https://doi.org/10.3311/PPee.12305

Periodica Polytechnica Electrical Engineering and Computer Science, 63(1), pp. 51–65, 2019

A Survey on Text-based Modeling in Model Evolution and Management

Ferenc A. Somogyi[1*], Mark Asztalos[1]

[1] Department of Automation and Applied Informatics, Faculty of Electrical Engineering and Informatics, Budapest University of Technology and Economics, H-1117 Budapest, Magyar Tudósok krt. 2., Hungary
[*] Corresponding author, e-mail: Somogyi.Ferenc@aut.bme.hu

Received: 05 April 2018, Accepted: 02 December 2018, Published online: 13 February 2019

Abstract

Model-driven software engineering methodologies like model-driven engineering aim to improve the productivity of software development by using graph-based models as the main artifacts during development, and generating the source code from these models. The models are usually displayed and edited using a graphical notation. However, they can also be described using a textual notation. This has some advantages and disadvantages compared to the graphical approach. For example, while editing the model, we can better focus on the details instead of a broad overview. Similarly to source code, models evolve rapidly during development. Handling and managing the evolution of models is an important task in model-driven methodologies and is an active research area today. However, there exist few research on text-based modeling approaches, compared to graph-based ones. This paper introduces the text-based modeling research field based on existing literature, and presents the state-of-the-art of the field related to model evolution and management. Our goal is to identify challenges and directions for future research in this field. The main topics covered are model differencing and merging, and the synchronization of the textual and graphical notations.

Keywords

review, survey, model-driven engineering, model-driven development, model-based engineering, domain-specific modeling, model evolution, model management, text-based modeling

1 Introduction

In this section, we briefly introduce text-based modeling, along with the main artifacts and processes involved in it. We also introduce some research fields related to text-based modeling, and summarize the goals and structure of this paper.

1.1 Introduction to text-based modeling

Model-driven software engineering methodologies [1, 2] like model-driven engineering (MDE) [3] aim to improve productivity by using graph-based models as the main artifacts during development. The models typically have a graph-like structure, containing nodes, edges, and other conventional model elements. They describe the problem at a higher level of abstraction, and aim to represent the target domain as accurately as possible. The models are usually defined in the context of a metamodel [4]. Metamodels are on a higher abstraction level than instance models. They describe the elements that the instance models can contain. Metamodels also define constraints that the instance models have to conform to. In this paper, we are focusing on MDE, but text-based modeling can also be applied to other model-driven methodologies that use graph-based models as the main artifacts.

The models can be manipulated in different ways, most often by using model transformations [5, 6]. There are various existing proposals with industrial applications, like VIATRA [7, 8] or ATL [9]. In most cases (with some exceptions, e.g. simulating the model), the goal is to generate a large percent of the source code from the models. Thus, MDE aims to improve traditional software development by requiring less effort and less attention to code details during the implementation of a software or a system. It aims to improve maintainability by using models that describe the problem at a higher level of abstraction. The automatic code generation also reduces the number of code defects during the development [1, 2, 10].

Displaying and editing the models in MDE is often performed by using a graphical (visual) notation, also known as

Cite this article as: Somogyi, F. A., Asztalos, M. "A Survey on Text-based Modeling in Model Evolution and Management", Periodica Polytechnica Electrical Engineering and Computer Science, 63(1), pp. 51–65, 2019. https://doi.org/10.3311/PPee.12305

the concrete syntax [1, 2] of the model. However, using a textual notation for displaying and editing the models also has some advantages. What we consider the main advantages of the graphical and textual notations are illustrated in Table 1. The work by Grönniger et al. [11] was one of the earliest works on text-based modeling, and it details some of these advantages. Outside of the context of modeling, the assumed superiority of graphical notations over textual notations was questioned by many researchers over time [12-14].

Many argue that using the textual and graphical notations in conjunction is the ideal solution, as we can keep the advantages of both [11, 15]. Although not directly related to MDE, successful industrial applications of using both a graphical and a textual notation in UI development also support this statement (e.g. WPF [16], Qt [17] and JavaFX [18]). However, using both notations together raises important questions regarding model evolution and management, most notably, the synchronization of the different notations. It is worth mentioning that while there are some approaches that embed textual information into the graphical notation [19], our focus is on using the textual notation as a stand-alone notation.

1.2 Processes and artifacts in text-based modeling

Using the textual notation for displaying and editing the models in practice is not as prevalent as using the graphical notation. The textual notation is often used in defining model constraints (e.g. OCL [20]), offline storage and model serialization (e.g. XMI [21]), or in the case of behavioral models. In this subsection, we introduce our definition of text-based modeling. We focus on the case where the model itself is described and edited in a textual form via a formal language [22]. The text is processed by the parser of this language, based on the grammar. The result of the parsing process is a parse tree or an abstract syntax tree (AST [23]). The parsing process can be considered a text-to-model (T2M) transformation. In practice, the parser is usually generated from the grammar by a parser generator, like ANTLR [24, 25], Bison [26], or Yacc [27, 28]. The inverse of the T2M transformation is the model-to-text (M2T) transformation [29, 30]. During this process, we generate the textual notation from the model. In this paper, we refer to the approach described in this paragraph as text-based modeling.

Fig. 1 contains an overview of the most common artifacts and processes used in text-based modeling. On Fig. 1, the model is the main artifact. The generated artifacts represent the automatically generated source code. Updating the

Table 1 Graphical and textual notations in domain-specific modeling

Graphical notation	Textual notation
Broad overview	Detailed view
Good readability	Good writability
Domain expert preference	Developer preference
Simulation support	Scalability (~model size)

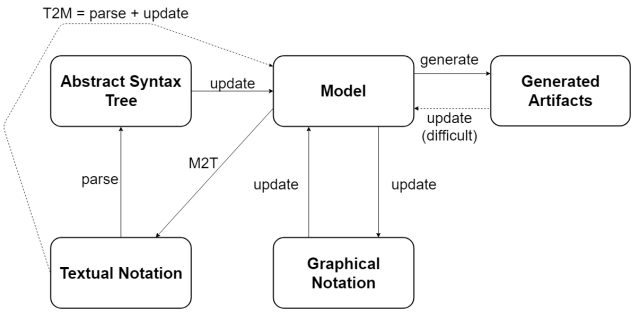

Fig. 1 Common processes and artifacts in text-based modeling

model from the generated artifacts is not a common practice in MDE, as it is a difficult task. This is usually referred to as round-trip engineering [31]. The model and the graphical notation are usually in a two-way association relationship, which means that they update each other when one of them changes. In practice, this is usually done with the aid of a view engine (like in VMTS [32-34], a visual and textual modeling framework), or another similar construct. We choose to omit this concept here, as the details of this process are not relevant to text-based modeling. The textual notation is processed by a parser, parsed into an AST, then, the model is updated based on this AST. This is the T2M transformation. Generating the textual notation from the model is the M2T transformation. The M2T transformation can be performed by directly generating the text, or by building an AST first, and then generating the text from the AST. It is worth noting that the relationship between the model and the generated artifacts, and between the model and the graphical notation are not exclusive to text-based modeling.

1.3 Research fields related to text-based modeling

We believe that the following research fields are the most important that are closely related to text-based modeling, and model evolution and management:

- **Synchronization.** Based on the overview in Fig. 1, we define two distinct synchronization-related challenges that are relevant to text-based modeling. It is worth mentioning that there are some similarities between the synchronization between the textual notation and the model, and the synchronization

between the generated artifacts (source code) and the model. Namely, both the textual notation and the generated code are in a textual form. Thus, results in this field can possibly be applied to the field of incremental code generation as well. We examine the following synchronization problems in detail in Section 3:
  - Synchronizing the graphical and textual notations so that they are always consistent with each other.
  - Synchronizing the textual notation and the model, after the model was edited via another approach (e.g. direct edit, editing via graphical notation).
- **Model differencing and merging (MDM).** Another issue that is relevant to model evolution and management is the differencing and merging of the graph-based models. This is different from source code differencing and merging, as our main artifacts are graph-based models instead of text-based source code. The most important application of MDM is model-based version control systems. We examine existing MDM approaches, and discuss the relevance and role of text-based modeling in this research field in Section 2.
- **Language workbench development.** Language workbenches – a term popularized by Martin Fowler [35] – are software development tools designed to build software using multiple, integrated domain-specific languages [36-38]. Most language workbenches are designed with the single goal of supporting language oriented programming [39]. Some tools, however, are more related to MDE, as the language they provide can be mapped to models. For example, Xtext [40, 41] languages are mapped to EMF [42, 43] models, or Fujaba [44, 45] maps Java code to UML [46] models. Some of these tools are closely related to text-based modeling, since they share the same scanner-parser approach [47]. Therefore, some challenges related to language workbenches are also related to text-based modeling. These challenges deal with the creation and evolution of domain-specific languages. The work published by the Language Workbench Challenge (LWC) community summarizes the open questions and challenges related to language workbench development [48, 49]. In Section 4, we further discuss this topic.

