be applied in a different order to the models. Changes might be addressed in exactly the *chronological order* in which they occurred. However, in case of *offline change collection*, information on this order is not available. Here, a resolution technique might either apply a *random order* or retrieve an order that is *optimal* for the resolution, e.g. by reducing the manual effort. For example, a metamodel might be evolved by, first, adding a mandatory property to a class and, second, deleting one of the subclasses of this class. Adding property instances might require human decisions on each added instance value. It would be a waste of effort to first add property values to instances of the deleted subclass, since these instances will be deleted during resolution of the second change. In this case, the chronological order would not be optimal for the resolution.

3.4.2 Supported Benefit Classes

A relevant factor, besides the discussed strategies to allow user control, is the ratio between the effort a user invests to specify a resolution strategy and the set of models that can be addressed with that strategy. In this survey, we consider this ratio in the simplified form of three *"benefit classes"*[18]:

1:1 A manual decision, such as configuring or selecting a resolution strategy, is required per model.

1:n One manual decision, such as the specification of a project specific resolution strategy, is required. This decision can be applied to all models to be migrated (e.g. within a company).

0:n No manual decision has to be made to resolve all models to be migrated. This happens during reuse of existing rules that are independent of concrete metamodels.

Which benefit classes can be supported by an approach depends on the implemented techniques for solution support. Therefore, we will later (in Section 4) investigate in detail how the different approaches support these benefit classes.

3.4.3 Solutions supported

Per change type, a resolution technique might support one or multiple resolution strategies for a metamodel change. A strategy is applicable to all changes conforming to its change type.

Single solution supported: There are two situations, where approaches in literature support only a single solution for a metamodel change. First, this often happens when approaches *predefine* a resolution strategy that does not include human intervention. Alternatively, approaches might (partially) generate a solution. Thus, single solutions are only supported in benefit class 0:n.

As a general advantage, single solution approaches enable a full automation of the resolution. However, the resolution results are not guaranteed to be the one desired for the concrete models at hand. For example, when a model is used as input of a code generation, automatically added instances with default values (as it is often done when attributes are added to metaclasses) can become problematic.

Multiple solutions supported: To give users a better control over the result, some approaches support multiple solutions per change. The first strategy is to allow users to freely *specify new resolution strategies*, which then might be applied to single or multiple models, e.g. that belong to the same project or company. Second, a set of alternative resolution strategies might be presented to the user to *select* from. Third, a provided resolution strategy might require a configuration of specified parameters. Multiple solutions can be supported in all benefit classes.

The gained control comes at the cost that user interaction is required during the resolution (leading to a reduced degree of automation). A way to improve the degree of automation is to make the *user interaction* optional, allowing to reduce the number of user decisions, without abandoning user control. This can be done by providing a solution strategy that can optionally be adapted. Finally, there is the special case of operator based approaches, especially the ones that work with *outreaching operators*. Here, the decision about the resolution strategy as well as configurations are moved to the moment where the metamodel change is applied. By providing different operators, multiple resolution strategies per metamodel change can be supported. After the metamodel change the model resolution requires no further decisions.

Manual solutions encouraged: Besides the two forms mentioned above, some approaches do not present single or multiple solutions, but encourage and allow for a manual creation of a resolution for a model. This approach is placed in benefit class 1:1.

18. Please note that the degree of automation is in literature often addressed with a classification of change types into "non-breaking", "breaking and resolvable" and "breaking and non-resolvable" change types. However, during this survey and a related one [76] we found some serious flaws in that classification and its application that made us reconsider its use.

The distinction between breaking and non-breaking is meaningful and clear. Nonetheless, there seem to be common sources of error, when assessing whether a change type is breaking or non-breaking. For example, in [76] we found that misclassifications seem to stem from situations where special cases of change types are not considered. In consequence, we found change types that are treated by some approaches as non-breaking while other approaches treated them as breaking and provided resolution strategies [76] .

Our skepticism with the distinction between resolvable and non-resolvable goes further. Given a precise enough restriction of the domain, each change type might be addressed with reusable and automated resolution strategies. For example, Cope/Edapt (https://www.eclipse.org/edapt/operations.php) [73] offers a configurable and automated operator that allows for resolution of the addition of string attributes to a metamodel (Split String Attribute), including mandatory string attributes. However, this is in contrast to most literature that considers the addition of mandatory attributes to be non-resolvable. For example, Burger and Gruschko [77] classify the addition of obligatory attributes as non-resolvable.

In consequence, we decided to not use this terminology and instead substitute it with the more precise notion of benefit classes, which concerns the resolution strategies instead of the change types.

4 Classification of Approaches

In this Section, we summarize and compare how the approaches of the different families support change collection, change identification and model resolution. Therefore, the taxonomy presented in Section 3 is applied.

4.1 Change Collection

As summarized in Column 3 in Table 2, 22 approaches address the challenge of change collection. The other 9 approaches are based on the assumption that a list of changes is given. Most of the approaches work with offline change collection. The exception are 5 online change collection approaches: The Cross-Layer Modeler uses a UI-preserving logging to get information about atomic changes [63], [64]. Further, the Cross-Layer Modeler relies on the approach of Iovino et al. to identify the impact of the metamodel change on the models [18]. Further 4 approaches (Krause et al. [29], ASIMOV [49], [50], Wittern [61], and COPE/Edapt [55]- [60]) are operator-based and, thus, UI-intrusive. For COPE/Edapt also an offline collection is possible [60].

Finally, one approach (Didonet et al. [36]) performs an offline matching of metamodels, without identifying changes subsequently.

4.2 Change Identification

In the following, existing proposals for change identification are discussed. Columns 4-6 in Table 2 summarize the results. Note that for approaches that do not support change identification the fields are empty (marked with "-").

One approach that is based on an offline collection of metamodels, but does not provide mechanisms for change identification, is the one of Didonet et al. [36]. In contrast to other approaches, Didonet et al. use the offline comparison of metamodels to identify similarities (and not differences). These similarities are used as the basis for the generation of the transformation.

4.2.1 Atomic change identification

Most (14) of the 15 approaches that allow for offline change collection also enable atomic change detection ([9], [10], [28], [32], [37], [39], [47], [53], [56], [62], [69], [70], [71], [72], see Table 2). To achieve that, most approaches (12 of 14) propose a calculation of the *difference* between the original and the evolved version of the metamodel, e.g. by using EMF Compare[19] (as in EMFMigrate [8], [26], [27], [28], Anguel et al. [47], and CBRMig [62]). Two approaches (Williams et al. [69] and MCL [30], [31], [32]) join atomic with complex change identification (as explained below).

4.2.2 Complex change identification with offline collection

We found 7 approaches that deal with automated or semi-automated identification of complex changes. In contrast, the approaches of Müller et al. [72] and COPE/Edapt, explicitly support the user in manually identifying complex changes. Many of the remaining approaches use complex changes, but do not specify appropriate identification mechanisms. In one case, it was not possible to determine whether complex changes are addressed: Brand et al. provide an explicit list of "detectable changes" that only includes atomic changes, but claim in their evaluation section that two complex changes had been identified [53].

Identification techniques: Complex change identification is mostly based on *pattern detection*, using patterns to define which constellations of atomic changes conform to a complex one. Patterns can also occur in form of "predicates" (CO-URE [10]) or "heuristics" (Garcés et al. [39]).

Williams et al. propose a *search-based technique* ([78]) to identify atomic and complex changes. The algorithm looks for different paths of possible changes to identify those paths that can lead from the initial to the new metamodel version [69].

Finally, two approaches from the family of *resolution strategy languages* do not provide extra mechanism to identify complex changes, but reuse the transformation language's or formalism's mechanism for rule application: the underlying graph formalism in MCL [32] and ATL in EMFMigrate [26]. While this approach seems natural, it reduces the control, since these reused mechanisms include a) no concepts for prioritization between contradicting complex changes and b) do not support manual choices.

Prioritization and correction: Three of the approaches address the need to prioritize between contradicting interpretations of complex changes in a list of atomic changes: Williams et al. propose the use of fitness functions [69], CO-URE prioritizes detected changes by size [10] and Vermolen et al. request a final decision from the user [70]. In contrast to a prioritization or choice between detected changes, Garcés et al. and COPE/Edapt allow for manual correction and enrichment of the set of detected complex changes ([39], [60]). Thus, only 4 approaches for complex change identification rely on user decisions. These are surprisingly few, considering the associated challenges and uncertainties discussed in Section 3.3.3.

Consideration of incompleteness: Only two approaches consider a possible *incompleteness* of the list of atomic changes derived by model differencing: the approaches presented by Vermolen et al. [70] and Garcés et al. [39] allow for detection of complex changes, even if parts of their effects are hidden by other changes.

4.2.3 Complex change identification with online collection

From the 5 approaches that are based on online change collection, only one addresses the identification of additional complex changes: COPE/Edapt [60] allow the user to manually correct the change lists. An explicit support for undo operations is given to relieve the user of manually searching these undos in the list of atomic changes (see discussion in Section 3.3.2). However, there is so far no approach for complex change detection that benefits from knowledge about the chronological order of changes.

4.3 Resolving Models

In this Section we summarize, which resolution orders are used and how benefit classes are supported by the different co-evolution approaches. Table 3 summarizes the results. Note, that also in this table some approaches have empty fields only (indicated by "-"). This is the case when approaches do not support the resolution of models directly.

19. EMF compare http://www.eclipse.org/emf/compare/

TABLE 2
Summary of change collection and change identification of the different approaches.

