Software and System Modeling manuscript No.
(will be inserted by the editor)

Blended Modeling in Commercial and Open-source Model-Driven Software Engineering Tools: A Systematic Study

Istvan David[1] · Malvina Latifaj[2] · Jakob Pietron[3] · Weixing Zhang[4,5] · Federico Ciccozzi[2] · Ivano Malavolta[6] · Alexander Raschke[3] · Jan-Philipp Steghöfer[4,5] · Regina Hebig[4,5]

Received: date / Accepted: date

Abstract Blended modeling aims to improve the user experience of modeling activities by prioritizing the seamless interaction with models through multiple notations over the consistency of the models. Inconsistency tolerance, thus, becomes an important aspect in such settings. To understand the potential of current commercial and open-source modeling tools to support blended modeling, we have designed and carried out a systematic study. We identify challenges and opportunities in the tooling aspect of blended modeling. Specifically, we investigate the user-facing and implementation-related characteristics of existing modeling tools that already support multiple types of notations and map their support for other blended aspects, such as inconsistency tolerance, and elevated user experience. For the sake of completeness, we have conducted a multivocal study, encompassing an academic review, and grey literature review. We have reviewed nearly 5,000 academic papers and nearly 1,500 entries of grey literature. We have identified 133 candidate tools, and eventually selected 26 of them to represent the current spectrum of modeling tools.

Keywords model-driven development · inconsistency tolerance · multi-view modeling · modeling tools · survey

This research was partially funded by the Rijksdienst voor Ondernemend Nederland (RVO) through the ITEA3 BUMBLE project (18006).

[1]DIRO – Université de Montréal, Canada
[2]Mälardalen University, Sweden
[3]Ulm University, Germany
[4]Chalmers University of Technology, Sweden
[5]University of Gothenburg, Sweden
[6]Vrije Universiteit Amsterdam, The Netherlands

1 Introduction

Model-driven engineering (MDE) advocates modeling the engineered system at high levels of abstraction before it gets realized. The resulting models serve crucial roles in ensuring the appropriateness (e.g., correctness, safety, optimality) of the system. To keep the cognitive flow of modeling effective and efficient, stakeholders shall be equipped with proper formalisms, notations, and supporting computer-aided mechanisms. This is especially important in the design of modern systems, as their complexity has been increasing exponentially over the past years [62]. Modeling does not remove complexity from the engineering process, but rather, it replaces the accidental complexity of complex systems with essential complexity that is easier to manage [3]. Nonetheless, as a consequence of the increasing complexity of modern systems, modeling itself is becoming more complex.

In this paper, we focus on a specific manifestation of this added complexity stemming from the need for an orchestrated ensemble of modeling notations, aiming to enable seamless interaction with models through any of the notations. Such a need has been reported in multiple academic [9] and industrial domains, e.g., automotive [41], avionics [39], cyber-physical systems [91], and product lines [71]. In such an approach, user experience may also be (temporarily) prioritized over the correctness of the described system, in an effort to enable a smooth process of expressing the stakeholder's cognitive models in terms of the modeling language. This approach is referred to as *blended modeling* [1].

1.1 What is blended modeling?

Blended modeling was first introduced by Ciccozzi et al. [15] as follows:

> *Blended modeling is the activity of interacting seamlessly with a single model (i.e., abstract syntax) through multiple notations (i.e., concrete syntaxes), allowing a certain degree of temporary inconsistencies.*

That is, blended modeling is characterized by the following three features.

Multiple notations. This is not to be confused with multiple *languages*. In our terminology, a language is composed of (i) a metamodel (abstract syntax), and (ii) a set of notations (concrete syntax). Blended modeling does not impose different metamodels.

Seamless interaction. Different notations have to be carefully integrated and orchestrated to allow for using the most appropriate notation for specific modeling tasks. This requires intuitive navigation between notations, proper change propagation between them, and in many cases, traceability.

Flexible consistency management. This aspect entails both vertical inconsistencies [83] (e.g., inconsistencies between the instance model and its metamodel); and horizontal inconsistencies (e.g., inconsistencies between two notations used to manipulate instances of the same metamodel).

1.2 What is *not* blended modeling?

Multi-view modeling is not blended modeling. As shown in Fig. 1, Multi-View Modeling (MVM) [87] and blended modeling share the trait of *multi-notation*. The main differences are, that (i) MVM further assumes *multiple languages*, while (ii) blended modeling assumes relaxed consistency rules instead. These differences stem from the different aims of the two approaches. MVM is concerned with constructing the appropriate views for stakeholders with varying backgrounds. Blended modeling focuses on the elevated UX with respect to an ensemble of notations, assuming a single underlying model. Prior work has reported challenges in relaxed consistency in multi-language settings such as MVM [66]. Blended modeling enables relaxed consistency by restricting the number of languages to one.

For example, the SCADE[1] tool suite provides the user with different languages for different purposes within the same model development environment. These languages facilitate multi-view modeling of the overall system and necessitate different abstract syntaxes. Therefore, working with SCADE cannot be considered blended modeling.

Multi-paradigm modeling is not blended modeling. In addition to assuming multiple languages, Multi-Paradigm Modeling (MPM) [58] further assumes potentially different semantics behind the languages, giving rise to *multi-formalism* (Fig. 1). This added complexity positions MPM even further from blended modeling, and vastly exacerbates consistency management, as reported in prior work [17].

For example, Matlab/Simulink is a typical combination of formalisms for system design, in which the overall system is graphically designed in Simulink[2], which follows causal block diagrams (CBD) semantics; and the low-level functions in the system are textually described in Matlab[3], which relies on matrix semantics for complex computations. While some level of navigation is provided between the two formalisms within the Matlab modeling and development environment, relaxed consistency is completely missing. Therefore, working with Matlab/Simulink cannot be considered blended modeling.

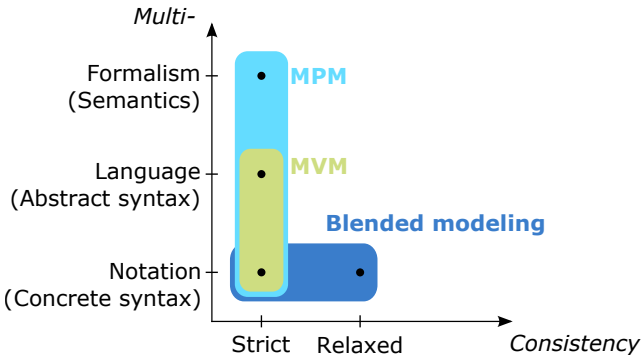


Fig. 1: Blended modeling in the context of MVM and MPM.

1.3 Motivation and aim

Blended modeling is an emerging new concept, thus, a map of current commercial and open-source tools is needed to properly position it in the research-and-development landscape.

[1] https://www.ansys.com/products/embedded-software/ansys-scade-suite

[2] https://se.mathworks.com/products/simulink.html

[3] https://www.mathworks.com/products/matlab.html

In this article, we report the design, execution, and results of our mapping study on tools that are prime candidates to support blended modeling. Our study shows that these are typically tools with multiple notations for a single underlying abstract syntax, but they lack proper inconsistency tolerance mechanisms or fail to leverage such features for an improved user experience. The aim of our study was to identify, classify, and analyze (i) the user-oriented, and (ii) the realization-oriented characteristics of these tools. To infer this information while ensuring external validity, we surveyed both the academic (peer-reviewed) literature and the grey literature [69], consisting of websites, blogs, and user manuals of engineering tools, following the guidelines for multi-vocal reviews in software engineering [33]. To be able to treat both types of literature uniformly, we made *tools* the primary units of our study, instead of papers. This is motivated by the inherent limitations of grey literature in terms of providing high-fidelity research data. Websites and end-user documentation do not aim to provide such information. We formulated a surveying protocol based on well-established guidelines and we have meticulously followed this protocol in the execution of our study. Eventually, we screened 4,975 academic papers and included 77 of them. Additionally, 1,494 grey literature entries were processed. Out of the academic papers, 68 distinct tools were extracted and complemented by 68 tools extracted from the grey literature. After removing duplicates, the set of 133 tools was reviewed according to the tool selection criteria (see Section 3) and we eventually identified 26 tools to be analyzed in detail. Although this list of tools is not exhaustive, we are reasonably confident about its representativeness of the domain of interest.

The results of this study provide a clear overview of the state of the art and practice of the domain of modeling tools closest to blended modeling. The tool characteristics reported in this paper can be particularly useful for tool providers in identifying the limitations of their tools in supporting blended modeling. Researchers of the three main dimensions of blended modeling (multi-notation, seamless integration of languages, inconsistency tolerance) could use this work to better contextualize their research, and position their work better in terms of applicability.

1.4 Structure

The rest of this paper is structured as follows. First, in Section 2, we give an overview of the background concepts of blended modeling and review the related work. In Section 3, we define the methodological framework for carrying out this study. In Section 4, we elaborate on the findings of this study, in particular on the results pertaining to the research questions of this study: the user-oriented and realization-oriented characteristics of the tools of interest. In Section 5, we provide orthogonal insights on the aggregated data. We discuss the results in Section 6, and the threats to validity in Section 7. Finally, we summarize this paper by drawing the conclusions in Section 8.

2 Background

In this section, we provide the foundational background concepts to contextualize our study. More specifically, we describe the core ingredients of blended modeling: multiple notations (Section 2.1), seamless interaction (Section 2.2), and flexibility in managing inconsistencies (Section 2.3). Additionally, we discuss the secondary literature related to our study (Section 2.4).

2.1 Multiple notations

Interacting with the (abstract) model through multiple notations (concrete syntaxes) is one of the three distinguishing features of blended modeling. A vast body of knowledge on the topic has been produced, especially in relation to multi-view modeling, and multi-paradigm modeling.

2.1.1 Multi-view modeling

Multi-view modeling (MVM) tackles the complexity of modeling heterogeneous systems by decomposing the models into multiple views, that are concerned with specific aspects of the system [87]. The ISO/IEC/IEEE 42010:2011 standard [42] defines a view as a set of concerns of specific stakeholders and viewpoints as the specification of conventions utilized to construct a view. The five mutually non-exclusive enabling mechanisms of multi-view modeling are (i) *synthetic*, where views are specified by means of different domain specific modeling languages and synthesized together; (ii) *separate*, a stricter version of synthetic, where synthesis does not take place; (iii) *projective*, where a single metamodel allows for the definition of multiple virtual views; (iv) *orthographic*, where views are orthographic projections of a single underlying model; or (v) *hybrid*, where views represent only a portion of the common metamodel [13]. MVM has been shown to be an effective approach in several complex domains, such as cyber-physical systems [62], and cloud-based software-intensive systems [16]. The principles of MVM are similar to those of blended modeling. However, its goal is

different. While MVM is oriented towards the identification of multiple views and the management of consistency between them, blended modeling focuses on enabling an elevated user experience while working with multiple notations at the same time.

2.1.2 Multi-Paradigm Modeling

Multi-paradigm modeling (MPM) advocates modeling every aspect of the system explicitly, at the most appropriate level of abstraction, and using the most appropriate formalism [58, 11]. As such, MPM facilitates the modeling of complex systems that could not be described through a single formalism and at a common level of abstraction due to the heterogeneity of the different components. It combines three research areas: (i) meta-modeling used for the specification of formalisms, (ii) multi-formalism used for the coupling of models specified in different formalisms and their transformations, and (iii) model abstraction used for the relationships among models described in different formalisms [81]. The principles of MPM are similar to those of blended modeling, as both approaches promote employing a variety of notations to model the problem at hand. However, MPM achieves this by employing a variety of separate formalisms, i.e., multiple notations with possibly different semantics. Blended modeling assumes a single abstract syntax, and therefore, single semantics. This simplification allows for greater flexibility in terms of temporarily inconsistent designs.

2.2 Seamless interaction

Usability in terms of the ability to seamlessly interact with models through multiple different notations is one of the three distinguishing features of blended modeling. In this section, we review how state-of-the-art approaches typically support seamless interaction. We focus on UML tools here since they have received significant attention from research and tool providers of the software engineering domain in the past. We also mention examples for other modeling languages where appropriate.

2.2.1 Text-based modeling with graphical visualizations

Umple [T25] is a modeling tool that supports the creation of UML models using both textual and graphical notations, where the synchronization between the two notations is automated and on the fly. However, the graphical editor does not offer full editing capabilities, and the existing editing capabilities are only available on class diagrams but not on state machines, composite structures, or feature diagrams. FXDiagram[4] is a JavaFX-based framework that can be integrated into Eclipse as well as intelliJ IDEA. It supports the creation of graph diagrams (nodes and edges) and it is typically used for graphical visualization of textual DSLs but does not provide editing functions. MetaUML[5] is a GNU GPL library for typesetting UML diagrams, using a textual notation. This notation is used for rendering read-only graphical UML diagrams. PlantUML[6] is very similar but supports also non-UML diagrams. ZenUML[7] supports sequence diagrams and flowcharts, again defined using a textual notation that is translated into read-only graphical views. The generation of the sequence diagrams is automatic, as the conversion happens on the browser. Excalibur [67] is a tool that relies on Xtext for textual specification and Sirius for graphical views of the textual specification. The model elements are defined using Messir textual DSL and the generated graphical visualization is read-only. Chart Mage[8] is a web-based tool that supports automatic and on-the-fly generation of sequence diagrams and flowcharts using a textual notation. DotUML[9] is a javascript application that supports the generation of a subset of UML diagrams (i.e., use-case, sequence, class, state, and deployment) from a textual notation. For all of the aforementioned tools, concrete syntaxes are predefined and not customizable, and the graphical notation is read-only, generated using the textual notation.

2.2.2 Mixed textual and graphical modeling

Addazi and Ciccozzi [1] present a proof-of-concept implementation for UML and UML profiles modeling using blended textual and graphical notations. The stack of technologies used includes Eclipse Modeling Framework (EMF)[10], Xtext[11], and Papyrus [T11]. Their solution includes a single underlying abstract syntax, two notations (i.e., graphical and textual), and one single persistent resource that is the UML resource. This architecture enables synchronization by means of serialization/deserialization operations across Xtext and UML models. In addition, the authors conduct an experiment to demonstrate that their solution on blended modeling

[4] https://jankoehnlein.github.io/FXDiagram
[5] https://github.com/ogheorghies/MetaUML
[6] https://plantuml.com
[7] https://www.zenuml.com
[8] http://chartmage.com/index.html
[9] https://dotuml.com
[10] https://www.eclipse.org/modeling/emf
[11] https://www.eclipse.org/Xtext

increases user performance compared to single notation modeling.

Maro et al. [52] introduce a solution that integrates graphical and textual editors for a specific UML profile-based DSL. Being that the graphical editor is already provided, this work focuses on obtaining the textual editor and switching between views (i.e., graphical and textual). To obtain the textual editor, the UML profile-based DSL is first transformed into an Ecore model using an ATL transformation, and then this Ecore model is consumed by the Xtext plugin to generate the textual editor. Switching between views is achieved by employing ATL transformations. Scheidgen [70] provides embedded textual editors for graphical editors as an add-on feature. For each selected model element that needs to be edited, the embedded textual editor creates an initial representation that can be changed by the user and using parsing operations, new edited model elements are created. However, the synchronization is on-demand as the changes in the underlying model are not carried out until they are committed by the user and the textual editor is closed.

Lazăr [51] makes use of the Eclipse modeling environment to integrate the existing UML tree-based editor with the textual editor for Alf language[12] and to create fUML[13] models. However, the synchronization is on-demand as the changes are carried out upon the occurrence of a save action by the user.

Charfi et al. [12] define a hybrid language that integrates textual and graphical notations in one concrete syntax. The contribution consists of a visual notation for the most used UML actions and an editor that supports the proposed notation. The hypothesis that the hybrid notation can perform better than the textual notation is backed by an experiment that takes into consideration the learnability of the hybrid notation, the prevented errors, and the circumstances in which the hybrid notation is a better fit than the textual notation. However, this approach is restricted to UML actions only.

Van Rest et al. [80] implement an approach for the robust synchronization of graphical editors generated with the Graphical Modeling Framework (GMF)[14] and textual editors generated with Spoofax[15]. This approach allows error recovery during synchronization and preserves the textual and graphical layout in case of errors. However, layout preservation is not supported at all times, as during cut-paste operations, the elements and their associated layouts are deleted and then recreated, therefore losing the original layout.

2.2.3 Projectional editing

Projectional editing is an approach where the abstract syntax tree (AST) is modified directly upon every editing action and bypasses the stages of the parser-based approach, where the parser must first check the correctness of the syntactic aspects, and then construct the AST based on the changes in the notation [86]. This course of action allows the definition of multiple notations (e.g., tables, diagrams, formulas) that cannot be supported by parser-based approaches, and supports multiple views of the same program, simultaneously. Moreover, a considerable amount of the ambiguities caused during the parsing process are tackled. Projectional editing is a realization of the intentional programming paradigm [73], and as such, it encourages the combination of a variety of different notations. Some of the state-of-the-art language workbenches that adopted this principle for providing domain-specific tool engineers with efficient tools [8] are JetBrains MPS[16] and MelanEE[17]. However, even though they provide a greater amount of notations, their support for textual notations is limited compared to parser-based approaches, as it is only a projection that resembles text. In particular, no possibly inconsistent intermediate states are allowed, which consequently restricts the user accustomed to classical text editors and their corresponding free editing features.