1.4 Goals and structure of the paper

This paper introduces the text-based modeling research field related to model evolution and management, and presents the state-of-the-art in this field. The main research topics covered are MDM approaches and synchronization. As models evolve, version control systems can be used to keep track of different versions of the models. Differencing and merging is an essential task in version control systems, and there exist little research on text-based MDM algorithms. During model evolution, it is also important to keep the different notations of the model synchronized with each other. The goal of this paper is to describe the text-based modeling research field regarding model evolution and management, and to identify challenges and open questions in this field. Another goal of the paper is to present our previous work in this research field, along with our research plans for the future.

The paper is structured as follows. In Section 2, we examine graph-based and text-based MDM approaches and their categorization, and identify directions for research in this field. Section 3 deals with the problem of synchronization in text-based modeling, where we identify some open questions related to synchronization. In Section 4, we discuss the main challenges in language workbench development. We present our own previous work in this research field in Section 5, and outline our main research plans for the future. Finally, Section 6 concludes the paper, highlighting our main findings.

2 Model differencing and merging (MDM)

In this section, we first give a brief introduction to MDM, and reason why it is needed. Afterwards, we outline the main motivations behind text-based MDM, and how it is different from graph-based MDM. Finally, we review the state of graph-based and text-based MDM algorithms in existing research.

2.1 Introduction to MDM

In traditional, source code-based software development, the code is in constant change. Similarly, models in MDE also undergo a lot of changes during their lifecycle. This process is called model evolution [50]. In order to handle the constantly changing source code, we use version control systems (VCS [51]) – like Git [52] or Subversion [53] – to manage the different versions of the same code. An important task in version control systems is the differencing and merging of different versions of the same code.

The concept of version control can also be applied to model-based methodologies [54, 55]. Using version control systems greatly improves the efficiency of teamwork in software development. However, differencing and merging text-based source code is different than

differencing and merging graph-based models. In source code differencing, it is difficult to use the semantics of the code during the process, as the code is usually split into multiple files. Even if the code is physically located in one file, semantically analyzing source code is not a trivial task [56]. Thus, it is difficult to judge that the code is semantically correct. By building an AST from the code, we can use some of the semantics of the code, but in most cases, the user of the VCS is still restricted to raw text differencing and merging [57]. During model differencing, the structure of the model holds most of the information. Differencing and merging graph-based models requires a different approach than source code differencing and merging. Although we can apply raw text differencing to the serialized form of the model (like XMI [21]), it is difficult to locate the precise differences between the two. In text-based modeling, there is a third option: differencing and merging the textual notations of the models. Text-based MDM shares similarities with both raw text-based and graph-based approaches. The characteristics of the main MDM problems are summarized in Fig. 2.

Text-based MDM can be considered a relatively new research field, as there are few existing algorithms. Text-based MDM approaches use text-based artifacts – similar to source code differencing – in addition to graph-based artifacts, which are usually the trees (AST) parsed from the texts. By using the AST, more semantic information can be extracted as opposed to using raw text differencing. This, of course, requires using the parser in order to get the tracing between the AST and the model. Since most modeling environments do not support saving incorrect models, it is also reasonable to demand that the textual notations – and the trees parsed from them – are syntactically and semantically correct. This means that the semantic information we extract from the trees is always correct.

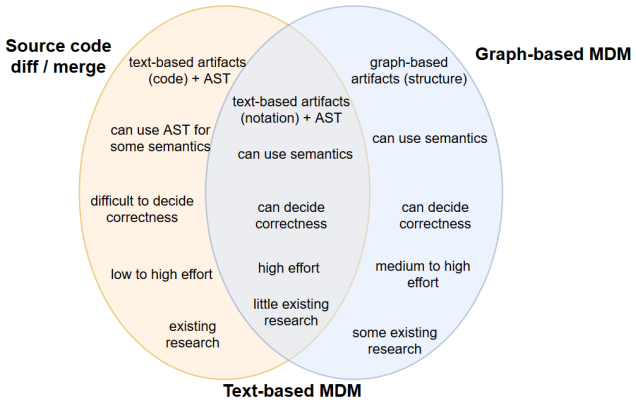

Fig. 2 The main diff / merge problems related to text-based modeling

In addition to being used in version control, MDM approaches can also be applied to other areas as well, like model transformation testing [58, 59]. This process consists of checking the result of the model transformation by using model differencing to compare it with the expected result. The expected result can then be constructed manually or automatically. It is also worth mentioning that there is research focusing on semantic model differencing [60]. These approaches are not solely dependent on the syntactic structure of the models, as they also use semantic diff witnesses to determine the differences.

2.2 Motivations behind text-based MDM

Although text-based MDM algorithms share similarities with other approaches, they also have some differences. We have summarized the main differences when using text-based MDM, compared to raw text differencing and merging, and graph-based MDM methods. These differences also serve as motivation behind researching text-based MDM. They are as follows:

- **Advantages over raw text differencing.** If we use a traditional text differencing and merging tool (e. g. KDiff [57]), we cannot recognize the differences between the models on a semantic level. We can recognize them on the level of the raw text, but not on the level of the model elements. This can result in confusing difference reports. By using a text-based MDM algorithm, and using the abstract syntax trees during the process, we can associate the differences with semantic meaning, for example, when two nodes are in a different order. Thus, using a text-based MDM algorithm is usually better than using raw text differencing and merging. This is illustrated in Fig. 3 and Fig. 4. In the example, we have the textual representations of two library metamodels. While raw differencing highlights the differences in the text, it is difficult to assign semantics to them. For example, it is difficult to notice that the *Title* attribute of the *BookMeta* node has changed. By using a text-based MDM algorithm (and using the AST parsed from the text), we can have a more accurate result.
- **Serialization support.** We can use the textual representations instead of a standard XML-like format like XMI [21] to serialize our models. This results in better readability of the text, especially during version control. Using a text-based MDM algorithm further supports this process.

