Software and Systems Modeling (2020) 19:693–720
https://doi.org/10.1007/s10270-019-00760-x

REGULAR PAPER

Systematic review of matching techniques used in model-driven methodologies

Ferenc Attila Somogyi[1] · **Mark Asztalos**[1]

Received: 21 March 2019 / Revised: 25 July 2019 / Accepted: 11 October 2019 / Published online: 1 November 2019

Abstract
In model-driven methodologies, model matching is the process of finding a matching pair for every model element between two or more software models. Model matching is an important task as it is often used while differencing and merging models, which are key processes in version control systems. There are a number of different approaches to model matching, with most of them focusing on different goals, i.e., the accuracy of the matching process, or the generality of the algorithm. Moreover, there exist algorithms that use the textual representations of the models during the matching process. We present a systematic literature review that was carried out to obtain the state-of-the-art of model matching techniques. The search process was conducted based on a well-defined methodology. We have identified a total of 3274 non-duplicate studies, out of which 119 have been included as primary studies for this survey. We present the state-of-the-art of model matching, highlighting the differences between different matching techniques, mainly focusing on text-based and graph-based algorithms. Finally, the main open questions, challenges, and possible future directions in the field of model matching are discussed, also including topics like benchmarking, performance and scalability, and conflict handling.

Keywords Model matching · Model comparison · Model differencing · Version control · Text-based modeling · Systematic literature review

1 Introduction

Model-driven methodologies like model-driven engineering (MDE) [131,143] or the Model-Driven Architecture (MDA) [61] of the Object Management Group (OMG) use graph-based models as the main artifacts during development. Models can be used for numerous purposes, like various types of formal analyses, design space exploration, requirements elicitation, or source code generation [74]. In the case of source code generation, the aim of model-driven methodologies is to increase productivity by requiring less attention to detail at lower abstraction levels [155]. The models represent the problem and solution at a higher abstraction level, making it easier to communicate with stakeholders [107]. Model transformation is the automated process of transforming a source model to a target model based on different rules. It is often used as a way of modifying and creating models. Model transformations are at the core of model-driven methodologies, as they are crucial in a lot of modeling tasks [23,43,135], and thus, even model management operations can be described using them [138]. Model-driven methodologies are increasingly more applied in the industry, although certain problems, like model evolution management and model versioning, impede their further spread [8,21].

Model evolution management is a difficult task and is an active research field today [121]. The field of model evolution includes multiple subfields, like version control or metamodel evolution management. Version control systems [15,141] are tools that support model evolution, and they greatly improve the productivity of software development in teamwork [22]. Model-based version control systems also boost productivity, although graph-based models are inher-

Communicated by Professor Jon Whittle.

This work was performed in the frame of FIEK_16-1-2016-0007 project, implemented with the support provided from the National Research, Development and Innovation Fund of Hungary, financed under the FIEK_16 funding scheme.

✉ Ferenc Attila Somogyi
Somogyi.Ferenc@aut.bme.hu

Mark Asztalos
Asztalos.Mark@aut.bme.hu

[1] Budapest University of Technology and Economics, Muegyetem rkp. 3, Budapest 1111, Hungary

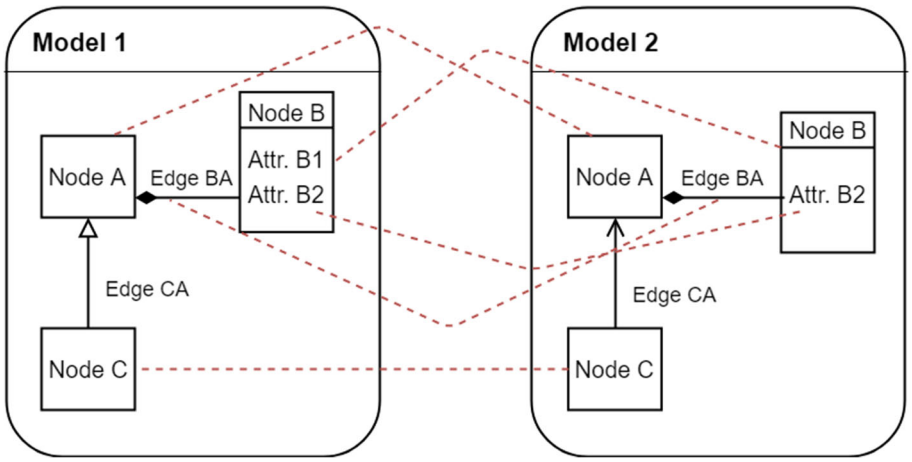

Fig. 1 Model matching illustrated on a simple example

ently different from text-based source code, and thus, they require different treatment during version control [9,30]. Supporting version control is one of our main motivations behind conducting a survey on model matching algorithms. It is worth mentioning that model matching can have other applications than version control, like in model transformation testing [87,95].

During optimistic concurrency handling in version control systems, differencing and merging the main artifacts is a necessary and crucial task [8,42,106]. Model matching is usually the first step of model differencing. The goal of model matching is to identify pairs of model elements in the input models that are considered to be matching, based on different considerations. Later in the process, differences between the models can be established using the matched pairs. Most matching approaches work directly with the graph-based structure of the models. However, there exist solutions that approach the problem differently, i.e., by working with the textual representations of the models. The textual notation can be mapped to the model in many ways, for example, by using a formal language and a parser [2], or by specifying it via a textual language like Alloy [70]. Describing models with a textual notation can have many advantages, such as (i) the off-line editing of models, (ii) better focus on details, thus, easier editing, especially in the case of large models, (iii) and they can offer a more readable serialization solution than XML-based approaches [67,68,144].

To demonstrate the problem of model matching, Fig. 1 presents a simple example. We try to match two imaginary models, *Model 1* and *Model 2*, with a custom model matching algorithm. Let us assume that the algorithm matches the following model elements: *Node A*, *Node C*, *Edge BA*, *Node B*, and *Attr. B2*. Thus, the output of the matching algorithm is the list of pairs containing the aforementioned model elements from the two input models. Our imaginary algorithm got these results by matching nodes and attributes based on their name, and matching edges based on their name, type, and the nodes they connect. We can see even in our simple example that model matching is not a trivial task. For example, a different matching algorithm might have matched *Edge CA* with its counterpart on the basis that the type of the edge is not as important as the nodes it connects. Let us assume that the correct decision in the case of *Edge CA* in our first modeling language would be to ignore the match, but in another, similar modeling language, the correct decision would be to match the edges in the two input models. If the matching algorithm would focus on generality, then it would aim to work with both modeling languages. If it also aims to be accurate, then it could be achieved by making the algorithm configurable in order to get the correct result for multiple languages. If the matching algorithm would focus only on accuracy instead, then it would likely focus on one modeling language, but it would have no configuration cost. This is only one example of the conflict between accuracy and generality, as there are other aspects to consider. In our survey, we aim to further investigate this topic.

As an important note, we would like to make a distinction between the type of model matching we are investigating in this paper, and between (graph) pattern matching that is also sometimes referred to as model matching in the literature. The first case is what we described in Fig. 1, where the goal is to find a matching pair for every model element between two or more models. In (graph) pattern matching, the goal is to find a homomorphic image (subgraph) of a pattern graph in the model [154]. While the two topics somewhat overlap in practice, due to the slightly different nature of the goals between the two cases, we decided not to include pattern matching in this survey, as it could possibly skew the statistics and conclusions drawn regarding the "classic" type of model matching.

This paper presents a systematic literature review (SLR) that was carried out in order to present the state-of-the-art of model matching approaches. This survey is different from others [9,30,57,134,146] as few other studies focus on the

categorization of model matching, and even fewer use a systematic approach, or focus on text-based model matching algorithms. We further elaborate on the differences between this survey and others in Sect. 3.1.1. According to our findings that we discuss later in this paper, there are even fewer that deal with the analysis of text-based model matching. Further reasoning behind the goals of this survey can be found in Sect. 3.1.2. The main results of this survey are (i) a collected body of research representing the state-of-the-art in model matching, (ii) identified correlations between different matching techniques, (iii) a summary of open questions related to the field of model matching, including topics like benchmarking, performance and scalability, conflict resolution and visualization, and the position of text-based algorithms in the literature.

The structure of the paper is as follows. Section 2 contains detailed background information on model matching, defining important concepts that we use in the survey. In Sect. 3, we present our SLR process in detail. We outline our motivations, detail the scope and goals of the survey, and present our research questions and search strategy. The study selection and data extraction processes are also discussed. Section 4 presents the results of the survey, answering the previously defined research questions. Since every survey is inherently subjective to a certain degree, we also discuss threats to the validity of this survey. Section 5 concludes the paper, highlighting the main results of the review.

2 Background on model matching

In this section, we discuss the main concepts related to model matching that are relevant to our survey. We use these concepts later, mostly during the categorization of matching algorithms. As we have discussed in Sect. 1, one of our main motivations behind surveying model matching algorithms is supporting model-based version control. In optimistic version control, the typical scenario is that the input models that are to be matched conform to the same metamodel. Dealing with evolving (changing) metamodels is a different topic with various proposed solutions [40,121,160], and is not the focus of this survey. Therefore, the algorithms included in our survey had to satisfy this criterion, namely, they had to support matching models that conform to the same metamodel. Fortunately, most model matching algorithms inherently satisfy this requirement.

Model matching is the process of finding a matching pair for every model element between two or more models, where the model elements can be nodes and edges, and the models are graph-based. The input consists of two or more graph-based models, while the output is the set of matching pairs between the model elements.

Kolovos et al. [86] analyzed existing model matching approaches, and split them into four categories, based on how the matching is conducted. Their work is one of the earliest takes on categorizing model matching algorithms, and many later studies were influenced by it [6,30,158,159]. This categorization also serves as the basis of our survey. The proposed categories are as follows, with the definitions being taken from the aforementioned research work [86]:

1. Static Identity-Based Matching is based on persistent and non-volatile unique identifiers that are associated with the model elements. This typically means a universally unique identifier (UUID). Static approaches are accurate, as the matching can always be determined. However, they are more difficult to generalize, as the reliance on unique identifiers may cause the algorithm to not work with more complex modeling languages.
2. Signature-Based (Dynamic) Matching approaches use a subset of the features of the models in order to match them. This is usually preceded by a configuration phase, during which a user-defined function is created to calculate the signature of model elements dynamically. As opposed to static matching, individually, the various features do not have to be unique. The downside of this form of matching is the effort required to specify the signature calculating functions.
3. Similarity-Based Matching differs from previous categories, as these algorithms do not give a binary yes or no answer to the question of whether two model elements are matching. Instead, a similarity degree is calculated, based on arbitrary features of the model elements. Different weighting values can be assigned to different features. If the similarity degree is above a certain (configurable) threshold, then the model elements are considered to be matching. Similarly to dynamic approaches, the model element features and weighting values can usually be configured by a user-defined function.
4. Custom Language-Specific Matching algorithms are tailored to a particular modeling language. They use the specific semantics of the language during matching, and thus, the process is more accurate. However, since they are tailored to a specific language, these approaches are not general.

During model differencing, a key difference between algorithms is the way they handle changes (differences) occurring between the input models. In the literature [69,82,83,85], there exist two main approaches:

– State-based approaches calculate changes by matching model elements in the input models with each other. Model matching—which is the main focus of our survey—explicitly occurs in this case. State-based

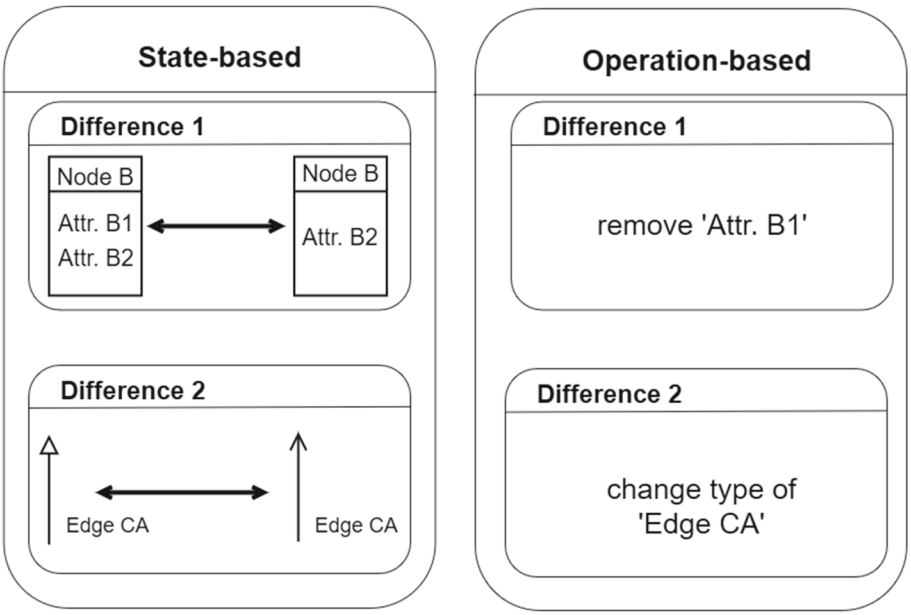

Fig. 2 Change tracking illustrated on our previous example

approaches usually store changes as instances of the metamodel of the input models, or as instances of a difference metamodel [41,127]. While there is often a significant computational complexity overhead (see the graph isomorphism problem [81]), these approaches are usually tool independent.
– Change-based approaches record changes as they occur, instead of calculating differences at a later time. Therefore, there is no need for model matching. The changes can often be run as model transformations on the initial model(s). In this case, we talk about **operation-based** approaches. These algorithms are usually easier to develop and comprehend [83], but are more tool dependent.

To further clarify what we mean by state-based and operation-based change tracking, let us demonstrate it using our previous example in Fig. 1. Figure 2 illustrates the differences our imaginary algorithm would have made if it used state-based or operation-based change tracking. Using a state-based approach, the algorithm would store the differences using an instance model of the metamodel, or a difference metamodel, as can be seen in the left side of the figure. If the algorithm would use an operation-based approach, it would record changes as they occur (in our example, let us assume that *Model 1* was the initial model), and we get a change log that can be applied to a model, as seen on the right side of the figure.

The focus of a model matching algorithm can vary based on what goals the algorithm aims to achieve. We consider accuracy and generality to be the main goals a matching algorithm can focus on, although other aspects, like performance can also be the goal. Some argue that striking a good balance between accuracy and generality of an algorithm greatly benefits the version control of models [33]. Our (informal) definitions for accuracy and generality are as follows:

– *Accuracy* the algorithm focuses on making the matching process accurate. This means finding as many correct matches and as few incorrect matches as possible. There are different metrics that can be used to measure accuracy. We investigate these metrics later in Sect. 4.2.
– *Generality* the algorithm focuses on being general, namely, that it can be used with as many modeling languages and tools, and with as few technology-related modifications to the algorithm as possible.

In Sect. 1, we mentioned that in addition to using their graph-based structure, models can also be matched, differenced, and merged using their textual representations. We consider a representation of a model text-based if (i) its notation is textual in nature and (ii) the representation is mapped to the model via a well-defined mechanism (i.e., a formal language with a parser [2]). We would like to stress that since models are inherently graph-based, their structure eventually has to be taken into account during the matching process. Therefore, we uphold the assumptions that raw text differencing tools [71] that are often used with source code, on their own, should not be used with models [30]. Instead, the textual representations represent another layer through which the matching and differencing processes are conducted. We do not consider the XML-based standard representation of a model (e.g., XMI [167]) to be a textual representation, unless the corresponding approach relies heavily on its textual nature, like when raw text differencing is used.

We consider a model matching algorithm to be text-based, if it uses the textual representations of the models during the matching process. Otherwise, we consider the algorithm to be graph-based.

3 Systematic literature review

Systematic literature reviews have a well-defined methodology that they follow, originating from the field of medicine [120]. Our survey also follows the recommendations proposed in the literature [25,78,79,122,168]. According to these recommendations, the process consists of the following phases:

1. *Planning phase* formulate the research questions, the search strategy, and define the inclusion and exclusion criteria (see Sect. 3.1).
2. *Conducting phase* carry out the survey based on the planning phase, and extract relevant data from the selected primary studies (see Sect. 3.2).
3. *Reporting phase* report and discuss the results of the survey, and draw conclusions (see Sect. 4).