2.3 Inconsistency management

Approaches, such as multi-view modeling (MVM) and multi-paradigm modeling (MPM) advocate modeling the engineered system using the most appropriate notations, formalisms, and abstractions. This allows multiple users to be involved in the modeling of the system, and thus, introduces parallelism, which is beneficial for the overall efficiency of the engineering endeavor. Parallelism, however, gives rise to inconsistencies between the design artifacts, compromising the ultimate correctness of the system. Inconsistency has been shown to be an effective heuristic for managing the ultimate correctness of the system [17]. Techniques, such as blended modeling, make use of this assertion by focusing on the early detection of inconsistencies [19] and establishing the proper tolerance mechanisms.

[12] https://www.omg.org/spec/ALF

[13] https://www.omg.org/spec/FUML

[14] https://www.eclipse.org/modeling/gmp

[15] http://strategoxt.org/Spoofax

[16] https://www.jetbrains.com/mps

[17] http://www.melanee.org

The notion of consistency models and their various alternatives have been well-researched already in early distributed systems. Lamport [50] is the first to describe how multi-processor systems should be constructed to ensure proper execution of programs. His notion of *sequential consistency* allows a relaxation of the locking model by assuming a total order of modifications that distributed nodes are guaranteed to observe. Adve and Gharachorloo [2] describe various relaxations of the sequential consistency model, based on architectural choices on the hardware and software level. *Eventual consistency* has been suggested by Vogel et al. [85] to enable a weaker notion of consistency between distributed participants, by embracing that real consistency can never be achieved. In such settings, distributed participants are characterized by the BASE properties: basic availability, soft state, and eventual consistency. Lately, *strong eventual consistency* (SEC) has been suggested [4] to combine the liveliness guarantees of eventual consistency with the safety guarantees of strong consistency. Conflict-free replicated data types [72] are the prime examples of their applications.

Inconsistencies are a well-researched area in software engineering [74], too. Consistency between models can be categorized into two orthogonal dimensions [26]: horizontal and vertical consistency; and syntactic and semantic consistency. Horizontal consistency is concerned with models on the same level of abstraction, whereas vertical consistency is defined between models on different levels of abstraction (typically in model-metamodel contexts) [82]. The majority of inconsistency management techniques rely on syntactic concepts, e.g. synchronization by bi-directional model transformations [75], triple-graph grammars [35], and by version control systems and related mechanisms [44, 45]. However, semantic techniques have been shown to be beneficial in heterogeneous engineering settings [83]. View consistency has been researched in the context of MVM, e.g., in the Vitruvius approach [48], which provides languages for consistency preservation, and defines a model-driven development process for enacting consistency rules.

Finkelstein et al. [28] suggest that inconsistencies are organic elements of any engineering process, and instead of simply removing them from the system, one should apply proper inconsistency management techniques [29]. Such inconsistency management techniques typically entail the activities of detecting, resolving, preventing, and tolerating inconsistencies [61]. Blended modeling heavily relies on the tolerance of inconsistencies. Balzer et al. [5] suggest augmenting inconsistency instances with a state. Inconsistency rules are first deconstructed into appearance and disappearance rules spanning a temporal interval; then, tolerance rules are put in place to trigger repair actions based on temporal constructs. Easterbrook et al. [24] propose a similar technique for temporal inconsistency tolerance in the context of MVM. Inconsistency tolerance is achieved via pairs of pre- and post-conditions relying on a user-defined consistency metric. David et al. [18] introduce various patterns of inconsistency tolerance for implementing such systems.

2.4 Related secondary literature

This paper reports on the first systematic study on blended modeling. There are, however, secondary studies close to our work that are similar in topic, but differ in terms of motivation and objectives, and are generally limited to a narrower scope.

Torres et al. [77] conduct a systematic literature review with the aim to identify a list of available tools to support model management and provide a categorization of these tools into (i) tools that can provide consistency checking on models of different domains, (ii) tools that can provide consistency checking on models of the same domain, and (iii) tools that do not provide any consistency checking. Furthermore, the authors identify the inconsistency types, strategies to keep the consistency between models of different domains, and the challenges to manage models of different domains. The information retrieved from the primary studies is also complemented with additional data sources (e.g., the official website of the tool). Our study focuses on a broader scope, especially multi-notation and seamless interaction. Torres et al. observe that 35% of their analyzed tools do not provide any consistency checking features, whereas in our study we observe that 64% of the analyzed tools do not support models inconsistencies. Moreover, Torres et al. identify different strategies that have been used to keep models consistent, e.g., by using standard file formats for the models, explicitly modeling dependencies among model elements, mapping model elements to a shared ontology, etc. Our study complements such results by highlighting which inconsistency management strategies involve a manual effort (like keeping a dependency matrix always up to date), a semi-automated procedure (e.g., by specifying a priori consistency constraints and checking them during development), or a completely automated one (e.g., via the automated application of inconsistency resolution procedures).

Iung et al. [43] conduct a systematic mapping study with the aim to identify tools, language workbenches, or frameworks for DSL development. The authors identify 59 tools and they use the feature model proposed by Erdweg et al. [27] for their comparison. The study

focuses on the technologies/tools used for DSL development, their license types, the application domains, and the features of the DSL creation process that these tools support. 48 tools support only one notation (graphical or textual), seven tools support two notations (graphical and textual), two tools support three notations, and two tools support four notations. Our study focuses on a broader scope, by extending the set of features on which the comparison is based with features such as synchronization mechanisms, collaborative features, or conformance relaxation. We also contextualize our work on a broader timeline, while the authors focus on the period between 2012–2019. In line with the results of our study, Iung et al. observed that the notations that were more frequently used in combination are *textual* and *graphical*, with the tabular one complementing them. In [43] two language workbenches are identified as particularly relevant for blended modeling: (i) GEMOC Studio, which provides real-time bidirectional synchronization in their generated editors, and (ii) the Whole Platform, which allows language engineers to choose among four different types of notation (i.e., textual, graphical, tabular, and symbolic), and to visualize the different translations among them at the model level.

Franzago et al. [30] and David et al. [20] map the state-of-the-practice of collaborative model-based software engineering. The authors identify and classify collaborative MDSE approaches based on the different categories such as characteristics of the collaborative model editing environments, model versioning mechanisms, model repositories, support for communication and decision making, and more. Additionally, the authors identify limitations and challenges with respect to the state of the art in collaborative MDSE approaches. Regarding model management, they provide a taxonomy for the management support of collaborative MDSE approaches, collaboration support, and communication support. This study covers some of the aspects that we cover in our systematic mapping study (e.g., conflict detection). However, while this study is mostly focused on the characteristics of the collaborative approaches, we aim toward a classification of tools based on a broader set of features such as synchronization mechanisms and their generation, or conformance relaxation. The results of Franzago et al. and David et al. for collaborative modeling that are confirmed in this study are about: (i) the types of notations, with graphical as the most supported one, followed by textual, (ii) the prevalence of custom/other modeling platforms with respect to Eclipse EMF, (iii) the growth of web-based approaches, (iv) the growth of preventive conflict management, and (v) the prevalence of mechanisms for (semi-)automatically resolving conflicts. We anticipate that 15 out of the 26 tools analyzed in this study support collaborative modeling, with the majority of tools providing off-line collaboration (i.e., a la Git), rather than real-time collaboration (i.e., a la Google Docs); this result is different for academia where, according to Franzago et al. and David et al., researchers focus primarily on real-time collaboration. Another difference with respect to the state of the art in collaborative modeling is that blended modeling tools are primarily parser-based, whereas collaborative modeling approaches tend to be equally distributed between parser-based and projectional approaches. Interestingly, while researchers are recently investigating more on eventual consistency for collaborative modeling [20], in our study we observe that blended modeling tools provide limited support for consistency tolerance that would allow deviations between different notations describing the same model.

Granada et al. [37] map model-based language workbenches that can be used to generate editors for visual DSLs and point out their features and functionalities. The authors identify eight language workbenches for the generation of editors for visual DSL. The features taken into consideration for their analysis are the following: scope, framework, the distinction between abstract and concrete syntax, abstract syntax, concrete syntax, editing capabilities, use of models, automation, usability, and methodological basis. The conclusions point out that the most complete commercial language workbenches are MetaEdit+[18] and ObeoDesigner[19], while the most complete open-source ones are Eugenia[20], GMF[21], Graphiti[22], and Sirius[23]. Our study differs in scope, as we focus on tools that provide multiple notations, not only on tools that can be used to develop editors for a single visual DSL. Indeed, none of the tools identified by Granada et al. support the definition of more than one (visual) concrete syntax for the same abstract syntax; this means that language engineers willing to develop blended modeling environments should either use a dedicated language workbench for blended modeling or suitably combine the languages produced by two or more of the language workbenches mentioned above.

Do Nascimento et al. [23] perform a large-scale systematic mapping study on DSLs and their related tools. The tools are categorized into (i) tools for using DSLs, (ii) tools for creating DSLs, and (iii) language work-

[18] https://www.metacase.com/products.html
[19] https://www.obeodesigner.com/en
[20] https://www.eclipse.org/epsilon/doc/eugenia
[21] https://www.eclipse.org/modeling/gmp
[22] https://www.eclipse.org/graphiti
[23] https://www.eclipse.org/sirius

benches. Our study differs from this work, as we focus on DSLs tool comparison, while the authors provide a brief categorization of DSL tools, and do not go into the details of conducting a comparison of the technical features. It is interesting to note that Do Nascimento et al. observed that tool support for a single DSL is well-studied in the literature, but at that time (2012) there was little knowledge about how to support multiple DSLs and notations in a single modeling environment. They claim that supporting multiple DSLs and multiple notations is fundamental when describing large-scale industrial systems and that methods and tool support are needed for the success of multi-DSL development. Based on the results of our study and the ones on multi-notation modeling (see Section 2.1, we can confirm that in the last years, the MDE scientific community actively worked and contributed to filling this research gap.

There are additional studies related to our research that are not systematic in nature, but their takeaways are still relevant. Negm et al. [59] compare 14 language workbenches based on (i) structure (grammar-driven or model-driven), (ii) editor (parser-based or projectional), (iii) language notations (textual, tabular, symbols, or graphical), (iv) semantics (translational or interpretive), and (v) composability language aspects. However, this study is limited to language workbenches and does not cover aspects such as synchronization mechanisms and their generation, or collaborative features. Some of the results obtained by Negm et al. are relevant for blended modeling as well. Firstly, out of nine analyzed parser-based language workbenches, only one (i.e., Ensõ) supports both textual and graphical concrete syntaxes; this capability is achieved by having a bidirectional mapping between tokens in the textual representation of the model and elements in the object graph. Moreover, all four considered projection-based language workbenches support multiple concrete syntaxes, with the *Whole* platform and MPS supporting four different syntaxes: textual, graphical, tabular, and symbolic. The main advantage of projection-based workbenches is that they can rely on a shared common representation of all modeling elements (e.g., the AST in MPS), whereas parser-based workbenches have a dedicated parser for each concrete syntax. One of the claimed advantages of parser-based language workbenches (especially the textual ones) is the flexibility with respect to the models' conformance; the textual representation of parser-based models can still be opened and inspected, whereas projectional editors work directly on the abstract representation of the model. Similarly, according to Negm et al., textual parser-based workbenches avoid tool lock-in since the modeler is not limited to using any specific editor and can be easily integrated with other tools.

Erdweg et al. [27] conduct a comparison study of 10 language workbenches participating in the 2013 edition of the Language Workbench Challenge (LWC). The comparison of the language workbenches is based on a feature model that includes: notation, semantics, editor support, validation, testing, and composability, where some of them support multiple notations (fully or partially). The conclusions state that no language workbench realizes all features. However, this study is limited to the language workbenches presented in LWC'13. For what concerns blended modeling, the results obtained by Erdweg et al. are in line with the ones reported by Negm et al. [59], where projectional language workbenches are better supporting the combination of different concrete syntaxes, with Enzõ and MPS again as the ones supporting all types of concrete syntaxes. Erdweg et al. also highlight the need for integrating "different notational styles", which is at the core of blended modeling.

Merkle [55] conduct a comparison study of textual language workbenches categorizing them into pure text-based and projectional-based with a textual projection. The language workbenches compared in this study are Xtext[24], TEF[25], EMFText[26], and MPS[27]. The language workbenches are compared based on workflow, abstract/concrete syntax, and editor. However, this study is limited to textual language workbenches, while our focus is on tools that provide multiple notations. In the study by Merkle, the only language workbench supporting a combination of concrete syntaxes is TEF (Textual Editing Framework[28]), an Eclipse-based language workbench focusing primarily on textual editors, but with the possibility of embedding them into other editors supporting other concrete syntaxes [70]. Internally, TEF follows the *background parsing* strategy for the textual concrete syntax, where textual models are always represented and edited as plain text, and their parsing is demanded by a background process. TEF also provides some basic form of blending, where modelers can bring up a textual editor from either a graphical or a tree-based editor (e.g., by opening a small overlay window); however, TEF-based modeling tools cannot be considered as blended since model updates the embedded textual editor is not seamlessly integrated into

[24] https://www.eclipse.org/Xtext

[25] https://www2.informatik.hu-berlin.de/sam/meta-tools/tef/tool.html

[26] https://github.com/DevBoost/EMFText

[27] https://www.jetbrains.com/mps

[28] http://www2.informatik.hu-berlin.de/sam/meta-tools/tef/tool.html

its host editor, and model updates are propagated on-demand to the host editor only when the modeler closes the textual editor.

3 Study design

The goal of this study is to characterize the state of the art and the state of the practice of modeling tools in relation to blended modeling. More specifically, we formulate such high-level goal by using the Goal-Question-Metric perspectives [7], shown in Table 1.

Purpose	Identify, classify, and analyze
Issue	the user-oriented and implementation-oriented characteristics of
Object	existing modeling tools
Context	in relation to the principles of blended modeling,
Viewpoint	from a researcher's and practitioner's point of view.

Table 1: Goal of this study.

3.1 Process

This research was carried out by following the process shown in Figure 2. Our process can be divided into three main phases, all well-established in systematic secondary studies [46, 90, 47, 64]: planning, conducting and documenting. In the following, we present the three phases of the process.

3.1.1 Planning

This phase aims at defining the plan for carrying out all the activities of this study. More specifically, we first identified *related secondary studies*, i.e., surveys and literature reviews with a scope similar to the current review's scope (Section 2.4). Subsequently, we formulated the *research questions* (Section 3.2), and compiled the *research protocol*.

The research protocol is a document reporting the methodological details of this study. Specifically, the research protocol contains a detailed description of all the steps we followed in the subsequent *Conducting* and *Documenting* phases. To mitigate potential threats to validity and any bias, the research protocol was defined *prior to* conducting the study, and it was reviewed by two experts. The experts were asked to provide feedback on the protocol, particularly on possible unidentified threats to validity, problems in the overall construction of the review, and the appropriateness of the proposed research protocol and final reports for the aim of this study. Both experts are well-established professors of Computer Science, with substantial experience in empirical research.

3.1.2 Conducting

In this phase, the mapping study is carried out according to the research protocol. More specifically, we carry out the following activities.

Reference set definition. The goal of this activity is to identify the modeling tools that could be part of the final set of modeling tools. This set will serve as a guideline for the subsequent steps of the study design, especially formulating the inclusion and exclusion criteria. The inclusion and exclusion criteria will be tested against this set, and thus, the reference set is subject to change until the criteria are not final. We identify the initial reference set based on (i) the modeling tools mentioned in related secondary studies (Section 2.4); (ii) the authors' experiences with tools partially supporting blended modeling (e.g., [15, 1]); (iii) searches in generic web search engines; and (iv) knowledge garnered from existing networks of experts, e.g., by accessing forums and mailing lists (e.g., the Eclipse EMF community forum[29]). The results of the subsequent *Search and selection* activity are eventually compared to the reference set for validation purposes. The eventual reference set is composed of the following tools: MagicDraw [T13], Eclipse Papyrus [T11], MetaEdit+[30], Umple [T25], and the Open Source AADL Tool Environment (OSATE) [T17].