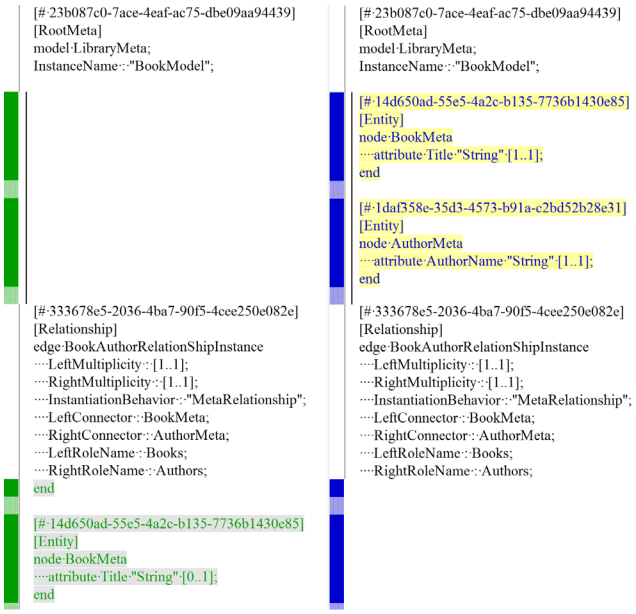

Fig. 3 Raw text differencing textual notations

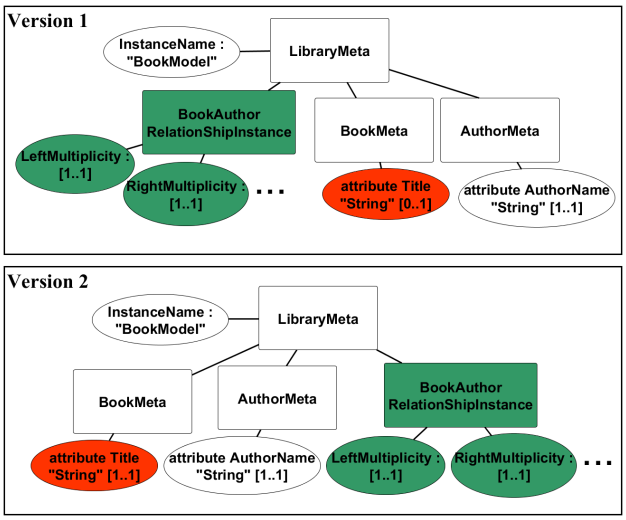

Fig. 4 Differencing textual notations using their AST

- **Synchronization support.** Text-based MDM algorithms can support synchronization by recognizing changes that occurred between two editing sessions. The changes can occur in different ways, e.g. directly editing the model, or editing it via the graphical notation. A text-based MDM algorithm can recognize differences between the newly generated representation, and the previously edited one. We are discussing synchronization in detail in Section 3.
- **Preserving non-semantic information.** It is beneficial to preserve the non-semantic information (e.g. comments, white space) in the textual representations between editing sessions. Text-based MDM methods support this, as we can use them to differentiate between semantic and non-semantic differences.
- **Fallback plan.** If for some reason, a text-based MDM algorithm fails to discover differences accurately, we can fall back to raw text differencing tools, like KDiff [57]. Some reasons for the failure are user error, configuration error, or some other unforeseen circumstances. By falling back to raw text differencing, the differences will not be accurately recognized, but the user is always informed of them. In our opinion, this is a very important advantage, as this makes text-based approaches less error-prone than graph-based approaches. It is more difficult to develop a fallback plan for graph-based approaches, as there is no easy way to compare two graph-based models based on a specific technology. Therefore, reaching 100 % accuracy is usually a difficult task. By using this fallback plan, we can reliably discover every difference, although this comes at the cost of comprehensibility and ease of use, as the differences would have to be interpreted and merged manually by the user.

2.3 Survey of existing approaches

The works by Alanen and Porres [61, 62] are considered by many to be the start of the MDM research field. They were among the first to propose a solution for the differencing and merging of graph-based models. The authors defined the difference and union (and thus, the merge) of two models based on MOF [63, 64] metamodels. Their approach uses operations (e.g. add, delete) to represent changes between two versions of a model. Future research directions mentioned in the paper include the need for more metamodel-specific solutions, and support for automatic merge conflict resolution. In addition, since the algorithm presented by the authors is dependent on MOF, at that time, there was also a need for more metamodel-independent approaches.

Graph-based model differencing and merging can be approached in numerous ways. The work presented in the paper by Kolovos et al. [65] focuses on the differencing in the MDM process. The first phase of differencing in MDM is matching the model elements in the two models based on some criteria. Their categorization of matching approaches covers the different matching strategies during the differencing process in a general way. The authors split model matching approaches into the following categories:

- **Static identity-based matching.** The matching is done based on static identifiers that must be unique for every model element. This approach can only be

used in simple cases, and the identifiers have to be maintained at all times. However, if it can be used, it is accurate and easy to implement.

- **Signature-based matching.** Similarly to static identity-based matching, these approaches also compare identifiers and give a true / false answer. However, the identifiers in this case are dynamic, as they can be a combination of the features of the model elements. This must be configured by a user-defined function, which increases the effort of implementing these approaches. The function has to be configured properly in order to achieve high accuracy.
- **Similarity-based matching.** The result of these matching approaches is not a true / false value, but a number that represents the similarity between the two model elements. If the number is above a certain threshold, the elements are considered to be matching. The similarity metric is calculated from the features of the model elements. The different features are weighted differently. The challenge in implementing this approach is finding the correct weight functions for the method in order to achieve high accuracy. This approach has the advantage of being generic (modeling language-independent), and if configured properly, it can achieve better accuracy than signature-based methods.
- **Custom language-specific matching algorithms.** These approaches are tailored to a specific modeling language in order to use the precise semantics of that language. Thus, they are very accurate, but are not general, and are usually difficult to implement.

Graph-based model differencing approaches usually fall into one (or in some cases, more) of these categories. We note that the differences between the different approaches can be measured in a trade-off between the following metrics:

- **Accuracy.** The percentage of correctly identified differences between the two versions.
- **Generality.** The number of modeling languages that the approach can be applied to.
- **Effort.** The time and effort required to implement the approach.
- **Performance.** The runtime performance of the algorithm.

Kolovos et al. [65] mention the difficulty of objectively and formally comparing the different approaches, which tends to be a recurring problem in this research field.

Altmanninger et al. [66] focused on the merging in the MDM process, and raised open questions that are still relevant today. The paper examines three-way (model) merging methods [67, 68]. The authors formalize the MDM process by splitting it into three distinct phases:

- **Change detection.** In this phase, the changes between the ancestor model V0 and the two modified versions V0' and V0'' are calculated. The detection can be done in a **state-based** (only the final states are considered) or in an **operation-based** (the model editor tracks the changes as operations) way [69, 70]. The authors differentiate between generic atomic (model independent operations like add), specific atomic (model dependent operations like rename), and specific composite (model dependent, complex operations, like refactor) changes. Detecting more complex changes improve the quality of the merged model.
- **Conflict detection.** Based on the result of the change detection phase, conflicting changes are identified. The authors differentiate between two conflict types: equivalent and contradicting conflicts. Equivalent conflicts (e.g. two distinct add operations) can be merged automatically, while contradicting conflicts (e.g. update and delete on the same model element) cannot be merged automatically in most cases.
- **Inconsistency detection.** This phase focuses on the inconsistencies between the merged model (after the conflict detection) and the metamodel. The authors categorize these inconsistencies into syntactic and semantic problems. Syntactic problems (e.g. cyclic inheritance) can be automatically detected based on the metamodel, while semantic problems (e.g. same concept implemented twice in the merged mode) are very difficult to detect automatically.

After evaluating four versioning systems (Subversion, IBM RSA [71], EMF Compare [72] and Unicase [73]), the authors defined key areas, where future research can be done. Most of these are still relevant today. They are as follows:

- **Benchmark availability.** There is a lack of detailed (formal) requirements and well-defined, expected run-time behavior of model versioning systems. In addition, there are no test cases for testing different capabilities of these systems. There have been some proposals since then, but there is still research to be done in this area [74].

- **Unreliable conflict detection.** There amount of false positives and false negatives during conflict detection in existing approaches is too high. There is a need for reliable (accurate) conflict detection approaches, especially in the case of model-independent (general) solutions.
- **Confusing difference report.** Differences are displayed differently in every tool. Moreover, they are usually not displayed in the concrete syntax of the model, but rather in an abstract tree or list representation. This results in worse readability.
- **Single diagram support.** Model-independent (general) model version tools are needed. While there are more general approaches now than before, they are still not very prevalent in practice.
- **Unreliable conflict resolution.** Automatic conflict resolution support for the existing tools is low. This issue still exists today, albeit to a lower degree.

Text-based MDM can be considered a young research field. We have mentioned that it shares similarities with source code differencing and merging, and graph-based MDM approaches. Since there is little existing research in this area, there is little information on what we can gain (e.g. performance, accuracy and generality) by using text-based approaches over graph-based ones. While we have outlined the main differences compared to graph-based MDM approaches in this section, there is still a need for more studies on this subject.

Van Rozen and van der Storm proposed TMDIFF [75, 76], a differencing approach for textual modeling languages. In the problem described by the authors, the models are created from the textual languages. Instead of the M2T transformation, origin tracking (a form of traceability [77]) is used to map the model to the text. Textual artifacts are the main artifacts instead of the model. Therefore, this problem is somewhat different from the text-based modeling we defined in this paper.

Finally, we would like to note that there are many existing approaches and solutions in MDM for different modeling languages. There are proposals for various UML diagrams [78-80], specific modeling environments [72], or ones introducing new approaches, like design-space exploration [81]. According to our experience, the number of graph-based approaches heavily outweighs the number of text-based approaches. Our future goal is to conduct a systematic literature review to prove this conjecture.