The survey was conducted in an iterative fashion. The planning phase was constantly refined based on the results of the conducting phase. For example, when we found new synonyms of keywords we have not thought of before, we added those to the search string that we defined in the planning phase. Then, we examined the newly found studies found during the conducting phase, and if we found more keywords, we updated the search strings again. We conducted a similarly iterative process when we defined the inclusion and exclusion criteria, which we are discussing later.

3.1 Planning the survey

Planning is very important in systematic reviews, as the accuracy and repeatability of the process greatly depends on it. In this section, we outline our motivations for conducting this survey (see Sect. 3.1.1), define the scope and goals of the survey (see Sect. 3.1.2), formulate our research questions (see Sect. 3.1.3), document our search strategy (see Sect. 3.1.5), and list our inclusion and exclusion criteria (see Sect. 3.1.4).

3.1.1 Motivations

While there are existing surveys that focus on model matching in some form [9,30,134,146], Altmanninger et al. [9] focused on providing a feature-based characterization of version control systems, focusing more on the architecture of versioning systems. They also discuss three-way merging in more detail. Their findings include a comprehensive state-of-the-art of version control systems, along with challenges that need to be addressed before parallel model development can become more applied in practice. Selonen [134] reviewed UML model comparison techniques, and found that most approaches aimed toward change detection, but were lacking in change resolution. The survey established some criteria that model comparison approaches can be compared by, including identifier independence, customizability, reliability, extensibility, and so on, most of which we are also using in our survey. However, the paper only reviewed 5 UML-based model comparison approaches; thus, our paper is meant to be more extensive. Stephan and Cordy [146] on the different methods and applications of model comparison. They differentiated between different types of models (structural, behavioral, and so on), which we are also doing in our survey to some extent. Moreover, they extended the same categorization system that we did, introduced by Kolovos et al. [86]. While their discussion and categorization of existing approaches is extensive, they covered 30 approaches, while our survey aims to make deductions based on a bigger statistical sample, identified by a systematic method, and also focuses on text-based model matching algorithms. Brosch et al. [30] did a very extensive introduction to model versioning that we recommend to everyone who is interested in this research field. They introduce the basic concepts of model versioning, including model differencing and merging. They categorize some model differencing approaches also including conflict detection and resolution. The paper also touches upon some open questions in the field of model matching, which we are discussing later in this paper. The main differences of our paper compared to this survey are again the systematic nature of the search process, and the narrower and more specific focus on model matching and text-based algorithms.

Based on the most popular and biggest surveys we found in this research field that we just discussed, we concluded that there exist few surveys focus on the categorization of matching techniques. Moreover, we found none that used a systematic approach to identify its key studies, and also found none focusing on text-based model matching algorithms. Analyzing these matching techniques can help researchers in identifying research trends and possible directions for future research. Summarized, our main motivations behind conducting this survey are as follows:

- Model matching is an important task in state-based model differencing and is useful in other areas as well, i.e., in model transformation testing.
- There exist few surveys that focus on the categorization of model matching algorithms, especially considering text-based algorithms.

– Finding out the reason why the number of text-based model matching algorithms is lower, and how they compare with graph-based algorithms.
– Analyzing the aforementioned topics can help researchers in identifying trends and open questions in the field of model matching.

3.1.2 Scope and goals

Kitchenham et al. [78] proposed that the PICO guidelines be used in order to frame research questions and to define the scope of the survey. This system originates from the medical domain [120], and has been updated for other sciences [122] like software engineering [25]. The extended version is the PICOC criteria system. This consists of the following (the definitions are taken from the work of Kitchenham et al. [78]) criteria:

– The **population** represents who or what benefits from the SLR. This can be a specific software engineering role, a category of software engineers, an application area, or an industrial group.
– The **intervention** is a software methodology/tool/technology/procedure that addresses a specific issue.
– The **comparison** software methodology/tool/technology /procedure with which the intervention is being compared.
– **Outcomes** relate to factors of importance to practitioners such as improved reliability or reduced production cost.
– The **context** in which the comparison takes place (e.g., academia or industry), the participants taking part in the study (e.g., practitioners or academics), and the tasks being performed (e.g., small scale or large scale).

Our PICOC criteria can be found in Table 1. Based on the criteria and our motivations outlined in Sect. 3.1.1, the goals of this survey are as follows:

Table 1 The PICOC criteria used in the survey

Population	Researchers and practitioners using model-driven methodologies
Intervention	Analysis of existing matching approaches, with an additional focus on text-based model matching
Comparison	Differences between different matching techniques, and the comparison of text-based and graph-based approaches
Outcomes	Better comprehension of the state-of-the-art, the identification of current research trends and possible avenues for future research
Context	Academic research related to, and industrial applications of model matching

– Present the state-of-the-art of model matching algorithms.
– Discover the fundamental differences between matching techniques in order to identify research trends and future research directions.
– Conduct a survey on text-based model matching approaches, and compare them to graph-based ones.
– Identify open questions and challenges in the field of model matching.

3.1.3 Research questions

Defining the research questions is arguably the most important task during the planning of a systematic literature review. Research questions drive the survey process, as the goal of the process is finding the answers to them. Inclusion and exclusion criteria (see Sect. 3.1.4) are also partially derived from research questions. Based on our motivations and the presented goals and scope of this survey, we formulated some research questions that we aimed to answer. The goal of this survey is to provide answers to these questions. They are as follows:

– *RQ1* What is the state-of-the-art of model matching?

  RQ1.1 How common are graph-based and text-based model matching approaches? How do they compare with each other?
  RQ1.2 What model matching techniques are most often used? What are the main differences between them?
  RQ1.3 How often are state-based and operation-based change tracking used? What are the main differences between the two, with regard to the focus of the algorithms using them?

– *RQ2* How can model matching approaches be evaluated? What are the main considerations for evaluation?
– *RQ3* What are the main open questions related to the field of model matching?
– *RQ4* How has interest in model matching developed over time?

3.1.4 Inclusion and exclusion criteria

Inclusion and exclusion criteria are used to identify suitable primary studies and to exclude unsuitable ones during the search process. A candidate study always has to fulfill these criteria. Every inclusion criteria has to be fulfilled in order for a study to be included, but if one or more exclusion criteria are violated, the study is excluded. Table 2 contains our inclusion and exclusion criteria that we used during the search process.

Table 2 Study selection (inclusion and exclusion) criteria

Inclusion	Exclusion
Study is a peer-reviewed journal article, conference or workshop paper, or a tech report	Study is not a peer-reviewed journal article, conference or workshop paper, or a tech report (e.g., books, masters theses, doctoral dissertations)
Study is related to model-driven methodologies	Study is not related to model-driven methodologies
Study contains research that is related to model matching. The presented algorithm must be able to support the matching of models that conform to the same metamodel. Alternatively, the study focuses on the evaluation of, or on open questions related to model matching	Study does not contain research related to model matching, or it presents an algorithm that does not support the matching of models that conform to the same metamodel. Moreover, it does not focus on the evaluation of, or on open questions related to model matching
Study contains research that answers at least one of our research questions	Study is not related to any of our research questions
–	Study is not a primary study, for example, it is a survey. However, references of related secondary studies are checked during the snowballing process
–	Study is not written in English
–	Study is not available online (for the authors of this survey)

3.1.5 Search strategy

During the search process, we acquired relevant primary studies from the following sources: (i) manual search of certain conference and workshop proceedings, (ii) automatic search of popular digital libraries, and (iii) the snowballing process.

The manual search was conducted on the proceedings of conferences and workshops that we knew of and contained relevant studies. The reason we conducted the manual search is to have an additional starting point besides the automatic search. We aimed to expand these starting points via the snowballing process, in order to get as many relevant papers screened as we could. During the manual search, we screened every paper in every year of the following conference and workshop proceedings:

- MODELS (International Conference on Model Driven Engineering Languages and Systems)
- ME @MODELS (Models and Evolution Workshop) and its predecessors **MoDSE** and **MoDSE-MCCM**
- MODELSWARD (International Conference on Model-Driven Engineering and Software Development)
- CVSM (Comparison and Versioning of Software Models)

During the automatic search, we used the ACM Guide to Computing Literature and IEEE Xplore. They are the largest digital libraries, and the majority of surveys related to software engineering use these two [168]. The ACM Guide to Computing Literature covers most other digital libraries[1], but the search strings can return more unrelated results. The results of IEEE Xplore can be better focused with more specialized search strings, while covering less studies. Due to this reasoning, we believe that these two databases together, along with the previously mentioned manual search, adequately covers most existing literature. Therefore, we used different search strings for the two digital libraries, which are as follows:

- *ACM DL (+text +model +diff) OR ((keywords.author.keyword:(+model +match) OR keywords.author.keyword:(+model +merge)*

  OR keywords.author.keyword:(+model +difference)

  OR keywords.author.keyword:(+model +compare)

  OR keywords.author.keyword:(+model +comparison)

  OR keywords.author.keyword:(+model +versioning)

  OR keywords.author.keyword:(+model +composition)

  OR keywords.author.keyword:(+model +union))

  AND domain-specific model-driven model-based UML)
- *IEEE Xplore ((model NEAR/2 differencing OR model NEAR/2 comparing OR "model comparison" OR model NEAR/2 matching OR model NEAR/2 merging OR "model composition" OR model NEAR/2 versioning OR model NEAR/2 union) AND ("UML" OR "model-driven" OR "model-based engineering" OR "domain-specific"))*

[1] The ACM Guide to Computing Literature contains 1,406,570 as of the writing of this paper, see https://libraries.acm.org/digital-library/acm-guide-to-computing-literature.

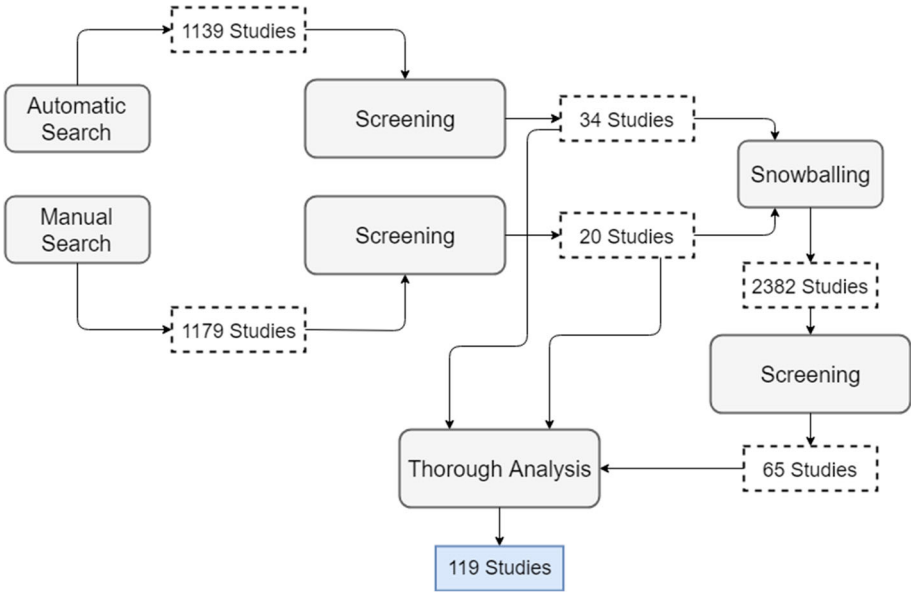

Fig. 3 Overview of the study selection process

The search strings were formulated according to our research questions, using terms and keywords found in previously known primary studies. The search strings were also continuously refined during the search process. It is worth noting that due to the search term limit of IEEE Xplore, the related search string was split into multiple search strings when we used it in practice. Keywords like *model merging* or *model composition* were included because during the search process, we realized that research related to these topics often deal with model matching in some form.

The last component of our search strategy is the snowballing process. The goal of the snowballing process [163] is to analyze the reference list of every included study from the manual and automatic search processes, and screen them using the same inclusion and exclusion criteria. The process is then repeated with the newly included studies, until we can no longer include new studies. The main goal of snowballing is to include related studies that our search strings might have missed, but the included primary studies reference them (e.g., other related studies, or follow-up studies).

3.2 Conducting the survey

The second phase of the survey is conducting the search based on the planning phase. We detail our study selection process with results displayed for each phase and present the structure of the data extracted from the studies.

3.2.1 Study selection process

The study selection (search) process was conducted based on the planning presented in Sect. 3.1.5. Figure 3 summarizes this. First, the automatic and manual searches were conducted (they can be run independently), which resulted in 34 and 20 included studies, respectively. The snowballing process was started with these 54 studies and finally resulted in an additional 65 included primary studies. Snowballing is an essential part of our study selection process, as over half of the included studies were included during this process. The screening process for a candidate study is depicted in Fig. 4. This process was applied to every study that we found during the manual, automatic, and snowballing processes. First, we read the title and abstract of the study, and excluded it if the inclusion and exclusion criteria were violated. After that, we read the full paper, and if the study was a related secondary study (namely, a survey), we included it for snowballing, but not as a primary study. Otherwise, if, after reading the full paper, the study fulfilled our inclusion and exclusion criteria, we included it as a primary study. Thus, the second inclusion and exclusion criteria check is there to check whether the study still fulfills the criteria after reading the paper.

Table 3 contains the detailed results of the entire study selection process. To sum it up, out of the 4693 studies examined, 1419 were duplicates, 2930 were excluded based on their title and abstract, and 219 were excluded after analyzing the paper. At the end of the process, 119 studies were included, with an additional 6 papers categorized as surveys (secondary studies). These surveys are not classified as primary studies, but their references were all checked during the snowballing process. We would like to note that for the sake of clarity, references examined during the snowballing process do not include studies that are not available online[2] and

[2] This criterion was applied in a total of 39 cases (of which 22 was during the snowballing process) and mostly consisted of situations when the paper in question was no longer available for anyone, as the web-

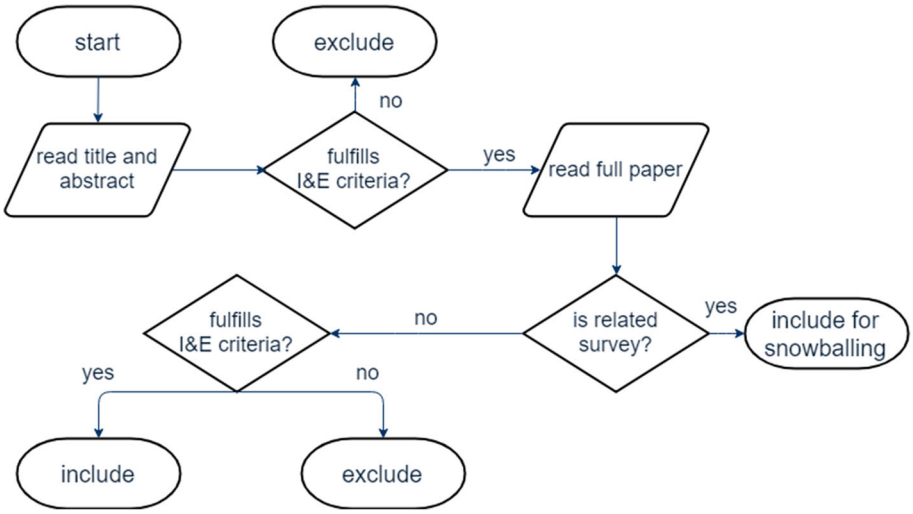

Fig. 4 The screening process for one study

Table 3 Summary of the study selection process

	Automatic search	Manual search	Snowballing	Overall
Total	1139	1179	2375	4693
Included	34	20	65	119
Surveys	0	1	5	6
Duplicates	95	15	1309	1419
Excluded (soft)	945	1102	883	2930
Excluded (hard)	65	41	113	219

Table 4 Extracted data structure with possible values

Type (RQ1.1)	Graph/text
Matching (RQ1.2)	Static/dynamic/similarity/custom
Change tracking (RQ1.3)	Operation-based/state-based
Focus (RQ1)	Accuracy/generality/both
Evaluation (RQ2)	Yes/no
Open questions (RQ3)	Yes/no
Year (RQ5)	–
Author(s)	–
Version control systems	Yes/no

are not peer-reviewed journal articles, conference or workshop papers, or tech reports.