Search and selection. (Section 3.3) The goal of this activity is to identify as many (possibly blended) modeling tools as possible. Two parallel activities are carried out: the *Academic literature review*, and the *Grey literature review*. In both search activities, we perform a combination of automated search, manual search, and backward-forward snowballing [89]. These activities yield two types of artifacts: (i) *Academic studies* (e.g., articles published in scientific journals, and proceedings of scientific conferences) and (ii) *Non-academic entries* (e.g., blog posts, technical reports). Because the subject of this study are the *tools* these artifacts describe, both types of artifacts are screened for a specific *Tool* in the *Tools identification* activity. Here, we manually analyze all academic studies and non-academic entries and

[29] https://www.eclipse.org/forums/index.php?t=thread&frm_id=108

[30] https://www.metacase.com/products.html

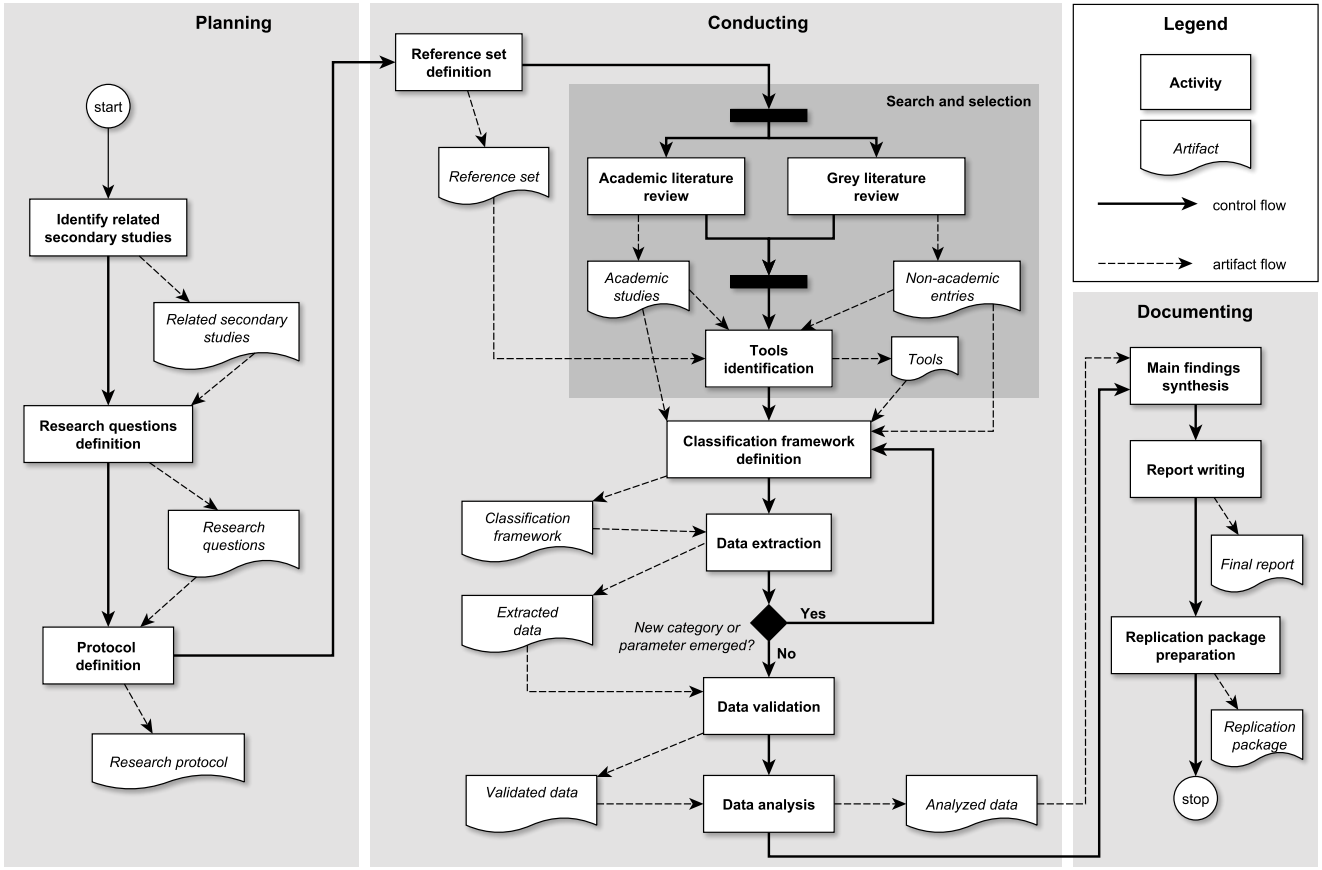


Fig. 2: Overview of the whole review process

identify every modeling tool mentioned in their contents. Moreover, in this activity, we keep track of pointers and links referring to the relevant documentation about each tool (e.g., its official documentation, its wiki-based knowledge base, etc.).

Classification framework definition. (Section 3.4) The goal of this activity is to define the set of categories and their possible values to classify the identified modeling tools.

Data extraction. (Section 3.5) The goal of this activity is to collect relevant information about each modeling tool. In this activity, multiple researchers collaboratively (i) read the full text of the relevant documentation of each modeling tool, and (ii) populate the data extraction form with the collected data. Upon the emergence of a new category or new possible value in the domain of previously defined categories, the classification framework can be dynamically adapted. In such cases, the previously extracted data entries are updated in accordance with the new framework.

Data validation. (Section 3.6) To ensure the validity of the extracted data, the tool vendors and knowledgeable experts are contacted to review the data extracted in the previous step.

Data analysis. (Section 3.7) The goal of this activity is to analyze the extracted data in accordance with the research questions. The activity involves both quantitative and qualitative analyses.

3.1.3 Documenting

The main activities performed in this phase are: (i) a thorough elaboration on the data analyzed in the previous phase with the aim of discovering the main findings of the study; (ii) reporting the possible threats to validity, especially the ones identified during the definition of the review protocol; and (iii) producing the final report. The final report is evaluated by external reviewers and forms the basis of this article.

The complete *replication package* is available online[31] to allow independent researchers to replicate and verify our study, and to reuse our data for other purposes. The replication package includes the research protocol, the list of all academic and non-academic en-

[31] `https://zenodo.org/record/6402743`

tries considered in the search and selection phase, the complete list of all identified tools, raw data, the scripts for data analysis, and the details on technical requirements.

3.2 Research questions

The research questions of this study are reported below.

RQ1. *What are the **user-oriented characteristics** of modeling tools most suitable for supporting blended modeling?*
Modeling tools are designed and developed to be adopted by specific users, application domains, and usage scenarios.
By answering this research question, we aim to identify the external characteristics of modeling tools, pertaining to their adoption and usage [15]. Typical examples include: supported (types of) notations, human-computer interfaces, application domains, and addressed user groups.
Practitioners can benefit from the answer to this research question by understanding how specific state-of-the-art tools address their problems, what are their limitations in terms of blended modeling, and how they can be improved.

RQ2. *What are the **realization-oriented characteristics** of modeling tools most suitable for supporting blended modeling?*
With the advent of model-based approaches and domain-specific modeling, in particular, several modeling tools are being developed to support certain levels of blending, formalisms, and semantics. Moreover, until the recent spread of mainstream language workbenches (e.g., Xtext[32], Sirius[33], MPS[34], etc.), the development of such modeling tools had been relatively ad-hoc.
By answering this research question, we aim to identify the internal characteristics of modeling tools, and that, in terms of (i) their features and (ii) the techniques employed to implement those features. Typical examples include: implementation platforms, consistency mechanisms, change propagation, traceability, and the linguistic level of model-to-model correspondence are investigated.
Researchers can benefit from the answer to this research question by understanding the state of the practice on the techniques of blended modeling tools, including the gaps to fill.

[32] https://www.eclipse.org/Xtext
[33] https://www.eclipse.org/sirius
[34] https://www.jetbrains.com/mps

The identified research questions drive the whole study, with a special influence on (i) the search and selection, (ii) data extraction (including the definition of the classification framework), and (iii) data and main findings synthesis.

3.3 Search and selection

The goal of the search and selection phase is to retrieve a representative set of modeling tools supporting multiple modeling notations, as demanded by the principles of blended modeling. First, we perform a *systematic review* of both the academic (i.e., scientific articles published at peer-reviewed academic venues) and grey literature (i.e., websites, online blogs, etc.), and discuss the results in Section 3.3.1. The output of these two activities (i.e., academic studies and non-academic entries) is then further analyzed in order to identify the modeling tools either considered, mentioned, or discussed in them (Section 3.3.2).

3.3.1 Systematic Reviews

We follow the same overall process when reviewing both the academic and grey literature. In this phase, it is fundamental to achieve a good trade-off between the coverage of existing results on the considered topic and having a manageable number of studies to be analyzed [46, 33]. To achieve the above-mentioned trade-off, our search and selection process has been designed as a multi-stage process; this gives us full control over the number and characteristics of the entries being either selected or excluded during the various stages. In the following, we present each step of our systematic review process. In the remainder of this report, we refer to both academic studies and non-academic entries as *primary studies*, unless specifically noted otherwise. The systematic review is divided into three subsequent and complementary steps of (i) automatic search, (ii) application of selection criteria, and (iii) snowballing.

Automatic search. In this step, we automatically inspect all the results returned from a query execution on (i) Google Scholar for academic studies and (ii) the Google Search engine for grey literature. The automatic searches for both academic and non-academic literature are executed in November 2020.

For the *academic literature*, we use *Google Scholar*. We use Google Scholar as the data source for the following main reasons: (i) it is one of the largest and most complete databases and indexing systems for scientific literature; (ii) as reported in [89], the adoption of this data source has proved to be a sound choice to

identify the initial set of literature studies for the snowballing process (Section 3.3.1), producing a reasonable number of false positives, but no false negatives (thus, no information is lost); (iii) the query results can be automatically processed via already existing tools.

Below we report the search string used in this study. In order to cover as many potentially relevant studies as possible, we defined the search string so that it includes academic studies on blended modeling. The search string can be divided into three main components: the first component captures the model-driven paradigm, the second one captures the focus on multiple entities (e.g., multiple notations) and blending, and the third one is used for ensuring that our results focus on software aspects. To keep the results of this initial search as focused as possible, the query has been applied to the title of the targeted studies.

```
("modeling" OR "modelling" OR "model based"
  OR "model driven")
    AND
("multi*" OR "blended")
    AND
("notation*" OR "syntax*" OR "editor"
  OR "tool" OR "software")
```

The search string has been tested by executing pilot searches on Google Scholar. At the time of writing, Google Scholar produced a total of 280 hits when searching with the reported search string.

For the *grey literature*, we target the regular *Google Search Engine*. The search engine is selected in accordance with the recommendations for including grey literature in software engineering multi-vocal reviews [33]. The search string used for the academic literature yields mostly academic results even in a general web search. We have, therefore, adapted our search strategy to find non-academic sources. In particular, we identified a number of relevant hits through manual searches early on. These manual hits could be classified as either *lists* (e.g., Wikipedia's "List of Unified Modeling Language tools") or *tool-specific pages* (e.g., tool vendor pages or blog posts about how specific tools are used).

We experimented with several search strings to ensure that we find all relevant hits. In particular, we tried to combine different modeling languages and diagram types into one large all-encompassing search string to simplify our search and make it easier to extract results. However, on prototyping this approach, we realized that the `OR` clauses that we used did not have the desired effect and we did not find the tools we expected, and in particular, not the lists that we expected. In comparison, a search string such as `(MARTE) AND (tool OR editor OR notation OR modelling)` yields 162 results on Google, whereas our combined search string that included MARTE[35] and many other languages only yielded 150 results.

Therefore, we decided to carry out an independent search for popular modeling languages. We ran the different searches independently and merged the results later on. We selected the relevant modeling languages using a mixture of expert knowledge, browsing the web pages of well-known modeling tools from the reference set and beyond (e.g., Eclipse Capella[36] and Enterprise Architect[37]), using lists such as Wikipedia's page on "Modeling Languages". We narrowed down the resulting list of around 40 potential modeling languages by searching for `(Language Name) AND (tool OR editor OR notation OR modelling)` in Google, and analyzing the first ten non-academic hits (i.e., search results that were not academic papers). Since the search term explicitly contains the terms "tool" and "editor", we expected that the Google search engine would include such a tool within the first ten non-academic hits if it exists, and has any practical relevance. Experiments where we checked later result pages for selected searches confirmed this expectation. We thus only included modeling languages for which Google does report a link to a modeling tool. Otherwise, we disregarded it.

To address the large number of hits we would get this way, we limited the search results for each included search string to the first 50 unique results (if less than 50 hits are reported, we collect all of them), which is based on the suggestion from Garousi et al. [33]. The eventual result set included 1,494 hits, typically containing blog posts, user manuals, websites, technical reports, white papers, academic articles, etc.

Application of Selection criteria. In this step, the identified potentially relevant entries undergo rigorous filtering based on the application of a set of selection criteria. Following the guidelines for systematic literature review for software engineering [46], we define the set of inclusion and exclusion criteria *a priori*, in order to reduce the likelihood of bias. The potentially relevant entries are rigorously examined by adopting multiple selection rounds in an adaptive reading depth fashion [63]. Specifically, in the first round, the title of the entry is examined. This first step enables us to discard all those papers or web pages that clearly do not fall within the scope of this study. In the second exclusion round, the introduction and conclusion sections are inspected (if present). Finally, the entries are further inspected by considering their full text, in order

[35] `https://www.omg.org/omgmarte`

[36] `https://www.eclipse.org/capella`

[37] `https://sparxsystems.com/products/ea`

to ensure that only the ones relevant to answering the research questions are selected. While processing the full text of a paper/web page, we also keep track of all the mentioned modeling tools and consider them in the tools' identification phase (Section 3.3.2).

In the following, we detail the set of inclusion and exclusion criteria that guide the selection of the academic and non-academic entries for our systematic review.[38] A potentially relevant entry is selected if it (i) satisfies *all* inclusion criteria and (ii) does not satisfy *any* of the exclusion criteria. The selection criteria are divided into three categories, namely: *generic* (i.e., they apply for both academic and non-academic studies), *academic-specific*, and *grey-specific*. The decision of adopting three categories of criteria originates from the different nature of the sources we considered (i.e., Google Scholar and the Google Search Engine). By defining three different sets, it is possible to design selection criteria specifically tailored to the specific characteristics of academic and non-academic entries, and hence, improve the overall quality of the selection process.

Generic inclusion criteria:

GEN-I1) Entries on modeling tools, i.e., where models are used as first-class entities and used as a substantial abstraction from the problem domain (e.g., OSATE [T17] for modeling hardware/software systems according to the AADL modeling language).

GEN-I2) Entries discussing at least two different notations (possibly for the same abstract syntax). The notations can be of the same type (e.g., both textual).

Generic exclusion criteria:

GEN-E1) Entries on non-modeling tools. For example, articles on IDEs, programming tools, drawing tools, etc.

GEN-E2) Entries that are not in English.

GEN-E3) Duplicates of already included entries.

GEN-E4) Entries that are not available, and hence not analyzable (e.g., the full text of a scientific article is not accessible or the link to a web page is broken).

Exclusion criteria specific to academic sources:

A-E1) Studies in the form of full proceedings and books since they are too broad for being thoroughly analyzed in this phase of the study.

A-E2) Studies that have not been peer-reviewed, as peer-reviewing is the *de facto* standard of quality assurance for scientific literature.

[38] The identifiers used in this section are consistent with those used in the replication package to enable better traceability.

Exclusion criteria specific to grey literature:

G-E1) Web pages reporting exclusively the basic principles of modeling techniques, without mentioning any modeling tool.

G-E2) Web pages reporting exclusively abstract best practices while applying modeling techniques.

G-E3) Web pages reporting an implementation without a discussion of its benefits and/or drawbacks.

G-E4) Academic literature, since such type of studies is considered by a different process in our protocol.

G-E5) Videos, podcasts, and webinars since they are too time-consuming to be considered for this phase of the study.

Snowballing. In this step, we complement the preliminary set of academic studies by applying the snowballing procedure [89]. To mitigate a potential bias with respect to the construct validity of the study, backward and forward snowballing is used to complement the automatic search of the academic literature [38]. In particular, this process is carried out by considering the scientific publications selected in the initial automatic search, and subsequently selecting relevant studies among those cited by one of the initially selected ones (backward snowballing). Then, we also perform forward snowballing, i.e., selecting relevant studies among those citing one of the initially selected academic studies [89]. In this context, the *Google Scholar*[39] bibliographic database is adopted to retrieve the studies citing the ones selected through the initial search phase. The final decision about the inclusion of the newly considered publications in the study is based on the application of the selection criteria presented in Section 3.3.

3.3.2 Tool identification

In the tool identification activity, each primary study is manually analyzed and the mentioned modeling tools are identified. This is achieved by investigating the full text of each primary study, and collecting every modeling tool mentioned in it, independently of whether it is blended or not. Then, the set of identified modeling tools is filtered for duplicates, which are subsequently merged, regardless of whether the tool originates from an academic or a non-academic source. After the merge, we obtained a total of 133 modeling tools. For each tool, we have collected the following information: (i) name, (ii) link/reference to official documentation, (ii) organization(s) implementing, maintaining, and supporting

[39] `https://scholar.google.com`

the tool, and (iii) tracing information towards all primary studies mentioning the tool.