2.4 Open questions

Based on the ideas presented in this section, we identify the following research directions related to text-based MDM:

- **Objective comparison and benchmarking.** A recurrent problem in research related to MDM is objectively comparing and classifying different algorithms. Objective comparison also calls for formalization. Moreover, there is a lack of benchmarking to use. This topic is not strictly related to text-based modeling, as these problems exist for graph-based MDM approaches as well. A difficult task is deciding what metrics we can apply to achieve an objective comparison. In addition, lots of MDM approaches are designed for one modeling language, making an objective, technology-independent classification a challenging task.
- **Adapting existing research.** A direction that is specific to text-based modeling is the adaptation of existing research for text-based MDM approaches. The main question is if key concepts and methods from research on graph-based MDM can be applied to research on text-based MDM. Since an AST can also be considered a graph, some of these concepts could – in theory – be applied. For example, applying similarity-based comparison on the trees parsed from the textual notations might make the algorithm more general, at the cost of reduced accuracy. Text-based MDM is still the ideal choice for text-based modeling, as most advantages it brings that we discussed before (e.g. preserving the non-semantic information in the text) during the differencing process greatly benefits text-based modeling. We consider a thorough examination of the pros and cons of applying these concepts a possible research direction.
- **General text-based MDM algorithms.** An important question is if a general text-based MDM algorithm can be developed. Developing a general text-based MDM algorithm is difficult, as we also have to tailor our approach to handle as many textual languages as possible. It can also be worthwhile to examine how the trade-offs are comparable to general graph-based MDM methods.
- **Evolution of the language.** Models evolving during development are one of the main motivations for research behind MDM. However, in text-based modeling, the language that describes the textual notation can also change over time. An interesting research direction would be to develop an algorithm

that adapts to these changes as much as possible. For example, if the syntax of our language changes, we want our algorithm to be compatible with the older textual notations as well. However, the semantics of the language can also change over time. Another interesting question is if we can define metrics in order to measure the flexibility that our algorithm has in this regard.

These are the main directions that we consider to be the most promising in this field. One of our main motivations behind researching text-based MDM is to examine how they compare to graph-based MDM. Two of the main research directions we listed above are closely related to this problem:

1. objective comparison and classification is needed in order to compare the algorithms, and
2. the adaptation of existing research might close the gap between graph-based and text-based MDM approaches.

3 Synchronization in text-based modeling

As models evolve, it is important to keep the different notations of the model up-to-date, or synchronized with each other. In this section, we propose two categories of synchronization problems in text-based modeling. We also examine existing solutions for synchronization, and identify areas where future research can be done.

3.1 Categorization of synchronization

During the examination of the processes and artifacts in text-based modeling in Section 1, we have identified two forms of synchronization:

- **Between the textual notation and the model.** The model and textual notation has to be updated when one of them changes. This can be done by the M2T and T2M transformations we discussed before. We also have to decide whether we want to continuously synchronize every change, or use a push-pull model instead. In some cases, the overhead in performance is not worth keeping the artifacts constantly synchronized. It is worth mentioning that the graphical notation also needs to be synchronized with the model, but the focus of our paper is on text-based modeling.
- **Between the graphical and textual notations**. When the content of one of the notations changes, the other one needs to be updated in order for the displayed information to remain consistent. Fig. 1 shows that the graphical and textual notations are usually independent of each other. This means that the model also has to be updated. Thus, this form of synchronization includes the previous one.

We would like to note that we mentioned incremental code generation before, which can be considered the synchronization process between the model and the generated source code. If it is two-ways, it is usually called round-trip engineering [31]. In this paper, we are not focusing on incremental code generation, though the results in this field could also be applied to the field of incremental code generation as well.

In this paper, and in our research, synchronization between the textual notation and the model is our focus, as it is most relevant to text-based modeling. We propose the (informal) definitions for two types of synchronization, depending on when and how often do we need to synchronize the textual notation and the model. They are illustrated in Fig. 5 and are as follows:

- **Online synchronization.** Changes that occur in the model or the textual notation need to be immediately reflected in the other. For example, when we are editing and updating the model using a textual editor.
- **Offline synchronization.** Changes that occurred between the model and the textual notation over an extended period of time have to be detected. For example, when we are opening the textual notation after the model changed through other means like direct editing, or editing via the graphical notation.

Offline synchronization is similar to state-based differencing [69, 70] in MDM. This means that we have to detect an unknown amount of changes that occurred over an unknown period of time. In online synchronization, we know exactly what changes occurred and in what order. It can be argued that complex operations in a textual editor (e.g. cutting and pasting a large chunk of text) counts

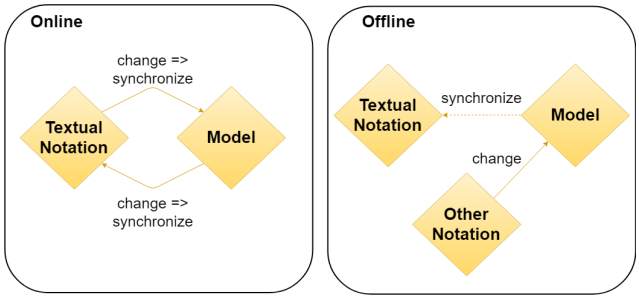

Fig. 5 Online and offline synchronization

as offline synchronization. Thus, the line between the two categories is not always clear.

3.2 Survey of existing approaches

Fairmichael and Kiniry [82] formalized the relationship between the textual and graphical notations of the Business Object Notation [83] modeling language. The textual and graphical notations are often loosely connected, or not connected at all, so formalizing the relationship between the two can be very helpful for future research. The authors state that this is a research direction where more research can be done. They also mention that one of the main applications for their approach is in MDM.

There are many existing proposals for the online and offline synchronization problems we defined [84-87]. In this paper, we are examining two of them, to represent each category.

Oskar van Rest et al. [88] proposed a solution for the online synchronization of the graphical and textual notations. Their approach recovers from errors during synchronization and preserves the layout of both notations. It was implemented to synchronize textual editors generated by Spoofax [89], and graphical editors generated by GMF [90]. They use model-to-tree transformations instead of the model-to-text transformations that we discussed previously.

Angyal et al. [91, 92] proposed an approach for the offline synchronization of the textual notation and the model, and thus, the textual and graphical notations. They implemented their prototype in the VMTS framework. The textual notation and the parser are generated by a metamodel-based approach. For every model element, the template attribute that maps a textual representation to the element has to be filled out. The synchronization is a three-way merge process, with the common ancestor being the stored textual notation. The differences are handled as edit scripts, thus, this is an operation-based approach.

3.3 Open questions

As opposed to online synchronization, offline synchronization tends to be less accurate and more reliant on the user. The reason for this is because offline synchronization is closely related to MDM. Offline synchronization and state-based model differencing are very similar, since in both cases:

1. differencing and merging is needed, and
2. an unknown amount of changes occur over an unknown period of time.

Thus, they share some problems that we discussed in Section 2, of which what we consider the most important are as follows:

- **Automatic conflict resolution.** It is not trivial to automatically solve conflicts that occur during the synchronization. This is mostly due to the inherent differences between the different notations.
- **General synchronization approaches.** Similarly to MDM, developing general algorithms for synchronization is a difficult task. Having such general algorithms is beneficial, since we do not have to develop a new algorithm for a new language.
- **Feedback and user involvement.** Similarly to version control systems, the result of the automatic synchronization is usually displayed for user supervision. The form of display and the ease of user involvement are areas where future research can be done.

4 Challenges in language workbench development

This section briefly reviews the state of language workbenches, and their relevance to text-based modeling. We also take a look at recent challenges in this research area. We consider some of these problems important to text-based modeling, as they deal with the editing and management of the textual notation.

Language workbenches (LW) are tools that specialize in building software using multiple, integrated domain-specific languages [35, 37]. The focus is on defining, processing and using these languages during software development.