3.2.2 Extracted data

For every study in the 119 included studies, we extracted relevant data that we used to answer our research questions. Table 4 summarizes the structure of the extracted data together with the related research questions and the potential values each data type column can take. In addition to the values depicted there, every data type can also take on the values *unknown* (when we could not decipher the value based on the paper), *other* (when the value is not of one listed in the table, but otherwise valid) and *none*. The detailed explanations for the potential values are as follows:

- *Type (RQ1.1)* The main type of the matching algorithm, namely, whether it works using the graph-based structure or the textual representation of the models. Our definition of a textual representation can be found in Sect. 2.
- *Matching (RQ1.2)* The matching technique used by the algorithm. We use the categorization proposed by Kolovos et al. [86] (see Sect. 2 for details).
- *Change tracking (RQ1.3)* State-based or operation-based change tracking, as discussed in Sect. 2.
- *Focus (RQ1)* The main goal and focus of the algorithm. We consider the main ones to be accuracy and generality (see Sect. 2 for details). If an algorithm focuses on both of them, the possible trade-off for doing so is also marked during data extraction. An algorithm can also focus on other aspects, like the performance or scalability of the algorithm.
- *Evaluation (RQ2)* Whether the study contains information on the evaluation of model matching approaches.
- *Open Questions (RQ3)* Whether the study contains information regarding open questions in the field of model matching. This excludes open questions that are only related to the presented algorithm in the paper, like specific improvements to the mentioned algorithm.
- *Year (RQ4)* The year the study was published.
- *Author(s)* The author(s) of the paper. Also used for identifying studies that discuss the same research work (see Sect. 4 and Table 5).
- *Version Control Systems* Whether the study focuses on model matching in the context of version control systems.

site containing it ceased to function. There were no significant major publication sources that were not included for this reason.

Table 5 Primary studies discussing the same algorithms

Algorithm	Studies
AMOR	S6, S18, S111
Chaouni et al.	S16, S63
Dijkman et al.	S36, S48, S64, S66, S103, S106
Ehrig et al.	S65, S97
EMFCompare	S81, S90, S98
Epsilon merging language	S68, S78, S116
Fleurey et al.	S7, S80, S92
Gerth et al.	S33, S34, S35, S56
Koegel et al.	S87, S88, S110, S112
Nejati et al.	S58, S59
Oda and Saeki	S52, S74
Ohst et al.	S42, S71
Oliveira et al.	S2, S76, S86
SiDiff	S9, S17, S39, S43, S113
Somogyi and Asztalos	S50, S70
UMLDiff	S44, S115
Zhang et al.	S26, S27

These data are only needed to reinforce our conception of version control systems being one of the primary motivating factors for researching model matching.

4 Results and discussion

In this section, we present the reporting phase of our survey. It contains the presentation and discussion of our results, namely, the answers to every research question formulated in Sect. 3.1.3. For a full list of included primary studies, please refer to Tables 6 and 7 that list every primary study with associated extracted data (see Sect. 3.2.2 for details). We included 119 primary studies and 6 surveys during the conduction of our survey. A total of 105 studies contained information regarding our first research question (RQ1), which means that the mentioned studies discuss a model matching algorithm. There are some research works that are split into multiple studies, for example, with the discussed algorithm being continually improved over time. The list of these algorithms can be found in Table 5. In order to normalize the effect these studies have in our analysis, we have decided to handle these papers as one study. We also updated the extracted data for every related study where the extracted values differed. For example, when the algorithm was improved over time (i.e., its focus changed from only accuracy to both accuracy and generality), we use the improved version in our categorization. At the end of this process, the number of studies containing answers to RQ1 was reduced to 72 from 105.

As for the statement that one of the primary motivations behind model matching is supporting version control, we found that 70 out of the included 119 primary studies mention it as one of the motivations behind their research. Thus, we believe this statement is mostly accurate. It is interesting to note that studies dealing with model composition (of which model matching can be a part of) and behavioral models were less likely to include the support of version control as one of their motivations.

4.1 State-of-the-art (RQ1)

Our first research question is aimed at identifying the state-of-the-art in model matching techniques. We split this question into multiple subquestions, all of which we are answering here: (i) comparison of text-based matching to graph-based matching, (ii) discovery of differences between main matching techniques, and the (iii) analysis of change tracking approaches, with regard to the focus of the algorithm.

4.1.1 Text-based model matching (RQ1.1)

RQ1.1: How common are graph-based and text-based model matching approaches? How do they compare with each other?

The studies we found often refer to text-based model matching as either raw text differencing [19,162,170] (e.g., GNU Diffutils [98] or KDiff [71]), or as to the XML-based serialization of the models [30,146] (e.g., XMI [167]). The former has very low accuracy in the case of models due to structural differences (compared with source code), while the latter is often dismissed due to its low abstraction level, which also makes it less accurate. In Sect. 2, we defined the textual representation of a model as a textual notation that is mapped to the model by well-defined mechanisms. We also excluded XML-based serialization from this definition. During the categorization of the studies, we used this definition to label algorithms as text-based. As we have discussed before, there are certain advantages of using the textual notation, like supporting off-line editing, better comprehensibility when focusing on details, or when compared with XMI [67,68,144].

Out of the 72 related primary studies, the number of text-based algorithms (8) is significantly lower than the number of graph-based algoritms (64). We believe that this 11% to 89% ratio is notable enough to conclude that text-based algorithms are significantly more rare than graph-based ones in the existing literature. In the following, we give a brief summary of every included text-based algorithm and then finish by comparing text-based algorithms to graph-based ones based on our findings.

Badreddin et al. [16] presented an approach where model matching is based on a traditional textual version control system (SVN [149]). The models are described in a textual form by Umple [58], which is a modeling tool and programming language family that aims at combining modeling and programming with each other. Inconsistencies (conflicting differences) are managed by the Umple editor. The main goal of the approach is to achieve the uniform versioning of model and code, for which the most feasible solution is to version models in their textual form. The authors succeeded in connecting raw text differencing with model-driven engineering. However, the difference representation is at a low abstraction level (per text-line); thus, the differences are more difficult to comprehend by the user during the merge process.

The algorithm presented by Alwanain et al. [11] focuses on the automated composition of sequence diagrams. Sequence diagrams are behavioral models, which differ from static models in that they are often more sensitive to model semantics when differenced and merged. The models used by the algorithm are described by Alloy [70], which is a textual language for describing structural properties. Alloy provides a modeling tool that uses first-order logic to define properties. Model matching is based on defining facts in Alloy, which is a dynamic approach to model matching. The main focus of the algorithm is accuracy, as it was specifically developed for sequence diagrams.

Maoz et al. [101] proposed *cddiff*, a semantic differencing operator for UML class diagrams. Similarly to Alwanain et al. [11], the models are described by Alloy. The matching is done using a custom, language-specific algorithm that heavily uses the semantics of class diagrams. The algorithm focuses on accuracy, although a future goal of the authors is "to apply the idea of semantic differencing and the computation of diff witnesses to other modeling languages as well".

Foucault et al. [59] aimed to enhance traditional text-based version control (more specifically, git [54]) with domain-specific semantics. The textual representations are described by a textual domain-specific modeling language using EBNF, and the matching is done based on static identifiers. The algorithm is integrated in a model-driven engineering simulation platform. The algorithm aims to achieve both accuracy and generality, although the use of static identifiers limits the latter.

The algorithm presented by Somogyi and Asztalos [136, 137] (previous work of the authors of this survey) differences and merges models based on their textual representations. The textual notation is described by an arbitrary textual language and must be mapped to the model by an associated parser [2]. The matching is done with a heavy reliance on the parser. The parser has to satisfy a number of requirements by the algorithm, which requires some effort to configure; however, in return, the algorithm can be both accurate and general. Thus, the algorithm aims to achieve both accuracy and generality, at the cost of fulfilling practical requirements that the parser has to satisfy.

The algorithm proposed by Barrett et al. [19] focuses on merging use cases models that are given in a textual form. Textual use cases are usually described in a natural language. The algorithm is operation-based, and as such, it records changes as they occur. Thus, there is no need for explicit model matching. Finite state machines are constructed from the textual representation, and then used during the merging process. The authors focus on the accuracy of merging use cases models given in a textual form—a task for which there was no proposed solution before—and present a unique algorithm for solving the problem.

TMDiff, proposed by van Rozen and van der Storm [157], is an algorithm that uses standard textual differencing during the origin tracking between the textual representation and the model. The algorithm is also operation-based. Therefore, while explicit matching is not needed due to the operation-based change tracking used by the algorithm, raw text differencing is still used as a means of origin tracking, namely, the mapping mechanism between text and model. The algorithm is implemented in Rascal [80], and the text is parsed into an abstract syntax tree by a parser [2]. The authors mention an interesting problem related to text-based model matching. Namely, non-semantic information (e.g., comments) in the text may influence text-based algorithms, and have to be handled appropriately.

Rivera and Vallecillo [127] chose to represent models in a textual form with Maude, which is a "high-level interpreter and compiler in the OBJ algebraic specification family [128]". The matching in their algorithm is based on both static identifiers and similarity metrics. Their algorithm also supports the models having different metamodels, and is highly configurable. The focus of the algorithm is both accuracy and generality, which comes at the cost of specifying the Maude-based representation of the models, and configuring the algorithm.

According to our findings, 3 text-based algorithms focused on both accuracy and generality, 4 algorithms focused only on accuracy, while 1 algorithm had special goals in mind (Badreddin et al. [16]). Aiming for both accuracy and generality often comes with the compromise of needing to map the textual representation for different modeling languages, and also with the configuration cost of the algorithm itself. We have seen examples of this configuration cost in two out of three algorithms that aimed to be both accurate and general at the same time [127,136,137]. Foucault et al. [59] is the exception to this, as their algorithm aims to be general by using static identifiers, which limits its practical applicability in the case of more complex modeling languages. We did not find a text-based algorithm which focuses only on generality. Raw text differencing might be considered to be such an approach, but due to the reasons explained in Sect. 2, such approaches

are not considered to be sufficient for graph-based models. Since the textual notation of a model is often considered a second-hand notation next to the graphical one, the effort required to use these algorithms with a new modeling language that do not already support the textual representation can be considerably high. However, if this problem can be efficiently solved, then text-based algorithms can provide an alternative to graph-based ones, as they can be easily integrated into existing version control systems, further bridging the gap between modeling and coding.

Graph-based and text-based algorithms can be successfully combined [127]. In addition to the textual representations, text-based algorithms still have to take into account the internal graph-based representation of the models in order to preserve accuracy and a high level of abstraction when calculating and presenting differences. Thus, often, text-based model matching algorithms can be viewed as an extension of graph-based algorithms. In addition, they also have the added benefit of supporting text-based modeling by solving the model matching problem on the textual representations [67].

As a final note, we found no text-based algorithms that relied on similarity-based matching, but found examples for every other matching technique. This can be explained by the fact that text-based algorithms often rely on the underlying mechanism that maps the model to text. This usually requires a higher configuration cost (dynamic matching), but there is no need for similarity-based matching if the underlying mechanism can give an exact answer to the question of whether or not two model elements are considered to be a match.

Due to space limitations, we do not summarize each of the 64 graph-based studies we included in our survey. Instead, we briefly summarize our findings. Figure 5 contains the summary of graph-based algorithms with regard to their focus. Statistically, we can conclude that graph-based algorithms are more varied than text-based ones. In contrast to text-based algorithms, similarity-based approaches are also more common among graph-based algorithms. Some of these graph-based algorithms will be discussed in more detail in later sections (e.g., when discussing open questions), as they have increased relevancy there.

Summarized, our key findings on comparing text-based model matching algorithms with graph-based ones are as follows:

- *Lack of text-based algorithms* There is a lack of text-based model matching algorithms, and an even greater lack of general text-based algorithms.
- *Motivations behind using text-based algorithms* The main motivations we found behind text-based model matching are (i) using the simplicity of raw text differencing and enhancing it to be suitable for models [16,59,157], (ii) aiming for better accuracy at a higher configuration cost [127,136], and (iii) supporting the textual notation of the models by solving a related, specific problem [19,157].
- *Generality comes with configuration cost* Text-based model matching algorithms tend to focus either on accuracy, or on both accuracy and generality. This often comes with a compromise in the form of an increased configuration cost of the algorithm and the mapping of the textual representation. Only when these costs are acceptable that text-based algorithms can be reliable used in a more general way in practice. Thus, they are either accurate (usually with a single language in mind), or have a higher configuration cost to be used generally. However, more research is needed on the analysis of the trade-off between the configuration cost and the generality/accuracy of the text-based algorithm, as no study we found dealt with this problem in detail. As a final note, the idea of supporting both accuracy and generality at a higher configuration cost is also present in graph-based algorithms.
- *Quantifying the configuration cost?* What metrics can we use to express the effort required to use a text-based matching algorithm with a particular modeling language? Can it be expressed using standard metrics? The studies we found did not discuss this problem.

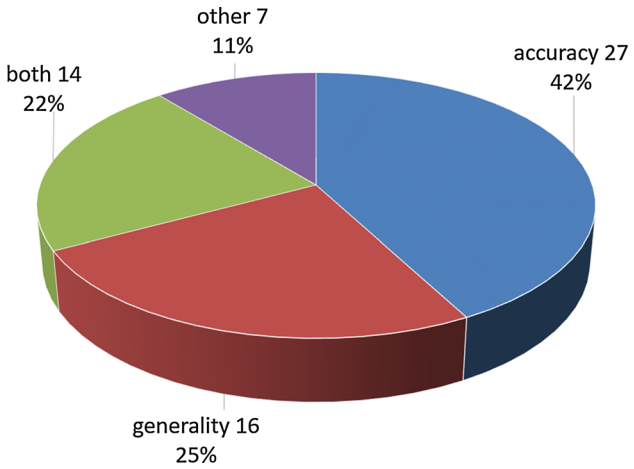

Fig. 5 Summary of the focus of graph-based algorithms

4.1.2 Matching techniques (RQ1.2)

RQ1.2: What model matching techniques are most often used? What are the main differences between them?

Figure 6 summarizes the main matching techniques used by the algorithms we found, according to the categorization detailed in Sect. 2. We would like to note that the sum of the

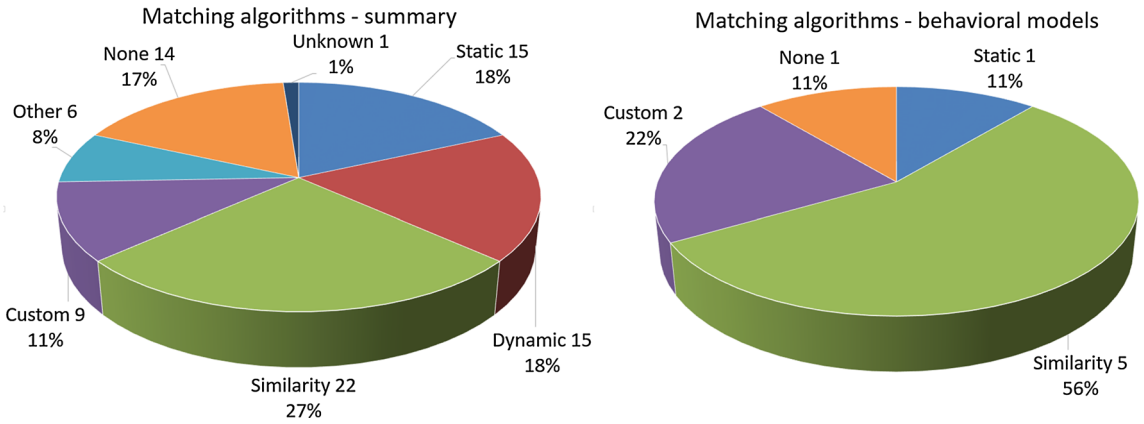

Fig. 6 Summary of the matching techniques used by identified model matching algorithms

numbers on the figure is greater than the number of unique algorithms. This is due to the fact that algorithms using more than one matching technique are counted multiple times for every used technique. We can see that similarity-based matching is the most popular, followed by static and dynamic matching. Operation-based algorithms usually use no matching technique (hence the category *none* on the figure). Please note that the number of operation-based algorithms (15, as detailed in Sect. 4.1.3) is higher than the number of algorithms using no matching technique (14), as one particular operation-based algorithm we found [157] uses raw text matching as a form of origin tracking between model and text.