In order to ensure that the identified tools support us in answering the research questions of this study, we further filter the list of all modeling tools according to a set of selection criteria. Below we report the inclusion and exclusion criteria.

TI1) The tool allows its users to edit the same model in multiple notations. The user can switch between these notations easily and without an extra processing step (i.e., the tool supports some level of blended modeling). The tool allows a certain degree of temporary inconsistencies. Notations like an overview tree for navigation purposes or any textual representation used for file persistency purposes only are not considered (e.g., XMI).
TI2) The tool is publicly available (either as an open-source or commercial product).
TI3) The documentation of the tool is publicly available.

TE1) The tool is a language workbench. (Our study focuses on modeling tools themselves.)
TE2) The tool is not available for download as a binary that can be run on current operating systems from an official website or an affiliated platform supporting it (e.g., a GitHub repository).
TE3) The documentation of the tool is not in English.

A potentially relevant modeling tool is included if it satisfies *all* inclusion criteria (TI1-TI3), and discarded if it satisfies *any* exclusion criterion (TE1-TE2).

To minimize bias, this activity is performed by five researchers and organized as follows. First, two researchers are randomly assigned to each of the potentially relevant tools. Then, the researchers independently apply the tool selection criteria to their assigned tools; each researcher could mark a tool as `included`, `excluded`, `maybe`. For the 12 of 133 tools where at least one researcher indicates an uncertainty (`maybe`), the conflicts are resolved with the intervention of a randomly–assigned third researcher and, when needed, discuss plenary among all researchers involved in this study.

After the final set of modeling tools has been established, we check whether each tool in the reference set is also included in this final set of tools. If all tools in the reference set are indeed included in the final set of tools, we continue with the subsequent phases of the protocol (i.e., data extraction). Otherwise, a dedicated meeting is set up, and a refinement of the systematic review process is designed and conducted again.
Eventually, the final list of modeling tools contains all tools of the reference set.

Figure 3 shows the different steps performed in the search and selection phase. Out of the 467 papers in the initial scientific search, 44 papers were included in the snowballing process. The snowballing was performed four times before no more new papers were included. During this process, a total of 2,134 cited and 3,623 referenced papers were reviewed. In summary, 68 distinct tools were extracted from the included papers. For the grey literature part, 30 relevant languages were identified as described above, for which the different search terms yielded 1,494 distinct websites. After applying the selection criteria, 68 tools were included in the tools set. Merging the academic and grey literature parts resulted in 133 distinct tools, of which 30 tools were selected according to the tool selection criteria. Two tools had to be excluded during the data extraction process due to lack of availability or semantically out of scope (see Section 3.4). Eventually, 26 modeling tools were sampled, shown in the Referred Tools section at the end of this paper.

3.4 Classification framework definition

Table 2 shows the classification framework of this study. The classification framework is composed of three distinct facets; the first facet is about generic characteristics of modeling tools (e.g., release dates, vendor, main motivation for blending notations); the second and third facets directly address research questions RQ1 and RQ2.

We partially reuse the results of previous work [15] related to blended modeling for defining the initial version of the classification framework. Then, as suggested in [90], the customization of the classification framework is performed as follows: (i) firstly we select a random sample of 10 modeling tools, (ii) then two researchers independently extract the data from the 10 modeling tools by using the initial version of the classification framework, (iii) the two researchers then discuss the results of the data extraction with a third researcher, with a special focus on too generic/abstract parameters, parameters which did not fully fit with the characteristics of the tools, parameters with redundant values, and recurrent missing concepts, (iv) the classification framework is customized according to the discussion, and lastly (v) the final version of the classification framework is applied to all remaining modeling tools. It is important to note that when analyzing the remaining 26 tools, the classification framework can still be enriched/updated based on the characteristics of the currently analyzed tool. The details about how we extracted data for each modeling tool are provided in the next section.

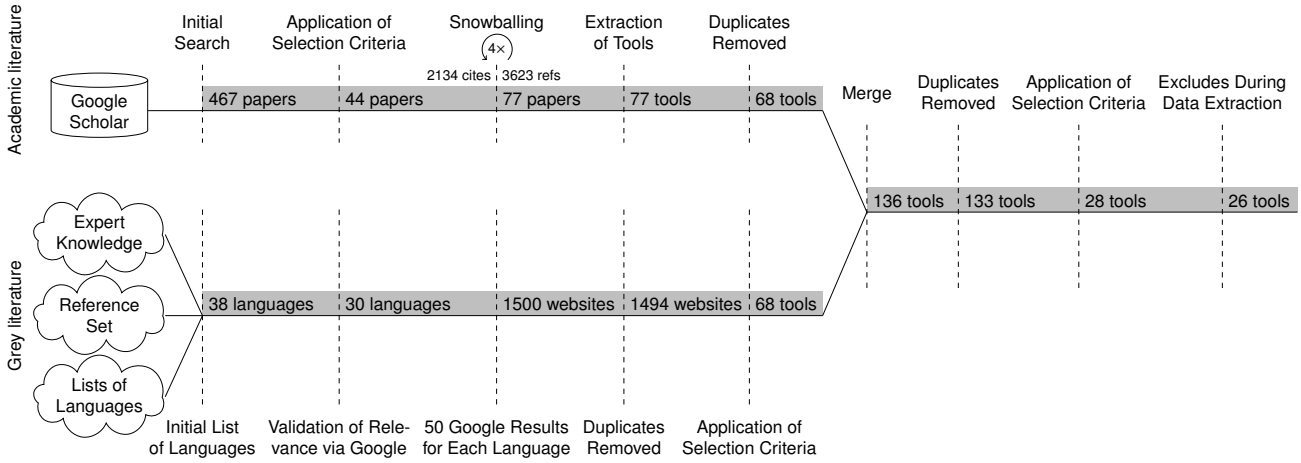


Fig. 3: Overview of the conducted search and selection steps

3.5 Data extraction

The main goal of this activity is to extract relevant data about each modeling tool for answering the research questions. The inputs to this activity are: (i) the set of 28 modeling tools, out of which 26 remained after excluding two additional tools during this phase; and (ii) the textual contents of the academic studies and non-academic entries referring to the tools, and the tools' official documentation (when publicly available). Moreover, when we are not able to collect all relevant data for some specific aspects of a tool (e.g., the internal consistency mechanisms of a proprietary tool) we perform a series of ad-hoc Web searches and contact the support team of the tool for collecting the missing data. For the sake of external verifiability, full tracing information is kept between the extracted data and the considered data sources and it is included in the replication package of the study.[31]

To carry out a rigorous data extraction process, and to ease the control and the subsequent analysis of the extracted data, a predefined data extraction form is designed prior to the data extraction process. The structure of the data extraction form is based on the various categories of the classification framework.

3.6 Data validation

To ensure the validity of the extracted data, the tool vendors are contacted and the data and the explanation of the reference framework are made available to them. If a tool does not have a clearly identified vendor, we identify knowledgeable experts who published scientific papers related to the tool. The vendors and experts are asked to identify any invalid data related to their tool. The contact is initiated via email with the vendors and experts having an option to ask and discuss the details with our research team. The majority of interactions happened in email. Some vendors and experts preferred a live discussion during a video call, which we also accommodated.

Eventually, we have contacted vendors and experts of 24 tools. The authors of this paper have developed or extensively contributed to the remaining 2 tools, and validated them internally. The validation phase ran for three weeks, between February 28 and March 22, 2022. 69% of tool vendors or experts replied either with minor change suggestions or with the approval of the extracted data. Based on their responses, 3.8% of the data (20 of 520 records) has been updated. The most changes, five, were observed in the model-level flexibility category.

3.7 Data analysis

The data analysis activity involves collating and summarizing the data, aiming at understanding, analyzing, and classifying the state of the art of modeling tools [47, § 6.5]. The data synthesis is divided into two main phases: vertical analysis and horizontal analysis. In both cases, we perform a combination of content analysis [31] (mainly for categorizing and coding tools under broad thematic categories) and narrative synthesis [68] (mainly for detailed explanation and interpretation of the findings coming from the content analysis). When performing *vertical analysis*, we analyze the extracted data to find trends and collect information about *each category* of the classification framework. When performing *horizontal analysis*, we analyze the extracted data to explore possible relations *across different categories* of the classification framework.

Table 2: Categories of the classification framework, and their domain.

Category	Definition	Type/Domain
GENERIC		
META		
Tool ID	The internally used ID of the tool.	"T"+[numeric]
Name	The name of the tool.	Free text
Analyzed release	The version of the release the analysis was carried out on.	Free text
TOOL		
First release	Date of the first available release.	Date
Latest release	Date of the latest available release.	Date
Motivation	The self-declared motivation of the tool.	Free text
Open-source	Whether the tool's sources are available openly.	{Yes, No}
Web-based	Whether the tool is web-based.	{Yes, No}
Collaboration	The degree and type of support for collaboration.	{No, Asynchronous, Synchronous}
RQ1: USER-ORIENTED CHARACTERISTICS		
NOTATIONS		
Notation types	Types of notations supported by the tool.	{Textual, Graphical, Tabular, Tree-based, Mixed textual-graphical}
Notation instances (number of)	Sum number of instances of notation types.	Numeric
Embedded notations	Whether there are notations that are embedded into each other.	{Yes, No}
Overlap	The degree of overlap between notations.	{None, Partial, Complete}
VISUALIZATION AND NAVIGATION		
Visualize multiple notations	The ability to visualize more than one notations.	{Yes, No}
Synchronous navigation	Whether the tool supports a synchronous navigation of multiple visualized notations.	{Yes, No}
Navigation among notations	The dynamics of navigation between different notations.	{Immediate, Complex}
FLEXIBILITY		
Flexibility - models	Whether the tool supports temporary inconsistency at the level of the instance models.	{Yes, No}
Flexibility - language	Whether the tool supports temporary inconsistency at the level of the language.	{Yes, No}
Flexibility - persistence	Whether the tools can persist inconsistent models.	{Yes, No}
RQ2: REALIZATION-ORIENTED CHARACTERISTICS		
MAPPING AND PLATFORMS		
Mapping	The way concrete and abstract syntax are mapped.	{Parser-based, Projectional}
Platform	The platform the tool is built on.	{Eclipse, Other}
CHANGE PROPAGATION AND TRACEABILITY		
Change propagation	The dynamics of propagating changes across notations.	{Sequential, Concurrent}
Traceability	Whether the tool supports explicit traceability between notations.	{Yes, No}
INCONSISTENCY MANAGEMENT		
Inconsistency visualization	The degree and way the tool visualizes inconsistencies.	{No, Internal, External}
Inconsistency management type	The way the tool manages inconsistencies.	{On-the-fly, On-demand, Preventive}
Inconsistency management automation	The degree of automation of inconsistency management activities.	{Manual, Partial, Automated, Not applicable}

3.7.1 Vertical analysis

Depending on the parameters of the classification framework, in this research, we apply both quantitative and qualitative synthesis methods, separately. When considering quantitative data, depending on the specific data to be analyzed, we apply descriptive statistics for a better understanding of the data. When considering qualitative data, we apply the *line of argument* synthesis [90], that is: firstly we analyze each tool individually to document it and tabulate its main features with respect to each specific parameter of the classification framework, then we analyze the set of tools as a whole, to reason on potential patterns and trends. When both quantitative and qualitative analyses are completed, we integrate their results to explain quantitative results by using qualitative results [47, § 6.5]. The results are discussed in Section 4.

3.7.2 Horizontal analysis

Following the best practice of previous secondary studies [20, 30, 22, 14], we explore significant phenomena across pairs of categories as well. We use contingency tables annotated with the Chi-square statistic at $\alpha = 0.05$, for identifying statistically significant cases. Following the directions of Haviland [40], we report the p-values of the conventional Chi-square test without Yates's correction for continuity. The results are discussed in Section 5.

4 Results

In this section, we elaborate on the findings of this study. First, we discuss the general findings in Section 4.1. Then, we elaborate on the two research questions of our study: the user-oriented characteristics (RQ1) and the realization-oriented characteristics (RQ2) of the sampled tools, in Section 4.2 and 4.3, respectively. In both cases, we contextualize our findings in terms of the three core blended modeling aspects: multi-notation, seamless interaction, and flexibility, as shown in Table 3.

4.1 Overview

In this section, we review some of the general findings regarding the analyzed blended modeling tools. The list of the included tools is shown in Table 4.

Project age and timeline. The tools and their respective projects spread over 25 years, with SOM/ADOxx [T20] being the oldest tool (first release in 1996) in our sample. On average, the age of the tool projects is 10.6 years ($\sigma = 5.9$). The means of the first and last releases are 2008.8 ($\sigma = 5.9$) and 2019.4 ($\sigma = 1.8$), respectively. These numbers suggest a sample of mature enough tools with sufficient recency in terms of the latest release. Fig. 4 provides a visual overview of the age and timeline of tool projects.

Fig. 4: Overview of the age of the tool projects, spanned by their respective first and last releases.

Motivations. The self-declared motivations of the tools vary greatly. We have recorded the mission statements of the tools and clustered them. General-purpose modeling tools are typical in our sample, usually offering multi-notation support for UML-based modeling, e.g. Modelio [T16], USE [T26], and Papyrus [T11]. Some of these tools are very specific about their intentions to combine or augment the traditional graphical notation of UML with textual elements, such as UMLet [T23] and ETAS ASCET Developer [T04]. Among the tools with specific modeling purposes are the ones aiming at process modeling (e.g., SOM/ADOxx [T20], ARIS [T03]), database modeling (e.g., DBDiagram [T10], QuickDBD [T18]), and enterprise architecture (e.g., Archi [T02], ADOIT [T01]).

Web-based implementation. We have found that the majority of the sampled tools, 17 of 26 (65%), are exclusively desktop-based applications, as shown in Table 5.

Table 5: The web-based nature of tools.

Web-based	**#Tools**	**Tools**
No	17 (65%)	[T02], [T03], [T04], [T06], [T07], [T09], [T11], [T12], [T13], [T14], [T15], [T16], [T17], [T20], [T22], [T23], [T26]
Yes	9 (35%)	[T01], [T05], [T08], [T10], [T18], [T19], [T21], [T24], [T25]

Open-source. Half of the sampled tools are released as open-source software (Table 6), allowing access to the source code of the tool.

Table 6: The open-source nature of tools.

Open-source	**#Tools**	**Tools**
No	13 (50%)	[T01], [T03], [T04], [T07], [T08], [T10], [T13], [T15], [T18], [T19], [T20], [T21], [T22]
Yes	13 (50%)	[T02], [T05], [T06], [T09], [T11], [T12], [T14], [T16], [T17], [T23], [T24], [T25], [T26]

Collaboration. Collaborative modeling is the joint creation of a shared representation of a system through means of modeling [30, 20]. Collaboration enables an orchestrated interplay among stakeholders of different domains, and thus, very often, collaboration raises the need for multiple different notations. In real-time collaborative settings, the groupwork of stakeholders happens synchronously. Off-line collaborative settings do not assume synchronicity, but rather stakeholders who

Table 3: Relationships between blended aspects (BA) and the research questions (RQ) of this study.

	RQ1: User-oriented characteristics (Section 4.2)	**RQ2: Realization-oriented characteristics (Section 4.3)**
BA1: Multi-notation	Notations (Section 4.2.1)	Mapping and platforms (Section 4.3.1)
BA2: Seamless interaction	Visualization and navigation (Section 4.2.2)	Change propagation, traceability (Section 4.3.2)
BA3: Flexibility	Model/language/persistence flexibility (Section 4.2.3)	Inconsistency management and tolerance (Section 4.3.3)

Table 4: The list of included tools.