Group	Approach	Change Collection	Change Identification		
			Atomic C.I.	Complex Change Identification	Degree of Automation
Resolution Strategy Languages	Vermolen & Visser 2008	-	-	-	-
	Sprinkle et al.	-	-	-	-
	Wimmer et al.	-	-	-	-
	EMFMigrate	offline	✓ MM-Diff	✓ *reuse of transformation technology's application decisions* • non-completeness NOT considered	automated
	Krause et al.	online (UI-intrusive)	-	-	-
	MCL	offline	✓ *reuse of transformation technology's application decisions* • non-completeness NOT considered		automated
	Epsilon Flock	-	-	-	-
Resolution Strategy Generation	Didonet et al.	offline	-	-	-
	de Geest et al.	offline	✓ MM-Diff	-	semi-automated
	Garcés et al.	offline	✓ MM-Diff	✓ Pattern Detection • non-completeness NOT considered	semi-automated
	Meyers et al.	-	-	-	-
	Mantz & Taentzer et al.	-	-	-	-
	Anguel et al. 2014	offline	✓ MM-Diff	-	automated
Predefined Resolution Strategies	Hößler et al.	-	-	-	-
	ASIMOV	online (UI-intrusive)	-	-	-
	Wachsmuth	-	-	-	-
	Cicchetti et al.	offline	✓ MM-Diff	-	automated
	Brand et al.	offline	✓ MM-Diff	?	automated
	Gruschko et al.	-	-	-	-
	COPE/Edapt	online (UI-intrusive) &	-	✓ optional manual unification of atomic changes • input is complete • input is ordered • explicit change undo supported	manual
		offline	✓ MM-Diff	✓ manual assignment of complex changes, • non-completeness NOT considered	atomic: automated, complex: manual
	Wittern	online (UI-intrusive)	-	-	-
Resolution Strategy Learning	CBRMig	offline	✓ MM-Diff	-	automated
Constrained Model Search	Cross-Layer Modeler	online (UI-preserving)	-	-	-
	CARE	offline	-	-	-
	Kessentini et al.	offline	-	-	-
	GraCoT	offline	-	-	-
Identify Complex Changes	Williams et al.	offline	✓ *Search-Based Technique* • non-completeness considered • prioritization: fitness function		automated
	Vermolen et al. 2012	offline	✓ MM-Diff	✓ Pattern Detection • non-completeness considered • prioritization: by user decision	semi-automated
	Langer et al.	offline	✓ MM-Diff	✓ Pattern Detection • non-completeness NOT considered	automated
	CO-URE	offline	✓ MM-Diff	✓ Pattern Detection • non-completeness NOT considered • prioritization: 'biggest' complex changes	automated
	Müller et al.	offline	✓ MM-Diff	✓ manual assignment of complex changes, • non-completeness NOT considered	atomic: automated, complex: manual

TABLE 3
Summary of model resolution support of the different approaches (n.d. = resolution order is not described; - = no hint on support identified).

Group	Approach	Resolution Order	0:n		1:n	1:1	
			single solution	multiple solutions	multiple solutions	multiple solutions	manual solutions
Resolution Strategy Languages	Vermolen & Visser 2008	n.d.	-	-	strategy specification	-	-
	Sprinkle et al.	optimized	-	-	strategy specification	-	-
	Wimmer et al.	n.d.	clean up only	-	strategy specification	-	-
	EMFMigrate	n.d.	-	-	strategy specification, free strategy manipulation	-	-
	Krause et al.	chronological	-	-	strategy specification	-	-
	MCL	n.d.	special cases (not explained)	-	strategy specification	-	-
	Epsilon Flock	n.d.	clean up only	-	strategy specification	-	-
Resolution Strategy Generation	Didonet et al.	all at once	-	-	free strategy manipulation	-	-
	de Geest et al.	n.d.	generated	-	free strategy manipulation	-	manual user action
	Garcés et al.	n.d.	generated	-	-	-	-
	Meyers et al.	chronological	-	default strategy (optional check)	strategy specification, free strategy manipulation	strategy configuration	-
	Mantz & Taentzer et al.	n.d.	post-processing for multiplicity changes	default strategy (optional check)	free strategy manipulation	-	-
	Anguel et al. 2014	n.d.	-	default strategy (optional check)	-	strategy configuration	-
Predefined Resolution Strategies	Hößler et al.	n.d.	predefined strategy	-	-	-	-
	ASIMOV	partly chronological, partly n.s.	-	predefined strategy (operator selection)	-	strategy specification	-
	Wachsmuth	n.d.	predefined strategy	-	strategy configuration	-	-
	Cicchetti et al.	optimized	predefined strategy	-	strategy configuration	strategy configuration	-
	Brand et al.	n.d.	predefined strategy	-	strategy configuration	-	manual user action
	Gruschko et al.	n.d.	predefined strategy	-	strategy specification, strategy configuration	-	-
	COPE/Edapt	online: chronological, offline: random	-	predefined strategy (operator selection)	strategy specification	strategy configuration	-
	Wittern	chronological	-	predefined strategy (operator selection)	strategy specification	-	-
Resolution Strategy Learning	CBRMig	n.d.	automatic selection	-	strategy specification, free strategy manipulation	- (mentioned possibility of strategy configuration)	-
Constrained Model Search	Cross-Layer Modeler	n.d.	-	-	-	strategy selection	identification of action need
	CARE	all at once	-	-	-	strategy selection	-
	Kessentini et al.	all at once	-	-	-	strategy selection	-
	GraCoT	chronological	-	-	-	strategy selection	-
Identify Complex Changes	Williams et al.	-	-	-	-	-	-
	Vermolen et al. 2012	-	-	-	-	-	-
	Langer et al.	-	-	-	-	-	-
	CO-URE	-	-	-	-	-	-
	Müller et al.	-	-	-	-	-	-

4.3.1 *Resolution Order*

Most approaches do not specify a resolution order explicitly. For example, the approach of Mantz & Taentzer et al. takes the metamodel change specifications directly as input [46]. Nonetheless, it is not specified that this input needs to be provided in chronological order.

However, some approaches are based on a chronological order: Krause et al. [29], Wittern [61], Meyers et al. [40], and GraCoT [65]. Further, ASIMOV seems to apply a chronological order, at least for the part of the changes that are resolved automatically. COPE/Edapt implies a chronological order when changes were collected online. For changes that had been collected offline a user specified order is applied [60]. In contrast, Sprinkle et al. propose the use of an optimized order [24] and Cicchetti et al. present an approach to analyze the dependencies between changes to retrieve an order for the resolution [15].

Finally, three approaches, CARE [68] and the ones by Kessentini et al. [66], [67] and Didonet et al. [36], handle all metamodel changes at once.

4.3.2 *Benefit class 0:n*

Model resolution happens in benefit class *0:n*, when no manual decisions are required for deriving a resolution strategy for a metamodel change.

Approaches in the family of *resolution strategy languages* usually do not support solutions in the benefit class 0:n. However, 2 approaches provide an automated *clean up* by automatically deleting model elements that are no longer conform to the new metamodel version: Wimmer et al. [25] and Epsilon Flock [33], [34], [35]. In both cases the clean up is a result of the chosen resolution technique, where model elements that should remain in the model need to be handled explicitly. As a result, non-conform elements are implicitly dropped. In addition, for MCL it is claimed, without providing further explanations, that special simple cases of atomic changes are solved automatically [32].

Geest et al. [37] and Garcés et al. [38], [39] are the only *resolution strategy generation* approaches that propose to generate a single supported migration strategy. All other approaches for *resolution strategy generation* (Meyers et al. [40], Mantz & Taentzer et al. [46], and Anguel et al. 2014 [47]) allow for the benefit class 0:n, by generating default strategies that can optionally be checked and adapted. For the special case of changing multiplicities of references in the metamodel Mantz & Taentzer et al. provide an approach that supports a single solution only. They use an instance generation technique to create missing reference instances during a post processing step [46].

Non-operator based approaches with *predefined resolution strategies*, support benefit class 0:n with single solutions per metamodel change or operator ("*predefined strategy*"). Due to the fact that a metamodel change can be supported by multiple operators, operator-based approaches with *predefined resolution strategies*, support benefit class 0:n with multiple solutions. Since the metamodel change includes the choice of the operators, the actual co-evolution does not require choices.

Finally, the resolution strategy learning approach CBR-Mig [62] supports situations where a single solution is automatically applied without user interaction. However, not a predefined strategy is used, but a learned solution that is selected from a list of previously applied solutions.

4.3.3 *Benefit class 1:n*

A resolution happens in benefit class *1:n*, when after a metamodel change a human decision is required to derive or configure a resolution strategy that can then be used to resolve all models to be co-evolved. We found approaches using three different strategies to support this benefit class.

Strategy Specification: Free specification of migration strategies is supported by all *resolution strategy languages* [23], [24], [25], [28], [29], [32], [35] as well as by individual approaches of other families: the *resolution strategy generation* approach of Meyers et al. [40], the *predefined resolution strategy* approaches of Gruschko et al. [14] and Wittern [61] as well as COPE/Edapt [60] and CBRMig [62].

Free Strategy Manipulation: Four approaches for *resolution strategy generation* allow for a free manipulation of the generated default strategies: de Geest et al. [37], Meyers et al. [40], Mantz & Taentzer et al. [46], and Didonet et al. [36]. While the first three approaches generate these strategies based on metamodel changes, the later approach uses information about metamodel similarities to reach the same goal. Also in the CBRMig approach some of the automatically selected default strategies are to be manipulated, when only a suboptimal match for a solution is found [62]. The same is true for EMFMigrate [26]. Here, the manipulated resolution strategy is manually selected from a library.

