Software & Systems Modeling
https://doi.org/10.1007/s10270-018-00713-w

OVERVIEW PAPER

Multi-view approaches for software and system modelling: a systematic literature review

Antonio Cicchetti[1] · Federico Ciccozzi[1] · Alfonso Pierantonio[2]

Received: 19 February 2018 / Revised: 14 October 2018 / Accepted: 20 December 2018

Abstract
Over the years, a number of approaches have been proposed on the description of systems and software in terms of multiple views represented by models. This modelling branch, so-called multi-view software and system modelling, praises a differentiated and complex scientific body of knowledge. With this study, we aimed at identifying, classifying, and evaluating existing solutions for multi-view modelling of software and systems. To this end, we conducted a systematic literature review of the existing state of the art related to the topic. More specifically, we selected and analysed 40 research studies among over 8600 entries. We defined a taxonomy for characterising solutions for multi-view modelling and applied it to the selected studies. Lastly, we analysed and discussed the data extracted from the studies. From the analysed data, we made several observations, among which: (i) there is no uniformity nor agreement in the terminology when it comes to multi-view artefact types, (ii) multi-view approaches have not been evaluated in industrial settings and (iii) there is a lack of support for semantic consistency management and the community does not appear to consider this as a priority. The study results provide an exhaustive overview of the state of the art for multi-view software and systems modelling useful for both researchers and practitioners.

Keywords Model-driven engineering · Multi-view modelling · Viewpoints · Views · Consistency

1 Introduction

Conventional wisdom on Model-Based Development (MBD) suggests that separation of concerns in modelling is crucial to manage the sheer complexity of modern software systems [1]. In fact, software engineering problems intrinsically involve many domains, each with their own experts, notations, and tooling [2]. Thus, providing these experts with views of the system that are specifically tailored to particular tasks enables superior clarity and simplicity by letting the experts focus on analysis and decision-making in their domain. Multi-view modelling favourably faces scalability from a clear separation of a number of dimensions. However, keeping these dimensions consistent is inherently difficult because of the hard problems of communication between the corresponding views [3].

The advent of Model-Driven Engineering [4], with its techniques, methods, and tools, eased the task of (partly or completely) automating the management of consistency across different dimensions [5]. The employment of uni- and bidirectional model transformations is commonplace in a wide range of applicative scenarios where changes applied to a view have to be automatically propagated to other views and related models synchronised [5]. Nonetheless, their employment is challenging as consistency restoration processes present difficulties and idiosyncrasies for both their foundations (e.g. [6]) and pragmatics (e.g. [7]). Furthermore, the separation can be based on heterogeneous criteria, including timescales, interface protocols, imposition of constraints, and any other mechanism to decompose the design problem into more tractable sub-problems [8]. For instance, in cyber-physical systems the distinction between cyber and physical

Communicated by Dr. Jeff Gray.

✉ Federico Ciccozzi
federico.ciccozzi@mdh.se

Antonio Cicchetti
antonio.cicchetti@mdh.se

Alfonso Pierantonio
alfonso.pierantonio@univaq.it

1 School of Innovation, Design and Engineering (IDT), Mälardalen University, Västerås, Sweden

2 DISIM, University of L'Aquila, L'Aquila, Italy

aspects of a system might require addressing the difficult task of translating models from different formalisms with possible semantic misalignments that are not easy to deal with.

While the potential of multi-view modelling is particularly attractive because of its conceptual simplicity and elegance, the transfer of solutions to practice may be severely hampered on account of the inherent complexity that they must encode. Thus, a systematic review of the existing literature (SLR) can be useful to drive researchers and practitioners in assessing current solutions and their applicability in real-life scenarios [9]. In fact, it permits the identification, classification, and interpretation of findings from relevant research related to a specific research question or topic by synthesising existing work according to a predefined strategy. Bruneliere et al. [10] have recently produced a survey on modelling views; differences and complementarities between this work and ours are discussed in Sect. 2.

Outline The remainder of this article is structured as follows. In Sect. 2, related work is presented and compared to our study. In Sect. 3, we introduce multi-view modelling by presenting its main characteristics. In Sect. 4, the research method is explained by defining the research questions, the selection criteria, data extraction, and finally synthesising the results. The subsequent section illustrates the publication trends, whereas Sect. 6 presents the technical characteristics of the analysed solutions. Sections 7 and 8 discuss the assessment and the limitations of the analysed research studies, respectively. Main findings and open challenges are given in Sects. 9 and 10 respectively, whereas the subsequent section discusses the threats to validity. Finally, Sect. 12 draws some conclusions.