Tool			Releases			Info	
ID	**Name**	**Vendor/Maintainer**	**First**	**Latest**	**Analyzed**	**Open-source**	**Self-declared motivation**
[T01]	ADOIT: Community Edition	BOC Products & Services AG	2003	2020	ADOIT:CE based on ADOIT 12.0	No	Enterprise architecture management
[T02]	Archi	Beauvoir, P and Sarrodie, JB	2010	2021	4.8.1	Yes	Enterprise architecture
[T03]	ARIS	Software AG	2009	2017	2.4d - 7.1.0.1161389	No	Business process modeling
[T04]	ASCET Developer	ETAS	2002	2020	7.6.0 Build ID 209	No	"easily combine texts and graphics suiting your programming needs"
[T05]	AToMPM	Université de Montréal	2013	2020	0.8.5	Yes	Multi-paradigm modeling on the web
[T06]	BlendedProfile	Mälardalen University	2018	2020	0.3	Yes	Blended modelling for UML profiles
[T07]	Boston	Viev	2015	2020	5.0	No	Fact-based modeling via Object-Role Modeling (ORM)
[T08]	Cardanit	ESTECO SpA	2013	2020	Online @07.04.2021.	No	Modeling BPMN with diagrams and tabular views
[T09]	Certware	NASA	2013	2016	2.0	Yes	Safety case modeling
[T10]	DBDiagrams	Holistics Software	2018	2021	Online @07.04.2021.	No	Visualize textual DB schema definition
[T11]	Eclipse Papyrus	The Eclipse Foundation	2008	2020	5.0.0	Yes	Generic-purpose MBSE tool, based on UML and providing support for DSLs via UML Profiles
[T12]	Eclipse Process Framework Project	The Eclipse Foundation	2006	2018	1.5.2	Yes	Software process modeling
[T13]	MagicDraw	CATIA No Magic	1998	2021	MagicDraw 2021x LTR Enterprise	No	Modelling tool that facilitates analysis and design of Object Oriented (OO) systems and databases. It provides code engineering mechanism (with full round-trip support for Java, C++, C#, CL (MSIL) and CORBA IDL programming languages), as well as database schema modeling, DDL generation and reverse engineering facilities.
[T14]	mbdeddr	itemis AG	2012	2018	2018.2.0 based on MPS 2018.2.6	Yes	"Boosting productivity and quality by using extensible DSLs, flexible notations and integrated verification tools."
[T15]	MEMO4ADO	OMiLAB	2015	2018	1.10	No	Multi-Perspective Enterprise Modeling
[T16]	Modelio	Modelisoft	2011	2020	4.1.0 (202001232131)	Yes	Generic modeling tool for UML, BPMN, ArchiMate, SysML, etc
[T17]	OSATE	Carnegie Mellon University	2004	2021	2.9.1	Yes	AADL is a language, with different representations. A textual representation provides a comprehensive view of all details of a system, and graphical if one want to hide some details, and allow for a quick navigation in multiple dimensions.
[T18]	QuickDataBaseDiagrams	Dovetail Technologies Ltd	2002	2021	Online @07.04.2021.	No	Modeling DB schemas by text and diagram
[T19]	SequenceDiagramOrg	-	2014	2021	Online - 9.1.1	No	Improve the efficiency when creating and working with sequence diagrams by combining text notation scripting and drawing by clicking and dragging in the same model.
[T20]	SOM/ADOxx	OMiLAB	1996	2014	SOM 3.0 on ADOxx 1.5	No	Semantic Object Model. Comprehensive approach for object-oriented and semantic modeling of business systems.
[T21]	Swimlanes	-	2014	2021	Online @07.04.2021.	No	Visualize sequence diagrams
[T22]	TopBraid Composer Maestro Edition	TopQuadrant, Inc	2006	2021	7.1.0	No	"TopBraid Composer™ Maestro Edition (TBC-ME) is a comprehensive Knowledge Graph modeling and SPARQL query tool. In use by thousands of commercial customers, Composer offers robust and comprehensive support for building and testing configurations of rich knowledge graphs."
[T23]	UMLet	TU Wien	2002	2018	14.3 Standalone	Yes	Allow textual+visual modeling of UML diagrams
[T24]	UMLetino	TU Wien	2013	2018	14.3	Yes	Allow textual+visual modeling of UML diagrams
[T25]	Umple	University of Ottawa	2008	2020	Online - 1.30.1.5099 .60569f335	Yes	Support the convenient modeling across different formalisms. No particular domain targeted, thus, it's a pretty abstract tool.
[T26]	USE	Universität Bremen	2007	2020	6.0.0	Yes	System modeling via a subset of UML + OCL

work on shared models at different times. As shown in Table 7, the majority of tools, 15 of 26 (58%), provides some means of collaboration. Specifically, off-line techniques are typical, accounting for 9 of 15 collaborative tools (60%) or 9 of 26 tools overall (35%), respectively. Finally, 11 of 26 sampled tools (42%) do not support any means of collaboration.

Table 7: Support for collaboration.

Collaboration	#Tools	Tools
No	11 (42%)	[T03], [T06], [T09], [T11], [T12], [T15], [T19], [T20], [T21], [T22] [T26]
Yes: Off-line	9 (35%)	[T02], [T04], [T13], [T14], [T16], [T17], [T23], [T24], [T25]
Yes: Real-time	6 (23%)	[T01], [T05], [T07], [T08], [T10], [T18]

4.2 User-oriented characteristics (RQ1)

In this section, we discuss the findings related to the user-oriented characteristics of the sampled tools. We contextualize our findings in terms of the three aspects of blended modeling tools: the support for multiple notations (Section 4.2.1), seamless interaction (Section 4.2.2), and flexibility (Section 4.2.3).

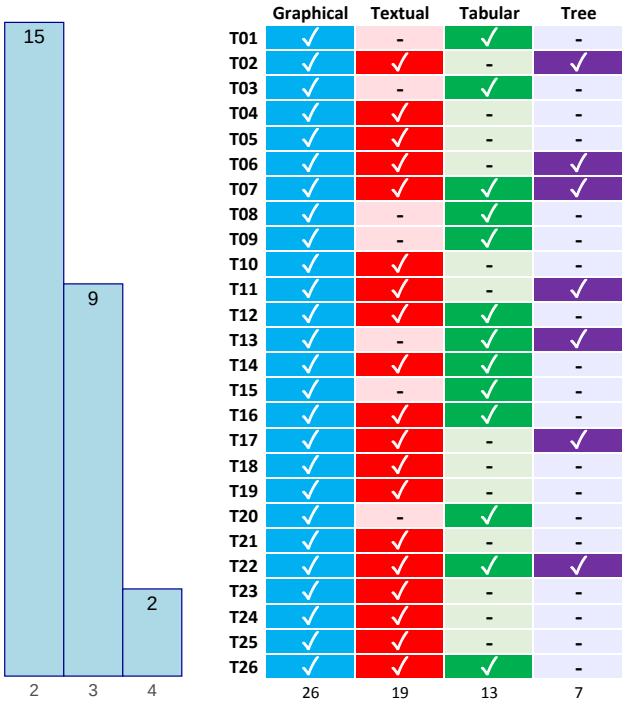
4.2.1 Notations

Notation types. As shown in Fig. 5a and Table 8, the majority of tools, 5 of 26 (19%), support two types of notation, with additional nine tools supporting three types, and two tools supporting four types.

Every tool, 26 of 26 (100%), features a graphical notation. Textual notations are supported by 19 tools. Additional 13 tools were found with a support for tabular notations, and seven with a support for tree-like notations. This information is detailed in Fig. 5b and Table 9.

Table 8: Number of supported notation types.

#Notation types	#Tools	Tools
2	15 (58%)	[T01], [T03], [T04], [T05], [T08], [T09], [T10], [T15], [T18], [T19], [T20], [T21], [T23], [T24], [T25]
3	9 (35%)	[T02], [T06], [T12], [T11], [T13], [T14], [T16], [T17], [T26]
4	2 (8%)	[T07], [T22]



(a) Number of notation types.

(b) Support for specific notation types.

Fig. 5: Number and combinations of notation types.

Table 9: Support for specific notation types.

Notation type	#Tools	Tools
Graphical	26 (100%)	[T01], [T02], [T03], [T04], [T05], [T06], [T07], [T08], [T09], [T10], [T11], [T12], [T13], [T14], [T15], [T16], [T17], [T18], [T19], [T20], [T21], [T22], [T23], [T24], [T25], [T26]
Textual	19 (73%)	[T02], [T04], [T05], [T06], [T07], [T10], [T12], [T11], [T14], [T16], [T17], [T18], [T19], [T21], [T22], [T23], [T24], [T25], [T26]
Tabular	13 (50%)	[T01], [T03], [T07], [T08], [T09], [T12], [T13], [T14], [T15], [T16], [T20], [T22], [T26]
Tree	7 (27%)	[T02], [T06], [T07], [T11], [T13], [T17], [T22]

Embedded notations. We found a single occurrence of embedded notations, i.e., a host notation being enriched by fragments of another notation (Table 10). While the host notation is prevalent during the entirety of the interaction, the embedded notation is accessible in a specific subset of the host notation. For example, in the `Statecharts + Class Diagrams` (SCCD) formalism [78], Class Diagram fragments are used to augment the Statecharts formalism, and provide structural information to compose complex systems.

Table 10: Support for embedded languages.

Embedded languages	#Tools	Tools
No	25 (96%)	[T01], [T02], [T03], [T04], [T05], [T06], [T07], [T08], [T09], [T10], [T11], [T12], [T13], [T15], [T16], [T17], [T18], [T19], [T20], [T21], [T22], [T23], [T24], [T25], [T26]
Yes	1 (4%)	[T14]

Overlap. The majority of tools, 21 of 26 (81%), comes with notations that are not fully overlapping. This means that different notations provide different modeling aspects in these tools. An example of full overlap is where a graphical state machine language can render a state machine model with every structural feature; whereas a table only shows which states have transitions to which states.

Table 11: Overlap between notations.

Overlap	#Tools	Tools
Partial	21 (81%)	[T01], [T02], [T03], [T04], [T05], [T06], [T07], [T08], [T09], [T11], [T12], [T13], [T15], [T16], [T17], [T20], [T22], [T23], [T24], [T25], [T26]
Complete	5 (19%)	[T10], [T14], [T18], [T19], [T21]

4.2.2 Visualization and navigation

Usability aspects in general are hard to measure. To gain reliable results, it is necessary to conduct a complex user study with concrete tasks, a larger number of participants, interviews and/or surveys, and a thorough evaluation of the answers. This is not feasible in the context of this study and therefore, we decided to focus on usability aspects that are i) easily measured objectively and ii) specific to blended modeling.

We do not consider the usability of modeling languages themselves as discussed in [57] and [6]. Instead, we focus on the usability of the tools in terms of the topics that are crucial for blended modeling. The idea of blended modeling is to use the notation that is best suited for the current task at hand. This makes it necessary to switch frequently between the available notations. Therefore, for pleasant usability with good support for the user, a tool must offer the possibility to visualize multiple notations side by side and/or provide seamless navigation between notations, or even synchronized navigation. To clarify this more focused view of usability, we use the term "seamless interaction".

Visualization of multiple syntaxes. In general, a blended modeling tool must have the ability to support multiple concrete syntaxes of the same abstract syntax. This parameter, in particular, addresses the possibility of simultaneously viewing multiple notations within a modeling tool, e.g., side-by-side or in an integrated manner such as projectional editors as mbeddr [T14] do. All 26 identified tools support the simultaneous view of two or more notations.

Synchronized navigation. In addition to the previous parameter, this parameter investigates whether the navigation across multiple notations in the models' editors is synchronized. For instance, this can be the case in a side-by-side view, if an element in one notation is selected, also its corresponding element in the other notation is selected. Another example of such synchronized navigation is the usage of the double click feature to jump between different views showing corresponding elements but belonging to different notations. As shown in Table 12, more than half of the tools, 16 of 26 (62%), provides synchronized navigation facilities.

Table 12: Support for synchronized navigation.

Sync'd navigation	#Tools	Tools
Yes	16 (62%)	[T02], [T03], [T05], [T06], [T07], [T08], [T10], [T12], [T13], [T14], [T16], [T18], [T19], [T21], [T22], [T25]
No	10 (38%)	[T01], [T04], [T09], [T11], [T15], [T17], [T20], [T23], [T24], [T26]

Navigation among notations. Blended modeling tools introduce the benefit that the same model can be viewed and modified using different notations. To enable a fluent modeling experience, the effort required to navigate across notations should be minimal. This binary parameter classifies the effort. It can be either *immediate* (e.g., a click or a keyboard shortcut), or it can involve more complex steps, such as the navigation through multiple (context) menus or wizards. The majority of tools, 20 of 26 (77%), provide immediate navigation from one notation to the other, suggesting a better user experience in terms of seamless interaction.

4.2.3 Flexibility

Flexibility is the user-related embodiment of tolerating vertical and horizontal inconsistencies [83] at various levels of abstraction in the modeling stack and various modeling facilities. In this study, we specifically consider three types of flexibility, as follows.

Table 13: Navigation among notations.

Navigation	#Tools	Tools
Immediate	20 (77%)	[T02], [T03], [T04], [T05], [T06], [T07], [T08], [T10], [T11], [T12], [T13], [T14], [T15], [T16], [T18], [T19], [T20], [T21], [T22], [T25]
Complex	6 (23%)	[T01], [T09], [T17], [T23], [T24], [T26]

Flexibility – models. As shown in Table 14, the majority of tools, 19 of 26 (73%), does not provide flexibility at the model level. This means that there are no inconsistency tolerance mechanisms in place that would allow deviations between different notations describing the same model. However, a small set of six tools support model-level flexibility.

Table 14: Support for model-level flexibility.

Flexibility: models	#Tools	Tools
No	19 (73%)	[T01], [T02], [T03], [T05], [T06], [T08], [T09], [T11], [T13], [T14], [T16], [T17], [T19], [T20], [T21], [T22], [T23], [T24], [T26]
Yes	7 (27%)	[T07], [T12], [T04], [T10], [T15], [T18], [T25]

Flexibility – language. The majority of tools, 22 of 26 (85%), does not provide flexibility at the language-level. (Table 15) This means that vertical inconsistencies between model and language (e.g., broken conformance or typing relationships) are not tolerated. We found three exceptions, which are, however, different from the ones with support for model-level flexibility discussed above: mbeddr [T14], OSATE [T17], TopBraid Composer [T22]. Only a single tool, Umple [T25], supports both model- and language-level flexibility.

Table 15: Support for language-level flexibility.

Flexibility: language	#Tools	Tools
No	22 (85%)	[T01], [T02], [T03], [T04], [T05], [T06], [T07], [T08], [T09], [T10], [T12], [T11], [T13], [T15], [T16], [T18], [T19], [T20], [T21], [T23], [T24], [T26]
Yes	4 (15%)	[T14], [T17], [T22], [T25]

Flexibility – persistence. The majority of tools, 22 of 26 (85%), does not support persisting inconsistent models. (Table 16) Out of the ones with support for persistence-level flexibility, ETAS ASCET Developer [T04] and Umple [T25] support model-flexibility and flexibility at both levels, respectively. The other two tools with support for persistence-level flexibility are MagicDraw [T13] and SequenceDiagramOrg [T19].

Table 16: Support for persistence flexibility.

Flexibility: persistence	#Tools	Tools
No	22 (85%)	[T01], [T02], [T03], [T05], [T06], [T07], [T08], [T09], [T10], [T11], [T12], [T14], [T15], [T16], [T17], [T18], [T20], [T21], [T22], [T23], [T24], [T26]
Yes	4 (15%)	[T04], [T13], [T19], [T25]

4.3 Realization-oriented characteristics (RQ2)

In this section, we discuss the findings related to the implementation characteristics of the sampled tools. We contextualize our findings in terms of the three aspects of blended modeling tools: the support for multiple notations (Section 4.3.1), seamless interaction (Section 4.3.2), and flexibility (Section 4.3.3).

4.3.1 Mapping and platforms

Mapping. The mapping between abstract syntax and notation is typically implemented either in a parser-based or in a projectional fashion. In *parser-based* approaches, the user modifies the models via different notations, and a parser produces the abstract syntax tree. In *projectional* approaches, however, the abstract syntax tree is modified directly. Since projectional editors bypass the stages of parser-based editors, they provide support for notations that cannot be easily parsed, but at the same time deliver a different editing experience for textual notations. As shown in Table 17, the majority of tools, 22 of 26 (85%), implement a parser-based editor, while four come with projectional facilities.

Table 17: Type of mapping.

Mapping	#Tools	Tools
Parser-based	22 (85%)	[T02], [T03], [T04], [T05], [T06], [T07], [T08], [T09], [T10], [T11], [T12], [T16], [T17] [T18], [T19], [T20], [T21], [T22], [T23], [T24], [T25], [T26]
Projectional	4 (15%)	[T01], [T13], [T14], [T15]

Platforms. Eclipse is the only frequently encountered platform in our sample. As shown in Table 18, 10 of 26 tools (38%) are built on top of Eclipse, and 18 are built on other, mainly custom platforms. mbdeddr [T14] is the only MPS-based tool in our sample. One tool, MagicDraw [T13], also supports more than one platform.

Table 18: Platforms of implementation.

Platform	**#Tools**	**Tools**
Other	17 (65%)	[T01], [T03], [T05], [T07], [T08], [T10], [T13], [T15], [T14], [T16], [T18], [T19], [T20], [T21], [T24], [T25], [T26]
Eclipse	10 (38%)	[T02], [T04], [T06], [T09], [T11], [T12], [T13], [T17], [T22], [T23]

4.3.2 Change propagation and traceability

Change propagation and traceability are the realization-oriented manifestations of the *seamless integration* blended modeling aspect. (See Table 3.) During the data extraction phase, however, we have failed to obtain any useful information in these two categories. In the vast majority of cases (exception for ADOIT [T01], ARIS [T03], the Eclipse Process Framework [T12], and MagicDraw [T13]), we have not found explicit discussions of these concerns, nor any evidence of these concerns being explicit in the tool.

We report these negative results to maintain the symmetry of our classification framework. We suggest replications of this study to be carried out in a conceptual way [21], i.e., attempting to answer the research questions using different methods.

4.3.3 Inconsistency management

Inconsistency visualization. As shown in Table 19, the majority of tools, 15 of 26 (58%), does not provide any visualization for inconsistencies. Out of the remaining 11 tools, eight implement an internal visualization mechanism, and three rely on external services.