Strategy Configuration: Finally, some approaches with *predefined resolution strategies* rely on restricted manipulations of the resolution strategies: there, predefined parameters need to be configured. This strategy configuration per metamodel evolution is supported by Wachsmuth [51], Cicchetti et al. [15], Brand et al. [53], and Gruschko et al. [14].

4.3.4 *Benefit class 1:1*

A resolution is in benefit class *1:1*, when human decisions are required for the resolution or each model.

Four approaches for *resolution strategy generation* and with *predefined resolution strategies* allow a *strategy configuration* that is valid per model: Meyers et al. [40], Anguel et al. 2014 [47], Cicchetti et al. [15], and COPE/Edapt [60]. The CBRMig approach [62] mentions the possibility to introduce such a model specific *strategy configuration*.

Further, ASIMOV relies on *strategy specification* per model [50]. In contrast, Brand et al. and de Geest et al. allow *manual changes* on the models when no other solution is possible [37], [53].

Finally, the approaches in the group "Constrained Model Search" are specialized on supporting users in manual migration. By reasoning on model constraints, alternative versions for a model are generated. In the approaches of this group, users select the right generated version. The "constrained model search" approaches differ slightly in the source of the constraints (ranging from the model-metamodel conformance relationship to the reuse of existing constraints as in Demuth et al. [63], [64]) and the applied algorithms to identify usable models. For example, the Cross-Layer Modeler proposes model resolutions with the help of a reasoning mechanism that calculates variations of the

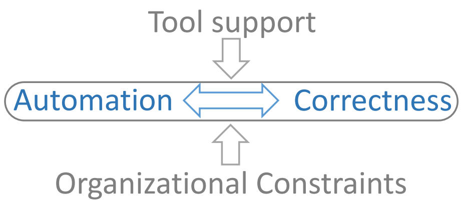

Fig. 4. Dimensions of the decision.

original models, so that they conform to the constraints. In contrast, CARE calculates permutations of possible changes and later excludes candidates that do not conform to the constraints. In addition, in the Cross-Layer Modeler constraints are used as automated support for indicating to the user the model parts that need to be adapted.

5 Decision Support and Discussion

In this section, we aggregate the findings of the survey. Our goal is to assist users in choosing between the existing approaches. We base this discussion on the trade-off between desired correctness of the resulting models and automation that we discussed in the introduction (illustrated in Figure 4). Besides this, trade-off decisions should also take tool support into account. Furthermore, there might be additional constrains given by the organizational context, e.g. missing control over the complete co-evolution process (e.g. in case of evolving standards) or missing expertise of the employees applying the co-evolution.

This decision support is a help for the user to navigate between the different characteristics considered in the classification above. Based on the consequences or challenges of the characteristics presented in the taxonomy, we identified questions that potential users should consider. These questions concern the required result correctness, the required degree of automation, the ability to expend effort on the approach adoption and application, as well as the organizational constraints that might impact the applicability of an approach. For each question, we propose criteria that can be used to identify approaches that are good candidates.

Since some approaches focus on either resolution or change identification only, we first of all summarize in Figure 5 which approaches can be used alone and which approaches should be combined. Please note, in most cases the following holds: if an approach can be used standalone, their change identification part can in theory be substituted with dedicated approaches for change identification (even thought it might require to implement an integration). However, for some approaches this is not possible: these are the constrained model search approaches that simply do not rely on metamodel changes. Furthermore, there are the approaches of Krause et al., ASIMOV, MCL, and Wittern, that inherently rely on the parallel application of metamodel change and model change[20].

20. This holds also for parts of the approach of COPE/Edapt. However, a variant of this approach allows for offline change collection. This part may benefit from external change identification approaches.

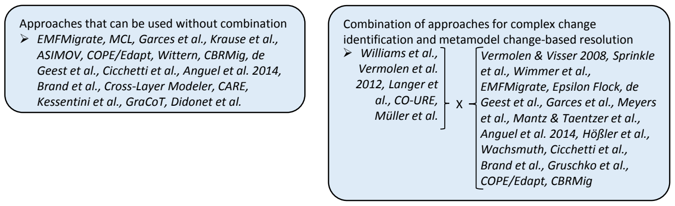
Approaches that can be used without combination
EMFMigrate, MCL, Garces et al., Krause et al., ASIMOV, COPE/Edapt, Wittern, CBRMig, de Geest et al., Cicchetti et al., Anguel et al. 2014, Brand et al., Cross-Layer Modeler, CARE, Kessentini et al., GraCoT, Didonet et al.
Combination of approaches for complex change identification and metamodel change-based resolution
Williams et al., Vermolen et al. 2012, Langer et al., CO-URE, Müller et al.
X
Vermolen & Visser 2008, Sprinkle et al., Wimmer et al., EMFMigrate, Epsilon Flock, de Geest et al., Garces et al., Meyers et al., Mantz & Taentzer et al., Anguel et al. 2014, Hößler et al., Wachsmuth, Cicchetti et al., Brand et al., Gruschko et al., COPE/Edapt, CBRMig

Fig. 5. Approaches that can be used alone or in combination.

In the remainder of this section, we go through the dimensions shown in Figure 4 and discuss under what circumstances different approaches might be preferable.

5.1 Correctness

Co-evolution can be performed with models at all stages of their life cycle; from models that are part of a software that is still subject to production or evolution, to models that are part of a legacy software that is only preserved. Depending on the point in the life cycle, there can be different needs on the co-evolution. When models are no more expected to be used as an input for code creation, but just as a documentation, it might be sufficient to keep models accessible, i.e. to ensure that a model is syntactically conform to a new metamodel, so that it can be opened with updated editors.

On the other hand, when models are still used as input, e.g. for code generation, it might not be sufficient to simply retrieve models that are conform to the syntax defined by the new metamodel, but to retrieve a semantically "correct" model. For example, a variant of the approach by Taentzer and Mantz creates references and objects, when lower bounds of multiplicities are increased [43]. While this can be required to make a model conform to a new metamodel version, a random creation of new references and objects might be harmful, when the model is used to generate code afterwards.

Similarly, the answer to the question, what a semantically "correct" model is, can differ from domain to domain, company to company, or even model to model. For example, when a type restriction is applied to a property in a metamodel, the generic solution implemented in the approaches of Meyers et al. and Edapt would delete property instances that are no longer conform. In contrast, the approach of Wachsmuth would allow the user to specify a type conversion. Thus, due to giving additional control to the user, the property values can be preserved in the models. Consequently, if there is the need for an outcome that deviates from generic solutions, approaches are required to provide the user with a certain amount of control about the results of the resolution.

Required result correctness: Due to the observation that approaches allow for different outcomes and different levels of control, we propose to potential users to ask this question: "Is a syntactical metamodel conformance a sufficient outcome for the co-evolution case or not?" If it is sufficient, all of the approaches presented above might be used.

However, as summarized in Figure 6, if a syntactical conformance is not sufficient, we suggest the user to consider approaches that allow for a full control of the result. Based on our classification this can be judged with 3 criteria:

1) Can the user reach each desired outcome, i.e. does he have free choice? We consider this criterion to be fulfilled, if the user can either choose freely from a complete set of possible solutions or if the user can freely manipulate a given solution or strategy, or specify a new resolution strategy.

2) The approach should never enforce the application of rules that have not been defined, selected, or been open to change by the user. We consider this criterion to be *not* fulfilled if an approach includes "single solutions" in benefit class 0:n. An exception are cases of *clean ups*, since then only non-conform elements will be deleted that have not been processed otherwise before. Thus, the user is still free to prevent this deletion.

3) When resolution strategies are applied, are they applied in the right situations? For approaches that use resolution strategies that are defined based on metamodel changes (i.e. all approaches, except the group constraint model search and the approach of Didonet et al. [36]), this depends on the correctness of the identified (complex) changes. We consider this criterion to be fulfilled if either of the three conditions is fulfilled:

- *No change information required:* an approach does not rely on metamodel change information (i.e. the group constraint model search),
- *Complex change collection:* the person changing the metamodel defines complex changes using online UI-intrusive change collection, or
- *Complex change identification:* if the complex changes are identified (semi-)automatically, the approach should be able to consider that collected atomic changes are potentially incomplete and/or allow the user to correct/define the result.

Generalization opportunities: From the taxonomy we know that there are different degrees of control in form of the two benefit classes 1:n (control about the strategy, but not per model) and 1:1 (control about the resolution per model). When selecting an approach that requires specific results the user should therefore consider what degree of control is required. To guide this decision, we propose to ask the following question: "Is it possible to generalize resolution per metamodel change within the given domain, company, or project?"

When such a generalization does not seem possible, we propose to use those approaches that fit the above listed criteria and allow for a free choice or manipulation of model specific solutions. Approaches that fulfill these criteria are: Cross-Layer Modeler [64], CARE [68], Kessentini et al. [67], GraCoT [65], and ASIMOV [50].