2 Related work

Separation of concerns in general and multi-view modelling in particular are well-investigated research topics. Indeed, there exist at least a hundred research contributions dealing with multi-view modelling (see Sect. 4 for more details). In 2001, a state-of-the-art survey on multi-view modelling was devoted to approaches dealing with inconsistencies management [11]. In 2009, a systematic literature review (SLR) reported on the mechanisms tackling the specific issues of UML diagrams consistency [12], while in 2017 another SLR targeted consistency requirements in the specific case of business processes modelling [13].

Only recently, it appeared review on the state of the art of modelling views [10]. In this work, Brueneliere et al. review multi-view approaches from a perspective that can be considered orthogonal to our SLR. In fact, their work focuses specifically on what the modelling views expose in terms of features, both at design and at run-time, when it comes to specifying and synchronising each of the views as considered in isolation. Our work instead categorises multi-view solutions by means of their overall technical constituents. In this respect, we also include features related to inter-artefact consistency management, inconsistency detection/resolution, and support for synthesis.

A major difference is also the process by which we derived our way to categorise characteristics of multi-view modelling approaches, (i.e. taxonomy described in Sect. 4.3) compared to theirs. In fact, we applied a bottom-up approach, by deriving the taxonomy by means of an entirely systematic process through the keywording performed during the data extraction phase. The taxonomy is indeed itself a result of the SLR. Differently, Bruneliere et al. defined their corresponding feature model through a top-down approach, "based on our own experiences working on/with model views in the past years and on a deep study of the related state of the art". This is reflected in several aspects carried by the two different categorisation artefacts. For instance, in our taxonomy we discern approaches according to the categories extracted from the literature (i.e. {orthographic, synthetic, separate, projective, hybrid}), while Bruneliere et al. decided to differentiate in terms of, for example, what they called "metamodel-arity". The fact that these two different categorisation approaches led to some common open problems, as described in the remainder of the paper, is a confirmation of the need to solve them, no matter from which angle multi-view modelling is analysed.

3 Multi-view modelling

Separation of concerns [14] is a well-established principle by means of which it is possible to tackle the complexity of a problem. In software and system modelling, this principle entails the partition of the target system into multiple problem domains, each of which dealing with system characteristics from a specific perspective. Notably, the standard ISO/IEC/IEEE 42010:2011 [15] encodes this principle for architectural descriptions as follows: "An architecture viewpoint frames one or more concerns. A concern can be framed by more than one viewpoint." and "A view is governed by its viewpoint: the viewpoint establishes the conventions for constructing, interpreting and analysing the view to address concerns framed by that viewpoint. Viewpoint conventions can include languages, notations, model kinds, design rules, and/or modelling methods, analysis techniques and other operations on views.". By extending these definitions to multi-view modelling, domain-specific languages (DSLs) are exploited to frame a particular set of system concerns, and the models produced by each DSL represent a view of the system. In general, viewpoints are defined based on the problem at hand; however, there exist domains of application for which viewpoints are well defined.

In particular, (data-intensive) Web applications rely on the model-view-controller architectural pattern that separates the problem into the definition of data behind the application, the specification of how data are shown through the pages of the application, and the definition of how information is appropriately stored and retrieved, respectively [16]. Other examples include the "4 + 1" view model [17], the Reference Model of Open Distributed Processing (RM-ODP) [18], and the Zachman's framework [19].

As any other modelling language, viewpoints can be specified through a selection/abstraction of available concepts in an existing base language (e.g. the UML), or created *ex-novo* as a new modelling language. The former approaches are typically referred to as *projective* and *homogeneous* while the latter as *synthetic* and *heterogeneous*. A special case of projective approaches is represented by *orthographic* multi-view, which relies on the principle of orthogonal viewpoints [20].

Viewpoints typically frame overlapping concerns, thus intrinsically raising consistency management issues [15]. In this respect, the standard prescribes consistency management as a necessary constituent of any architectural framework: "an architecture description shall record any known inconsistencies across its architecture models and its views.". Moreover, the standard suggests the exploitation of correspondence rules as a possible way *"to express, record, enforce and analyse consistency"*. Quite interestingly, the standard does not dictate the immediate resolution of all existing inconsistencies; on the contrary, it notes that in practical scenarios it might be worth tolerating part of the inconsistencies until the development has reached an adequate maturity stage. Since separation of concerns encourages distributed development, it is also likely to face conflicts when concurrent modifications should be propagated to other viewpoints. In this respect, analogously to consistency management, the standard does not prescribe any specific management strategy, while distributed modelling literature proposed mechanisms to tolerate inconsistencies whenever possible, hence delaying the decision about which modification should prevail [21].