Table 19: Support for inconsistency visualization.

Inconsistency visualization	**#Tools**	**Tools**
No	15 (58%)	[T01], [T03], [T05], [T06], [T08], [T09], [T12], [T13], [T17], [T19], [T20], [T21], [T23], [T24], [T26]
Internal	8 (31%)	[T02], [T07], [T04], [T10], [T14], [T16], [T18], [T25]
External	3 (11%)	[T11], [T15], [T22]

Inconsistency management type. The two fundamental approaches to manage inconsistencies are prevention, and allow-and-resolve [17]. Preventive techniques effectively prohibit the emergence of inconsistencies, either by serializing user operations (e.g., via locking), or by constructing the underlying data structures in a way that they can never be inconsistent (e.g., in conflict-free replicated data types (CRDT) [72]. Allow-and-resolve approaches embrace the existence of inconsistencies [28] instead of preventing them. This allows treating inconsistencies with highly sophisticated operations for tolerance [5, 18], and resolution [54, 60, 34]. As shown in Table 20, half of the tools, 13 of 26 (50%), prevent inconsistencies. The remaining tools either manage inconsistencies on-the-fly (11 of 26 – 42%) or on-demand (2 of 26 – 8%).

Table 20: Support for different inconsistency management types.

Inconsistency mgmt type	**#Tools**	**Tools**
Preventive	13 (50%)	[T01], [T03], [T06], [T08], [T12], [T13], [T15], [T16], [T21], [T23], [T24], [T25], [T26]
On-the-fly	11 (42%)	[T02], [T04], [T05], [T07], [T09], [T10], [T14], [T18], [T19], [T20], [T22]
On-demand	2 (8%)	[T11], [T17]

Inconsistency management automation. As shown in Table 21, 13 of 26 tools (50%) do not provide inconsistency resolution due to their preventive inconsistency management approach. These tools are identical to the ones of the *preventive* category in Table 20. Out of the remaining 13 tools, 11 provide some level of automation for resolving inconsistencies, while two tools rely on manual resolution.

Table 21: Level of automation of inconsistency management.

Inconsistency automation	**#Tools**	**Tools**
Not applicable	13 (50%)	[T01], [T03], [T06], [T08], [T12], [T13], [T15], [T16], [T21], [T23], [T24], [T25], [T26]
Fully automated	6 (23%)	[T07], [T17], [T18], [T19], [T20], [T22]
Semi-automated	5 (19%)	[T02], [T09], [T10], [T14], [T11]
Manual	2 (8%)	[T04], [T05]

5 Orthogonal findings

We have analyzed the extracted data for horizontal findings, orthogonal to the vertical analysis reported in the previous section. Specifically for this purpose, we have generated contingency tables for each pair of categories of the classification framework and looked for relevant emerging correlations. In this section, we discuss these findings and contextualize them in terms of the aspects of blended modeling: the support for multiple notations (Section 5.1), seamless interaction (Section 5.2), and flexibility and inconsistency management (Section 5.3); and in the additional aspect of technological trends that are independent from the blended aspects (Section 5.4).

5.1 Number of notation types and Overlap of notations

As shown by the data in Section 4.2, the sampled tools support 2.5 types of notation on average. In about 81% of the cases, the overlap between the specific notations is only partial, thus providing a richer way to build models.

Notation types count vs Web-based nature. The number of types of notation tends to be higher in desktop tools. Tools with more than two types of notation are exclusively desktop-based. While every web-based tool in our sample provides a maximum of two types of notations, 11 of 17 desktop tools (65%) provide three or more types of notations. We have measured a statistically significant difference at $p = 0.0064$.[40]

Overlap of notations vs Web-based nature. We found significantly more completely overlapping notations in web-based tools than in desktop-based tools. 4 of 9 web-based tools (44%) come with completely overlapping notations. This ratio is 5.9% in desktop-based tools ($p = 0.0176$). This is in line with the previous observation of web-based tools typically providing fewer types of notation. It is plausible to assume that in desktop tools, the higher number of notation types might result in relevant differences between the notations and, thus, less overlap among them.

[40] For the remainder of the paper, $\alpha = 0.05$, unless specifically noted otherwise. Following the directions of Haviland [40], we report the p-values of the conventional Chi-square test without Yates's correction for continuity.

Notation types count vs open-source nature. The number of notation types tends to be higher in open-source tools than in commercial ones. Three or more types of notations are supported in 8 of 13 open-source tools (62%), while this number is only 3 of 13 (23%) in commercial tools. However, a deeper look also reveals that the only two tools supporting four types of notations are commercial ones ([T07], [T22]). While 2 of 13 commercial tools (15%) provide four types of notations, 10 of 13 (77%) of them support only two. These differences are significant at $p = 0.0105$. It is plausible to assume that while commercial tool vendors have the capabilities to develop sophisticated tools with many types of notations, they still opt for a more streamlined user experience either due to explicit user requirements, or to minimize the technological risks and improve the maintainability of the tools.

5.2 Seamless interaction

In terms of seamless interaction, we have found significant relationships between the navigation among notations, their synchronicity, and the presence of inconsistency visualization.

Navigation among notations vs Synchronous navigation. We have observed a statistically significant difference (p =4E-4) between the *complexity of navigation among notations*, and the *synchronicity of navigation*. The two features go hand in hand. 16 of 16 tools (100%) with support for synchronous navigation also support immediate navigation across different notations. In contrast, only 4 of 10 tools (40%) without synchronous navigation support immediate navigation. That is, in over half of such tools, navigation between notations becomes a complex and tedious task, significantly impacting the user experience in terms of seamless interaction. Synchronous navigation is more frequently observed in tools with completely overlapping notations. While 5 of 5 tools (100%) with completely overlapping notations operate with synchronous navigation, this ratio is only 11 of 21 (52%) in tools with partially overlapping notations.

Navigation among notations vs Inconsistency visualization. We observed that 11 of 11 tools (100%) that support inconsistency visualization operate with immediate navigation; while tools without inconsistency visualization support immediate navigation only in 9 of 15 cases (60%). The difference is significant at $p = 0.0168$.

5.3 Flexibility and inconsistency management

As discussed in Section 4.2, flexibility, in general, is sporadically supported by the tools we have sampled. We have found that the three types of flexibility features (model-level, language-level, persistence-level), often correlate with inconsistency management aspects.

Model-level flexibility vs Inconsistency visualization. Inconsistency visualization is significantly better supported in tools with model-level flexibility. We have found that 7 of 7 tools (100%) with model-level flexibility also support inconsistency visualization, while this ratio drops to 4 of 19 (21%) in tools without model-level flexibility ($p = 3E{-}4$). It is plausible to assume that inconsistency visualization is an enabler to model-level flexibility. Visualizing inconsistencies certainly helps the stakeholders to keep track of inconsistencies and reason about the most appropriate time and approach to resolving them.

Inconsistency visualization vs collaboration. Tools with internal inconsistency visualization features are also collaborative tools. This holds for 8 of 26 tools (31%). Conversely, the 11 of 26 tools (42%) without collaborative features do not support internal means of inconsistency visualization. The ratio of collaborative and non-collaborative tools is split almost evenly when inconsistency visualization is not present. 15 of 26 tools (58%) come without inconsistency visualization, out of which seven (27%) support collaboration and eight (31%) lack collaborative features. These relationships are significant at $p = 0.0047$. These observations can be explained by the strong relationship between collaboration and inconsistencies: as the lack of collaboration might severely reduce the cases when inconsistencies can appear, tools vendors whose tools do not support collaboration might be less interested in developing internal inconsistency visualization techniques.

5.4 Technological trends

We have further identified some purely technological trends, orthogonal to the three facets of blended modeling, mainly related to the web-based nature of tools (Table 5), their collaborative features (Table 7), and their platforms of implementation (Table 18).

Collaboration on the web. The type of collaboration tends to correlate with the type of client software. 5 of 6 tools (83%) that operate with synchronous (real-time) collaboration, are implemented as web-based tools. In contrast, 7 of 9 tools (78%) that operate with asynchronous (off-line) collaboration, are implemented as desktop tools. The type of client software is nearly evenly split in collaborative tools between web clients (7 of 15 – 47%) and desktop clients (8 of 15 – 53%). However, 9 of 11 non-collaborative tools (82%) are built as desktop applications, and we found only two web-based non-collaborative tools. These differences are significant at $p = 0.0164$. These observations are in line with the observations of our previous work [20], especially on the apparent mobilization of collaborative modeling.

"Modeling" platforms are primarily desktop-based. We observed that neither of the web-based tools in our sample is implemented on a platform that explicitly aims to provide *modeling* capabilities. In contrast, 11 of 18 desktop tools (61%) are implemented on top of a modeling platform, such as Eclipse (10 of 18 – 56%), JetBrains MPS (1 of 18 – 6%), and other, custom platforms (7 of 18 – 39%). While the web-based tools in our sample leverage web frameworks that provide reusable elements to build front-end and back-end functionality, the lack of *modeling* frameworks tailored to the web are apparent. These differences are significant at $p = 0.0097$.

6 Discussion

The corpus of this paper consists of 68 academic papers and 68 entries of grey literature survey, which eventually resulted in 26 identified tools. Based on the rigorously constructed research protocol, we are reasonably confident in the representativeness of our sample for the field under study.

6.1 Takeaways

The main takeaway of our investigation is that the state-of-the-art and state-of-the-practice tools only provide *partial and accidental support for blended modeling*. This is not a surprising result, considering the novel and emerging nature of the concept of blended modeling. We have found adequately scaling tools in terms of the number of supported notation types. 11 of 26 tools (42%) provide more than the minimal two notation types (Table 8). Various aspects related to flexibility, however, pose a potentially serious obstacle for multi-notation tools to become true blended modeling tools. Only 7 of 26 tools (27%) provide flexibility at the instance model level, i.e., tolerance of horizontal inconsistencies between models (Table 14). 4 of 26 tools (15%) support flexibility at the language level, i.e., tolerance of vertical inconsistencies, such as conformance

or type discrepancies (Table 15). In terms of user experience (UX), and especially seamless interaction, we noticed encouraging signs in cross-notation navigability and inconsistency management automation. 16 of 26 tools (62%) support a synchronized navigation across their supported notations (Table 12) and, in 20 of 26 tools (77%), immediate navigation is also available (Table 13). This enables a better concert of notations, allowing using them in a truly complementary fashion. 11 of 13 tools (85%) that allow inconsistencies to occur treat them with a substantial level of automation; only 2 of 13 (15%) of such tools (a grand total of 8% – 2 of 26) rely on manual resolution of inconsistencies (Table 21).

In terms of *user-oriented characteristics* (RQ1), we observed a strong dominance of graphical notations, supported by 26 of 26 tools (100%), followed by textual (19 of 26 – 73%), tabular (13 of 26 – 50%), and tree-based ones (7 of 26 – 27%) (Table 9 and Fig. 5b). Only 5 of 26 tools (19%) feature a combination of notations that are completely overlapping in terms of modeling language concepts (Table 11). This means that multi-notation tools tend to leverage the complementary nature of different types of notation. This is a welcome direction as it opens up for opportunities of a richer modeling experience, paramount in approaches such as MVM and MPM and, as such, it motivates the efforts of blended modeling.

In terms of *realization-oriented characteristics (RQ2)*, we observed the dominance of parser-based solutions, employed in 22 of 26 tools (85%) (Table 17). Evidence suggests that projectional editors align better with multi-view and multi-notation principles [8, 84, 86], which are now the typical modeling settings for complex systems [62]. The average age of tools in our sample is 10.6 years ($\sigma = 5.9$), dating the typical modeling tool earlier than the uptick in research interest in projectional editors.[41] We foresee the support for projectional editors to grow as modeling tools are becoming more complex in their denotational and semantic functionalities. We observed a relatively high support for automation of inconsistency management (Table 21). Inconsistency management, and tolerance in particular (Tables 14-16), are key enablers to the flexibility of modeling tools. Only 2 of 26 tools (8%) come without some level of automation in resolving conflicts and these are either research tools, such as [T05], or tools that are explicitly not supporting groupwork, such as [T04].

[41] A directed search on Google Scholar using the `(intitle:"projectional editing" OR intitle:"projectional editor" OR intitle:"projectional editors") OR ("projectional editing" OR "projectional editor" OR "projectional editors")` search string suggests an increasing publication output starting from 2013.

6.2 Challenges and opportunities

By mapping the state-of-the-art and state-of-the-practice, we have identified challenges and opportunities related to the concept of blended modeling in relation to tools.

Multi-formalism. Our study assumed one single underlying abstract syntax and a single underlying formalism, but even with this simplification, the support for multi-notation is sporadic. Multi-formalism, and especially multi-semantics, exacerbates this problem as we anticipate the interest in blended modeling gradually shifting towards more complex domains [13, 58, 83]. We see an opportunity for tool builders and integrators in complex engineering domains that inherently work in an MVM/MPM setup, such as mechatronics, automotive, and robotics, to incorporate blendedness as an enabling concept into their existing tool ecosystems. However, this should be preceded by academic research on extending blended modeling, especially on topics such as coordination between models of different languages [25], and synchronization of abstract and concrete syntax in DSLs [65]. Nevertheless, we expect an early maturation and rapid take-off of blended modeling techniques in an array of applied modeling settings. Therefore, we advise technology transfer entities to closely follow academic and semi-academic advancements to propel the transition of the concept to applied industrial settings.

Seamless interaction. As a primary user experience (UX) concern, seamless interaction can make a substantial difference in user satisfaction [88] towards modeling tools. The user-oriented aspects of our study (Section 4.2.2) show that current tools are often equipped with related features (e.g., synchronous navigation among notations). Such tools have the opportunity to provide holistic support for blended modeling. The evaluation and comparison of realization-oriented aspects, however, is certainly a challenge, as demonstrated in Section 4.3.2. The scope of our study did not include the development of methods that would allow extracting information about user experience and seamless integration of the different modeling paradigms in blended modeling tools. In general, the evaluation of such user-facing aspects remains a challenge. We encourage researchers to develop methods suitable for extracting the types of information outlined in Section 4.3.2; and to further enrich the user-facing aspects, based on Section 4.2.2. We suggest facilitating dedicated evaluation

events, e.g., hands-on workshops at major conferences, where crowdsourcing models for hands-on experimentation and evaluation are feasible because of the volume of the co-located participants and their significant expertise, such as the Hands-on Workshop on Collaborative Modeling (HoWCoM)[42], and the workshop on Human Factors in Modeling / Modeling of Human Factors (HuFaMo)[43] at MODELS[44], as well as the Conference on Human Factors in Computing Systems (CHI)[45]. Explicitly modeled user interfaces [76] and API protocols [79] provide especially good foundations for developing software tools that allow seamless switching between notations. Seamless interaction across textual and graphical notations is especially challenging [25] due to the differences between their respective grammar-based and metamodel-based approaches [36]. Projectional editing [86] provides appropriate means to overcome these limitations, thus, we advise researchers to investigate seamless interaction from this standpoint as well.

Flexibility. The flexibility of modeling tools in terms of (temporarily) tolerating inconsistencies, such as violations of well-formedness rules and inter-notation/interview discrepancies, is best approached by employing state-of-the-art inconsistency models, such as eventual and strong eventual consistency [72]. Although the scope of this study does not entail the particularities of inconsistency management, we have identified traces and patterns of shortcomings in this aspect. While the majority of tools operate in a preventive inconsistency management fashion (Section 4.3.3), they implement prevention in the traditional way, i.e., by prohibiting consistency-breaking operations. Such approaches stem from the limitations of strict consistency, whereas novel developments in the field offer much better inconsistency management and, by extension, better flexibility. Strong eventual consistency (SEC) [72], for example, offers a convenient trade-off between the strictness of strong consistency and the guarantees of eventual consistency. As such, SEC is especially well-suited for tools whose developers are more comfortable with preventive inconsistency management models. Such avenues have been explored in multiple collaborative modeling frameworks, such as lowkey[46], and C-Praxis [56]. We see an opportunity in developing advanced inconsistency tolerance methods that work at the semantic level of models, especially if blended modeling is extended to support multiple abstract syntaxes or multiple semantics. Recently, inconsistency management between the data and (meta)model level has been investigated, e.g., by Zaher et al. [92]. Such directions align well with the persistence flexibility aspect of modeling tools, which is sporadically supported currently. In general, we encourage tool builders to treat inconsistencies as first-class citizens and, instead of overspending on resources to prevent them, we suggest appropriately managing them [28, 17].