If a generalization is possible, we propose to use those approaches that fit the above listed criteria and allow for an unrestricted specification of resolution strategies for metamodel changes: Krause et al. [29], COPE/Edapt [60], Wittern [61], and Didonet et al. [36], as well as the combinations of the change identification approaches of Williams et al. [69], Vermolen et al. [70], and Mü)ler et al. [72] with one of the following resolution approaches: Vermolen & Visser 2008 [23], Sprinkle et al. [24], Wimmer et al. [25], EMFMigrate

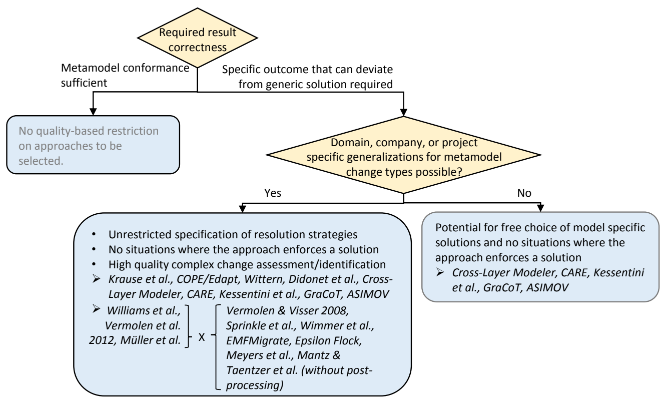
Required result correctness
Metamodel conformance sufficient
Specific outcome that can deviate from generic solution required
No quality-based restriction on approaches to be selected.
Domain, company, or project specific generalizations for metamodel change types possible?
Yes
No
• Unrestricted specification of resolution strategies
• No situations where the approach enforces a solution
• High quality complex change assessment/identification
➢ Krause et al., COPE/Edapt, Wittern, Didonet et al., Cross-Layer Modeler, CARE, Kessentini et al., GraCoT, ASIMOV
➢ Williams et al., Vermolen et al. 2012, Müller et al.
X
Vermolen & Visser 2008, Sprinkle et al., Wimmer et al., EMFMigrate, Epsilon Flock, Meyers et al., Mantz & Taentzer et al. (without post-processing)
Potential for free choice of model specific solutions and no situations where the approach enforces a solution
➢ Cross-Layer Modeler, CARE, Kessentini et al., GraCoT, ASIMOV

Fig. 6. Required correctness of the resulting models.

[28], Epsilon Flock [35], Meyers et al. [40], Mantz & Taentzer et al. (without post-processing) [46]. However, even if a generalization is possible, a company does not need to use this. Thus, also the approaches Cross-Layer Modeler [64], CARE [68], Kessentini et al. [67], GraCoT [65], and ASIMOV [50] can be used.

5.2 Automation

The second major subject of trade-off when deciding between approaches is the automation support, since the running costs of co-evolution are caused by the manual effort that needs to be invested. For example, decisions that are made per model allow a high control over the result. However, making such individual decisions can turn out to be infeasible, when hundreds of models need to be co-evolved.

Therefore, we propose to ask the following question, when selecting an approach: "Is the main priority to fully automate the resolution if possible?"

If yes, approaches that should be adopted are those that provide full automation for change identification and include resolutions in the benefit class 0:n. These are the approaches of Garcés et al. [39], ASIMOV [50], COPE/Edapt [60], and Wittern [61]. Furthermore, the change identification approaches of Williams et al. [69], Langer et al. [71], and CO-URE [10] might be used in combination with the following resolution approaches: de Geest et al. [37], Meyers et al. [40], Mantz & Taentzer et al. [46], Anguel et al. 2014 [47], Hößler et al. [48], Wachsmuth [51], Cicchetti et al. [15], Brand et al. [53], Gruschko et al. [14], COPE/Edapt [60], and CBRMig [62].

Automation - support manual tasks: As mentioned above, manual tasks during co-evolution determine the running costs of applying an approach.

Therefore, even if a full automation is not necessarily prioritized over correctness, when manual tasks are taken into consideration, we propose to consider the following question:: "Is support required to reduce manual effort?" If yes, the user should take approaches into account that help to reduce manual effort. In looking at the approaches in our survey, we identified the following strategies to reduce effort:

1) An approach might include full automation of change identification or resolutions in benefit class 0:n.

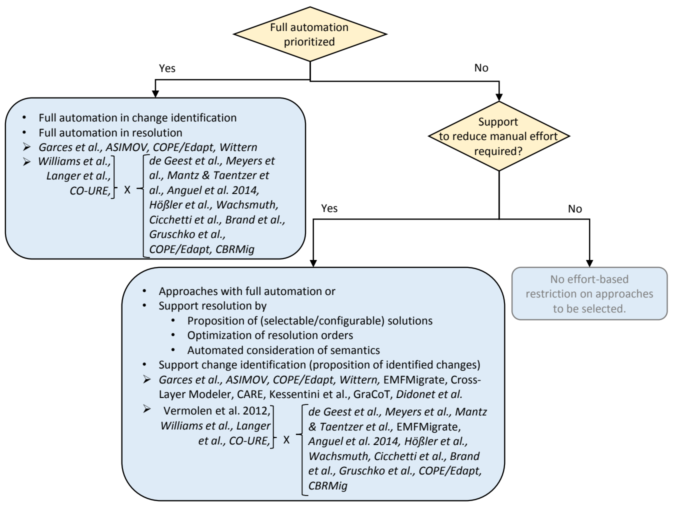
Full automation prioritized
Yes
No
Full automation in change identification
Full automation in resolution
Garces et al., ASIMOV, COPE/Edapt, Wittern
Williams et al., Langer et al., CO-URE
X
de Geest et al., Meyers et al., Mantz & Taentzer et al., Anguel et al. 2014, Hößler et al., Wachsmuth, Cicchetti et al., Brand et al., Gruschko et al., COPE/Edapt, CBRMig
Support to reduce manual effort required?
Yes
No
Approaches with full automation or
Support resolution by
Proposition of (selectable/configurable) solutions
Optimization of resolution orders
Automated consideration of semantics
Support change identification (proposition of identified changes)
Garces et al., ASIMOV, COPE/Edapt, Wittern, EMFMigrate, Cross-Layer Modeler, CARE, Kessentini et al., GraCoT, Didonet et al.
Vermolen et al. 2012, Williams et al., Langer et al., CO-URE
X
de Geest et al., Meyers et al., Mantz & Taentzer et al., EMFMigrate, Anguel et al. 2014, Hößler et al., Wachsmuth, Cicchetti et al., Brand et al., Gruschko et al., COPE/Edapt, CBRMig
No effort-based restriction on approaches to be selected.

Fig. 7. Automation and support for manual tasks

2) An approach might support the creation of resolution strategies or model resolutions by providing multiple strategies/solutions that can be selected or single strategies that only need to be configured. This way, also users who are not familiar with a change specification language can be enabled to specify their own strategies.

3) An approach can optimize the order of change resolutions to save the user form unnecessary manual tasks.

4) An approach might consider, in addition to the metamodel conformance, constraints arising from the model's context, in order to reduce the number of solutions that are presented to the user for selection.

5) Finally, an approach for the identification of complex changes might support the user by automatically proposing a list of identified changes, from which the user can select.

These strategies can be found within the following approaches (as shown in Figure 7): Garcés et al. [39], ASIMOV [50], COPE/Edapt [60], Wittern [61], EMFMigrate [28], Cross-Layer Modeler [64], CARE [68], Kessentini et al. [67], GraCoT [65], and Didonet et al. [36]. Again, the change identification approaches of Vermolen et al. 2012 [70], Williams et al. [69], Langer et al. [71], and CO-URE [10] might be combined with the following resolution approaches: de Geest et al. [37], Meyers et al. [40], Mantz & Taentzer et al. [46], EMFMigrate [28], Anguel et al. 2014 [47], Hößler et al. [48], Wachsmuth [51], Cicchetti et al. [15], Brand et al. [53], Gruschko et al. [14], COPE/Edapt [60], or CBRMig [62].

5.3 Tool Support

The approaches we surveyed have been partially implemented already and some of these implementations have even been made publicly available. When a company's ability to invest is low, the direct reuse of a tool might be required, since an approach for which an implementation can be reused is easier to adopt. Therefore, we propose to ask the following questions when adopting an approach (as summarized in Figure 8): "Can or should a proprietary tool be built?".

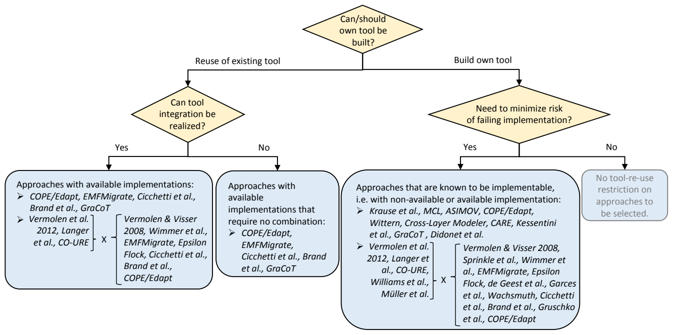
Can/should own tool be built?
Reuse of existing tool
Build own tool
Can tool integration be realized?
Yes
No
Need to minimize risk of failing implementation?
Yes
No
Approaches with available implementations:
COPE/Edapt, EMFMigrate, Cicchetti et al., Brand et al., GraCoT
Vermolen et al. 2012, Langer et al., CO-URE
X
Vermolen & Visser 2008, Wimmer et al., EMFMigrate, Epsilon Flock, Cicchetti et al., Brand et al., COPE/Edapt
Approaches with available implementations that require no combination:
COPE/Edapt, EMFMigrate, Cicchetti et al., Brand et al., GraCoT
Approaches that are known to be implementable, i.e. with non-available or available implementation:
Krause et al., MCL, ASIMOV, COPE/Edapt, Wittern, Cross-Layer Modeler, CARE, Kessentini et al., GraCoT , Didonet et al.
Vermolen et al. 2012, Langer et al., CO-URE, Williams et al., Müller et al.
X
Vermolen & Visser 2008, Sprinkle et al., Wimmer et al., EMFMigrate, Epsilon Flock, de Geest et al., Garces et al., Wachsmuth, Cicchetti et al., Brand et al., Gruschko et al., COPE/Edapt
No tool-re-use restriction on approaches to be selected.