Separation of concerns has been practised in industry at length, very often by adopting development in silos for keeping the consistency among the different parts of a system. However, the sharp separation guaranteed by hardware in the past is being increasingly jeopardised by software, which is replacing most of the hardware. Indeed, the malleability of software breaks the boundaries between functionalities of the system, making them smarter but at the same time introducing relevant issues due to cross-disciplinary development. Therefore, multi-view modelling provides value not only for alleviating the complexity of the development, but also for anticipating unintended interactions between system's features. In this respect, a challenge will be represented by need of interconnecting views at different levels of abstraction. Moreover, portion of the domain knowledge is expected to be moved in the definition and characterisation of correspondence relationships.

4 Research method

To carry out this research, we followed the process depicted in Fig. 1, which can be divided into three main phases, well established when it comes to systematic literature reviews [22]: planning, conducting, and documenting.

Planning The objective of this phase was to: establish the need for a review of multi-view modelling strategies (see Sect. 2), identify the main research questions (see Sect. 4.1), and define the protocol to be followed by the research team.

Conducting In this phase, we performed the review itself by following all the steps defined in the review protocol:

- *Search and selection* we performed exhaustive forward and backward snowballing for identifying the set of potentially relevant approaches for multi-view modelling. Next, identified candidate studies were filtered in order to obtain the final list of primary studies to be considered in later activities of the review.
- *Data extraction form definition and taxonomy* we defined the set of parameters to compare and classify the primary studies based on the research questions. This was done systematically by applying a keywording process [23]. The set of parameters has been formalised into a taxonomy, described in Sect. 4.3. The taxonomy was used for data extraction and can be used for the classification of arbitrary multi-view modelling approaches.
- *Data extraction* we extracted relevant data from each primary study based on the data extraction form and the taxonomy.
- *Data synthesis and analysis* during this activity, we analysed and summarised the extracted data with the aim of answering our research questions.

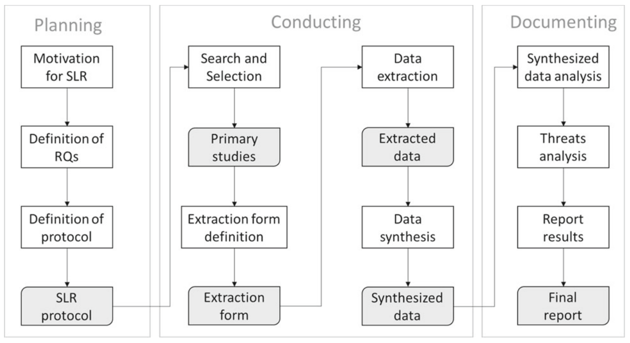


Fig. 1 Overview of the SLR process

Table 1 Goal of this study

Purpose	Identify, classify, and evaluate
Issue	The publication trends, characteristics, provided evidence, and limitations
Object	Of existing approaches for multi-view software and system modelling
Viewpoint	From a researcher's and practitioner's point of view

Documenting In this phase, we thoroughly elaborated the analysed and synthesised data and performed an accurate analysis of possible threats to validity. Eventually, we wrote the report describing the performed study.

To enable easy replication of our study, we provide a complete *replication package*,[1] which includes the complete list of selected studies, the extracted data, as well as the synthesis scripts implemented in R[2] that we defined and used for synthesising our findings.

4.1 Goal and research questions

A clear definition of the research questions is a pivotal task for systematic studies [24]. Before looking into the details of the research questions, we first formulated the *goal of our study* by using the Goal-Question-Metric perspectives [25] (see Table 1).

To achieve the goal, we needed to answer the following research questions:

- *RQ1—What are the publication trends of research studies in the context of multi-view modelling?* Objective: to classify primary studies in order to assess interest, relevant venues, and contribution types; depending on the number of primary studies, trends will be assessed over the years.
- *RQ2—What are the characteristics of the existing approaches for multi-view modelling?* Objective: to identify and classify existing approaches for multi-view modelling.
- *RQ3—What kind of evidence is used to assess existing approaches for multi-view modelling?* Objective: to find out what evidence is available when it comes to the application of multi-view modelling.
- *RQ4—What are the limitations of existing approaches for multi-view modelling?* Objective: to identify current gaps and limitations with respect to the state of the art in multi-view modelling.

1 http://www.mrtc.mdh.se/SLR_multiview.

2 https://www.r-project.org.

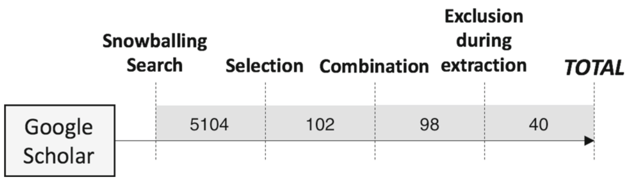


Fig. 2 Search and selection process

The results of this study are meant to be useful for both (i) researchers, to further contribute to this research area, and (ii) practitioners, to better understand existing approaches and thereby to be able to adopt the one that better suits their business goals.