*The many facets of web-based tools.*The interconnected nature of web-based tools and the advanced communication and networking standards of the Internet align well with building collaborative modeling tools. We observed a tendency of tool builders to use web technologies more in collaborative tools (Section 5.4). However, we also observed that web-based tools come with significantly less types of notations (Section 5.1), and that *modeling* platforms and frameworks are built for desktop applications (Section 5.4). It is possible that the shortage of modeling frameworks and language workbenches with a web-based focus limits the ability of tool vendors to provide rich modeling tools with numerous types of notations and advanced modeling facilities. Modeling platforms such as Eclipse already started providing support for deploying modeling tools onto the web, but this is merely a workaround. We foresee an increasing industrial interest in web-based modeling frameworks, such as WebGME [53], providing researchers of language engineering and language workbenches with opportunities.

Tools performance assessment. The current generation of modeling tools is facing challenges to manage large-scale complex models [10, 49]. Given the presence of multiple different notations in blended modeling, estimating tool performance when dealing with large-scale and complex models is crucial for the future technical sustainability of blended modeling. However, in our data analysis, we did not observe that tool builders discuss the performance of their blended modeling tools. We conjecture that this lack of communication is mainly because (i) tool performance is still an open problem in MDE [10], and (ii) there are still no standard benchmarks for objectively and fairly comparing the performance of different modeling tools. We suggest that researchers investigate a shared and open benchmark for assessing the performance of modeling tools when dealing with models of different levels of size and complexity (i.e., from a few up to millions of modeling elements). To avoid bias concerning specific DSMLs or application domains, populating such benchmark should be a community effort, where researchers and tool builders coming from different domains collaborate and contribute

[42] http://howcom2021.github.io/
[43] https://www.monash.edu/it/humanise-lab/hufamo21
[44] http://www.modelsconference.org/
[45] https://chi2021.acm.org/
[46] https://github.com/geodes-sms/lowkey

their models, language definitions, and requirements (e.g., expected time to open a model with 1M elements, expected time to propagate a model change from a visual syntax to the corresponding textual one, etc.). Having such shared benchmarks will provide practitioners with an evidence-based instrument for comparing similar modeling tools and choosing the best one according to their project and organizational needs. Also, a shared benchmark will help MDE researchers in designing and conducting empirical studies assessing the performance of (blended) modeling tools, thus providing objective and replicable knowledge for addressing the grand challenge of scalability in Model-Driven Engineering [10].

7 Threats to validity

The study reported in this paper has been carried out based on a carefully designed protocol. To minimize the threats to validity, we have designed our protocol based on well-established guidelines for systematic studies in software engineering [47, 90, 93] and those for including grey literature by Garousi et al. [32].

We have assessed the quality of our study following the guidelines by Petersen et al. [64] and achieved a 63.6% result. This score is significantly higher than the median and absolute maximum scores (33% and 48%, respectively) reported in [64]. This high score can be mainly attributed to the detailed search strategy; the involvement of external senior consultants in the study design phase; and the involvement of multiple authors in the screening phase, minimizing the number of false inclusions and exclusions.

In the following, we discuss the possible threats to the validity of our study and elaborate on how we have mitigated them.

7.1 External validity

External validity concerns the generalizability of the results [90] and it is primarily associated with the sampling method. The most severe threat to external validity is the lack of representativeness of the selected tools to the field of interest in general. We have mitigated this threat by an appropriately constructed protocol with two orthogonal concerns. First, our search strategy included manual and automated search steps, with exhaustively iterative backward and forward snowballing. Second, we have carried out this search both for the academic and the grey literature [32].

Another class of threats to external validity can be attributed to the inclusion and exclusion criteria used in the screening. To mitigate these threats, we defined exclusion criteria specific to the type of literature (white or grey) being surveyed. Some threats remain, for example, due to the exclusion of non-peer-reviewed academic material (`A-E2` in Section 3), and the exclusion of proprietary tools that do not allow experimentation with at least a trial version (`GEN-E4`). We consider these threats minimal.

7.2 Internal validity

Internal validity is the extent to which claims are supported by data and it is primarily associated with the study design. We have mitigated this risk by the thorough construction and validation of our protocol. The protocol has been developed by multiple authors with relevant expertise on the topics related to blended modeling. Additionally, the protocol has been validated by an external reviewer with significant expertise in empirical research. We have employed rigorous descriptive statistical methods for orthogonal analysis and validation of the data to further mitigate the threats.

7.3 Construct validity

Construct validity is concerned with the generalizability of the measures of the study to the investigated concepts, and it is primarily associated with the categories and parameters employed during the data extraction and the subsequent analysis. We have mitigated the threats by mapping the research questions to typical parameters before constructing our search strategy. Consequently, we are reasonably confident about the construction validity of the search strings used in the automatic search steps. We have further minimized the threats in the screening phase by refining the inclusion and exclusion criteria in multiple iterations, to reach unambiguous definitions. Each study was assigned to two researchers randomly, and a third researcher was involved to oversee the results and make the final decisions on the inclusion.

7.4 Conclusion validity

Conclusion validity is the degree of credibility of the conclusions, based on the relationship between cause and effect. Specifically, in our case, conclusion validity is concerned with the relationship between the conclusions communicated in Sections 4.2–6 and the extracted data. We mitigated the main threats in two steps. First, considering that different researchers might interpret

the same data in different ways, we have documented our research protocol in great detail and made it available along with our datasets and statistical analysis scripts in the publicly available replication package.[31] Second, we have constructed conclusions based only on the available data. Any hypotheses and conjunctures were explicitly marked as such.

8 Conclusions

In this paper, we have reported the results of our systematic, multi-vocal study on the potential, opportunities, and challenges of the emerging approach of blended modeling. We have reviewed nearly 5,000 academic papers, and nearly 1,500 entries of grey literature. Based on these, we have identified 133 candidate tools, and eventually selected 26 state-of-the-art and state-of-the-practice modeling tools which represent the current spectrum of modeling tools. We defined a classification framework for these tools which we used to map their support for other blended aspects, such as navigation and inconsistency tolerance.

Our findings show that current tooling only provides partial support for the features of blended modeling, in particular for inconsistencies between different notations of the same model. The existing support for automated consistency management is encouraging. We also observe that the overlap between notations is not complete. Projectional editing seems to be a promising avenue for future blended modeling, but most existing tools we reviewed are not projectional. Concerning the challenges, we observe that support for multi-formalism and multi-semantics is still largely lacking. We also see opportunities for improvements when it comes to the seamless integration of the different modeling notations and the evaluation of the user experience. Finally, we identify incorporating "softer" models of consistency that directly use the semantics of the models to achieve eventual consistency as a promising area of future research.

We foresee a new generation of modeling tools that will take blended modeling further by introducing semantic techniques that will allow basing the modeling workflow on multiple different abstract syntaxes.

As for future work, we are working on implementing a generator that produces blended modeling tools for arbitrary domain-specific languages. These tools will be based on the takeaways of this study as well as on a prototype implementation that already embraces the blended principles by Addazi et al. [1]. We intend to keep our dataset up-to-date and report increments on the efforts made on improving blended modeling. Finally, we plan to develop methods for the evaluation of the user experience of blended modeling tools based on hands-on events and workshops.

Acknowledgements

The authors would like to thank Patricia Lago and Matthias Tichy for reviewing the protocol, and their constructive remarks. The authors would like to thank the reviewers for their constructive remarks that helped improve the initial manuscript significantly. Last but not least, the authors would like to thank the tool vendors and experts who helped us validate the data in this paper, including Martin Auer, Dominik Bork, Lola Burgueño, Frank Hoffmann, Trevor Jobling, Timothy Lethbridge, Victor Morgante, Staffan Persson, Irene Polikoff, Alessandro Turco, Tamas Szabo, the dbdiagram.io team, the ETAS team (sales.de@etas.com).

Referred Tools

T01. BOC Products & Services AG (2021) ADOIT:Community Edition. https://www.adoit-community.com/en/, Retrieved: 22/05/2021.
T02. Beauvoir, P and Sarrodie, JB (2021) Archi. https://www.archimatetool.com/, Retrieved: 22/05/2021.
T03. Software AG (2021) ARIS. https://www.ariscommunity.com/, Retrieved: 22/05/2021.
T04. ETAS (2021) ASCET Developer. https://www.etas.com/en/products/ascet-developer.php, Retrieved: 22/05/2021.
T05. Université de Montréal (2021) AToMPM. https://atompm.github.io/, Retrieved: 22/05/2021.
T06. Mälardalen University (2021) Blended Profile. http://www.es.mdh.se/ModComp/demo.html, Retrieved: 09/08/2021.
T07. Viev (2021) Boston Professional. https://www.viev.com/index.php/products-menu/boston-professional, Retrieved: 22/05/2021.
T08. ESTECO SpA (2021) Cardanit. https://www.cardanit.com/, Retrieved: 22/05/2021.
T09. NASA (2021) certware. https://nasa.github.io/CertWare/, Retrieved: 22/05/2021.
T10. Holistics Software (2021) DBDiagram. https://dbdiagram.io/home, Retrieved: 22/05/2021.
T11. The Eclipse Foundation (2021) Eclipse Papyrus. https://www.eclipse.org/papyrus/, Retrieved: 22/05/2021.
T12. The Eclipse Foundation (2021) Eclipse Process Framework Project. https://projects.eclipse.org/projects/technology.epf, Retrieved: 22/05/2021.

T13. CATIA No Magic (2021) MagicDraw. https://www.3ds.com/products-services/catia/products/no-magic/, Retrieved: 22/05/2021.
T14. itemis AG (2021) mbeddr. http://mbeddr.com/, Retrieved: 22/05/2021.
T15. OMiLAB (2021) MEMO4ADO. https://austria.omilab.org/psm/content/memo4ado/info, Retrieved: 08/09/2021.
T16. Modelisoft (2021) Modelio. https://www.modelio.org/, Retrieved: 22/05/2021.
T17. Carnegie Mellon University (2021) OSATE. https://osate.org/, Retrieved: 22/05/2021.
T18. Dovetail Technologies Ltd (2021) QuickDataBaseDiagrams. https://www.quickdatabasediagrams.com/, Retrieved: 22/05/2021.
T19. - (2021) SequenceDiagram.org. https://sequencediagram.org, Retrieved: 22/05/2021.
T20. OMiLAB (2021) SOM/ADOxx. https://austria.omilab.org/psm/content/som/info?view=home, Retrieved: 22/05/2021.
T21. - (2021) Swimlanes.io. https://swimlanes.io/, Retrieved: 22/05/2021.
T22. TopQuadrant, Inc (2021) TopBraid Composer. https://www.topquadrant.com/products/topbraid-composer/, Retrieved: 22/05/2021.
T23. TU Wien (2021) UMLet. https://www.umlet.com/, Retrieved: 22/05/2021.
T24. TU Wien (2021) UMLetino 14.3. https://www.umletino.com/, Retrieved: 22/05/2021.
T25. University of Ottawa (2021) Umple. https://cruise.umple.org/umple/, Retrieved: 22/05/2021.
T26. Universität Bremen (2021) USE – The UML-based Specification Environment. http://useocl.sourceforge.net/w/index.php/Main_Page, Retrieved: 22/05/2021.

References

1. Addazi L, Ciccozzi F (2021) Blended graphical and textual modelling for UML profiles: A proof-of-concept implementation and experiment. J Syst Softw 175:110,912, DOI 10.1016/j.jss.2021.110912
2. Adve SV, Gharachorloo K (1996) Shared Memory Consistency Models: A Tutorial. Computer 29(12):66–76, DOI 10.1109/2.546611
3. Atkinson C, Kühne T (2008) Reducing accidental complexity in domain models. Softw Syst Model 7(3):345–359, DOI 10.1007/s10270-007-0061-0
4. Balegas V, Duarte S, Ferreira C, Rodrigues R, Preguiça NM, Najafzadeh M, Shapiro M (2015) Putting consistency back into eventual consistency. In: Proceedings of the Tenth European Conference on Computer Systems, EuroSys 2015, ACM, pp 6:1–6:16, DOI 10.1145/2741948.2741972
5. Balzer R (1991) Tolerating Inconsistency. In: Proceedings of the 13th International Conference on Software Engineering, IEEE/ACM, pp 158–165
6. Barisic A, Amaral V, Goulão M (2012) Usability Evaluation of Domain-Specific Languages. In: 8th International Conference on the Quality of Information and Communications Technology, QUATIC 2012, IEEE, pp 342–347, DOI 10.1109/QUATIC.2012.63
7. Basili VR, Caldiera G, Rombach HD (1994) The Goal Question Metric Approach. In: Encyclopedia of Software Engineering, vol 2, Wiley, pp 528–532
8. Berger T, Völter M, Jensen HP, Dangprasert T, Siegmund J (2016) Efficiency of projectional editing: a controlled experiment. In: Proceedings of the 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering, FSE 2016, ACM, pp 763–774, DOI 10.1145/2950290.2950315
9. Broy M (2012) Software and System Modeling: Structured Multi-view Modeling, Specification, Design and Implementation. In: Conquering Complexity, Springer, pp 309–372, DOI 10.1007/978-1-4471-2297-5_14
10. Bucchiarone A, Cabot J, Paige RF, Pierantonio A (2020) Grand challenges in model-driven engineering: an analysis of the state of the research. Softw Syst Model 19(1):5–13, DOI 10.1007/s10270-019-00773-6
11. Carreira P, Amaral V, Vangheluwe H (2020) Foundations of Multi-Paradigm Modelling for Cyber-Physical Systems. Springer Nature
12. Charfi A, Schmidt A, Spriestersbach A (2009) A Hybrid Graphical and Textual Notation and Editor for UML Actions. In: Model Driven Architecture - Foundations and Applications, 5th European Conference, ECMDA-FA 2009, Springer, LNCS, vol 5562, pp 237–252, DOI 10.1007/978-3-642-02674-4_17
13. Cicchetti A, Ciccozzi F, Pierantonio A (2019) Multi-view approaches for software and system modelling: a systematic literature review. Softw Syst Model 18(6):3207–3233, DOI 10.1007/s10270-018-00713-w
14. Ciccozzi F, Malavolta I, Selic B (2019) Execution of UML models: a systematic review of research and practice. Softw Syst Model 18(3):2313–2360, DOI 10.1007/s10270-018-0675-4
15. Ciccozzi F, Tichy M, Vangheluwe H, Weyns D (2019) Blended Modelling - What, Why and How.

In: 22nd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion, MODELS Companion 2019, IEEE, pp 425–430, DOI 10.1109/MODELS-C.2019.00068
16. Corley J, Syriani E, Ergin H, Van Mierlo S (2016) Modern Software Engineering Methodologies for Mobile and Cloud Environments, IGI Global, chap Cloud-based Multi-View Modeling Environments, pp 120–139. 7
17. David I (2019) A Foundation for Inconsistency Management in Model-Based Systems Engineering. PhD thesis, University of Antwerp, Belgium, Middelheimlaan 1, 2020 Antwerpen, Belgium
18. David I, Syriani E, Verbrugge C, Buchs D, Blouin D, Cicchetti A, Vanherpen K (2016) Towards Inconsistency Tolerance by Quantification of Semantic Inconsistencies. In: Proceedings of the 1st International Workshop on Collaborative Modelling in MDE (COMMitMDE 2016) co-located with ACM/IEEE 19th International Conference on Model Driven Engineering Languages and Systems (MoDELS 2016), CEUR-WS.org, CEUR Workshop Proceedings, vol 1717, pp 35–44
19. David I, Denil J, Vangheluwe H (2018) Process-oriented Inconsistency Management in Collaborative Systems Modeling. In: 16th International Industrial Simulation Conference 2018, ISC 2018, Eurosis, pp 54–61
20. David I, Aslam K, Faridmoayer S, Malavolta I, Syriani E, Lago P (2021) Collaborative Model-Driven Software Engineering: A Systematic Update. In: 24th International Conference on Model Driven Engineering Languages and Systems, MODELS 2021, IEEE, pp 273–284, DOI 10.1109/MODELS50736.2021.00035
21. Dennis AR, Valacich JS (2015) A replication manifesto. AIS Trans Replication Res 1:1, DOI 10.17705/1atrr.00001
22. Di Francesco P, Lago P, Malavolta I (2019) Architecting with microservices: A systematic mapping study. J Syst Softw 150:77–97, DOI 10.1016/j.jss.2019.01.001
23. do Nascimento LM, Viana DL, Neto P, Martins D, Garcia VC, Meira S (2012) A systematic mapping study on domain-specific languages. In: The Seventh International Conference on Software Engineering Advances (ICSEA 2012), pp 179–187
24. Easterbrook S, Finkelstein A, Kramer J, Nuseibeh B (1994) Coordinating Distributed ViewPoints: the anatomy of a consistency check. Concurrent Engineering 2(3):209–222, DOI 10.1177/1063293X9400200307
25. Engelen L, van den Brand M (2010) Integrating Textual and Graphical Modelling Languages. Electron Notes Theor Comput Sci 253(7):105–120, DOI 10.1016/j.entcs.2010.08.035
26. Engels G, Küster JM, Heckel R, Groenewegen L (2001) A methodology for specifying and analyzing consistency of object-oriented behavioral models. In: Proceedings of the 8th European Software Engineering Conference held jointly with 9th ACM SIGSOFT International Symposium on Foundations of Software Engineering 2001, ACM, pp 186–195, DOI 10.1145/503209.503235
27. Erdweg S, et al (2015) Evaluating and comparing language workbenches: Existing results and benchmarks for the future. Comput Lang Syst Struct 44:24–47, DOI 10.1016/j.cl.2015.08.007
28. Finkelstein A (2000) A Foolish Consistency: Technical Challenges in Consistency Management. In: Database and Expert Systems Applications, 11th International Conference, DEXA 2000, Springer, LNCS, vol 1873, pp 1–5, DOI 10.1007/3-540-44469-6_1
29. Finkelstein A, Gabbay DM, Hunter A, Kramer J, Nuseibeh B (1994) Inconsistency Handling in Multperspective Specifications. IEEE Trans Software Eng 20(8):569–578, DOI 10.1109/32.310667
30. Franzago M, Ruscio DD, Malavolta I, Muccini H (2018) Collaborative Model-Driven Software Engineering: A Classification Framework and a Research Map. IEEE Trans Software Eng 44(12):1146–1175, DOI 10.1109/TSE.2017.2755039
31. Franzosi R (2010) Quantitative narrative analysis. 162, Sage
32. Garousi V, Fernandes JM (2016) Highly-cited papers in software engineering: The top-100. Inf Softw Technol 71:108–128, DOI 10.1016/j.infsof.2015.11.003
33. Garousi V, Felderer M, Mäntylä MV (2019) Guidelines for including grey literature and conducting multivocal literature reviews in software engineering. Inf Softw Technol 106:101–121, DOI 10.1016/j.infsof.2018.09.006
34. Gausemeier J, Schäfer W, Greenyer J, Kahl S, Pook S, Rieke J (2009) Management of cross-domain model consistency during the development of advanced mechatronic systems. In: DS 58-6: Proceedings of ICED 09, the 17th International Conference on Engineering Design, ICED, vol 6, pp 1–12
35. Giese H, Wagner R (2006) Incremental Model Synchronization with Triple Graph Grammars. In: Model Driven Engineering Languages and Systems, 9th International Conference, MoDELS 2006, Springer, LNCS, vol 4199, pp 543–557, DOI 10.