We identified six studies that used matching techniques that did not fit into our categorization. Schwägerl et al. [132], van den Brand et al. [159], Kehrer and Kelter [73], and Barrett et al. [17] all use configurable matching, which means that any matching technique can be used as long as it fulfills specific criteria. Badreddin et al. [16], along with van Rozen and van der Storm [157], use raw text differencing in some form. Thus, there were two categories that did not fit our existing categorization based on the work of Kolovos et al. [86]: (i) configurable matching algorithms, and (ii) raw text differencing.

Figure 6 also presents matching techniques used by algorithms working with behavioral models. Behavioral models describe a process, like state charts [75,108], or business process models (BPM) [49,50,93]. We found that the semantics of the models greatly influence the model matching, as opposed to static models. In the case of behavioral models, similarity-based matching is the most popular. Moreover, algorithms focusing on behavioral models tend to focus on accuracy instead of generality.

Figure 7 depicts the different matching techniques with regard to the main focus of the algorithm. Static matching is often used in achieving both accuracy and generality, although the use of static identifiers usually severely limits the latter. Similarity-based matching is often employed when the accuracy of the algorithm is the main focus. It is also worth mentioning that behavioral algorithms contribute greatly to this statistic, as seen in Fig. 6. In the case of dynamic matching, custom, and other algorithms, we found no distinguishable correlations.

4.1.3 Change tracking (RQ1.3)

> RQ1.3: How often are state-based and operation-based change tracking used? What are the main differences between the two, with regard to the focus of the algorithms using them?

Out of the 72 algorithms the survey identified, the majority (75%) were state-based, some were operation-based (~21%), while a small number of algorithms (~4%) used both change tracking approaches. Figure 8 depicts the main focus of state-based and operation-based algorithms. Contrary to our preliminary expectations, the operation-based algorithms we found are split evenly with regard to the focus of the algorithm. State-based algorithms tend to focus more often on accuracy (50%). Thus, in general, we can draw two main conclusions from these results: (i) state-based approaches are more commonly used, and (ii) there are no significant correlations between the used change tracking method and the main focus of the algorithm.

Regarding behavioral models, Soto and Münch [140] argue that operation-based approaches are not suitable for process models, stressing that manual change logs are especially difficult to use. Our findings support their claims, as the number of state-based behavioral algorithms (7) we found significantly outweighs the number of operation-based (1) algorithms, even if we count the number of approaches where both change tracking methods were used (1).

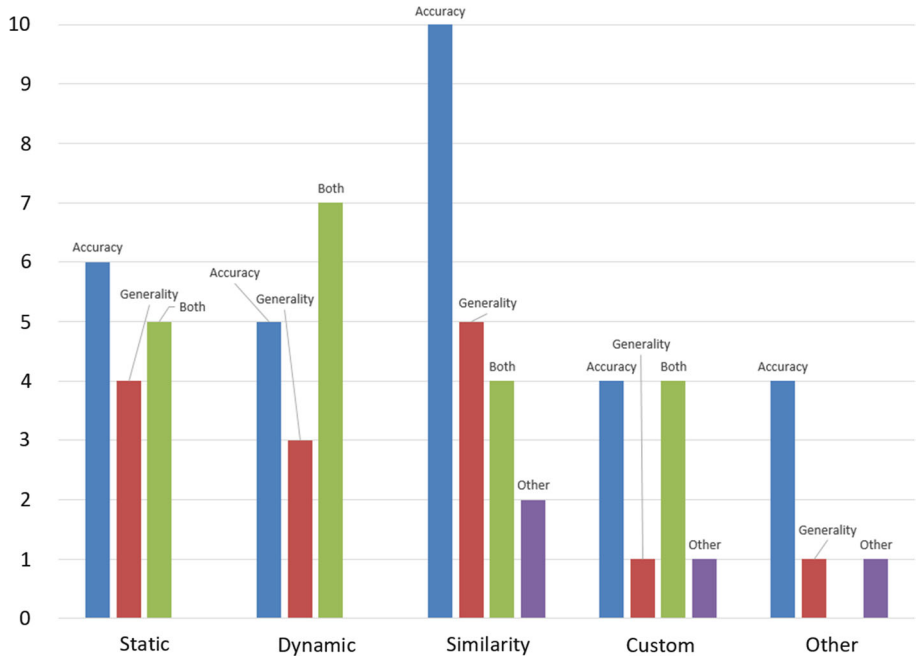

Fig. 7 Correlations between matching types and the main focus of the algorithm

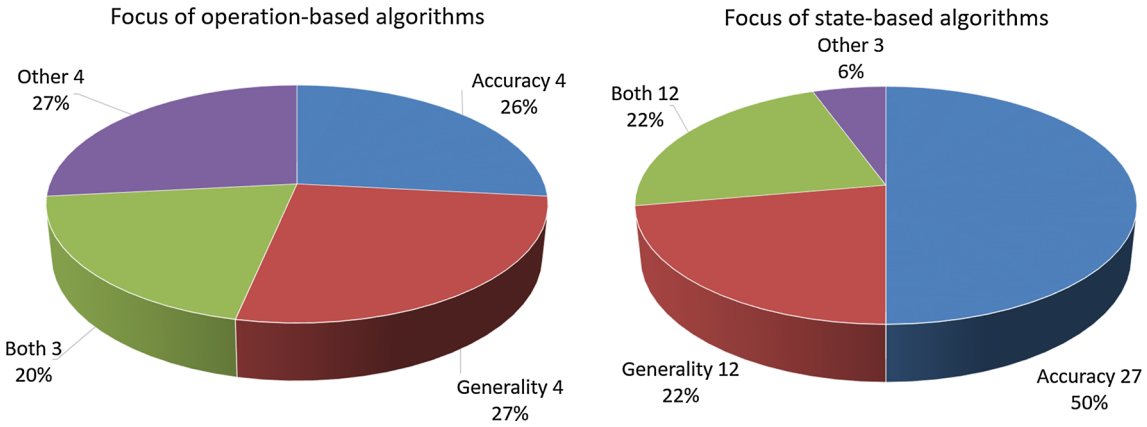

Fig. 8 Summary of the main focus of operation-based and state-based algorithms

4.2 Evaluation techniques (RQ2)

RQ2: How can model matching approaches be evaluated? What are the main considerations for evaluation?

The evaluation and benchmarking of model matching, differencing, and merging algorithms is considered to be a difficult task in the literature [8,152]. Most approaches that we found focus on the accuracy of the matching process in some way [1,4–7,27,44,47,48,108,109,152,158,161,169]. Metrics like precision, recall, or *F*-measure that are known from pattern recognition and machine learning [26] are often used to measure accuracy. Precision is usually the measure of exactness or fidelity; it is equivalent to the ratio of correctly detected matching pairs to all detected pairs. Recall is often used as the measure of completeness; it is equivalent to the ratio of correctly detected matching pairs to the number of all correct matching pairs. Instead of these values, the mean average precision and recall are often used [47,48]. These metrics are regularly used in the case of structural models, but they can also be defined for behavioral models. In addition to being used to measure matching accuracy, these metrics can also be used for measuring the accuracy of conflict detection or resolution. The concept is similar, finding the ratio of correctly identified matching pairs or conflicts [169]. As a final note on the topic of conflict detection, Koegel et al. [84] evaluated operation-based conflict detection approaches by running the changes in a batch-mode on predefined models, and evaluating their results manually. Conflicting and non-conflicting changes were separated in their approach.

Another common consideration regarding the evaluation of matching algorithms is the runtime of the algorithm [1,47,152,158]. Scalability—based on the number of model elements in the input models—is also an important factor that mostly influences large models. Runtime performance can be expressed with a simple time metric, i.e., in milliseconds. Scalability can be expressed by using space and time complexity [14], which, similarly to other fields where algorithms are considered, are used in the field of model matching as well [108,109]. As with most algorithms in computer science, the dilemma of whether to focus on accuracy or performance is also present with model matching. This is especially relevant when matching large-scale models, although few studies that we found focused on this problem [151].

Benchmarking algorithms is considered to be an area where further research is required in this field [8,123]. Having standardized benchmark model datasets that can be used to evaluate algorithms is beneficial to researchers, as it is then possible to compare different algorithms objectively, regardless of their inner workings or the technology used to implement the algorithm. Unfortunately, there are very few such benchmark datasets available. It would also be useful if such benchmark datasets could automatically be generated instead of creating them manually, especially in the case of large-scale test models. The benchmark dataset used by van den Brand et al. [158] contains both manually and automatically created EMF [145] models. Using the datasets, the authors successfully compare EMFCompare [111] and RCVDiff [156] with each other, based on performance and accuracy. However, using the dataset with algorithms that are incompatible with EMF models by default would be difficult, as the implementation is technology dependent. Dijkman et al. [47] used the SAP reference model, while Koegel et al. [84] used two manually defined real projects during their evaluation. Both datasets have the same problems when we try to extend them for more general use—they are technology dependent. Dijkman et al. [44,47] compared different matching algorithms used with business process models, based on accuracy and performance. Both experiments concluded that the greedy algorithm was the best in both accuracy and performance. The testing models were derived from the local government domain and from the SAP reference model. To conclude, the automatic generation of sufficient benchmark datasets remains to be a problem in the field of model matching.

Another way we can evaluate matching algorithms is the conduction of user studies in order to gather empirical information on various aspects of an algorithm. Melnik et al. [104] carried out a user study in order to find out how the different parameters of their algorithm affected matching results, and how much overall manual labor was saved when using the algorithm. De Lucia et al. [97] conducted an empirical study carried out by students in order to evaluate the effectiveness of their algorithm with regard to the automatic conflict resolution and the manual intervention required. Although proven to be efficient in evaluating certain aspects of algorithms that would otherwise be difficult to quantify, we found that empirical user studies are few in number in the literature.

We identified frameworks whose purpose is the evaluation of model matching/differencing/merging algorithms [62,152]. MCTest [62] evaluates Eclipse-based model matching algorithms. The framework is implemented in Java, highly configurable, and homogenizes results for every approach. The accuracy and performance of the algorithms can be evaluated with the framework. The framework proposed by Uhrig and Schwägerl [152] is also an Eclipse-based framework for the evaluation of matching algorithms. The framework accounts for user involvement, has automatic error calculation (based on a manually defined match model), and evaluates the accuracy and performance of the matching algorithms.

Although slightly out-of-scope for our survey, evaluating the correctness of the merge process is a direction that some approaches take [108]. This evaluation often includes mapping the model to a mathematical formalism, i.e., mapping behavioral models to labeled transition systems (LTS). On the topic of merging, La Rosa et al. [93] examined some basic properties of their merge operator: idempotence, commutativity, and associativity. They also analyzed their compression factor (size of the merged model compared with the input models), and evaluated the scalability of the merging process as opposed to manual man-hours needed to accomplish the merge.

To sum it up, the two main problems we found regarding evaluation during our survey are (i) the technology dependence of benchmark data sets and evaluation tools (which are mostly Eclipse-based), and (ii) the manual verification of the result of the matching algorithm compared with the optimal result [44,50,62,152,161]. The latter is especially problematic in the case of large-scale models, for which we found a lack of solutions regarding evaluation.

Summarized, our key findings on the evaluation of model matching algorithms are as follows:

- *Measuring metrics are established* Matching accuracy is often used to evaluate matching algorithms. It can be measured with metrics borrowed from pattern recognition: precision, recall, and F-score. The accuracy of conflict detection and resolution can also be measured using these metrics. Performance and scalability are also common factors, although large-scale models are somewhat neglected in this regard.
- *Lack of automated benchmarking* Benchmarking is often mentioned as one of the main challenges in the field of model matching. We found that there is a need for the

automatic generation of technology-independent benchmarking model datasets.
- *Lack of empirical studies* Empirical user studies are rare, although they can be used to verify certain aspects of an algorithm (e.g., human effort required to use) that would otherwise be difficult to quantify.
- *Lack of technology-independent evaluation frameworks* There are some existing evaluation frameworks for model matching algorithms, but they are usually technology dependent [62,152].

4.3 Open questions (RQ3)

RQ3: What are the main open questions related to the field of model matching?

In the earlier years of research in model differencing and merging, there was a clear need for general algorithms that did not rely on static identifiers [3,8]. Our findings confirmed that this is no longer the case, as there are numerous algorithms that focus only on generality and do not rely on static identifiers (16 that we found, see Sect. 4.1.2 for details).

Although loosely connected to model matching, the most frequently mentioned open questions that we found were (i) the lack of reliable automatic conflict resolution [3,8,16,33] and (ii) the need for better difference and conflict visualization [8,33,95,102]. The former problem is important, since introducing manual user choice can make the algorithm be more prone to errors, as user input is now a factor. Although it can be argued that knowledge from domain experts is useful in supplementing the algorithm, a more automatic, and also reliable conflict resolution can greatly ease the process. However, a faulty automatic conflict resolution mechanism can lead to inconsistencies in the model. Therefore, reliable and accurate automatic conflict resolution is important in model differencing and merging. A frequently mentioned solution to the latter problem is the specification of differences and conflicts in the concrete syntax of the model in question. Our findings confirmed that these problems are still open questions, as we found few approaches that focused on them [37,97,114,142].

Brosch et al. [30] conducted an extensive survey that serves as an introduction to the model versioning research field. The main open questions they identified are (i) intention-aware and (ii) semantics-aware model versioning. The former problem is related to model merging. The authors argue that ideally, the merged version should represent the summary of the intentions each developer had in mind when performing their operations. Essentially, this means the preservation of the original goals the developers had, instead of merely automatically combining every non-conflicting operation. The solution proposed by the authors is detecting the application of composite operations like refactoring, allowing developers to annotate intention. Moreover, Barrett et al. [18] also argue that there is no clear definition on what should happen during model merge. The second problem states that non-conflicting syntactical operations can still cause semantical conflicts (e.g., deadlocks in behavioral models). Also, at the time, there were no currently accepted formal semantics for often-used languages like UML. [3] Moreover, the authors argue that existing approaches are specific to modeling languages and language constructs.

Semantic model matching and differencing is a relatively new research direction [10,55,99–102,119]. It tries to incorporate the semantic meaning of the model elements during the matching process. Maoz et al. [102] identified several open questions related to semantic model differencing: (i) semantic diff witness computation is algorithmically difficult and the witnesses may be sound, but incomplete, (ii) diff witness presentation is language-specific (see previous open questions), (iii) integration with syntactic differencing is lacking [95]. As for the integration, the authors propose that semantic matching should use the results of the previously ran syntactic matching in order to localize and improve its performance.

Badreddin et al. [16] identified several open questions and challenges in the field of model matching. They are as follows: (i) model entities and diagram layouts are not separated sufficiently, (ii) low tool adoption of existing approaches (lack of mature tools), (iii) existing approaches are not suitable for large-scale models (few approaches consider them), and (iv) the non-determinism and unpredictability of matching, especially for similarity-based approaches. The first problem refers to standard model representations (i.e., XMI [167]) containing diagram-specific information, like the coordinates of model elements on the diagram. This could inherently influence the matching process. A way to solve this is to avoid using only the XMI representation for the matching process. For example, the algorithm could work with an abstract graph built from the XMI that omits diagram layout data, or with a structure built from a textual representation. The second problem, namely, lack of tool support, is still a relevant problem. The proposed algorithms usually have a working prototype, but according to our findings, industrial tools are much more rare. The third problem, large-scale model matching, is still an open question. Since the graph isomorphism problem is NP-complete [81], algorithms that deal with large-scale models have to be heavily optimized [34,151]. This results in a heavy focus on performance, which could come at the cost of accuracy. The last stated problem is the non-determinism of existing approaches. As the authors stated, this often

[3] Regarding UML, this is no longer the case, see https://www.omg.org/spec/FUML/.

occurs when the matching algorithms are configurable or similarity-based. Indeed, two separate configurations can lead to different results on the same input. However, it can also be argued that the strength of these approaches is in their high degree of configuration, which makes them more general. Therefore, we do not consider this a huge problem.