Language workbenches are usually sorted [49] into one of the following categories:

- **Graphical** workbenches support languages that use the graphical notation. Some examples are VMTS [32-34], MetaEdit+ [93] and GME [94].
- **Textual** workbenches support textual languages. In this case, the DSL is defined and processed by a formal language. This is often referred to as the scanner / parser approach. Textual language workbenches often make use of advances in IDE and editor technology, as editors are usually generated along with the parser. Some examples are Xtext [40, 41] and Spoofax [89].
- **Projectional** workbenches move away from the scanner / parser approach by using syntax-directed projectional editors. Using these editors, the user can directly edit the abstract syntax, and define the concrete syntax separately. This is a more language-oriented approach, and enables the mix of textual and non-textual

notations. Some examples are JetBrains MPS [95] and the Intentional Domain Workbench [96].

In text-based modeling, we often use formal languages to describe and process the textual notation during the M2T and T2M transformations. Due to the use of the scanner / parser approach, textual language workbenches have much in common with text-based modeling. It is worth mentioning that some of these workbenches (like Xtext) also map the language to a domain-specific model. Therefore, advances in textual language workbenches – and consequently, IDE and editor technologies – are also beneficial to text-based modeling. Thus, we consider some of the challenges and open questions in this field to be relevant to text-based modeling.

The annual Language Workbench Challenge (LWC) was launched in 2011 to allow researchers in this field to compare their approaches [48]. The first four challenges were issued to solve specific problems related to language workbenches by implementing a different language each year. They also proposed a feature model aimed to describe the features a workbench can have. These features are split into categories like notation, semantics, editor, validation, or composability.

In 2015, the LWC community defined benchmark problems for language workbenches and called for solutions for these problems [49]. Briefly summarized, their categorization is as follows:

- **Notation**. These problems address issues that are relevant to the notation of languages. Some of the problems included here (but not limited to) are related to metadata annotations, computed properties, optional hiding. The most important problem related to text-based modeling is the support for multiple notations.
- **Evolution and reuse**. These problems concern the modular extension and the evolution of languages over time. The evolution of the formal language is a relevant problem in text-based modeling as well. Moreover, keeping the new version as compatible as possible with older textual notations can be useful during the MDM process.
- **Editing**. These problems are related to the editor of the language. Solving these problems advances IDE and editor technology as well. Some examples mentioned here are editing incomplete programs (or the textual notation in the case of text-based modeling), referencing missing items, restructuring, and formatting preservations. Making the editor of the textual notation as user-friendly as possible greatly increases the ease of use of text-based modeling.

Out of the challenges mentioned by the LWC, we consider the following to be the most relevant to text-based modeling:

- **Supporting multiple notations**. By supporting the graphical and textual notations as equivalent and views of the model, synchronization issues can be solved more easily. However, offline synchronization would still remain an issue as we could still modify the model directly through its persistent structure.
- **Syntax migration**. When the syntax of the language changes (e.g. changing a keyword), we would like our old textual notations to be as compatible as possible with the new syntax. This is related to one of our open questions in Section 2.
- **Structure migration**. Similar to syntax migration, but instead of the syntax, the underlying structure of the AST is changed instead. The question is how can existing textual notations be migrated to the new representation and in what ways does this affect the non-semantic information in the text.
- **Editing problems**. As we have discussed earlier, solving problems related to the editor advances text-based modeling. We believe the two most interesting problems are formatting preservation and end-user defined formatting. The former deals with refactoring and quick-fixes; these should not alter the formatting of the textual notation. It is also something that we strive for during offline synchronization and text-based MDM. The latter specifies a need for formatters that the users can customize to their own needs.

5 Previous work and personal research plans

In this section, we present our own previous work in the text-based modeling research field. After that, we briefly present our research plans for the future, according to the open questions discussed in this paper.

In previous work, we developed a text-based MDM algorithm [97] that operates on the textual representations of VMTS [32-34] models. The models are described by a textual language called VMDL (Visual Model Definition Language) that was also created by us. The synchronization between model and text is done in an offline way (as described in Section 3), as they are synchronized once the notation is saved. The mapping between model and text is done by a formal language developed with

ANTLR [24, 25]. We also formally verified the algorithm based on certain aspects [98].

Currently, our algorithm works only with VMTS models, but can – in theory – support other modeling languages. This is achieved by using the parser of the textual language during the MDM process, namely, demanding certain requirements from it. For example, when trying to match two model elements with each other, we ask the parser if they can be considered a match based on the AST. This is considered to be a dynamic (signature-based) approach as described by Kolovos et al. [65] and presented in Section 2.

Based on the research presented in this paper, we briefly present our own research plans for the future:

- **Systematic review.** We plan to conduct a systematic literature review (SLR) to further support the conclusions of this paper, especially regarding the ones presented in Section 2.
- **Comparing text-based and graph-based MDM.** Based on the reasoning presented in Section 2, our goal is to provide a classification system that we can use to compare the different MDM algorithms. We aim to introduce a formal model that can be used as a basis during the comparison.
- **Develop a general text-based MDM method.** We aim to improve our text-based MDM algorithm to be as general as possible. We also aim to examine the trade-offs that we have to make, and compare it to general graph-based MDM methods.
- **Automatic conflict resolution.** We intend to improve our algorithm so that it can automatically discover and resolve most conflicts that arise during the MDM process. Achieving this greatly improves the usability of an MDM algorithm, provided that the automatic conflict detection and resolution are proven to be correct at all times. Otherwise, the user intervention effort can even be higher than without automatic conflict resolution.

6 Conclusion

In this paper, we introduced the text-based modeling research field, and presented the state-of-the-art related to this field. We focused on two areas relevant to text-based modeling: model differencing and merging (MDM), and synchronization. We also discussed that challenges in language workbench development can have some in text-based modeling as well.

We discussed that text-based MDM is a relatively new direction in the field of MDM. We showed how text-based MDM is different from raw text differencing and merging, and from graph-based MDM. We outlined our main motivations behind researching text-based MDM. We concluded that the lack of objective comparison and benchmarking are a recurring problem in this field, and identified several directions where future research can be done.

In model evolution, keeping the textual and graphical notations and the model synchronized is an important task. We categorized synchronization problems into two distinct categories: online and offline synchronization. We concluded that offline synchronization is very similar to state-based MDM, and thus, they share some open questions as well.

We presented the state-of-the-art of, and identified the main challenges in language workbench development, based on the work of the Language Workbench Challenge (LWC) community. Language workbenches are related to text-based modelling, as the two fields have common challenges that deal with the editing and management of the textual notation.

Finally, we presented our previous work in this research field and outlined our main plans for the future. These plans are closely related to the open questions we discussed earlier.

References

[1] Beydeda, S., Book, M., Gruhn, V. "Model-Driven Software Development", 1st ed., Springer-Verlag, Berlin, Germany, 2005. https://doi.org/10.1007/3-540-28554-7

[2] Kelly, S., Tolvanen, J.-P. "Domain-Specific Modeling: Enabling Full Code Generation", 1st ed., Wiley-IEEE Computer Society Press, Los Alamitos, USA, 2007. https://doi.org/10.1002/9780470249260

[3] Schmidt, D. C. "Guest Editor's Introduction: Model-Driven Engineering", Computer, 39(2), pp. 25–31, 2006. https://doi.org/10.1109/MC.2006.58

[4] Paige, R. F., Kolovos, D. S., Polack, F. A. C. "A tutorial on metamodelling for grammar researchers", Science of Computer Programming, 96(4), pp. 396–416, 2014. https://doi.org/10.1016/j.scico.2014.05.007

[5] Ehrig, H., Ehrig, K., Prange, U., Taentzer, G. "Fundamentals of Algebraic Graph Transformation", 1st ed., Springer-Verlag, Berlin, Germany, 2006. https://doi.org/10.1007/3-540-31188-2

[6] Sendall, S., Kozaczynski, W. "Model transformation: The heart and soul of model-driven software development"", IEEE Software, 20(5), pp. 42–45, 2003. https://doi.org/10.1109/MS.2003.1231150

[7] 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 Theory and Practice of Model Transformations, L'Aquila, Italy, 2015, pp. 101–110. https://doi.org/10.1007/978-3-319-21155-8_8

[8] Eclipse Foundation, "VIATRA", [online] Available at: http://www.eclipse.org/viatra/ [Accessed: 03 April 2018]

[9] Jouault, F., Kurtev, I. "Transforming Models with ATL", In: MoDELS 2005 International Workshops Doctoral Symposium, Educators Symposium, Montego Bay, Jamaica, 2005, pp. 128–138. https://doi.org/10.1007/11663430_14

[10] Selic, B. "The pragmatics of model-driven development", IEEE Software, 20(5), pp. 19–25, 2003. https://doi.org/10.1109/MS.2003.1231146

[11] Grönninger, H., Krahn, H., Rumpe, B., Schindler, M., Völkel, S. "Textbased Modeling", presented at 4th International Workshop on Software Language Engineering, Nashville, USA, Oct. 2007.