Fig. 8. Tool support

In case an existing tool should be reused, the criterion for the selection of approaches should be that implementations are made available already[21].

Tool support - integration: As mentioned above, only some of the approaches can be used standalone. For situations where it is required to combine change identification and resolution approaches, it is necessary to integrate these tools, even if implementations already exist. Therefore, we propose to further ask the refining question: "Can an integration of multiple tools be realized?".

If an integration can be build, it is possible to use all single approaches and combinations of change identification and resolution approaches for which the above mentioned criterion - that implementations are publicly available - is fulfilled. Thus, the approaches COPE/Edapt [60], EMFMigrate [28], Cicchetti et al. [15], Brand et al. [53], or GraCoT [65] might be used. Furthermore, the change identification approaches by Vermolen et al. 2012 [70], Langer et al. [71], or CO-URE [10] can be used in combination with one of the following resolution approaches: Vermolen & Visser 2008 [23], Wimmer et al. [25], EMFMigrate [28], Epsilon Flock [35], Cicchetti et al. [15], Brand et al. [53], and COPE/Edapt [60].

However, if also the effort of building an integration is not desired, we propose to focus on approaches that can be used standalone and fulfill the above mentioned criterion of having publicly available implementation. These are: COPE/Edapt [60], EMFMigrate [28], Cicchetti et al. [15], Brand et al. [53], GraCoT [65].

Tool support - risk of first implementation: Building a tool for a research approach can be risky, especially when the approach is of theoretical nature and has not been implemented before. Therefore, we propose to ask the following refining question, when it is decided that an own tool should be build: "Is there a need to minimize the risk of failing the attempt to implement the tool?".

If there is this need, we propose to the user to consider approaches that are at least known to be implementable, i.e. for which an implementation is publicly available or at least is mentioned or described in the corresponding publications. Thus, one of the following approaches could be used: Krause et al. [29], MCL [32], ASIMOV [50], Wittern

21. Of course, it is also a possibility to contact authors of other approaches to request whether they are willing to make their implementations available. However, here we only consider approaches for which these implementations are already published.

[61], Cross-Layer Modeler [64], CARE [68], Kessentini et al. [67], GraCoT [65], and Didonet et al. [36]. Alternatively, the change identification approaches by Vermolen et al. 2012 [70], Langer et al. [71], or CO-URE [10], Williams et al. [69], Müller et al. [72] could be used in combination with one of the following resolution approaches: Vermolen & Visser 2008 [23], Sprinkle et al. [24], Wimmer et al. [25], EMFMigrate [28], Epsilon Flock [35], de Geest et al. [37], Garcés et al. [39], Wachsmuth [51], Cicchetti et al. [15], Brand et al. [53], Gruschko et al. [14], or COPE/Edapt [60].

5.4 Organizational Constraints

Finally, as illustrated in Figure 4, there are some organizational constraints that should be taken into account, in addition to the considerations about required correctness and automation. As mentioned above, these constraints concern the control about the co-evolution process as well as the skills/expertise of the person performing the corresponding activities.

Control about the process: One special characteristic of the co-evolution of metamodels and models is that its subtasks do not necessarily have to be performed within the same organizational units or even company. Leading to the situation that a company, which wants to co-evolve models after a metamodel change, might not have any influence on the decision on how the metamodel changes are applied and, thus, on how the changes can be collected. For example, when a metamodel is used that is changed outside a company (e.g. as for standards like BPMN), approaches that are based on online change detection are often not applicable.

Consequently, if the metamodel change is not under the control of the company, online change collection might not happen[22].

In consequence, a user should consider the following question: "Are the metamodel changes performed under the control of the company or will the results of the online change collection be available?" If not, change identification can only be performed with approaches that do not rely on online change collection as summarized in Figure 9.

This means that the following approaches that use offline change identification can be used: EMFMigrate [28], MCL [32], de Geest et al. [37], Garcés et al. [39], Anguel et al. [47], Cicchetti et al. [15], Brand et al. [53], COPE/Edapt (offline change collection) [60], CBRMig [62], CARE [68], Kessentini et al. [67], GraCoT [65], and Didonet et al. [36]. Furthermore, combinations of approaches with the approaches from the group "identify complex changes" (Williams et al. [69], Vermolen et al. 2012 [70], Langer et al. [71], CO-URE [10], Müller et al. [72]) can be used.

Skills of the person changing the metamodel: As argued above, UI-intrusive change collection approaches require the user who changes the metamodel to be aware of up to 60 or more operators and to strictly apply them, even though simpler valid ways of changing the metamodel in the intended way might exists (e.g. by using atomic operators instead of complex ones). However, mistakes in the use of these approaches lead to wrong identification of

22. To the best of our knowledge, online change collection is so far not happening (or results are not provided) for the big standard metamodels UML and BPMN

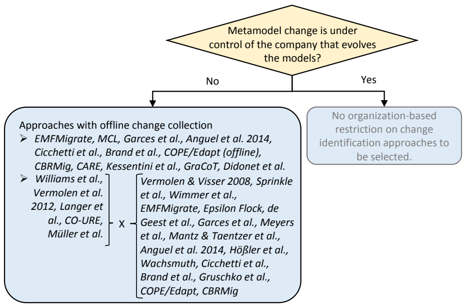
Metamodel change is under control of the company that evolves the models?
No
Yes
Approaches with offline change collection
➢ EMFMigrate, MCL, Garces et al., Anguel et al. 2014, Cicchetti et al., Brand et al., COPE/Edapt (offline), CBRMig, CARE, Kessentini et al., GraCoT, Didonet et al.
➢ Williams et al., Vermolen et al. 2012, Langer et al., CO-URE, Müller et al.
X
Vermolen & Visser 2008, Sprinkle et al., Wimmer et al., EMFMigrate, Epsilon Flock, de Geest et al., Garces et al., Meyers et al., Mantz & Taentzer et al., Anguel et al. 2014, Hößler et al., Wachsmuth, Cicchetti et al., Brand et al., Gruschko et al., COPE/Edapt, CBRMig
No organization-based restriction on change identification approaches to be selected.

Fig. 9. Organizational Constraints: control about the process.

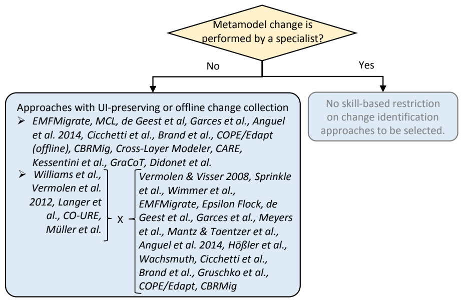
Metamodel change is performed by a specialist?
No
Yes
Approaches with UI-preserving or offline change collection
➢ EMFMigrate, MCL, de Geest et al, Garces et al., Anguel et al. 2014, Cicchetti et al., Brand et al., COPE/Edapt (offline), CBRMig, Cross-Layer Modeler, CARE, Kessentini et al., GraCoT, Didonet et al.
➢ Williams et al., Vermolen et al. 2012, Langer et al., CO-URE, Müller et al.
X
Vermolen & Visser 2008, Sprinkle et al., Wimmer et al., EMFMigrate, Epsilon Flock, de Geest et al., Garces et al., Meyers et al., Mantz & Taentzer et al., Anguel et al. 2014, Hößler et al., Wachsmuth, Cicchetti et al., Brand et al., Gruschko et al., COPE/Edapt, CBRMig
No skill-based restriction on change identification approaches to be selected.

Fig. 10. Organizational Constraints: skills of person changing the metamodel.

changes and in consequence to the application of model resolution rules that are not the intended ones (leading to a lower correctness of the result).

Therefore, we propose to ask the following question (as summarized in Figure 10): "Is the person who is changing the metamodel an expert in the considered co-evolution approach?" If not, we propose to prefer offline or UI-preserving change identification approaches over UI-intrusive change collection approaches.

In consequence, following approaches that are based on offline or UI-preserving change collection can be applied: EMFMigrate [28], MCL [32], de Geest et al. [37], Garcés et al. [39], Anguel et al. [47], Cicchetti et al. [15], Brand et al. [53], COPE/Edapt (offline change collection) [60], CBRMig [62], Cross-Layer Modeler [64], CARE [68], Kessentini et al. [67], GraCoT [65], and Didonet et al. [36]. Again, also combinations of approaches with the approaches from the group "identify complex changes" (Williams et al. [69], Vermolen et al. 2012 [70], Langer et al. [71], CO-URE [10], Müller et al. [72]) can be used.

Skills of the person co-evolving the models: The resolution of models most often includes manual tasks, either on the level of resolution strategy definitions or on the level of decision making for single models. However, defining resolution strategies for a domain or company requires a basic understanding of the co-evolution approach. Similarly, making resolution decisions per model requires knowledge about the models as well as the software and the projects they belong to.