The question of how we can measure the quality of model differences can arise, i.e., what can be considered a "good" model difference [123]. There is also a clear need of standardized benchmark model sets [8,123], as we have discussed in Sect. 4.2.

We examined the differences between the matching of structural and behavioral models in Sect. 4.1.2. In the case of structural models, a wide variety of matching approaches were used in the primary studies we found. However, in the case of behavioral models, we found that general approaches are much more rare. We concluded that the reason for this is that behavioral models rely on semantics more than structural models [75]. Thus, more specific matching algorithms are needed for behavioral models in order to achieve an acceptable degree of accuracy. Soto and Münch [140] identified several open questions related to process model matching and differencing. They are as follows: (i) filtering and representing results for multiple user groups (i.e., developers, managers) is difficult, (ii) process models are often specific to organizations; thus, algorithms have to respect their schema, and (iii) the problem of whether state-based or operation-based approaches are better in the case of process models. The first question is similar to the difference representation found in the case of static models. The second problem further reinforces the fact that general approaches are less useful in the case of behavioral models. We discussed the third problem in Sect. 4.1.3, during our discussion on change tracking. As a final note on process model matching, Soto and Münch also argue that raw text differencing is useful for process models, but only for certain parts of the model, like long text-based descriptions.

Finally, we found the following recurring questions that we aimed to answer in our survey: (i) what criteria can be used to identify correspondences between models, and how these criteria can be quantified [55], (ii) how should model elements be identified [18], (iii) what are the fundamental limitations of the matching process (i.e., complexity) [95], and (iv) are published model differencing approaches substantially different [95]. We believe that the answers to our research questions in this section cover these questions.

Summarized, the major open questions and challenges in the field of model matching that we identified are as follows:

- *Relatively low industrial appliance* Low tool adoption of existing algorithms. Most algorithms have a working prototype, although industrial applications are rare.
- *Inadequate change visualization* Although loosely related to the field of model matching, difference and conflict visualization is not adequate enough, the concrete syntax of the model should more often be used for this purpose.
- *Intention-aware model versioning* Intention-aware model versioning is a possible future research direction where the result of a model merge is based on the intentions of the developers committing the changes.
- *Semantic model matching* Semantic model matching is a relatively new research direction with multiple problems: algorithmic complexity, difference representation, and integration with syntactic matching.
- *Lack of benchmarking datasets* Benchmarking datasets are uncommon and are usually technology dependent. Automatically generated and technology-independent datasets are needed in order to more objectively evaluate algorithms.
- *Large-scale model matching* Large-scale model matching needs more research. The trade-off between performance and accuracy also merits further research.
- *General algorithms for behavioral models?* General model matching algorithms do not seem to perform well regarding accuracy, as behavioral models seem to rely too heavily on semantics. Can more general algorithms be developed for behavioral models?
- *Further analysis of text-based algorithms* The investigation of the differences between text-based and graph-based algorithms, and identifying and quantifying the aspects that they differ in merits further research. This also includes the analysis of the correlation between the configuration cost and accuracy/generality of the text-based algorithm discussed in Sect. 4.1.1.

4.4 Interest in model matching (RQ4)

RQ4: How has interest in model matching developed over time?

Figure 9 illustrates every included primary study per year, from the earliest published paper (2001) to the latest one (2018). We can see that the apex of research related to model matching was between 2007 (16 studies) and 2009 (20 studies). From then, a steady decline in the frequency of published papers can be observed. Nevertheless, we believe that the open questions and challenges we identified in this survey are still relevant and merit further research.

Figure 10 contains every graph-based and text-based algorithm, on a per-year basis. This excludes primary studies that do not present a particular algorithm. Text-based algorithms are significantly fewer in number. They started to emerge

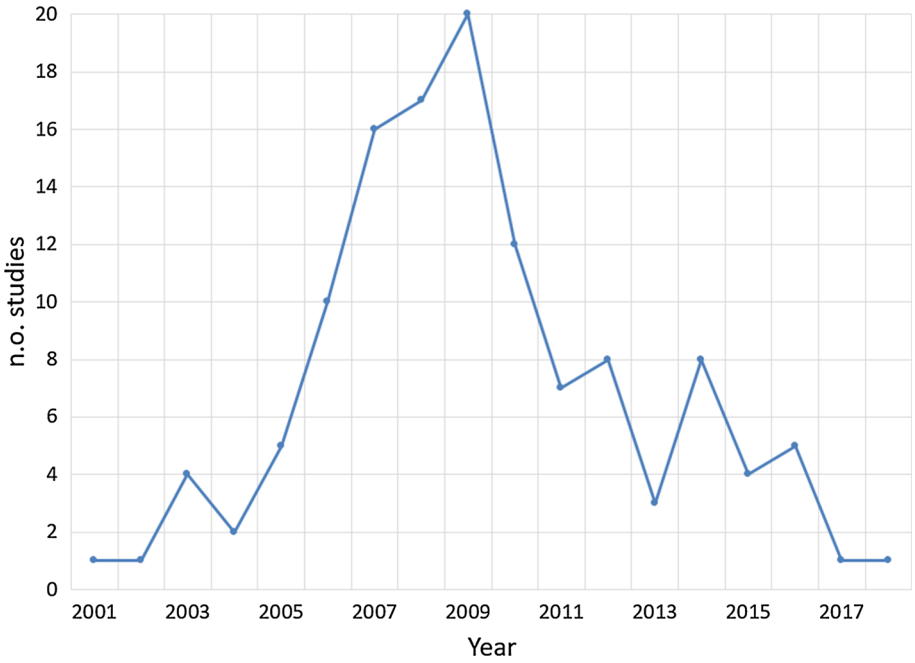

Fig. 9 Number of included primary studies per year

later as well, from 2008. Moreover, the steady decline that is present in the case of graph-based algorithms is not present here. This can mean that although interest in text-based model matching algorithms is low, it is not declining. However, due to the low sample size of text-based algorithms, we are not confident in this conclusion.

As a final note, in previous charts, multiple studies that discuss the same algorithm were handled separately, and are not merged into one study like we discussed in Sect. 4. We believe this represents interest in model matching more accurately, as iterations on existing algorithms are also important in this regard.

4.5 Threats to validity

Systematic literature reviews follow well-defined guidelines in order to be accurate and repeatable. However, even when closely following these guidelines, every survey is inherently subjective to some degree. In this subsection, we list the main threats to the validity of our survey, and what steps we took to mitigate them.

We consider the following to be the main threats to the objectivity and repeatability of our survey: (i) the study selection strategy we used and (ii) researcher bias. To mitigate the first one, we followed the guidelines for systematic literature reviews presented in the literature as closely as possible [78,79,163]. We aimed for the best coverage of included studies by using broader search terms during automatic search, and by manually searching venues that we knew had promising studies we can include. The goal of the snowballing process was to find related studies that we missed during the previous two searches. That said, we do acknowledge that there may be some missing studies that our search strategy did not manage to find. Furthermore, by also doing a manual search process, the selected venues might be overrepresented in the final result of the survey. We concluded that this was an acceptable compromise to increase the number of included primary studies.

Out of the two main threats mentioned, we consider researcher bias to be the primary threat to the validity of our survey, since it was conducted by two researchers. Different researchers might interpret a given study differently, especially regarding some aspects that had to be examined for our research questions. Although we aimed to give the exact definitions and instructions for extracting data from the included studies in Sect. 3.2.2, some values might be subjective to researcher bias. We acknowledge that there may be misinterpretations during the analysis of these approaches, especially due to the low number of researchers conducting the survey. For example, determining the focus of a model matching algorithm can be subjective when little information is given in the examined study. However, since we conducted the survey based on the guidelines and planning presented in Sect. 3.1, we believe that our survey is repeatable, and the repeated results would closely resemble the ones we got.

Moreover, research works that discuss the same algorithm can be split into multiple studies, for example, if the algorithm has improved over time. In order to handle cases like this as objectively as possible, we merged these studies into one study. Therefore, when answering relevant research ques-

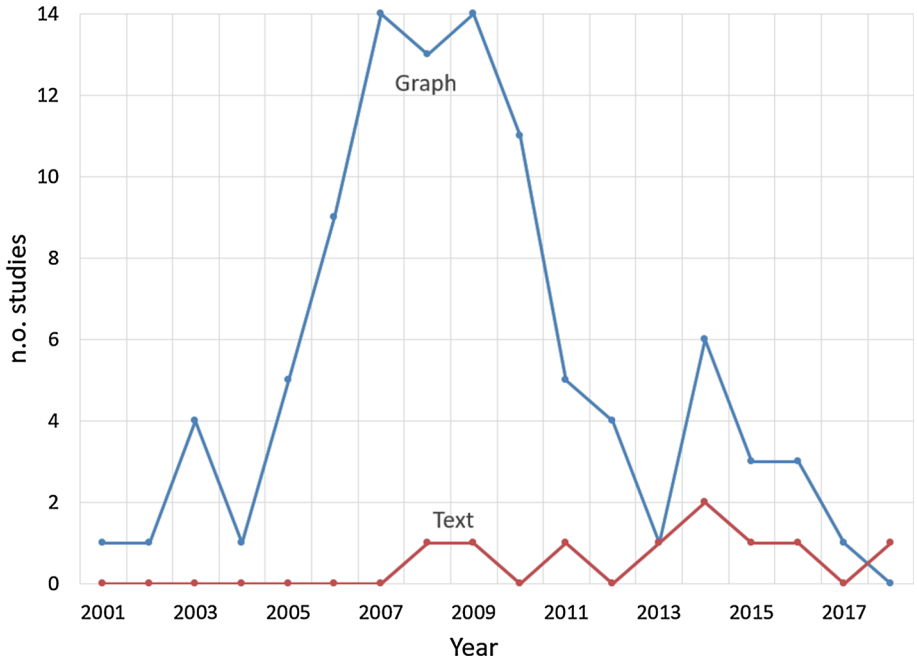

Fig. 10 Number of identified graph-based and text-based model matching algorithms per year

tions, statistics are less skewed as opposed to handling them as multiple studies. Section 4 and Table 5 discuss and summarize these merged studies. In the case of improving algorithms (for example, when an algorithm focused only on accuracy, but was improved to also be general in the next study), we used the improved properties of the later publications. Thus, we believe that we have managed to minimize the impact of this threat to our survey.

5 Conclusions

This survey consists of a systematic literature review (SLR) on matching techniques used in model-driven methodologies. We conducted the survey as a planned, well-defined process. It was carried out to acquire the state-of-the-art of model matching approaches in the existing literature. We identified the boundaries of the research field of model matching in model-driven methodologies. We focused on discovering differences between matching techniques, and on comparing text-based model matching approaches with graph-based ones. We have identified a total of 4693 studies, of which 1419 were duplicates, and 119 were included as primary studies, with an additional 6 papers categorized as relevant survey papers[4]. After thoroughly analyzing the data extracted from the selected primary studies, we identified research trends in model matching along with possibilities for future research. We also identified the main open questions related to the field of model matching, and summarized the state-of-the-art of evaluating model matching, differencing, and merging approaches. We compared text-based and graph-based algorithms and concluded that the former are much more rare, and usually require a higher configuration cost to use. The conclusions presented in the survey can help researchers and practitioners in the field of model matching in identifying gaps in existing research and in focusing their future work.

Acknowledgements Open access funding provided by Budapest University of Technology and Economics (BME). The authors would like to thank the anonymous reviewers. Their insightful comments led to a much improved paper.

Appendix

See Tables 6 and 7.

[4] The extracted data (including relevant venues of publications) can be found here: https://github.com/Fsomogyi/Systematic-Review-of-Matching-Techniques-Used-in-Model-Driven-Methodologies.

Table 6 Categorization of selected primary studies (pt. I)

ID	Type	Matching	Change	Focus	Eval.	Open Q.	VCS
S1 [129]	G	C	S	B	–	–	Y
S2 [55]	G	D+SI	S	B	–	Y	–
S3 [12]	G	C	S	O	–	–	–
S4 [51]	G	N	OP	B	–	–	Y
S5 [100]	G	N	OP	G	–	–	Y
S6 [148]	G	ST+SI	B	B	–	–	Y
S7 [56]	G	D	S	B	–	–	–
S8 [103]	G	ST	S	G	–	–	Y
S9 [76]	G	D+SI	S	G	–	–	Y
S10 [132]	G	O	S	A	–	–	Y
S11 [124]	G	N	OP	G	–	–	Y
S12 [118]	G	D	S	A	–	–	–
S13 [102]	–	–	–	–	–	Y	–
S14 [16]	T	O	S	O	–	Y	Y
S15 [110]	G	ST	S	G	–	–	Y
S16 [35]	G	C	S	A	–	–	–
S17 [72]	G	D+SI	S	G	–	–	–
S18 [31]	G	ST+SI	B	B	–	–	Y
S19 [99]	G	C	S	A	–	–	Y
S20 [44]	–	–	–	–	Y	–	–
S21 [123]	–	–	–	–	–	Y	–
S22 [158]	–	–	–	–	Y	–	–
S23 [11]	T	D	S	A	–	–	–
S24 [101]	T	C	S	A	–	–	Y
S25 [75]	–	–	–	–	–	Y	–
S26 [37]	G	N	OP	A	–	–	Y
S27 [169]	G	N	OP	A	Y	–	Y
S28 [27]	G	SI	S	A	Y	–	–
S29 [97]	G	ST	S	A	Y	–	Y
S30 [147]	G	C	B	B	–	–	Y
S31 [20]	G	N	OP	A	–	–	–
S32 [92]	G	N	OP	O	–	–	Y
S33 [91]	G	C	S	B	–	–	Y
S34 [65]	G	C	S	B	–	–	Y
S35 [63]	G	C	S	B	–	–	Y
S36 [46]	G	SI	S	A	–	–	–
S37 [77]	G	ST	S	A	–	–	Y
S38 [3]	G	ST	S	A	–	Y	Y
S39 [151]	G	D+SI	S	G	Y	–	Y
S40 [66]	G	D+SI	S	A	–	–	Y
S41 [114]	G	ST+D	S	G	–	–	Y
S42 [115]	G	ST+D	S	A	–	–	Y
S43 [24]	G	D+SI	S	G	–	–	Y
S44 [166]	G	SI	S	A	–	–	Y
S45 [86]	–	–	–	–	Y	–	Y
S46 [96]	G	ST+SI	S	G	–	–	Y

Table 6 continued

ID	Type	Matching	Change	Focus	Eval.	Open Q.	VCS
S47 [59]	T	ST	S	B	–	–	Y
S48 [45]	G	SI	S	A	–	–	–
S49 [159]	G	O	S	G	–	–	–
S50 [136]	T	D	S	B	–	–	Y
S51 [28]	G	D	S	B	–	–	–
S52 [112]	G	N	OP	B	–	–	Y
S53 [47]	–	–	–	–	Y	–	–
S54 [130]	G	N	OP	O	–	–	Y
S55 [33]	–	–	–	–	–	Y	Y
S56 [64]	G	C	S	B	–	–	Y
S57 [41]	G	unknown	S	G	–	–	Y
S58 [108]	G	SI	S	A	Y	–	–
S59 [109]	G	SI	S	A	Y	–	–
S60 [153]	G	N	OP	O	–	–	–

G graph; *T* text; *B* both; *O* other; *N* none; *ST* static; *SI* similarity; *D* dynamic; *C* custom; *S* state-based; *OP* operation-based; *A* accuracy; *G* generality; *Y* yes

Table 7 Categorization of selected primary studies (pt. II)