[12] Green, T. R. G., Petre, M., Bellamy, R. K. E. "Comprehensibility of Visual and Textual Programs: A Test of Superlativism Against the 'Match-Mismatch' Conjecture", In: Koenemann-Belliveau, J., Moher, T. G., Robertson, S. P. (eds.) Proceedings of the Fourth Annual Workshop on Empirical Studies of Programmers, 91(743), 1991, pp. 121–146. [online] Available at: https://www.researchgate.net/publication/238987815_Comprehensibility_of_visual_and_textual_programs_A_test_of_superlativism_against_the_%27match-mismatch%27_conjecture [Accessed: 03 April 2018]

[13] Petre, M. "Why looking isn't always seeing: readership skills and graphical programming", Communications of the ACM, 38(6), pp. 33–44, 1995. https://doi.org/10.1145/203241.203251

[14] Green, T. R. G., Petre, M. "When Visual Programs are Harder to Read than Textual Programs", In: 6th European Conference on Cognitive Ergonomics, Balatonfüred, Hungary, 1992, pp. 167–180. [online] Available at: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.57.1633 [Accessed: 03 April 2018]

[15] Pérez Andrés, F., de Lara, J., Guerra, E. "Domain Specific Languages with Graphical and Textual Views", In: Third International Symposium, Kassel, Germany, 2007, pp. 82–97. https://doi.org/10.1007/978-3-540-89020-1_7

[16] Jones, A., Freeman, A. "Windows Presentation Foundation", In: Visual C# 2010 Recipes, 1st ed., Apress, New York City, USA, 2010, pp. 789–904. https://doi.org/10.1007/978-1-4302-2526-3_17

[17] The Qt Company "Qt", [online] Available at: https://www.qt.io/ [Accessed: 03 April 2018]

[18] Topley, K. "JavaFX Developer's Guide", 1st ed., Addison-Wesley, Boston, USA, 2010.

[19] Scheidgen, M. "Textual Modelling Embedded into Graphical Modelling", In: 4th European Conference on Model Driven Architecture - Foundations and Applications, Berlin, Germany, 2008, pp. 153–168. https://doi.org/10.1007/978-3-540-69100-6_11

[20] Warmer, J. B., Kleppe, A. G. "The Object Constraint Language: Precise Modelling with UML", 1st ed., Addison-Wesley, Boston, USA, 1998.

[21] Object Management Group (OMG), "About the XML Metadata Interchange Specification Version 2.1.1", 2007. [online] Available at: https://www.omg.org/spec/XMI/2.1.1/About-XMI/ [Accessed: 03 April 2018]

[22] Moll, R. N., Arbib, M. A., Kfoury, A. J. "An Introduction to Formal Language Theory", 1st ed., Springer-Verlag, New York, USA, 1988. https://doi.org/10.1007/978-1-4613-9595-9

[23] Aho, A. V., Lam, M. S., Sethi, R., Ullman, J. D. "Compilers: Principles, Techniques, and Tools", 2nd ed., Addison-Wesley, Boston, USA, 2006.

[24] Parr, T. "The Definite ANTLR 4 Reference", 2nd. ed., Pragmatic Bookshelf, Raleigh, USA, 2013.

[25] Parr, T. "ANTLR", 2014. [online] Available at: http://www.antlr.org/ [Accessed: 03 April 2018]

[26] Donnelly, C., Stallman, R. "GNU Bison - The Yacc-compatible Parser Generator", Free Software Foundation, Cambridge, 2015. [online] Available at: https://www.gnu.org/software/bison/manual/ [Accessed: 03 April 2018]

[27] Merrill, G. H. "Parsing Non-LR (k) grammars with yacc", Software: Practice and Experience, 23(8), pp. 829–850, 1993. https://doi.org/10.1002/spe.4380230803

[28] Johnson, S. C. "Yacc: Yet Another Compiler-Compiler", [online] Available at: http://dinosaur.compilertools.net/yacc/ [Accessed: 03 April 2018]

[29] Czarnecki, K., Helsen, S. "Classification of Model Transformation Approaches (2003)", In: OOPSLA'03 Workshop on Generative Techniques in the Context of Model-Driven Architecture, Anaheim, USA, 2003, pp. 1–17. [online] Available at: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.122.8124 [Accessed: 03 April 2018]

[30] Rose, L. M., Matragkas, N., Kolovos, D. S., Paige, R. F. "A feature model for model-to-text transformation languages", In: 4th International Workshop on Modeling in Software Engineering (MISE), Zurich, Switzerland, 2012, pp. 57–63. https://doi.org/10.1109/MISE.2012.6226015

[31] Hettel, T., Lawley, M., Raymond, K. "Model Synchronisation: Definitions for Round-Trip Engineering", In: 1st International Conference on Theory and Practice of Model Transformations, Zurich, Switzerland, 2008, pp. 31–45. https://doi.org/10.1007/978-3-540-69927-9_3

[32] Levendovszky, T., Lengyel, L., Mezei, G., Charaf, H. "A Systematic Approach to Metamodeling Environments and Model Transformation Systems in VMTS", Electronic Notes in Theoretical Computer Science, 127(1), pp. 65–75, 2005. https://doi.org/10.1016/j.entcs.2004.12.040

[33] Angyal, L., Asztalos, M., Lengyel, L., Levendovszky, T., Madari, I., Mezei, G., Mészáros, T., Siroki, L., Vajk, T. "Towards a Fast, Efficient and Customizable Domain-Specific Modeling Framework", In: IASTED International Conference on Software Engineering, Innsbruck, Austria, 2009, pp. 11–16. [online] Available at: https://www.actapress.com/Abstract.aspx?paperId=34623 [Accessed: 03 April 2018]

[34] Visual Modeling Group "Visual Modeling and Transformation System", [online] Available at: http://vmts.aut.bme.hu [Accessed: 03 April 2018]

[35] Fowler, M. "Language Workbenches: The Killer-App for Domain Specific Languages?", 2005. [online] Available at: http://www.martinfowler.com/articles/languageWorkbench.html [Accessed: 03 April 2018]

[36] van Deursen, A., Klint, P., Visser, J. "Domain-specific languages: an annotated bibliography", ACM SIGPLAN Notices, 35(6), pp. 26–36, 2000. https://doi.org/10.1145/352029.352035

[37] Mernik, M., Heering, J., Sloane, A. M. "When and how to develop domain-specific languages", ACM Computing Surveys (CSUR), 37(4), pp. 316–344, 2005. https://doi.org/10.1145/1118890.1118892

[38] Kosar, T., Bohra, S., Mernik, M. "Domain-Specific Languages: A Systematic Mapping Study", Information and Software Technology, 71, pp. 77–91, 2016. https://doi.org/10.1016/j.infsof.2015.11.001

[39] Ward, M. P. "Language-Oriented Programming", Software-Concepts and Tools, 15(4), pp. 147–161, 1994. [online] Available at: https://www.researchgate.net/publication/234125675_Language_Oriented_Programming [Accessed: 03 April 2018]

[40] Eysholdt, M., Behrens, H. "Xtext: implement your language faster than the quick and dirty way", In: International Conference Companion on Object Oriented Programming Systems Languages and Applications Companion (OOPSLA '10), Reno/Tahoe, Nevada, USA, 2010, pp. 307–309. https://doi.org/10.1145/1869542.1869625

[41] Eclipse Foundation "Xtext", [online] Available at: https://eclipse.org/Xtext/ [Accessed: 03 April 2018]

[42] Steinberg, D., Budinsky, F., Paternostro, M., Merks, E. "EMF: Eclipse Modeling Framework", 2nd ed., Addison-Wesley, Boston, USA, 2008.