Nonetheless, a domain expert who has to specify resolution strategies is not necessarily a specialist in model resolution technologies and a developer who performs the model migration is not necessarily entirely familiar with the

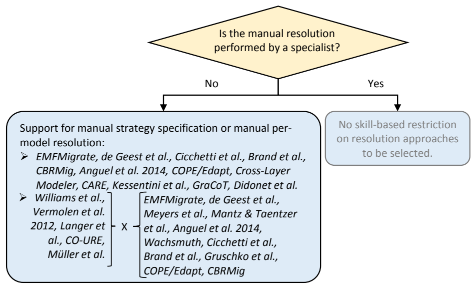
Is the manual resolution performed by a specialist?
No
Yes
Support for manual strategy specification or manual per-model resolution:
EMFMigrate, de Geest et al., Cicchetti et al., Brand et al., CBRMig, Anguel et al. 2014, COPE/Edapt, Cross-Layer Modeler, CARE, Kessentini et al., GraCoT, Didonet et al.
Williams et al., Vermolen et al. 2012, Langer et al., CO-URE, Müller et al.
X
EMFMigrate, de Geest et al., Meyers et al., Mantz & Taentzer et al., Anguel et al. 2014, Wachsmuth, Cicchetti et al., Brand et al., Gruschko et al., COPE/Edapt, CBRMig
No skill-based restriction on resolution approaches to be selected.

Fig. 11. Organizational Constraints: skills of the person co-evolving the models.

original models.

Therefore, we propose to companies to consider the question whether the persons applying or configuring the approach have the above mentioned expertise. If not, we propose to apply the following criteria when selecting an approach:

1) For the need to define resolution strategies, approaches studied in this survey provide support by giving default resolution strategies that can be configured or freely manipulated.

2) For resolution decisions per model, we found support given in 2 forms: (a) by providing resolution strategies that can be configured per model, or (b) by providing a choice between multiple resolutions per model.

The following approaches fulfill at least one of the two criteria (summarized in Figure 11): EMFMigrate [28], de Geest et al. [37], Cicchetti et al. [15], Brand et al. [53], CBRMig [62], Anguel et al. [47], COPE/Edapt [60], Cross-Layer Modeler [64], CARE [68], Kessentini et al. [67], GraCoT [65], and Didonet et al. [36]. Furthermore, combinations can be used that include the above mentioned or the following additional approaches: Meyers et al. [40], Mantz & Taentzer et al. [46], Wachsmuth [51], and Gruschko et al. [14].

5.5 How to use the questions for decision making?

To determine which approaches are good candidates for the company's or the project's case, a user might take the different decision diagrams shown in this section and walk through the questions to get lists of approaches that fit his constraints.

For example, consider the case of a (hypothetical) company that uses and maintains its own domain specific modeling language (DSML). The metamodel change is performed within the company by those experts that developed the DSML, but who are not familiar with co-evolution technology. Model resolution is performed by the developers that created the models and related software. Thus, the company has one organizational constraint. In addition, the language developers formulate the need for domain specific resolutions, but also predict that common strategies can be used for the whole language. While a full automation is not the main priority, support for manual tasks is desired to increase the degree of automation. Finally, the company is willing to build a tool for the resolution, if it is shown that the used techniques can be implemented.

In a second step, the user should compare the outcomes of the different decision diagrams to identify approaches that fit all his needs.

For our exemplary company, this would mean that the approaches of Didonet et al., Cross-Layer Modeler, CARE, Kessentini et al., and GraCoT could be used. An alternative choice could be a combination of one of the change identification approaches of Williams et al. and Vermolen et al. 2012 with the resolution parts of EMFMigrate or COPE/Edapt.

The result might also be used to reflect upon the question whether compromises are required, e.g. if there is no or only a, unsatisfying overlap between the approaches fulfilling the different needs.

In the case of our exemplary company, the decision makers might recognize that the list contains COPE/Edapt only for its resolution part. This is the case, since the developers of the DSML are not experts in co-evolution technologies. However, this is something that could be changed by training these developers. This willingness to eliminate this organizational constraint would allow the company to add COPE/Edapt as a standalone approach to the list of candidates as well as the approach of Wittern.

6 Related Work

In the last 6 years, a handful of surveys focussing either on approaches or tools for model migration, i.e. metamodel-model co-evolution, have been published. In the following, we give an overview of the goals and perspectives of these surveys and explain how our survey complements these existing overviews.

In 2009, Rose et al. presented a small summary of approaches for model migration [79]. Already back then, they identified that operator based approaches have to deal with problems, such as the user's challenge to find the right operator and the need of an integration to editors. They also discussed that per metamodel change alternative migration strategies are feasible. Overall, the discussion focusses mainly on the detection of changes.

A year later, Rose et al. presented a comparison of four model migration tools [80]. The tools were applied to example tasks, considering aspects as, e.g. conciseness (i.e. the required number of operators and additional lines of code), clarity (simplicity/understandability of migration strategies), supported modeling technologies, e.g. Ecore or XML, and performance of model migration execution. They summarize whether a tool allows to change the offered migration strategy, without discussing how this happens. Similarly, they approach a first simple comparison of the expressiveness of the different tools.

Another tool comparison was performed by a broader team around Rose et al., as a contest of 9 different transformation tools [81]. Besides tools for model and graph transformations, only two of the considered tools where specialized in providing model migration strategies for metamodel changes. Again the comparison was performed on examples of metamodel migrations and all participants of the contest evaluated the tools. Some additional criteria were considered, compared to [80], e.g. tool maturity and the question to what extent the given benchmark problem could be solved. The findings suggest that the participants considered model migration tools more appropriate for the given tasks than the transformation tools.

Herrmannsdoerfer and Wachmuth provide a survey on 10 approaches for the coupled evolution of metamodel and models [82]. A focus is put on transformation characteristics, such as the used transformation language and the question whether the model migration is performed as an in-place transformation. They distinguish between approaches that allow to overwrite existing "couplings of metamodel changes" with model migration rules (i.e. change a strategy) and approaches that allow to "extend the set of couplings" (i.e. that allow specification of migration rules for new metamodel changes). Finally, they make a rough differentiation, whether metamodel changes are defined by a user, recorded directly, or detected.

Anguel et al. presented another overview paper in 2015, comparing 8 approaches with regard to specification and source of the evolution information, language of the resolution specification and extensibility [83].

Most recently, Paige et al. published a summary of state of the art and future challenges in model evolution. In contrast to the focused in-depth investigation in this paper, Paige's work gives a broader, high-level overview of diverse topics around the evolution of models, e.g. including model versioning [84]. Nonetheless, Paige et al. provide a nice short summary of 9 approaches for co-evolution of models with evolving metamodels. They differentiate between operator-based and inference approaches, i.e. offline approaches.

To summarize, the existing surveys provide interesting perspectives on approaches that directly support metamodel-model co-evolution, but also on approaches that might just be used to apply the co-evolution. However, the following points are so far not addressed:

- *Balance between degree of automation and user control:* Existing surveys already address the question, whether an approach allows for changes of resolution strategies. However, until now there is no overview on the techniques and abilities provided by the different approaches to balance the degree of automation and user control. Further, there is so far no systematic summary of approaches and techniques that also support the detection of complex changes.
- *Prerequisites:* None of the surveys summarizes or discusses the implicit prerequisites that techniques define on skills and the organizational structure. Thus, we have so far no support for determining the applicability of approaches in industrial contexts.

None of the overview papers discussed here was created based on a systematic literature review [21]. This is reflected in the fact that the three papers that were published in 2014 and 2015 are outdated even for their time, since they include no approaches that have been published after 2011. The set of todays 31 existing approaches is nearly twice as big as the set of approaches considered in the related work listed above.

7 Conclusion and Future Work

In this paper, we presented an extensive taxonomy and survey on approaches that support metamodel-model co-evolution. 31 approaches from literature have been studied and classified with respect to the taxonomy. Based on the survey's insights, we presented a decision support that can be used by practitioners to choose the co-evolution approaches that are good candidates for their case. Needs for automation and user control are taken into account as well as prerequisites that are implied by the approaches. This decision support is summarized in Section 5.

Challenges: We observe that only 6 approaches address all three benefit classes. However, none of the approaches fully exploits the existing possibilities. It seems to still be an open question how the different techniques to support the three benefit classes can be combined into one approach. We saw that the question whether a resolution strategy has been generated or is predefined still plays an important role when it comes to the way how user control and automation are balanced. It still seems a challenge to provide optional user interaction in approaches with predefined strategies. A possible reason might be the missing ability to allow for free strategy manipulation. It is an open question whether strategies with configuration points can be generated and whether such configurations might be optional. Future approaches might aim at overcoming these limitations.

Further, most approaches still focus on syntax in their automation. Approaches that work towards improving the automation, while not decreasing semantic quality, are still rare. This remains a challenge for future work.

Perspectives: Providing open and accessible tools for the generation of resolution strategies is a step that can change the landscape of research on co-evolution in the future. Similarly, it would be interesting to see first implementations and more mature forms of approaches for the learning of resolution strategy, as proposed in CBRMig [62].

We are aware that the evolution of metamodels also requires the co-evolution of other artifacts, such as transformations or constraints. Our insights on metamodel change collection and identification can definitely be useful for such other cases of co-evolution on metamodels. In the future, it might be interesting to investigate similarities between resolution techniques for different artifacts. For example, it will be useful to compare how different approaches to metamodel-constraints and metamodel-transformation co-evolution deal with resolution and whether there are similarities with model-resolution strategies. Further, it would be interesting to investigate whether these co-evolution techniques might partially be reused for cases of software evolution, i.e. when different software artifacts, such as models and code, co-evolve.

In the decision support we propose to ask the question "Is it possible to generalize resolution per metamodel change within the given domain, company, or project?" (Figure 6). In fact, it requires expert knowledge about the domain, company, or project at hand to answer this question. It would be interesting to see more studies in the future that investigate for different domains, whether 0:n solutions provided by the different approaches are sufficient to cover most needs or not. This knowledge could further support users in making their choice.