ID	Type	Matching	Change	Focus	Eval.	Open Q.	VCS
S61 [4]	G	SI	S	A	Y	–	–
S62 [62]	–	–	–	–	Y	–	Y
S63 [36]	G	C	S	A	–	–	–
S64 [49]	G	SI	S	A	–	–	–
S65 [50]	G	SI	S	A	Y	–	–
S66 [93]	G	SI	S	A	Y	–	–
S67 [170]	G	ST	S	A	–	–	Y
S68 [88]	G	D	S	B	–	–	–
S69 [19]	T	N	OP	A	–	–	Y
S70 [137]	T	D	S	B	–	–	Y
S71 [116]	G	ST+D	S	A	–	–	Y
S72 [126]	G	D	S	A	–	–	–
S73 [133]	G	C	S	G	–	–	Y
S74 [113]	G	N	OP	B	–	–	Y
S75 [17]	G	O	S	A	–	–	Y
S76 [119]	G	D+SI	S	B	–	Y	–
S77 [165]	G	SI	S	B	–	–	–
S78 [87]	G	D	S	B	–	–	–
S79 [95]	–	–	–	–	–	Y	Y
S80 [125]	G	D	S	A	–	–	–
S81 [111]	G	SI	S	G	–	–	Y
S82 [18]	–	–	–	–	–	Y	Y
S83 [38]	G	N	OP	O	–	–	–
S84 [89]	G	ST+D	S	B	–	–	–
S85 [94]	G	SI	B	O	–	–	–
S86 [117]	G	D+SI	S	B	–	–	–
S87 [84]	G	N	OP	B	Y	–	Y

Table 7 continued

ID	Type	Matching	Change	Focus	Eval.	Open Q.	VCS
S88 [82]	G	N	OP	B	–	–	Y
S89 [157]	T	O	OP	A	–	–	Y
S90 [150]	G	SI	S	G	–	–	Y
S91 [140]	–	–	–	–	–	Y	Y
S92 [60]	G	D	S	A	–	–	–
S93 [127]	T	ST + SI	S	B	–	–	Y
S94 [90]	G	N	OP	G	–	–	Y
S95 [105]	G	SI	S	G	–	–	Y
S96 [34]	G	SI	S	O	–	–	–
S97 [29]	G	SI	S	A	–	–	–
S98 [1]	G	SI	S	G	Y	–	Y
S99 [10]	G	D	S	B	–	–	Y
S100 [5]	G	SI	S	A	Y	–	–
S101 [6]	G	SI	S	A	Y	–	–
S102 [104]	G	SI	S	G	Y	–	–
S103 [48]	G	SI	S	A	Y	–	–
S104 [53]	G	C	S	A	–	–	–
S105 [142]	G	ST	S	A	–	–	Y
S106 [161]	G	SI	S	A	Y	–	–
S107 [152]	–	–	–	–	Y	–	–
S108 [39]	G	N	OP	G	–	–	Y
S109 [13]	G	D	S	G	–	–	–
S110 [69]	G	N	OP	B	–	–	Y
S111 [32]	G	ST + SI	B	B	–	–	Y
S112 [85]	G	N	OP	B	–	–	Y
S113 [162]	G	D + SI	S	G	–	–	Y
S114 [7]	G	SI	S	A	Y	–	–
S115 [164]	G	SI	S	A	–	–	Y
S116 [52]	G	D	S	B	–	–	Y
S117 [139]	G	ST	S	B	–	–	Y
S118 [73]	G	O	S	A	–	–	Y
S119 [8]	–	–	–	–	–	Y	Y

G graph; *T* text; *B* both; *O* other; *N* none; *ST* static; *SI* similarity; *D* dynamic; *C* custom; *S* state-based; *OP* operation-based; *A* accuracy; *G* generality; *Y* yes

References

1. Addazi, L., Cicchetti, A., Rocco, J.D., Ruscio, D.D., Iovino, L., Pierantonio, A.: Semantic-based model matching with emfcompare. In: Mayerhofer, T., Pierantonio, A., Schätz, B., Tamzalit, D. (eds.) 10th Workshop on Models and Evolution, pp. 40–49. CEUR-WS (2016). http://www.es.mdh.se/publications/4468-
2. Aho, A.V., Sethi, R., Ullman, J.D.: Compilers: Principles, Techniques, and Tools. Addison-Wesley Longman Publishing Co., Inc., Boston (1986)
3. Alanen, M., Porres, I.: Difference and union of models. In: Stevens, P., Whittle, J., Booch, G. (eds.) UML 2003—The Unified Modeling Language. Modeling Languages and Applications, pp. 2–17. Springer, Berlin (2003). https://doi.org/10.1007/978-3-540-45221-8_2
4. Al-Khiaty, M.A.R., Ahmed, M.: Matching uml class diagrams using a hybridized greedy-genetic algorithm. In: 2017 12th International Scientific and Technical Conference on Computer Sciences and Information Technologies (CSIT), vol. 1, pp. 161–166 (2017). https://doi.org/10.1109/STC-CSIT.2017.8098759
5. Al-Khiaty, M.A.R., Ahmed, M.: Similarity assessment of uml class diagrams using a greedy algorithm. In: 2014 International Computer Science and Engineering Conference (ICSEC), pp. 228–233 (2014). https://doi.org/10.1109/ICSEC.2014.6978199
6. Al-Khiaty, M.A.R., Ahmed, M.: Similarity assessment of uml class diagrams using simulated annealing. In: 2014 IEEE 5th International Conference on Software Engineering and Service Science, pp. 19–23 (2014). https://doi.org/10.1109/ICSESS.2014.6933505
7. Al-Rhman Al-Khiaty, M., Ahmed, M.: Uml class diagrams: similarity aspects and matching. Lect. Notes Softw. Eng. **4**, 41–47 (2016). https://doi.org/10.7763/lnse.2016.v4.221
8. Altmanninger, K., Brosch, P., Kappel, G., Langer, P., Seidl, M., Wieland, K., Wimmer, M.: Why model versioning research is

needed!? an experience report. In: Proceedings of the MoDSE-MCCM 2009 Workshop @ MoDELS 2009 (2009)
9. Altmanninger, K., Seidl, M., Wimmer, M.: A survey on model versioning approaches. Int. J. Web Inf. Syst. **5**(3), 271–304 (2009). https://doi.org/10.1108/17440080910983556
10. Altmanninger, K., Schwinger, W., Kotsis, G.: Semantics for accurate conflict detection in smover: specification, detection and presentation by example. Int. J. Enterp. Inf. Syst. **6**(1), 68–84 (2010). https://doi.org/10.4018/jeis.2010120206
11. Alwanain, M., Bordbar, B., Bowles, J.K.F.: Automated composition of sequence diagrams via alloy. In: 2014 2nd International Conference on Model-Driven Engineering and Software Development (MODELSWARD), pp. 384–391 (2014). https://doi.org/10.5220/0004715003840391
12. Anwar, A., Dkaki, T., Ebersold, S., Coulette, B., Nassar, M.: A formal approach to model composition applied to vuml. In: 2011 16th IEEE International Conference on Engineering of Complex Computer Systems, pp. 188–197 (2011). https://doi.org/10.1109/ICECCS.2011.26
13. Anwar, A., Ebersold, S., Nassar, M., Coulette, B., Kriouile, A.: Towards a generic approach for model composition. In: 2008 The Third International Conference on Software Engineering Advances, pp. 83–90 (2008). https://doi.org/10.1109/ICSEA.2008.38
14. Arora, S., Barak, B.: Computational Complexity: A Modern Approach, 1st edn. Cambridge University Press, New York (2009)
15. Babich, W.A.: Software Configuration Management: Coordination for Team Productivity. Addison-Wesley Longman Publishing Co., Inc., Boston (1986)
16. Badreddin, O., Lethbridge, T.C., Forward, A.: A novel approach to versioning and merging model and code uniformly. In: 2014 2nd International Conference on Model-Driven Engineering and Software Development (MODELSWARD), pp. 254–263 (2014). https://doi.org/10.5220/0004699802540263
17. Barrett, S.C., Butler, G., Chalin, P.: Mirador: a synthesis of model matching strategies. In: Proceedings of the 1st International Workshop on Model Comparison in Practice, IWMCP '10, pp. 2–10. ACM, New York (2010). https://doi.org/10.1145/1826147.1826151
18. Barrett, S.C., Chalin, P., Butler, G.: Model merging falls short of software engineering needs. In: MoDSE '08: International Workshop on Model-Driven Software Evolution (2008)
19. Barrett, S., Sinnig, D., Chalin, P., Butler, G.: Merging of use case models: semantic foundations. In: 2009 Third IEEE International Symposium on Theoretical Aspects of Software Engineering, pp. 182–189 (2009). https://doi.org/10.1109/TASE.2009.34
20. Bartelt, C.: Consistence preserving model merge in collaborative development processes. In: Proceedings of the 2008 International Workshop on Comparison and Versioning of Software Models, CVSM '08, pp. 13–18. ACM, New York (2008). https://doi.org/10.1145/1370152.1370157
21. Bendix, L., Emanuelsson, P.: Diff and merge support for model based development. In: Proceedings of the 2008 International Workshop on Comparison and Versioning of Software Models, CVSM '08, pp. 31–34. ACM, New York (2008). https://doi.org/10.1145/1370152.1370161
22. Berczuk, S.P., Appleton, B.: Software Configuration Management Patterns: Effective Teamwork, Practical Integration. Addison-Wesley Longman Publishing Co., Inc., Boston (2002)
23. Bergmann, G., Dávid, I., Hegedüs, Á., Horváth, Á., Ráth, I., Ujhelyi, Z., Varró, D.: Viatra 3 : a reactive model transformation platform. In: 8th International Conference on Model Transformations. Springer, L'Aquila(2015). https://doi.org/10.1007/978-3-319-21155-8_8
24. Berlik, S., Fathi, M.: Differences of structured documents—improving their quality. In: 2007 IEEE International Conference on Information Reuse and Integration, pp. 486–491 (2007). https://doi.org/10.1109/IRI.2007.4296667
25. Biolchini, J., Mian, P.G., Natali, A.C.C., Travassos, G.H.: Systematic review in software engineering. Technical report. Systems Engineering and Computer Science Department, COPPE UFRJ, Rio de Janeiro (2005)
26. Bishop, C.M.: Pattern Recognition and Machine Learning (Information Science and Statistics). Springer, Berlin (2006)
27. Bogdanov, K., Walkinshaw, N.: Computing the structural difference between state-based models. In: 2009 16th Working Conference on Reverse Engineering, pp. 177–186 (2009). https://doi.org/10.1109/WCRE.2009.17
28. Boronat, A., Carsí, J.A., Ramos, I., Letelier, P.: Formal model merging applied to class diagram integration. Electron. Notes Theor. Comput. Sci. **166**, 5–26 (2007). https://doi.org/10.1016/j.entcs.2006.06.013
29. Brockmans, S., Ehrig, M., Koschmider, A., Oberweis, A., Studer, R.: Semantic alignment of business processes. In: Proceedings of the Eighth International Conference on Enterprise Information Systems—Volume 3: ICEIS, pp. 191–196. INSTICC, SciTePress (2006). https://doi.org/10.5220/0002495601910196
30. Brosch, P., Kappel, G., Langer, P., Seidl, M., Wieland, K., Wimmer, M.: An introduction to model versioning. In: Proceedings of the 12th International Conference on Formal Methods for the Design of Computer, Communication, and Software Systems: Formal Methods for Model-driven Engineering, SFM'12, pp. 336–398. Springer, Berlin (2012). https://doi.org/10.1007/978-3-642-30982-3_10
31. Brosch, P., Kappel, G., Seidl, M., Wieland, K., Wimmer, M., Kargl, H., Langer, P.: Adaptable model versioning in action. In: Modellierung 2010, 24.-26. März 2010, Klagenfurt, Österreich, pp. 221–236 (2010). http://subs.emis.de/LNI/Proceedings/Proceedings161/article5545.html
32. Brosch, P., Langer, P., Seidl, M., Wimmer, M.: Towards end-user adaptable model versioning: the by-example operation recorder. In: Proceedings of the 2009 ICSE Workshop on Comparison and Versioning of Software Models, CVSM '09, pp. 55–60. IEEE Computer Society, Washington (2009). https://doi.org/10.1109/CVSM.2009.5071723
33. Brosch, P.: Improving conflict resolution in model versioning systems. In: 2009 31st International Conference on Software Engineering—Companion Volume, pp. 355–358 (2009). https://doi.org/10.1109/ICSE-COMPANION.2009.5071020
34. Byrne, B., Fokoue, A., Kalyanpur, A., Srinivas, K., Wang, M.: Scalable matching of industry models: a case study. In: Proceedings of the 4th International Conference on Ontology Matching—Volume 551, OM'09, pp. 1–12. CEUR-WS.org, Aachen (2009)
35. Chaouni, S.B., Fredj, M., Mouline, S.: A rules-based system for model composition. In: 2015 IEEE/ACS 12th International Conference of Computer Systems and Applications (AICCSA), pp. 1–8 (2015). https://doi.org/10.1109/AICCSA.2015.7507217
36. Chaouni, S.B., Fredj, M., Mouline, S.: MDA based-approach for UML models complete comparison. Int. J. Comput. Sci. Issues (IJCSI) (2011). arXiv:1105.6128
37. Chong, H., Zhang, R., Qin, Z.: Composite-based conflict resolution in merging versions of uml models. In: 2016 17th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), pp. 127–132 (2016). https://doi.org/10.1109/SNPD.2016.7515890
38. Cicchetti, A., Di Ruscio, D., Pierantonio, A.: Model patches in model-driven engineering. In: Proceedings of the 2009 International Conference on Models in Software Engineering, MODELS'09, pp. 190–204. Springer, Berlin 2010). https://doi.org/10.1007/978-3-642-12261-3_19