[43] Eclipse Foundation "Eclipse Modeling Framework (EMF)", [online] Available at: https://eclipse.org/modeling/emf [Accessed: 03 April 2018]

[44] Nickel, U., Niere, J., Zündorf, A. "The FUJABA environment", In: International Conference on Software Engineering, ICSE 2000, Limerick, Ireland, 2000, pp. 742–745. https://doi.org/10.1145/337180.337620

[45] Fujaba Core Development Group "Fujaba", 2012. [online] Available at: http://www.fujaba.de/ [Accessed: 03 April 2018]

[46] Fowler, M. "UML Distilled: A Brief Guide to the Standard Object Modeling Language", 3rd ed., Addison-Wesley, Boston, USA, 2003.

[47] Merkle, B. "Textual Modeling Tools: Overview and Comparison of Language Workbenches", In: International Conference Companion on Object Oriented Programming Systems Languages and Applications Companion (OOPSLA '10), Reno/Tahoe, Nevada, USA, 2010, pp. 139–148. https://doi.org/10.1145/1869542.1869564

[48] Erdweg, S., van der Storm, T., Völter, M., Boersma, M., Bosman, R., Cook, W. R., Gerritsen, A., Hulshout, A., Kelly, S., Loh, A., Konat, G., Molina, P. J., Palatnik, M., Pohjonen, R., Schindler, E., Schindler, K., Solmi, R., Vergu, V., Visser, E., van der Vlist, K., Wachsmuth, G., van der Woning, J. "The State of the Art in Language Workbenches: Conclusions from the Language Workbench Challenge", In: 6th International Conference on Software Language Engineering (SLE 2013), Indianapolis, IN, USA, 2013, pp. 197–217. https://doi.org/10.1007/978-3-319-02654-1_11

[49] Erdweg, S., van der Storm, T., Völter, M., Tratt, L., Bosman, R., Cook, W. R., Gerritsen, A., Hulshout, A., Kelly, S., Loh, A., Konat G., Molina, P. J., Palatnik, M., Pohjonen, R., Schindler, E., Schindler, K., Solmi, R., Vergu, V., Visser, E., van der Vlist, K., Wachsmuth, G., van der Woning, J. "Evaluating and comparing language workbenches: Existing results and benchmarks for the future", Computer Languages, Systems and Structures, 44(A), pp. 24–47, 2015. https://doi.org/10.1016/j.cl.2015.08.007

[50] Paige, R. F., Matragkas, N., Rose, L. M. "Evolving models in Model-Driven Engineering: State-of-the-art and future challenges", Journal of Systems and Software, 111, pp. 272–280, 2016. https://doi.org/10.1016/j.jss.2015.08.047

[51] Spinellis, D. "Version control systems", IEEE Software, 22(5), pp. 108–109, 2005. https://doi.org/10.1109/MS.2005.140

[52] GIT "GIT", [online] Available at: https://git-scm.com/ [Accessed: 03 April 2018]

[53] Collins-Sussman, B., Fitzpatrick, B. W., Pilato, C. M. "Version Control with Subversion - The Official Guide and Reference Manual", 2nd ed., CreateSpace, Paramount, CA, USA, 2009.

[54] Brosch, P., Kappel, G., Langer, P., Seidl, M., Wieland, K., Wimmer, M. "An Introduction to Model Versioning", In: 12th International School on Formal Methods for the Design of Computer, Communication and Software Systems (SFM 2012), Bertinoro, Italy, 2012, pp. 336–398. https://doi.org/10.1007/978-3-642-30982-3_10

[55] Brosch, P., Langer, P., Seidl, M., Wieland, K., Wimmer, M., Kappel, G. "The Past, Present, and Future of Model Versioning", In: Rech, J., Bunse, C. (eds.) Emerging Technologies for the Evolution and Maintenance of Software Models, 1st ed., IGI Global, Hershey, PA, USA, 2012, pp. 410–443. https://doi.org/10.4018/978-1-61350-438-3.ch015

[56] Falleri, J.-R., Morandat, F., Blanc, X., Martinez, M., Monperrus, M. "Fine-grained and accurate source code differencing", In: 29th International Conference on Automated Software Engineering (ASE '14), Vasteras, Sweden, 2014, pp. 313–324. https://doi.org/10.1145/2642937.2642982

[57] Eibl J. "KDiff3", 2014. [online] Available at: http://kdiff3.sourceforge.net/ [Accessed: 03 April 2018]

[58] Lin, Y., Zhang, J., Gray, J. "Model comparison: A key challenge for transformation testing and version control in model driven software development (2004)", In: OOPSLA/GPCE Workshop on Best Practices for Model-Driven Software Development, Vancouver, Canada, 2004, pp. 219–236. [online] Available at: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.91.5709 [Accessed: 03 April 2018]

[59] Kolovos, D. S., Paige, R. F., Polack, F. A. C. "Model Comparison: A Foundation for Model Composition and Model Transformation Testing", In: International Workshop on Global Integrated Model Management (GaMMa '06), Sanghai, China, 2006, pp. 13–20. https://doi.org/10.1145/1138304.1138308

[60] Maoz, S., Ringert, J. O., Rumpe, B. "A Manifesto for Semantic Model Differencing", In: International Conference on Model Driven Engineering Languages and Systems (MODELS 2010), Oslo, Norway, 2010, pp. 194–203. https://doi.org/10.1007/978-3-642-21210-9_19

[61] Alanen, M., Porres, I. "Difference and Union of Models", In: International Conference on the Unified Modeling Language (UML 2003), San Francisco, CA, USA, 2003, pp. 2–17. https://doi.org/10.1007/978-3-540-45221-8_2

[62] Porres, I. "Union and Difference of Models, 10 years later", In: 16th International Conference on Model Driven Engineering Languages and Systems (MODELS 2013), Miami, USA, 2013, pp. 1–5. [online] Available at: http://ceur-ws.org/Vol-1115/talk1.pdf [Accessed: 03 April 2018]

[63] Tang, W. "Meta Object Facility", In: Liu, L., Özsu, M. T. (eds.) Encyclopedia of Database Systems, 1st ed., Springer, Boston, MA, USA, 2009. https://doi.org/10.1007/978-0-387-39940-9_914

[64] Object Management Group (OMG) "Meta-Object Facility", [online] Available at: http://www.omg.org/mof [Accessed: 03 April 2018]

[65] 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: 2009 ICSE Workshop on Comparison and Versioning of Software Models, Vancouver, BC, USA, 2009, pp. 1–6. https://doi.org/10.1109/CVSM.2009.5071714

[66] Altmanninger, K., Brosch, P., Kappel, G., Langer, P., Seidl, M., Wieland, K., Wimmer, M. "Why Model Versioning Research is Needed!? An Experience Report (2009)", In: MoDSE-MCCM Workshop in MoDELS, Denver, Colorado, USA, 2009, pp. 1–12. [online] Available at: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.514.424 [Accessed: 03 April 2018]

[67] Mens, T. "A state-of-the-art survey on software merging", IEEE Transactions on Software Engineering, 28(5), pp. 449–462, 2002. https://doi.org/10.1109/TSE.2002.1000449

[68] Westfechtel, B. "A formal approach to three-way merging of EMF models", In: 1st International Workshop on Model Comparison in Practice (IWMCP 2010), Malaga, Spain, 2010, pp. 31–41. https://doi.org/10.1145/1826147.1826155

[69] Koegel, M., Herrmannsdoerfer, M., Helming, J., Li, Y. "State-based vs. Operation-based Change Tracking", In: Joint MODELS '09 Workshop on ModelDriven Software Evolution (MoDSE) and Model CoEvolution and Consistency Management (MCCM), Denver, Colorado, USA, 2009, pp. 132–141. [online] Available at: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.460.5090&rep=rep1&type=pdf [Accessed: 03 April 2018]

[70] Koegel, M., Herrmannsdoerfer, M., Li, Y., Helming, J., David, J. "Comparing State- and Operation-Based Change Tracking on Models", In: 14th IEEE International Enterprise Distributed Object Computing Conference, Vitoria, Brazil, 2010, pp. 163–172. https://doi.org/10.1109/EDOC.2010.15

[71] IBM "IBM Rational Software Architect Designer", [online] Available at: https://www.ibm.com/developerworks/downloads/r/architect/ [Accessed: 03 April 2018]