1007/11880240_38
36. Gjøsæter T, Prinz A, Scheidgen M (2008) Metamodel or Grammar? Methods and Tools for the Formal Definition of Languages. In: Nordic Workshop on Model Driven Engineering (NW-MoDE 2008), pp 67–82
37. Granada D, Vara JM, Blanco FJP, Marcos E (2017) Model-based Tool Support for the Development of Visual Editors - A Systematic Mapping Study. In: Proceedings of the 12th International Conference on Software Technologies, ICSOFT 2017, SciTePress, pp 330–337, DOI 10.5220/0006430503300337
38. Greenhalgh T, Peacock R (2005) Effectiveness and efficiency of search methods in systematic reviews of complex evidence: audit of primary sources. BMJ 331(7524):1064–1065
39. Gu Z, Wang S, Kodase S, Shin KG (2003) An end-to-end tool chain for multi-view modeling and analysis of avionics mission computing software. In: Proceedings of the 24th IEEE Real-Time Systems Symposium (RTSS 2003), 3-5 December 2003, Cancun, Mexico, IEEE Computer Society, pp 78–81, DOI 10.1109/REAL.2003.1253256
40. Haviland MG (1990) Yates's correction for continuity and the analysis of 2× 2 contingency tables. Statistics in medicine 9(4):363–367, DOI 10.1002/sim.4780090403
41. Huning L, Osterkamp T, Schaarschmidt M, Pulvermüller E (2021) Seamless integration of hardware interfaces in UML-based MDSE tools. In: Proceedings of the 16th International Conference on Software Technologies, ICSOFT 2021, Online Streaming, July 6-8, 2021, SCITEPRESS, pp 233–244, DOI 10.5220/0010575802330244
42. ISO/IEC/IEEE (2011) Systems and software engineering – architecture description. ISO/IEC/IEEE 42010:2011(E) (Revision of ISO/IEC 42010:2007 and IEEE Std 1471-2000) pp 1–46
43. Iung A, Carbonell J, Marchezan L, Rodrigues EM, Bernardino M, Basso FP, Medeiros B (2020) Systematic mapping study on domain-specific language development tools. Empir Softw Eng 25(5):4205–4249, DOI 10.1007/s10664-020-09872-1
44. Kehrer T, Kelter U, Taentzer G (2013) Consistency-preserving edit scripts in model versioning. In: 2013 28th IEEE/ACM International Conference on Automated Software Engineering, ASE 2013, IEEE, pp 191–201, DOI 10.1109/ASE.2013.6693079
45. Kelly S (2017) Collaborative modelling with version control. In: Software Technologies: Applications and Foundations - STAF 2017 Collocated Workshops, Springer, LNCS, vol 10748, pp 20–29, DOI 10.1007/978-3-319-74730-9_3
46. Kitchenham BA, Brereton P (2013) A systematic review of systematic review process research in software engineering. Inf Softw Technol 55(12):2049–2075, DOI 10.1016/j.infsof.2013.07.010
47. Kitchenham BA, Charters S (2007) Guidelines for performing systematic literature reviews in software engineering, Version 2.3. EBSE Technical Report EBSE-2007-01, Keele University and University of Durham
48. Klare H, Kramer ME, Langhammer M, Werle D, Burger E, Reussner RH (2021) Enabling consistency in view-based system development - the vitruvius approach. J Syst Softw 171:110,815, DOI 10.1016/j.jss.2020.110815
49. Kolovos DS, Rose LM, Matragkas ND, Paige RF, Guerra E, Cuadrado JS, de Lara J, Ráth I, Varró D, Tisi M, Cabot J (2013) A research roadmap towards achieving scalability in model driven engineering. In: Proceedings of the Workshop on Scalability in Model Driven Engineering, ACM, p 2, DOI 10.1145/2487766.2487768
50. Lamport L (1979) How to Make a Multiprocessor Computer That Correctly Executes Multiprocess Programs. IEEE Trans Computers 28(9):690–691, DOI 10.1109/TC.1979.1675439
51. Lazăr CL (2011) Integrating Alf editor with Eclipse UML editors. Studia Universitatis Babes-Bolyai, Informatica 56(3)
52. Maro S, Steghöfer J, Anjorin A, Tichy M, Gelin L (2015) On integrating graphical and textual editors for a UML profile based domain specific language: an industrial experience. In: Proceedings of the 2015 ACM SIGPLAN International Conference on Software Language Engineering, SLE 2015, ACM, pp 1–12
53. Maróti M, Kecskés T, Kereskényi R, Broll B, Völgyesi P, Jurácz L, Levendovszky T, Lédeczi Á (2014) Next generation (meta) modeling: web- and cloud-based collaborative tool infrastructure. MPM@ MoDELS 1237:41–60
54. Mens T, Straeten RVD, D'Hondt M (2006) Detecting and Resolving Model Inconsistencies Using Transformation Dependency Analysis. In: Model Driven Engineering Languages and Systems, 9th International Conference, MoDELS 2006, Genova, Italy, October 1-6, 2006, Proceedings, Springer, LNCS, vol 4199, pp 200–214, DOI 10.1007/11880240_15
55. Merkle B (2010) Textual modeling tools: overview and comparison of language workbenches. In: Companion to the 25th Annual ACM SIGPLAN Confer-

ence on Object-Oriented Programming, Systems, Languages, and Applications, SPLASH/OOPSLA 2010, ACM, pp 139–148, DOI 10.1145/1869542.1869564
56. Michaux J, Blanc X, Shapiro M, Sutra P (2011) A semantically rich approach for collaborative model edition. In: Proceedings of the 2011 ACM Symposium on Applied Computing (SAC), ACM, pp 1470–1475, DOI 10.1145/1982185.1982500
57. Moody DL (2009) The "Physics" of Notations: Toward a Scientific Basis for Constructing Visual Notations in Software Engineering. IEEE Trans Software Eng 35(6):756–779, DOI 10.1109/TSE.2009.67
58. Mosterman PJ, Vangheluwe H (2004) Computer Automated Multi-Paradigm Modeling: An Introduction. Simul 80(9):433–450, DOI 10.1177/0037549704050532
59. Negm E, Makady S, Salah A (2019) Survey on Domain Specific Languages Implementation Aspects. International Journal of Advanced Computer Science and Applications 10
60. Nentwich C, Emmerich W, Finkelstein A (2003) Consistency Management with Repair Actions. In: Proceedings of the 25th International Conference on Software Engineering, IEEE, pp 455–464, DOI 10.1109/ICSE.2003.1201223
61. Nuseibeh B, Easterbrook SM, Russo A (2001) Making inconsistency respectable in software development. J Syst Softw 58(2):171–180, DOI 10.1016/S0164-1212(01)00036-X
62. Persson M, Törngren M, Qamar A, Westman J, Biehl M, Tripakis S, Vangheluwe H, Denil J (2013) A characterization of integrated multi-view modeling in the context of embedded and cyber-physical systems. In: Proceedings of the International Conference on Embedded Software, EMSOFT 2013, IEEE, pp 10:1–10:10, DOI 10.1109/EMSOFT.2013.6658588
63. Petersen K, Feldt R, Mujtaba S, Mattsson M (2008) Systematic mapping studies in software engineering. In: 12th International Conference on Evaluation and Assessment in Software Engineering, EASE 2008, BCS, Workshops in Computing
64. Petersen K, Vakkalanka S, Kuzniarz L (2015) Guidelines for conducting systematic mapping studies in software engineering: An update. Inf Softw Technol 64:1–18, DOI 10.1016/j.infsof.2015.03.007
65. Ráth I, Ökrös A, Varró D (2010) Synchronization of abstract and concrete syntax in domain-specific modeling languages - By mapping models and live transformations. Softw Syst Model 9(4):453–471, DOI 10.1007/s10270-009-0122-7
66. Reineke J, Stergiou C, Tripakis S (2019) Basic problems in multi-view modeling. Softw Syst Model 18(3):1577–1611, DOI 10.1007/s10270-017-0638-1
67. Ries B, Capozucca A, Guelfi N (2018) Messir: a text-first DSL-based approach for UML requirements engineering (tool demo). In: Proceedings of the 11th ACM SIGPLAN International Conference on Software Language Engineering, SLE 2018, ACM, pp 103–107, DOI 10.1145/3276604.3276614
68. Rodgers M, Sowden A, Petticrew M, Arai L, Roberts H, Britten N, Popay J (2009) Testing methodological guidance on the conduct of narrative synthesis in systematic reviews: effectiveness of interventions to promote smoke alarm ownership and function. Evaluation 15(1):49–73, DOI 10.1177/1356389008097871
69. Rothstein HR, Hopewell S (2009) Grey literature. The handbook of research synthesis and meta-analysis 2:103–125, DOI 10.1002/0470870168.ch4
70. Scheidgen M (2008) Textual Modelling Embedded into Graphical Modelling. In: Model Driven Architecture - Foundations and Applications, 4th European Conference, ECMDA-FA 2008, Springer, LNCS, vol 5095, pp 153–168, DOI 10.1007/978-3-540-69100-6_11
71. Schulze M, Weiland J, Beuche D (2012) Automotive model-driven development and the challenge of variability. In: 16th International Software Product Line Conference, SPLC '12, Salvador, Brazil - September 2-7, 2012, Volume 1, ACM, pp 207–214, DOI 10.1145/2362536.2362565
72. Shapiro M, Preguiça NM, Baquero C, Zawirski M (2011) Conflict-Free Replicated Data Types. In: Stabilization, Safety, and Security of Distributed Systems - 13th International Symposium, SSS 2011, Springer, Lecture Notes in Computer Science, vol 6976, pp 386–400, DOI 10.1007/978-3-642-24550-3_29
73. Simonyi C (1995) The Death of Computer Languages, The Birth of Intentional Programming. Tech. Rep. MSR-TR-95-52
74. Spanoudakis G, Zisman A (2001) Inconsistency management in software engineering: Survey and open research issues. In: Handbook of Software Engineering and Knowledge Engineering: Volume I: Fundamentals, World Scientific, pp 329–380
75. Stevens P (2020) Maintaining consistency in networks of models: bidirectional transformations in the large. Softw Syst Model 19(1):39–65, DOI 10.1007/s10270-019-00736-x
76. Syriani E, Riegelhaupt D, Barroca B, David I (2021) Generation of Custom Textual Model Ed-

itors. Modelling 2(4):609–625
77. Torres W, van den Brand MGJ, Serebrenik A (2021) A systematic literature review of cross-domain model consistency checking by model management tools. Softw Syst Model 20(3):897–916, DOI 10.1007/s10270-020-00834-1
78. Van Mierlo S, Van Tendeloo Y, Meyers B, Exelmans J, Vangheluwe H (2016) SCCD: SCXML extended with class diagrams. In: Proceedings of the Workshop on Engineering Interactive Systems with SCXML, vol 2, pp 1–2
79. Van Mierlo S, Van Tendeloo Y, David I, Meyers B, Gebremichael A, Vangheluwe H (2018) A multi-paradigm approach for modelling service interactions in model-driven engineering processes. In: Proceedings of the Model-driven Approaches for Simulation Engineering Symposium, SpringSim (Mod4Sim) 2018, ACM, pp 6:1–6:12
80. van Rest O, Wachsmuth G, Steel JRH, Süß JG, Visser E (2013) Robust Real-Time Synchronization between Textual and Graphical Editors. In: Theory and Practice of Model Transformations - 6th International Conference, ICMT@STAF 2013, Springer, LNCS, vol 7909, pp 92–107, DOI 10.1007/978-3-642-38883-5_11
81. Vangheluwe H, de Lara J, Mosterman PJ (2002) An introduction to multi-paradigm modelling and simulation. In: Proceedings of the AIS'2002 conference (AI, Simulation and Planning in High Autonomy Systems), pp 9–20
82. Vanherpen K (2018) A Contract-based Approach for Multi-viewpoint Consistency in the Concurrent Design of Cyber-physical Systems. PhD thesis, University of Antwerp, Belgium, Middelheimlaan 1, 2020 Antwerpen, Belgium
83. Vanherpen K, Denil J, David I, Meulenaere PD, Mosterman PJ, Törngren M, Qamar A, Vangheluwe H (2016) Ontological reasoning for consistency in the design of cyber-physical systems. In: 1st International Workshop on Cyber-Physical Production Systems, CPPS@CPSWeek 2016, IEEE, pp 1–8, DOI 10.1109/CPPS.2016.7483922
84. Voelter M (2011) Language and IDE Modularization and Composition with MPS. In: Generative and Transformational Techniques in Software Engineering IV, International Summer School, GTTSE 2011, Springer, LNCS, vol 7680, pp 383–430, DOI 10.1007/978-3-642-35992-7_11
85. Vogels W (2009) Eventually consistent. Commun ACM 52(1):40–44, DOI 10.1145/1435417.1435432
86. Völter M, Siegmund J, Berger T, Kolb B (2014) Towards User-Friendly Projectional Editors. In: Software Language Engineering - 7th International Conference, SLE 2014, Springer, LNCS, vol 8706, pp 41–61, DOI 10.1007/978-3-319-11245-9_3
87. von Hanxleden R, Lee EA, Motika C, Fuhrmann H (2012) Multi-view Modeling and Pragmatics in 2020 – Position Paper on Designing Complex Cyber-Physical Systems. In: Large-Scale Complex IT Systems. Development, Operation and Management - 17th Monterey Workshop 2012, Springer, LNCS, vol 7539, pp 209–223, DOI 10.1007/978-3-642-34059-8_11
88. Wixom BH, Todd PA (2005) A Theoretical Integration of User Satisfaction and Technology Acceptance. Inf Syst Res 16(1):85–102, DOI 10.1287/isre.1050.0042
89. Wohlin C (2014) Guidelines for snowballing in systematic literature studies and a replication in software engineering. In: 18th International Conference on Evaluation and Assessment in Software Engineering, EASE '14, ACM, pp 38:1–38:10, DOI 10.1145/2601248.2601268
90. Wohlin C, Runeson P, Höst M, Ohlsson MC, Regnell B (2012) Experimentation in Software Engineering. Springer, DOI 10.1007/978-3-642-29044-2
91. Yang G, Zhou X, Lian Y (2017) Constraint-Based Consistency Checking for Multi-View Models of Cyber-Physical System. In: 2017 IEEE International Conference on Software Quality, Reliability and Security Companion, QRS-C 2017, IEEE, pp 370–376, DOI 10.1109/QRS-C.2017.68
92. Zaheri M, Famelis M, Syriani E (2021) Towards Checking Consistency-Breaking Updates between Models and Generated Artifacts. In: ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion, MODELS 2021 Companion, IEEE, pp 400–409, DOI 10.1109/MODELS-C53483.2021.00063
93. Zhang H, Babar MA, Tell P (2011) Identifying relevant studies in software engineering. Information and Software Technology 53(6):625–637, DOI 10.1016/j.infsof.2010.12.010