39. Cicchetti, A., Muccini, H., Pelliccione, P., Pierantonio, A.: Towards a framework for distributed and collaborative modeling. In: 18th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises, WETICE 2009, Groningen, Proceedings. The Netherlands, 29 June–1 July 2009, pp. 149–154 (2009). https://doi.org/10.1109/WETICE.2009.48
40. Cicchetti, A., Ruscio, D.D., Eramo, R., Pierantonio, A.: Automating co-evolution in model-driven engineering. In: 2008 12th International IEEE Enterprise Distributed Object Computing Conference, pp. 222–231 (2008). https://doi.org/10.1109/EDOC.2008.44
41. Cicchetti, A., Di Ruscio, D., Pierantonio, A.: Managing model conflicts in distributed development. In: Czarnecki, K., Ober, I., Bruel, J.M., Uhl, A., Völter, M. (eds.) Model Driven Engineering Languages and Systems, pp. 311–325. Springer, Berlin (2008). https://doi.org/10.1007/978-3-540-87875-9_23
42. Conradi, R., Westfechtel, B.: Version models for software configuration management. ACM Comput. Surv. **30**(2), 232–282 (1998). https://doi.org/10.1145/280277.280280
43. Czarnecki, K., Helsen, S.: Feature-based survey of model transformation approaches. IBM Syst. J. **45**(3), 621–645 (2006). https://doi.org/10.1147/sj.453.0621
44. Dijkman, R., Dumas, M., Garcia-Banuelos, L., Kaarik, R.: Aligning business process models. In: Proceedings of the 2009 IEEE International Enterprise Distributed Object Computing Conference, EDOC '09, pp. 45–53. IEEE Computer Society, Washington (2009). https://doi.org/10.1109/EDOC.2009.11
45. Dijkman, R.M.: Feedback on differences between business processes. Technical report. Eindhoven University of Technology, The Netherlands (2007)
46. Dijkman, R.: Diagnosing differences between business process models. In: Dumas, M., Reichert, M., Shan, M.C. (eds.) Business Process Management, pp. 261–277. Springer, Berlin (2008). https://doi.org/10.1007/978-3-540-85758-7_20
47. Dijkman, R., Dumas, M., García-Bañuelos, L.: Graph matching algorithms for business process model similarity search. In: Dayal, U., Eder, J., Koehler, J., Reijers, H.A. (eds.) Business Process Management, pp. 48–63. Springer, Berlin (2009). https://doi.org/10.1007/978-3-642-03848-8_5
48. Dijkman, R., Dumas, M., van Dongen, B., Käärik, R., Mendling, J.: Similarity of business process models: metrics and evaluation. Inf. Syst. **36**(2), 498–516 (2011). https://doi.org/10.1016/j.is.2010.09.006
49. Dongen, B., Dijkman, R., Mendling, J.: Measuring similarity between business process models. In: Proceedings of the 20th International Conference on Advanced Information Systems Engineering, CAiSE '08, pp. 450–464. Springer, Berlin (2008). https://doi.org/10.1007/978-3-540-69534-9_34
50. Ehrig, M., Koschmider, A., Oberweis, A.: Measuring similarity between semantic business process models. In: Proceedings of the Fourth Asia-Pacific Conference on Comceptual Modelling—Volume 67, APCCM '07, pp. 71–80. Australian Computer Society, Inc., Darlinghurst (2007). http://dl.acm.org/citation.cfm?id=1274453.1274465
51. Ehrig, H., Ermel, C., Taentzer, G.: A formal resolution strategy for operation-based conflicts in model versioning using graph modifications. In: Giannakopoulou, D., Orejas, F. (eds.) Fundamental Approaches to Software Engineering, pp. 202–216. Springer, Berlin (2011). https://doi.org/10.1007/978-3-642-19811-3_15
52. Engel, K.D., Paige, R.F., Kolovos, D.S.: Using a model merging language for reconciling model versions. In: Proceedings of the Second European Conference on Model Driven Architecture: Foundations and Applications, ECMDA-FA'06, pp. 143–157. Springer, Berlin (2006). https://doi.org/10.1007/11787044_12
53. Eshuis, R., Grefen, P.: Structural matching of bpel processes. In: Fifth European Conference on Web Services (ECOWS'07), pp. 171–180 (2007). https://doi.org/10.1109/ECOWS.2007.22
54. Evans, S.: GIT: For Starters. CreateSpace Independent Publishing Platform, Scotts Valley (2016)
55. Farias, K., Breitman, K.K., de Oliveira, T.C.: A flexible strategy-based model comparison approach: bridging the syntactic and semantic gap. J. Univers. Comput. Sci. **15**, 2225–2253 (2009). https://doi.org/10.3217/jucs-015-11-2225
56. Fleurey, F., Baudry, B., France, R., Ghosh, S.: A generic approach for automatic model composition. In: Giese, H. (ed.) Models in Software Engineering, pp. 7–15. Springer, Berlin (2008). https://doi.org/10.1007/978-3-540-69073-3_2
57. Förtsch, S., Westfechtel, B.: Differencing and merging of software diagrams—state of the art and challenges. In: Proceedings of the Second International Conference on Software and Data Technologies—Volume 2: ICSOFT, pp. 90–99. INSTICC, SciTePress (2007). https://doi.org/10.5220/0001342900900099
58. Forward, A., Badreddin, O., Lethbridge, T.C., Solano, J.: Model-driven rapid prototyping with umple. Softw. Pract. Exp. **42**(7), 781–797 (2012). https://doi.org/10.1002/spe.1155
59. Foucault, M., Barbier, S., Lugato, D.: Enhancing version control with domain-specific semantics. In: Proceedings of the 5th International Workshop on Modeling in Software Engineering, MiSE '13, pp. 31–36. IEEE Press, Piscataway (2013). https://doi.org/10.1109/mise.2013.6595293. http://dl.acm.org/citation.cfm?id=2662737.2662745
60. France, R., Fleurey, F., Reddy, R., Baudry, B., Ghosh, S.: Providing support for model composition in metamodels. In: 11th IEEE International Enterprise Distributed Object Computing Conference (EDOC 2007), pp. 253–253 (2007). https://doi.org/10.1109/EDOC.2007.55
61. Frankel, D.: Model Driven Architecture: Applying MDA to Enterprise Computing. Wiley, New York (2002)
62. Garcia-Diaz, V., G-Bustelo, B.C.P., Sanjuan-Martinez, O., Valdez, E.R.N., Lovelle, J.M.C.: Mctest: towards an improvement of match algorithms for models. IET Softw. **6**(2), 127–139 (2012). https://doi.org/10.1049/iet-sen.2011.0040
63. Gerth, C., Luckey, M., Küster, J.M., Engels, G.: Detection of semantically equivalent fragments for business process model change management. In: 2010 IEEE International Conference on Services Computing, pp. 57–64 (2010). https://doi.org/10.1109/SCC.2010.38
64. Gerth, C., Küster, J.M., Engels, G.: Language-independent change management of process models. In: Schürr, A., Selic, B. (eds.) Model Driven Engineering Languages and Systems, pp. 152–166. Springer, Berlin (2009). https://doi.org/10.1007/978-3-642-04425-0_12
65. Gerth, C., Küster, J.M., Luckey, M., Engels, G.: Detection and resolution of conflicting change operations in version management of process models. Softw. Syst. Model. **12**(3), 517–535 (2013). https://doi.org/10.1007/s10270-011-0226-8
66. Girschick, M.: Difference detection and visualization in uml class diagrams. Technical report. Department of Computer Science, TU Darmstadt (2006)
67. Grönniger, H., Krahn, H., Rumpe, B., Schindler, M., Völkel, S.: Text-based modeling (2014). arXiv:1409.6623
68. Heijstek, W., Kühne, T., Chaudron, M.R.V.: Experimental analysis of textual and graphical representations for software architecture design. In: 2011 International Symposium on Empirical Software Engineering and Measurement, pp. 167–176 (2011). https://doi.org/10.1109/ESEM.2011.25
69. Herrmannsdoerfer, M., Koegel, M.: Towards a generic operation recorder for model evolution. In: Proceedings of the 1st International Workshop on Model Comparison in Practice, IWMCP

'10, pp. 76–81. ACM, New York (2010). https://doi.org/10.1145/1826147.1826161
70. Jackson, D.: Alloy: a lightweight object modelling notation. ACM Trans. Softw. Eng. Methodol. **11**(2), 256–290 (2002). https://doi.org/10.1145/505145.505149
71. KDiff3 (2003). http://kdiff3.sourceforge.net/
72. Kehrer, T., Kelter, U., Pietsch, P., Schmidt, M.: Adaptability of model comparison tools. In: Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering, ASE 2012, pp. 306–309. ACM, New York (2012). https://doi.org/10.1145/2351676.2351731
73. Kehrer, T., Kelter, U.: Versioning of ordered model element sets. Softwaretechnik-Trends **34**(2) (2014)
74. Kelly, S., Tolvanen, J.P.: Domain-Specific Modeling. Wiley, New York (2007)
75. Kelter, U., Schmidt, M.: Comparing state machines. In: Proceedings of the 2008 International Workshop on Comparison and Versioning of Software Models, CVSM '08, pp. 1–6. ACM, New York (2008). https://doi.org/10.1145/1370152.1370154
76. Kelter, U., Wehren, J., Niere, J.: A generic difference algorithm for uml models. Softw. Eng. **64**(105–116), 4–9 (2005)
77. Kindler, E., Könemann, P., Unland, L.: Diff-based model synchronization in an industrial mdd process. Technical report. Technical University of Denmark, DTU Informatics (2008)
78. Kitchenham, B.A., Charters, S.M.: Guidelines for performing systematic literature reviews in software engineering. Technical report. School of Computer Science and Mathematics, Keele University, Keele, United Kingdom and Department of Computer Science, University of Durham, Durham (2007)
79. Kitchenham, B.A., Dyba, T., Jorgensen, M.: Evidence-based software engineering. In: Proceedings of the 26th International Conference on Software Engineering, ICSE '04, pp. 273–281. IEEE Computer Society, Washington (2004). https://doi.org/10.1109/ICSE.2004.1317449. http://dl.acm.org/citation.cfm?id=998675.999432
80. Klint, P., van der Storm, T., Vinju, J.J.: Rascal: a domain specific language for source code analysis and manipulation. In: Ninth IEEE International Working Conference on Source Code Analysis and Manipulation (SCAM), pp. 168–177. IEEE Computer Society (2009). https://doi.org/10.1109/SCAM.2009.28
81. Köbler, J., Schöning, U., Torán, J.: The Graph Isomorphism Problem: Its Structural Complexity. Birkhauser Verlag, Basel (1993)
82. Koegel, M., Helming, J., Seyboth, S.: Operation-based conflict detection and resolution. In: Proceedings of the 2009 ICSE Workshop on Comparison and Versioning of Software Models, CVSM '09, pp. 43–48. IEEE Computer Society, Washington (2009). https://doi.org/10.1109/CVSM.2009.5071721
83. Koegel, M., Herrmannsdoerfer, M., Li, Y., Helming, J., David, J.: Comparing state- and operation-based change tracking on models. In: 2010 14th IEEE International Enterprise Distributed Object Computing Conference, pp. 163–172 (2010). https://doi.org/10.1109/EDOC.2010.15
84. Koegel, M., Herrmannsdoerfer, M., von Wesendonk, O., Helming, J.: Operation-based conflict detection. In: Proceedings of the 1st International Workshop on Model Comparison in Practice, IWMCP '10, pp. 21–30. ACM, New York (2010). https://doi.org/10.1145/1826147.1826154
85. Kögel, M.: Towards software configuration management for unified models. In: Proceedings of the 2008 International Workshop on Comparison and Versioning of Software Models, CVSM '08, pp. 19–24. ACM, New York (2008). https://doi.org/10.1145/1370152.1370158
86. Kolovos, D.S., Di Ruscio, D., Pierantonio, A., Paige, R.F.: Different models for model matching: An analysis of approaches to support model differencing. In: Proceedings of the 2009 ICSE Workshop on Comparison and Versioning of Software Models, CVSM '09, pp. 1–6. IEEE Computer Society, Washington (2009). https://doi.org/10.1109/CVSM.2009.5071714
87. Kolovos, D.S., Paige, R.F., Polack, F.A.: Model comparison: a foundation for model composition and model transformation testing. In: Proceedings of the 2006 International Workshop on Global Integrated Model Management, GaMMa '06, pp. 13–20. ACM, New York (2006). https://doi.org/10.1145/1138304.1138308
88. Kolovos, D.S., Paige, R.F., Polack, F.A.C.: Merging models with the epsilon merging language (eml). In: Nierstrasz, O., Whittle, J., Harel, D., Reggio, G. (eds.) Model Driven Engineering Languages and Systems, pp. 215–229. Springer, Berlin (2006). https://doi.org/10.1007/11880240_16s
89. Könemann, P.: Model-independent differences. In: Proceedings of the 2009 ICSE Workshop on Comparison and Versioning of Software Models, CVSM '09, pp. 37–42. IEEE Computer Society, Washington (2009). https://doi.org/10.1109/CVSM.2009.5071720
90. Kuryazov, D., Winter, A.: Representing model differences by delta operations. In: 2014 IEEE 18th International Enterprise Distributed Object Computing Conference Workshops and Demonstrations, pp. 211–220 (2014). https://doi.org/10.1109/EDOCW.2014.39
91. Küster, J.M., Gerth, C., Förster, A., Engels, G.: Detecting and resolving process model differences in the absence of a change log. In: Dumas, M., Reichert, M., Shan, M.C. (eds.) Business Process Management, pp. 244–260. Springer, Berlin (2008). 10.1007/978-3-540-85758-7_19
92. Küster, J.M., Gerth, C., Engels, G.: Dependent and conflicting change operations of process models. In: Paige, R.F., Hartman, A., Rensink, A. (eds.) Model Driven Architecture—Foundations and Applications, pp. 158–173. Springer, Berlin (2009). https://doi.org/10.1007/978-3-642-02674-4_12
93. La Rosa, M., Dumas, M., Uba, R., Dijkman, R.: Merging business process models. In: Meersman, R., Dillon, T., Herrero, P. (eds.) On the Move to Meaningful Internet Systems: OTM 2010, pp. 96–113. Springer, Berlin (2010). https://doi.org/10.1007/978-3-642-16934-2_10
94. Li, C., Reichert, M., Wombacher, A.: On measuring process model similarity based on high-level change operations. In: Li, Q., Spaccapietra, S., Yu, E., Olivé, A. (eds.) Conceptual Modeling—ER 2008, pp. 248–264. Springer, Berlin (2008). https://doi.org/10.1007/978-3-540-87877-3_19
95. Lin, Y., Zhang, J., Gray, J.: Model comparison: a key challenge for transformation testing and version control in model driven software development. In: Control in Model Driven Software Development. OOPSLA/GPCE: Best Practices for Model-Driven Software Development, pp. 219–236. Springer (2004)
96. Lin, Y., Gray, J., Jouault, F.: Dsmdiff: a differentiation tool for domain-specific models. Eur. J. Inf. Syst. **16**(4), 349–361 (2007). https://doi.org/10.1057/palgrave.ejis.3000685
97. Lucia, A.D., Fasano, F., Scanniello, G., Tortora, G.: Concurrent fine-grained versioning of uml models. In: 2009 13th European Conference on Software Maintenance and Reengineering, pp. 89–98 (2009). https://doi.org/10.1109/CSMR.2009.35
98. MacKenzie, D., Eggert, P., Stallman, R.: GNU Diffutils Reference Manual. Samurai Media Limited, London (2015)
99. Maoz, S., Ringert, J.O., Rumpe, B.: Addiff: semantic differencing for activity diagrams. In: Proceedings of the 19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering, ESEC/FSE '11, pp. 179–189. ACM, New York (2011). https://doi.org/10.1145/2025113.2025140
100. Maoz, S., Ringert, J.O.: A framework for relating syntactic and semantic model differences. Softw. Syst. Model. **17**(3), 753–777 (2018). https://doi.org/10.1007/s10270-016-0552-y
101. Maoz, S., Ringert, J.O., Rumpe, B.: Cddiff: semantic differencing for class diagrams. In: Mezini, M. (ed.) ECOOP 2011—Object-