REFERENCES

[1] J. Hutchinson, J. Whittle, M. Rouncefield, and S. Kristoffersen, "Empirical assessment of mde in industry," in *Proceedings of the 33rd International Conference on Software Engineering*. ACM, 2011, pp. 471–480.

[2] G. Liebel, N. Marko, M. Tichy, A. Leitner, and J. Hansson, "Assessing the state-of-practice of model-based engineering in the embedded systems domain," in *Model-Driven Engineering Languages and Systems*. Springer, 2014, pp. 166–182.

[3] S. Kelly and J.-P. Tolvanen, *Domain-specific modeling: enabling full code generation*. John Wiley & Sons, 2008.

[4] J.-P. Tolvanen and S. Kelly, "Metaedit+: defining and using integrated domain-specific modeling languages," in *Proceedings of the 24th ACM SIGPLAN conference companion on Object oriented programming systems languages and applications*. ACM, 2009, pp. 819–820.

[5] H. Behrens, M. Clay, S. Efftinge, M. Eysholdt, P. Friese, J. Köhnlein, K. Wannheden, and S. Zarnekow, "Xtext user guide," *https://eclipse.org/Xtext/documentation/1_0_1/xtext.pdf*, 2008.

[6] X. Blanc, I. Mounier, A. Mougenot, and T. Mens, "Detecting model inconsistency through operation-based model construction," in *Software Engineering, 2008. ICSE'08. ACM/IEEE 30th International Conference on*. IEEE, 2008, pp. 511–520.

[7] A. Reder and A. Egyed, "Incremental consistency checking for complex design rules and larger model changes," in *Proceedings of the 15th international conference on Model Driven Engineering Languages and Systems*. Springer-Verlag, 2012, pp. 202–218.

[8] D. Di Ruscio, L. Iovino, and A. Pierantonio, "Evolutionary togetherness: how to manage coupled evolution in metamodeling ecosystems," in *Graph Transformations*. Springer, 2012, pp. 20–37.

[9] A. Cicchetti, D. Di Ruscio, R. Eramo, and A. Pierantonio, "Automating co-evolution in model-driven engineering," in *Enterprise Distributed Object Computing Conference, 2008. EDOC'08. 12th International IEEE*. IEEE, 2008, pp. 222–231.

[10] J. García, O. Diaz, and M. Azanza, "Model transformation co-evolution: A semi-automatic approach," in *Software Language Engineering*. Springer, 2013, pp. 144–163.

[11] K. Garcés, J. M. Vara, F. Jouault, and E. Marcos, "Adapting transformations to metamodel changes via external transformation composition," *Software & Systems Modeling*, vol. 13, no. 2, pp. 789–806, 2014.

[12] A. Kusel, J. Etzlstorfer, E. Kapsammer, W. Retschitzegger, W. Schwinger, and J. Schonbock, "Consistent co-evolution of models and transformations," in *Model Driven Engineering Languages and Systems (MODELS), 2015 ACM/IEEE 18th International Conference on*. IEEE, 2015, pp. 116–125.

[13] M. Herrmannsdoerfer, S. Benz, and E. Juergens, "Cope-automating coupled evolution of metamodels and models," in *ECOOP 2009–Object-Oriented Programming*. Springer, 2009, pp. 52–76.

[14] B. Gruschko, D. Kolovos, and R. Paige, "Towards synchronizing models with evolving metamodels," in *Proceedings of the International Workshop on Model-Driven Software Evolution*, 2007.

[15] A. Cicchetti, D. Di Ruscio, and A. Pierantonio, "Managing dependent changes in coupled evolution," in *Theory and Practice of Model Transformations*. Springer, 2009, pp. 35–51.

[16] D. E. Khelladi, R. Hebig, R. Bendraou, J. Robin, and M.-P. Gervais, "Detecting complex changes during metamodel evolution," in *Advanced Information Systems Engineering*. Springer, 2015, pp. 263–278.

[17] R. F. Paige, P. J. Brooke, and J. S. Ostroff, "Metamodel-based model conformance and multiview consistency checking," *ACM Trans. Softw. Eng. Methodol.*, vol. 16, no. 3, Jul. 2007. [Online]. Available: http://doi.acm.org/10.1145/1243987.1243989

[18] L. Iovino, A. Pierantonio, and I. Malavolta, "On the impact significance of metamodel evolution in mde." *Journal of Object Technology*, vol. 11, no. 3, pp. 1–33, 2012.

[19] V. Lifschitz, "What is answer set programming?." in *AAAI*, vol. 8, 2008, pp. 1594–1597.

[20] M. Alanen and I. Porres, *Difference and union of models*. Springer, 2003.

[21] B. Kitchenham, "Procedures for performing systematic reviews," *Keele, UK, Keele University*, vol. 33, no. 2004, pp. 1–26, 2004.

[22] B. Cornelissen, A. Zaidman, A. Van Deursen, L. Moonen, and R. Koschke, "A systematic survey of program comprehension through dynamic analysis," *IEEE Transactions on Software Engineering*, vol. 35, no. 5, pp. 684–702, 2009.

[23] S. Vermolen and E. Visser, "Heterogeneous coupled evolution of software languages," in *Model Driven Engineering Languages and Systems*. Springer, 2008, pp. 630–644.

[24] J. Sprinkle and G. Karsai, "A domain-specific visual language for domain model evolution," *Journal of Visual Languages & Computing*, vol. 15, no. 3, pp. 291–307, 2004.

[25] M. Wimmer, A. Kusel, J. Schönböck, W. Retschitzegger, W. Schwinger, and G. Kappel, "On using inplace transformations for model co-evolution," in *Proceedings of the 2nd International Workshop on Model Transformation with ATL (MtATL) at TOOLS*, vol. 10, 2010, pp. 65–78.

[26] D. Di Ruscio, L. Iovino, and A. Pierantonio, "What is needed for managing co-evolution in mde?" in *Proceedings of the 2nd International Workshop on Model Comparison in Practice*. ACM, 2011.

[27] J. Di Rocco, L. Iovino, and A. Pierantonio, "Bridging state-based differencing and co-evolution," in *Proceedings of the 6th International Workshop on Models and Evolution*. ACM, 2012, pp. 15–20.

[28] D. Wagelaar, L. Iovino, D. Di Ruscio, and A. Pierantonio, "Translational semantics of a co-evolution specific language with the emf transformation virtual machine," in *Theory and Practice of Model Transformations*. Springer, 2012, pp. 192–207.

[29] C. Krause, J. Dyck, and H. Giese, "Metamodel-specific coupled evolution based on dynamically typed graph transformations," in *Theory and Practice of Model Transformations*. Springer, 2013, pp. 76–91.

[30] A. Narayanan, T. Levendovszky, D. Balasubramanian, and G. Karsai, "Automatic domain model migration to manage metamodel evolution," in *Model Driven Engineering Languages and Systems*. Springer, 2009, pp. 706–711.

[31] D. Balasubramanian, T. Levendovszky, A. Narayanan, and G. Karsai, "Continuous migration support for domain-specific languages," in *Proceedings of the 9th OOPSLA workshop on domain-specific modeling, Orlando*, vol. 2, 2009, pp. 311–322.

[32] T. Levendovszky, D. Balasubramanian, A. Narayanan, F. Shi, C. van Buskirk, and G. Karsai, "A semi-formal description of migrating domain-specific models with evolving domains," *Software & Systems Modeling*, vol. 13, no. 2, pp. 807–823, 2014.

[33] L. M. Rose, D. S. Kolovos, R. F. Paige, and F. A. Polack, "Enhanced automation for managing model and metamodel inconsistency," in *Automated Software Engineering, 2009. ASE'09. 24th IEEE/ACM International Conference on*. IEEE, 2009, pp. 545–549.

[34] ——, "Model migration with epsilon flock," in *Theory and Practice of Model Transformations*. Springer, 2010, pp. 184–198.

[35] L. M. Rose, D. S. Kolovos, R. F. Paige, F. A. Polack, and S. Poulding, "Epsilon flock: a model migration language," *Software & Systems Modeling*, vol. 13, no. 2, pp. 735–755, 2014.

[36] M. D. Del Fabro and P. Valduriez, "Semi-automatic model integration using matching transformations and weaving models," in *Proceedings of the 2007 ACM symposium on Applied computing*. ACM, 2007, pp. 963–970.

[37] G. de Geest, S. Vermolen, A. van Deursen, and E. Visser, "Generating version convertors for domain-specific languages," in *Reverse Engineering, 2008. WCRE '08. 15th Working Conference on*. IEEE, 2008, pp. 197–201.

[38] K. Garces, F. Jouault, P. Cointe, and J. Bézivin, "Adaptation of Models to Evolving Metamodels," inria, Research Report RR-6723, 2008. [Online]. Available: https://hal.inria.fr/inria-00338695

[39] K. Garcés, F. Jouault, P. Cointe, and J. Bézivin, "Managing model adaptation by precise detection of metamodel changes," in *Proceedings of the 5th European Conference on Model Driven Architecture-Foundations and Applications*. Springer, 2009, pp. 34–49.

[40] B. Meyers, M. Wimmer, A. Cicchetti, and J. Sprinkle, "A generic in-place transformation-based approach to structured model co-evolution," *Electronic Communications of the EASST*, vol. 42, 2012.

[41] G. Taentzer, F. Mantz, and Y. Lamo, "Co-transformation of graphs and type graphs with application to model co-evolution," in *Graph Transformations*. Springer, 2012, pp. 326–340.