[72] Eclipse Foundation "EMF Compare", [online] Available at: https://www.eclipse.org/emf/compare/ [Accessed: 03 April 2018]

[73] Koegel, M., Helming, J., Seyboth, S. "Operation-based conflict detection and resolution", In: 2009 ICSE Workshop on Comparison and Versioning of Software Models, Vancouver, BC, USA, 2009, pp. 43–48. https://doi.org/10.1109/CVSM.2009.5071721

[74] Pietsch, P., Yazdi, H. S., Kelter, U., Kehrer, T. "Assessing the Quality of Model Differencing Engines", Softwaretechnik-Trends, 32(4), pp. 47–48, 2012. [online] Available at: http://pi.informatik.uni-siegen.de/stt/32_4/08_Sonderteil_Positionspapiere/cvsm2012_pietsch.pdf [Accessed: 03 April 2018]

[75] van Rozen, R., van der Storm, T. ""Model Differencing for Textual DSLs", In: BENEVOL 2014 - Belgian-Netherlands Evoluation Workshop, Amsterdam, Netherlands, 2014. [online] Available at: https://hal.inria.fr/hal-01110856 [Accessed: 03 April 2018]

[76] van Rozen, R., van der Storm, T. "Origin Tracking + Text Differencing = Textual Model Differencing", In: 8th International Conference on Theory and Practice of Model Transformations (ICMT 2015), L'Aquila, Italy, 2015, pp. 18–33. https://doi.org/10.1007/978-3-319-21155-8_2

[77] Olsen, G. K., Oldevik, J. "Scenarios of Traceability in Model to Text Transformations", In: 3rd European Conference on Model Driven Architecture - Foundations and Applications (ECMDA-FA 2007), Haifa, Israel, 2007, pp. 144–156. https://doi.org/10.1007/978-3-540-72901-3_11

[78] Xing, Z., Stroulia, E. "UMLDiff: An Algorithm for Object-Oriented Desing Differencing", In: 20th International Conference on Automated Software Engineering (ASE '05), Long Beach, CA, USA, 2005, pp. 54–65. https://doi.org/10.1145/1101908.1101919

[79] Kelter, U., Wehren, J., Niere, J. "A Generic Difference Algorithm for UML Models", In: Software Engineering 2005, Fachtagung des GI-Fachbereichs Softwaretechnik, Essen, Germany, 2005, pp. 105–116. [online] Available at: https://www.researchgate.net/publication/221232282_A_Generic_Difference_Algorithm_for_UML_Models [Accessed: 03 April 2018]

[80] Brosch, P., Seidl, M., Widl, M. "Semantics-Aware Versioning Challenge: Merging Sequence Diagrams along with State Machine Diagrams", Softwaretechnik-Trends, 33(2), pp. 84–86, 2013. https://doi.org/10.1007/s40568-013-0058-5

[81] Debreceni, C., Ráth, I., Varró, D., De Carlos, X., Mendialdua, X., Trujillo, S. "Automated Model Merge by Design Space Exploration", In: 19th International Conference on Fundamental Approaches to Software Engineering (FASE 2016), Eindhoven, The Netherlands, 2016, pp. 104–121. https://doi.org/10.1007/978-3-662-49665-7_7

[82] Fairmichael, F., Kiniry, J. R. "Verified Visualisation of Textual Modelling Languages", Electronic Communications of the EASST, 36, pp. 1–18, 2010. https://doi.org/10.14279/tuj.eceasst.36.446.431

[83] Nerson, J.-M. "Applying Object-Oriented Analysis and Design", Communications of the ACM, Special Issue on Analysis and Modeling in Software Development, 35(9), pp. 63–74, 1992. https://doi.org/10.1145/130994.130997

[84] Giese, H., Hildebrandt, S., Neumann, S. "Model Synchronization at Work: Keeping SysML and AUTOSAR Models Consistent", In: Engels, G., Lewerentz, C., Schäfer, W., Schürr, A., Westfechtel, B. (eds.) Graph Transformations and Model-Driven Engineering: Essays Dedicated to Manfred Nagl on the Occasion of his 65th Birthday, Springer, Berlin, Heidelberg, Germany, pp. 555–579, 2010. https://doi.org/10.1007/978-3-642-17322-6_24

[85] Giese, H., Wagner, R. "From model transformation to incremental bidirectional model synchronization", Software and Systems Modeling, 8(1), pp. 21–43, 2009. https://doi.org/10.1007/s10270-008-0089-9

[86] Giese, H., Wagner, R. "Incremental Model Synchronization with Triple Graph Grammars", In: 9th International Conference on Model Driven Engineering Languages and Systems (MODELS 2006), Genova, Italy, 2006, pp. 543–557. https://doi.org/10.1007/11880240_38

[87] Xiong, Y., Liu, D., Hu, Z., Zhao, H., Takeichi, M., Mei, H. "Towards Automatic Model Synchronization from Model Transformations", In: 22nd International Conference on Automated Software Engineering (ASE '07), Atlanta, Georgia, USA, 2007, pp. 164–173. https://doi.org/10.1145/1321631.1321657

[88] van Rest, O., Wachsmuth, G., Steel, J. R. H., Süß, J. G., Visser, E. "Robust Real-Time Synchronization between Textual and Graphical Editors", In: 6th International Conference on Theory and Practice of Model Transformations (ICMT 2013), Budapest, Hungary, 2013, pp. 92–107. https://doi.org/10.1007/978-3-642-38883-5_11

[89] Kats, L. C. L., Visser, E. "The Spoofax Language Workbench: Rules for Declarative Specification of Languages and IDEs", ACM SIGPLAN Notices - OOPSLA '10, 45(10), pp. 444–463, 2010. https://doi.org/10.1145/1932682.1869497

[90] Eclipse Foundation "Graphical Modeling Framework", [online] Available at: https://www.eclipse.org/modeling/gmp/ [Accessed: 03 April 2018]

[91] Angyal, L., Lengyel, L., Charaf, H. "A Synchronizing Technique for Syntactic Model-Code Round-Trip Engineering", In: 15th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems (ECBS 2008), Belfast, UK, 2008, pp. 463–472. https://doi.org/10.1109/ECBS.2008.33

[92] Angyal, L., Lengyel, L. "Synchronization of textual and visual representations of evolving information in the context of model-based development", In: IEEE Eurocon 2009, Saint Petersburg, Russia, 2009, pp. 420–425. https://doi.org/10.1109/EURCON.2009.5167666

[93] Kelly, S., Lyytinen, K., Rossi, M. "MetaEdit+ A fully configurable multi-user and multi-tool CASE and CAME environment", In: 8th International Conference on Advanced Information System Engineering (CAiSE 1996), Heraklion, Crete, Greece, 1996, pp. 1–21. https://doi.org/10.1007/3-540-61292-0_1

[94] Ledeczi, A., Maroti, M., Bakay, A., Karsai, G., Garrett, J., Thomason, C., Nordstrom, G., Sprinkle, J., Volgyesi, P. "The generic modeling environment", In: Workshop on Intelligent Signal Processing, Budapest, Hungary, 2001. [online] Available at: https://www.researchgate.net/publication/233757687_The_Generic_Modeling_Environment [Accessed: 03 April 2018]

[95] Voelter, M., Pech, V. "Language modularity with the MPS language workbench", In: 34th International Conference on Software Engineering (ICSE), Zurich, Switzerland, 2012, pp. 1449–1450. https://doi.org/10.1109/ICSE.2012.6227070

[96] Simonyi, C., Christerson, M., Clifford, S. "Intentional software", ACM SIGPLAN Notices, 41(10), pp. 451–464, 2006. https://doi.org/10.1145/1167515.1167511

[97] Somogyi, F., Asztalos, M. "Merging textual representations of software models - a practical approach", In: XVIII KKIO Software Engineering Conference, Wroclaw, Poland, 2016, pp. 113–128. [online] Available at: http://www.dbc.wroc.pl/dlibra/docmetadata?id=36511&from=publication [Accessed: 03 April 2018]

[98] Somogyi, F. A., Asztalos, M. "Formal Description and Verification of a Text-based Model Differencing and Merging Method", In: 6th International Conference on Model-Driven Engineering and Software Development, Madeira, Portugal, 2018, pp. 657–667. https://doi.org/10.5220/0006728006570667