Oriented Programming, pp. 230–254. Springer, Berlin (2011). https://doi.org/10.1007/978-3-642-22655-7_12
102. Maoz, S., Ringert, J.O., Rumpe, B.: A manifesto for semantic model differencing. In: Dingel, J., Solberg, A. (eds.) Models in Software Engineering, pp. 194–203. Springer, Berlin (2011). https://doi.org/10.1007/978-3-642-21210-9_19
103. Mehra, A., Grundy, J., Hosking, J.: A generic approach to supporting diagram differencing and merging for collaborative design. In: Proceedings of the 20th IEEE/ACM International Conference on Automated Software Engineering, ASE '05, pp. 204–213. ACM, New York (2005). https://doi.org/10.1145/1101908.1101940
104. Melnik, S., Garcia-Molina, H., Rahm, E.: Similarity flooding: a versatile graph matching algorithm and its application to schema matching. In: Proceedings of the 18th International Conference on Data Engineering, ICDE '02, p. 117. IEEE Computer Society, Washington (2002). https://doi.org/10.1109/ICDE.2002.994702. http://dl.acm.org/citation.cfm?id=876875.879024
105. Melnik, S., Rahm, E., Bernstein, P.A.: Rondo: a programming platform for generic model management. In: Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data, SIGMOD '03, pp. 193–204. ACM, New York (2003). https://doi.org/10.1145/872757.872782
106. Mens, T.: A state-of-the-art survey on software merging. IEEE Trans. Softw. Eng. **28**(5), 449–462 (2002). https://doi.org/10.1109/TSE.2002.1000449
107. Mohagheghi, P., Aagedal, J.: Evaluating quality in model-driven engineering. In: Proceedings of the International Workshop on Modeling in Software Engineering, MISE '07. IEEE Computer Society, Washington (2007). https://doi.org/10.1109/MISE.2007.6
108. Nejati, S., Sabetzadeh, M., Chechik, M., Easterbrook, S., Zave, P.: Matching and merging of statecharts specifications. In: Proceedings of the 29th International Conference on Software Engineering, ICSE '07, pp. 54–64. IEEE Computer Society, Washington (2007). https://doi.org/10.1109/ICSE.2007.50
109. Nejati, S., Sabetzadeh, M., Chechik, M., Easterbrook, S., Zave, P.: Matching and merging of variant feature specifications. IEEE Trans. Softw. Eng. **38**(6), 1355–1375 (2012). https://doi.org/10.1109/TSE.2011.112
110. Nguyen, T.N.: A novel structure-oriented difference approach for software artifacts. In: 30th Annual International Computer Software and Applications Conference (COMPSAC'06), vol. 1, pp. 197–204 (2006). https://doi.org/10.1109/COMPSAC.2006.13
111. Obeo, C.B., Pierantonio, A.: Model differences in the eclipse modeling framework. UPGRADE Eur. J. Inf. Prof. **9**(2), 29–34 (2008)
112. Oda, T., Saeki, M.: Generative technique of version control systems for software diagrams. In: 21st IEEE International Conference on Software Maintenance (ICSM'05), pp. 515–524 (2005). https://doi.org/10.1109/ICSM.2005.49
113. Oda, T., Saeki, M.: Meta-modeling based version control system for software diagrams. IEICE Trans. **89**(D), 1390–1402 (2006)
114. Ohst, D., Welle, M., Kelter, U.: Difference tools for analysis and design documents. In: Proceedings of the International Conference on Software Maintenance, ICSM '03, p. 13. IEEE Computer Society, Washington (2003). https://doi.org/10.1109/icsm.2003.1235402. http://dl.acm.org/citation.cfm?id=942800.943567
115. Ohst, D., Welle, M., Kelter, U.: Differences between versions of uml diagrams. In: Proceedings of the 9th European Software Engineering Conference Held Jointly with 11th ACM SIGSOFT International Symposium on Foundations of Software Engineering, ESEC/FSE-11, pp. 227–236. ACM, New York (2003). https://doi.org/10.1145/940071.940102
116. Ohst, D., Welle, M., Kelter, U.: Merging uml documents. Technical report, University of Siegen (2004)
117. Oliveira, K., Breitman, K., Oliveira, T.: Ontology aided model comparison. In: 2009 14th IEEE International Conference on Engineering of Complex Computer Systems, pp. 78–83 (2009). https://doi.org/10.1109/ICECCS.2009.55
118. Oliveira, K.S.F., de Oliveira, T.C.: A guidance for model composition. In: International Conference on Software Engineering Advances (ICSEA 2007), pp. 27–27 (2007). https://doi.org/10.1109/ICSEA.2007.5
119. Oliveira, K.S.F., de Oliveira, T.C.: Model comparison: a strategy-based approach. In: Proceedings of the Twentieth International Conference on Software Engineering & Knowledge Engineering (SEKE'2008), San Francisco, CA, USA, July 1–3, 2008, pp. 912–917 (2008)
120. Pai, M., Mcculloch, M., Gorman, J.D., Pai, N.P., Enanoria, W.T.A., Kennedy, G.C., Tharyan, P., Colford, J.M.: Systematic reviews and meta-analyses: an illustrated, step-by-step guide. Natl. Med. J. India **17**(2), 86–95 (2004)
121. Paige, R.F., Matragkas, N., Rose, L.M.: Evolving models in model-driven engineering. J. Syst. Softw. **111**(C), 272–280 (2016). https://doi.org/10.1016/j.jss.2015.08.047
122. Petticrew, M., Roberts, H.: Systematic Reviews in the Social Sciences: A Practical Guide, 1st edn. Blackwell Publishing, Hoboken (2006)
123. Pietsch, P., Shariat Yazdi, H., Kelter, U., Kehrer, T.: Assessing the quality of model differencing engines. Softwaretechnik-Trends **32**, 47–48 (2013)
124. Rajbhoj, A., Reddy, S.: A graph-pattern based approach for metamodel specific conflict detection in a general-purpose model versioning system. In: Moreira, A., Schätz, B., Gray, J., Vallecillo, A., Clarke, P. (eds.) Model-Driven Engineering Languages and Systems, pp. 422–435. Springer, Berlin (2013). https://doi.org/10.1007/978-3-642-41533-3_26
125. Reddy, R., France, R., Ghosh, S., Fleurey, F., Baudry, B.: Model composition—a signature-based approach. In: Proceedings of the AOM Workshop at MODELS'05, Montego Bay (2005)
126. Rhouma, T.B., Tessier, P., Terrier, F.: Merging uml2 composite structures of software product lines. In: 2012 IEEE 17th International Conference on Engineering of Complex Computer Systems, pp. 77–85 (2012)
127. Rivera, J.E., Vallecillo, A.: Representing and operating with model differences. In: Paige, R.F., Meyer, B. (eds.) Objects, Components, Models and Patterns, pp. 141–160. Springer, Berlin (2008). https://doi.org/10.1007/978-3-540-69824-1_9
128. Romero, J.R., Rivera, J.E., Durán, F., Vallecillo, A.: Formal and tool support for model driven engineering with maude. J. Object Technol. **6**(9), 187–207 (2007). https://doi.org/10.5381/jot.2007.6.9.a10
129. Rutle, A., Rossini, A., Lamo, Y., Wolter, U.: A category-theoretical approach to the formalisation of version control in mde. In: Chechik, M., Wirsing, M. (eds.) Fundamental Approaches to Software Engineering, pp. 64–78. Springer, Berlin (2009). https://doi.org/10.1007/978-3-642-00593-0_5
130. Schmidt, M., Wenzel, S., Kehrer, T., Kelter, U.: History-based merging of models. In: Proceedings of the 2009 ICSE Workshop on Comparison and Versioning of Software Models, CVSM '09, pp. 13–18. IEEE Computer Society, Washington (2009). https://doi.org/10.1109/CVSM.2009.5071716
131. Schmidt, D.C.: Guest editor's introduction: model-driven engineering. Computer **39**, 25–31 (2006). https://doi.org/10.1109/MC.2006.58
132. Schwägerl, F., Uhrig, S., Westfechtel, B.: A graph-based algorithm for three-way merging of ordered collections in emf models. Sci. Comput. Program. **113**(P1), 51–81 (2015). https://doi.org/10.1016/j.scico.2015.02.008
133. Selonen, P., Kettunen, M.: Metamodel-based inference of inter-model correspondence. In: Proceedings of the 11th European Conference on Software Maintenance and Reengineering, CSMR

'07, pp. 71–80. IEEE Computer Society, Washington (2007). https://doi.org/10.1109/CSMR.2007.31
134. Selonen, P.: A review of uml model comparison approaches. In: Staron, M. (ed.) Workshop Proceedings of the 5th Nordic Workshop on Model Driven Engineering. Research report, 27–29 August 2007, Ronneby, Sweden. Blekinge Institute of Technology, pp. 37–51 (2007)
135. Sendall, S., Kozaczynski, W.: Model transformation: the heart and soul of model-driven software development. IEEE Softw. **20**(5), 42–45 (2003). https://doi.org/10.1109/MS.2003.1231150
136. Somogyi, F.A., Asztalos, M.: Formal description and verification of a text-based model differencing and merging method. In: Proceedings of the 6th International Conference on Model-Driven Engineering and Software Development—Volume 1: AMARETTO, pp. 657–667. INSTICC, SciTePress (2018). https://doi.org/10.5220/0006728006570667
137. Somogyi, F.A.: Merging textual representations of software models. In: Kékesi, T. (ed.) The Publications of the MultiScience—XXX. microCAD International Multidisciplinary Scientific Conference. Miskolc (2016)
138. Song, G., Zhang, K., Kong, J.: Model management through graph transformation. In: 2004 IEEE Symposium on Visual Languages—Human Centric Computing, pp. 75–82 (2004). https://doi.org/10.1109/VLHCC.2004.37
139. Soto, M., Münch, J.: Using model comparison to maintain model-to-standard compliance. In: Proceedings of the 2008 International Workshop on Comparison and Versioning of Software Models, CVSM '08, pp. 35–40. ACM, New York (2008). https://doi.org/10.1145/1370152.1370162
140. Soto, M., Münch, J.: Process model difference analysis for supporting process evolution. In: Richardson, I., Runeson, P., Messnarz, R. (eds.) Software Process Improvement, pp. 123–134. Springer, Berlin (2006). https://doi.org/10.1007/11908562_12
141. Spinellis, D.: Version control systems. IEEE Softw. **22**(5), 108–109 (2005). https://doi.org/10.1109/MS.2005.140
142. Sriplakich, P., Blanc, X., pierre Gervais, M.: Supporting collaborative development in an open mda environment. In: 2006 22nd IEEE International Conference on Software Maintenance, pp. 244–253 (2006). https://doi.org/10.1109/ICSM.2006.64
143. Stahl, T., Voelter, M., Czarnecki, K.: Model-Driven Software Development: Technology, Engineering, Management. Wiley, New York (2006)
144. Steel, J., Raymond, K.: Generating human-usable textual notations for information models. In: Proceedings Fifth IEEE International Enterprise Distributed Object Computing Conference, pp. 250–261 (2001). https://doi.org/10.1109/EDOC.2001.950444
145. Steinberg, D., Budinsky, F., Paternostro, M., Merks, E.: EMF: Eclipse Modeling Framework 2.0, 2nd edn. Addison-Wesley Professional, Boston (2009)
146. Stephan, M., Cordy, J.R.: A survey of methods and applications of model comparison. Technical report, School of Computing, Queens University, Kingston, Ontario, Canada (2012)
147. Taentzer, G., Ermel, C., Langer, P., Wimmer, M.: Conflict detection for model versioning based on graph modifications. In: Ehrig, H., Rensink, A., Rozenberg, G., Schürr, A. (eds.) Graph Transformations, pp. 171–186. Springer, Berlin (2010). https://doi.org/10.1007/978-3-642-15928-2_12
148. Taentzer, G., Ermel, C., Langer, P., Wimmer, M.: A fundamental approach to model versioning based on graph modifications: from theory to implementation. Softw. Syst. Model. **13**(1), 239–272 (2014). https://doi.org/10.1007/s10270-012-0248-x
149. Tortoise SVN (2004). http://tortoisesvn.net/
150. Toulmé, A.: Presentation of emf compare utility. In: Eclipse Modeling Sympossium, pp. 1–8 (2006)
151. Treude, C., Berlik, S., Wenzel, S., Kelter, U.: Difference computation of large models. In: Proceedings of the the 6th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on The Foundations of Software Engineering, ESEC-FSE '07, pp. 295–304. ACM, New York (2007). https://doi.org/10.1145/1287624.1287665
152. Uhrig, S., Schwägerl, F.: Tool support for the evaluation of matching algorithms in the eclipse modeling framework. In: Proceedings of the 1st International Conference on Model-Driven Engineering and Software Development—Volume 1: MODELSWARD,, pp. 101–110. INSTICC, SciTePress (2013). https://doi.org/10.5220/0004310801010110
153. Uhrig, S.: Matching class diagrams: with estimated costs towards the exact solution? In: Proceedings of the 2008 International Workshop on Comparison and Versioning of Software Models, CVSM '08, pp. 7–12. ACM, New York (2008). https://doi.org/10.1145/1370152.1370155
154. Valiente, G., Martínez, C.: An algorithm for graphpattern-matching. In: Proceedings of the 4th South American Workshopon String Processing, Volume 8 of International Informatics Series, pp. 180–197. Carleton University Press (1997)
155. Vallecillo, A.: A journey through the secret life of models. In: Aßmann, U., Bézivin, J., Paige, R., Rumpe, B., Schmidt, D.C. (eds.) Perspectives Workshop: Model Engineering of Complex Systems (MECS), no. 08331 in Dagstuhl Seminar Proceedings. Schloss Dagstuhl—Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany (2008). http://drops.dagstuhl.de/opus/volltexte/2008/1601
156. van den Brand, M., Protic, Z., Verhoeff, T.: Rcvdiff—a standalone tool for representation, calculation and visualization of model differences. In: ME 2010—International Workshop on Models and Evolution (Oslo, Norway, October 3, 2010; co-located with ACM/IEEE 13th International Conference on Model Driven Engineering Languages and Systems). Association for Computing Machinery, Inc, New York (2011)
157. van Rozen, R., van der Storm, T.: Origin tracking $$+$$+ text differencing $$=$$= textual model differencing. In: Proceedings of the 8th International Conference on Theory and Practice of Model Transformations—Volume 9152, pp. 18–33. Springer, New York (2015). https://doi.org/10.1007/978-3-319-21155-8_2
158. van den Brand, M., Hofkamp, A., Verhoeff, T., Protić, Z.: Assessing the quality of model-comparison tools: a method and a benchmark data set. In: Proceedings of the 2nd International Workshop on Model Comparison in Practice, IWMCP '11, pp. 2–11. ACM, New York (2011). https://doi.org/10.1145/2000410.2000412
159. van den Brand, M., Protić, Z., Verhoeff, T.: Fine-grained metamodel-assisted model comparison. In: Proceedings of the 1st International Workshop on Model Comparison in Practice, IWMCP '10, pp. 11–20. ACM, New York (2010). https://doi.org/10.1145/1826147.1826152
160. Wachsmuth, G.: Metamodel adaptation and model co-adaptation. In: Proceedings of the 21st European Conference on Object-Oriented Programming, ECOOP'07, pp. 600–624. Springer, Berlin (2007). http://dl.acm.org/citation.cfm?id=2394758.2394797
161. Weidlich, M., Dijkman, R., Mendling, J.: The icop framework: identification of correspondences between process models. In: Proceedings of the 22nd International Conference on Advanced Information Systems Engineering, CAiSE'10, pp. 483–498. Springer, Berlin (2010). https://doi.org/10.1007/978-3-642-13094-6_37. http://dl.acm.org/citation.cfm?id=1883784.1883832
162. Wenzel, S., Hutter, H., Kelter, U.: Tracing model elements. In: 2007 IEEE International Conference on Software Maintenance, pp. 104–113 (2007). https://doi.org/10.1109/ICSM.2007.4362623

163. Wohlin, C.: Guidelines for snowballing in systematic literature studies and a replication in software engineering. In: Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering, EASE '14, pp. 38:1–38:10. ACM, New York (2014). https://doi.org/10.1145/2601248.2601268
164. Xing, Z., Stroulia, E.: Umldiff: an algorithm for object-oriented design differencing. In: Proceedings of the 20th IEEE/ACM International Conference on Automated Software Engineering, ASE '05, pp. 54–65. ACM, New York (2005). https://doi.org/10.1145/1101908.1101919
165. Xing, Z.: Model comparison with genericdiff. In: Proceedings of the IEEE/ACM International Conference on Automated Software Engineering, ASE '10, pp. 135–138. ACM, New York (2010). https://doi.org/10.1145/1858996.1859020
166. Xing, Z., Stroulia, E.: Differencing logical uml models. Autom. Softw. Eng. **14**(2), 215–259 (2007). https://doi.org/10.1007/s10515-007-0007-3
167. XML Metadata Interchange (XMI) Specification (2007). https://www.omg.org/spec/XMI/
168. Zhang, H., Babar, M.A., Tell, P.: Identifying relevant studies in software engineering. Inf. Softw. Technol. **53**(6), 625–637 (2011). https://doi.org/10.1016/j.infsof.2010.12.010. http://www.sciencedirect.com/science/article/pii/S0950584910002260. Special Section: Best papers from the APSEC
169. Zhang, Z., Zhang, R., Qin, Z.: Composite-level conflict detection in uml model versioning. Math. Prob. Eng. **2015**, 1–9 (2015). https://doi.org/10.1155/2015/650748
170. Zündorf, A., Wadsack, J.P., Rockel, I.: Merging graph-like object structures. In: Proc. of the 10th International Workshop on Software Configuration Management (SCM-10), Toronto, Canada. (ICSE 2001 Workshop 14) (2001)

Ferenc Somogyi has received his master's degree in 2016 from Budapest University of Technology and Economics and is currently working on his Ph.D. His main topics of interest are compiler and programming language design, model-based software development, and multi-level modeling.

Mark Asztalos has received his master's degree in 2007 and his Ph.D. in 2013 from Budapest University of Technology and Economics. Now, he is an associate professor at the same university. His main topics of interest are model-based software development including automated model processing and code generation, programming language design and static analysis of code.