[42] F. Mantz, S. Jurack, and G. Taentzer, "Graph transformation concepts for meta-model evolution guaranteeing permanent type conformance throughout model migration," in *Applications of Graph Transformations with Industrial Relevance*. Springer, 2012, pp. 3–18.

[43] G. Taentzer, F. Mantz, T. Arendt, and Y. Lamo, "Customizable model migration schemes for meta-model evolutions with multiplicity changes," in *Model-Driven Engineering Languages and Systems*. Springer, 2013, pp. 254–270.

[44] F. Mantz, G. Taentzer, and Y. Lamo, “Customizing model migrations by rule schemes,” in *Proceedings of the 2013 International Workshop on Principles of Software Evolution*. ACM, 2013, pp. 1–10.

[45] ——, “Co-transformation of type and instance graphs supporting merging of types with retyping,” *GCM 2012*, pp. 47–58, 2012.

[46] ——, “Well-formed model co-evolution with customizable model migration,” *Electronic Communications of the EASST*, vol. 58, 2013.

[47] F. Anguel, A. Amirat, and N. Bounour, “Using weaving models in metamodel and model co-evolution approach,” in *Computer Science and Information Technology (CSIT), 2014 6th International Conference on*. IEEE, 2014, pp. 142–147.

[48] J. Hößler, M. Soden, and H. Eichler, “Coevolution of models, metamodels and transformations,” *Models and Human Reasoning. Wissenschaft und Technik Verlag, Berlin*, pp. 129–154, 2005.

[49] H. Florez, M. Sánchez, J. Villalobos, and G. Vega, “Coevolution assistance for enterprise architecture models,” in *Proceedings of the 6th International Workshop on Models and Evolution*. ACM, 2012, pp. 27–32.

[50] H. A. F. Fernandez, “Adapting models in metamodels composition processes,” *Vínculos*, vol. 10, no. 1, pp. 96–108, 2013.

[51] G. Wachsmuth, “Metamodel adaptation and model co-adaptation,” in *ECOOP 2007–Object-Oriented Programming*. Springer, 2007, pp. 600–624.

[52] A. Cicchetti, D. Di Ruscio, R. Eramo, and A. Pierantonio, “Meta-model differences for supporting model co-evolution,” in *Proceedings of the 2nd Workshop on Model-Driven Software Evolution-MODSE*, 2008.

[53] M. Van Den Brand, Z. Protić, and T. Verhoeff, “A generic solution for syntax-driven model co-evolution,” in *Objects, Models, Components, Patterns*. Springer, 2011, pp. 36–51.

[54] S. Becker, B. Gruschko, T. Goldschmidt, and H. Koziolek, “A process model and classification scheme for semi-automatic meta-model evolution,” in *1st Workshop MDD, SOA und IT-Management (MSI), GI, GiTO-Verlag*, 2007, pp. 35–46.

[55] M. Herrmannsdoerfer, S. Benz, and E. Juergens, “Automatability of coupled evolution of metamodels and models in practice,” in *Model Driven Engineering Languages and Systems*. Springer, 2008, pp. 645–659.

[56] M. Herrmannsdoerfer, “Operation-based versioning of metamodels with cope,” in *Proceedings of the 2009 ICSE Workshop on Comparison and Versioning of Software Models*. IEEE Computer Society, 2009, pp. 49–54.

[57] M. Herrmannsdoerfer and M. Koegel, “Semantics-preserving model migration,” in *International Workshop on Models and Evolution*, 2010.

[58] M. Herrmannsdoerfer and D. Ratiu, “Limitations of automating model migration in response to metamodel adaptation,” in *Models in Software Engineering*. Springer, 2010, pp. 205–219.

[59] M. Herrmannsdoerfer and M. Koegel, “Towards a generic operation recorder for model evolution,” in *Proceedings of the 1st International Workshop on Model Comparison in Practice*. ACM, 2010, pp. 76–81.

[60] M. Herrmannsdoerfer, “Cope a workbench for the coupled evolution of metamodels and models,” in *Software Language Engineering*, B. Malloy, S. Staab, and M. van den Brand, Eds. Springer Berlin Heidelberg, 2011, vol. 6563, pp. 286–295.

[61] H. Wittern, “Determining the necessity of human intervention when migrating models of an evolved dsl.” in *EDOC Workshops*, 2013, pp. 209–218.

[62] F. Anguel, A. Amirat, and N. Bounour, “Towards models and metamodels co-evolution approach,” in *Programming and Systems (ISPS), 2013 11th International Symposium on*. IEEE, 2013, pp. 163–167.

[63] A. Demuth, R. E. Lopez-Herrejon, and A. Egyed, “Co-evolution of metamodels and models through consistent change propagation.” in *ME@MoDELS*. Citeseer, 2013, pp. 14–21.

[64] A. Demuth, M. Riedl-Ehrenleitner, R. E. Lopez-Herrejon, and A. Egyed, “Co-evolution of metamodels and models through consistent change propagation,” *Journal of Systems and Software*, vol. 111, pp. 281–297, 2016.

[65] P. Gomez, M. E. Sánchez, H. Florez, and J. Villalobos, “An approach to the co-creation of models and metamodels in enterprise architecture projects.” *Journal of Object Technology*, vol. 13, no. 3, pp. 1–29, 2014.

[66] W. Kessentini, “Automated metamodel/model co-evolution using a multi-objective optimization approach,” in *Proceedings of the ACM Student Research Competition at MODELS 2015 co-located with the ACM/IEEE 18th International Conference MODELS 2015*, 2015.

[67] W. Kessentini, H. Sahraoui, and M. Wimmer, “Automated metamodel/model co-evolution using a multi-objective optimization approach,” in *12th European Conference on Modelling Foundations and Applications (ECMFA 2016)*, 2016.

[68] J. Schoenboeck, A. Kusel, J. Etzlstorfer, E. Kapsammer, W. Schwinger, M. Wimmer, and M. Wischenbart, “Care: a constraint-based approach for re-establishing conformance-relationships,” in *Proceedings of the Tenth Asia-Pacific Conference on Conceptual Modelling-Volume 154*. Australian Computer Society, Inc., 2014, pp. 19–28.

[69] J. R. Williams, R. F. Paige, and F. A. Polack, “Searching for model migration strategies,” in *Proceedings of the 6th International Workshop on Models and Evolution*. ACM, 2012, pp. 39–44.

[70] S. D. Vermolen, G. Wachsmuth, and E. Visser, “Reconstructing complex metamodel evolution,” in *Software Language Engineering*. Springer, 2012, pp. 201–221.

[71] P. Langer, M. Wimmer, P. Brosch, M. Herrmannsdörfer, M. Seidl, K. Wieland, and G. Kappel, “A posteriori operation detection in evolving software models,” *Journal of Systems and Software*, vol. 86, pp. 551–566, 2013.

[72] K. Müller and B. Rumpe, “User-driven adaptation of model differencing results,” in *Proc. International Workshop on Comparison and Versioning of Software Models (CVSM14)*, 2014.

[73] M. Herrmannsdoerfer, S. Benz, E. Juergens *et al.*, “Cope: A language for the coupled evolution of metamodels and models,” in *1st International Workshop on Model Co-Evolution and Consistency Management*, 2008.

[74] M. Koegel, M. Herrmannsdoerfer, Y. Li, J. Helming, and J. David, “Comparing state-and operation-based change tracking on models,” in *Enterprise Distributed Object Computing Conference (EDOC), 2010 14th IEEE International*. IEEE, 2010, pp. 163–172.

[75] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design patterns: elements of reusable object-oriented software*. Pearson Education, 1994.

[76] R. Hebig, D. E. Khelladi, and R. Bendraou, “Surveying the corpus of model resolution strategies for metamodel evolution,” in *Asia-Pacific Software Engineering Conference (APSEC)*, 2015.

[77] E. Burger and B. Gruschko, “A change metamodel for the evolution of mof-based metamodels.” in *Modellierung*, 2010, pp. 285–300.

[78] M. Harman and B. F. Jones, “Search-based software engineering,” *Information and software Technology*, vol. 43, no. 14, pp. 833–839, 2001.

[79] L. M. Rose, R. F. Paige, D. S. Kolovos, and F. A. Polack, “An analysis of approaches to model migration,” in *Proceedings of the Joint MoDSE-MCCM Workshop*, 2009, pp. 6–15.

[80] L. M. Rose, M. Herrmannsdoerfer, J. R. Williams, D. S. Kolovos, K. Garcés, R. F. Paige, and F. A. Polack, “A comparison of model migration tools,” in *Model Driven Engineering Languages and Systems*. Springer, 2010, pp. 61–75.

[81] L. M. Rose, M. Herrmannsdoerfer, S. Mazanek, P. Van Gorp, S. Buchwald, T. Horn, E. Kalnina, A. Koch, K. Lano, B. Schätz *et al.*, “Graph and model transformation tools for model migration,” *Software & Systems Modeling*, vol. 13, no. 1, pp. 323–359, 2014.

[82] M. Herrmannsdörfer and G. Wachsmuth, “Coupled evolution of software metamodels and models,” in *Evolving Software Systems*, T. Mens, A. Serebrenik, and A. Cleve, Eds. Springer Berlin Heidelberg, 2014, pp. 33–63.

[83] A. A. Fouzia Anguel and N. Bounour, “Comparison study of metamodels and models co-evolution approaches,” in *Symposium on Complex Systems and Intelligent Computing (CompSIC)*, 2015.

[84] R. F. Paige, N. Matragkas, and L. M. Rose, “Evolving models in model-driven engineering: State-of-the-art and future challenges,” *Journal of Systems and Software*, vol. 111, pp. 272–280, 2016.