4.2 Search and selection strategy

Through the search and selection process, we retrieved the set of research studies that are relevant and representative for multi-view modelling. Figure 2 shows our search and selection process.

Our search was based on exhaustive snowballing (backward and forward), according to the guidelines by Wohlin et al. [26], which we complemented with a sanity check by an expert in the research topic. In the selection phase, each potentially relevant study was evaluated against a set of selection criteria (see Sect. 4.2.1). In order to handle those cases in a cost-effective way, we used the adaptive reading depth [23], as the full-text reading of clearly excluded approaches was unnecessary. Each potentially relevant approach was classified by all three researchers either as *relevant*, *uncertain*, or *irrelevant*; any approach classified as *irrelevant* was directly excluded, whereas the others were discussed more in depth.

When going through a primary study in detail for extracting information, researchers could agree that the currently analysed study was semantically out of the scope of this study, and so it was excluded (*Exclusion during Data Extraction*).

1. Snowballing-based search In this stage, we performed a search based on exhaustive snowballing. The choice of snowballing-based search was driven by the fact that the set of basic keywords (i.e. "multi-view" and "modelling") are heavily used in other research areas outside the scope of our work (e.g. computer graphics).

The snowballing-based search workflow is depicted in Fig. 3 and is composed of three main steps, described in the following.

Step 1. Literature search for identification of the start set As starting point of our search and selection process, we manually selected a set of core studies to function as *start set* of our snowballing-based search. Studies of the start set were selected based on the authors knowledge of the targeted research domain and on a preliminary search on the avail-

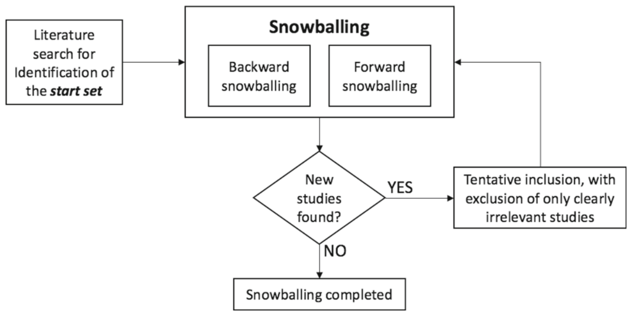

Literature search for Identification of the *start set*
Snowballing
Backward snowballing
Forward snowballing
New studies found?
YES
Tentative inclusion, with exclusion of only clearly irrelevant studies
NO
Snowballing completed

Fig. 3 Snowballing-based search

able literature about the research topic in Google Scholar (to avoid publisher bias). In order to perform search on Google Scholar, we defined a search string based on a set of keywords extracted from the research questions. More specifically, the search string[3] we used was the following:

```
(''view–based'' OR ''view based'' OR ''multiple view'' OR ''
    multiple–view'' OR ''view point'' OR ''viewpoint'' OR ''
    view–point'' OR ''multi view'' OR ''multiview'' OR ''multi–
    view'')
AND (''model*'' OR ''specification'')
AND (''software'' OR ''system'')
```

Listing 1 Search string used for search on Google Scholar

Out of 8610 results, we identified ten studies forming the start set and fulfilling the selection criteria of our SLR (see Sect. 4.2.1); they are depicted in Table 2. According to the guidelines by Wohlin et al. [26], we are confident of the goodness of our start set since:

- ten studies represent a not too small set;
- since many papers were found initially, we identified the most relevant ones by looking at: publication forum (software and system engineering in computer science) as well as number of citations;
- it covers several different publishers, years, and authors; thereby ensuring diversity;
- keywords were extracted partially from the research questions, and we included synonyms, alternative ways to address the topic as well as different spellings (British and American English). Doing so we avoided only capturing studies using a specific terminology.

The start set was used both as the starting point for the snowballing search, as well as to overall validate our search and selection strategy.

[3] When collecting the start set of studies, we realised that often "specification" was used as synonym of "model". Our decision was to be inclusive in order to avoid missing potentially relevant works, and thereby we included "specification" as possible alternative token to "model*".

Step 2. Snowballing Starting from the start set, we performed an exhaustive backward and forward snowballing search [26]. From a practical point of view, we went through each selected study (starting from the identified *start set*) by focusing on its references, and we extracted additional studies to be included in the set of relevant studies. For the forward snowballing, Google Scholar was used to obtain those studies citing the current one [26]. It is worth noting that duplicates were removed at each iteration of the snowballing activity. After the snowballing was completed, we had gathered 5104 studies.

Step 3. Tentative inclusion In both backward and forward snowballing, if a non-empty set of possible relevant studies was found, we screened each study based on its title and decide upon its tentative inclusion. Then, we only excluded studies that were clearly irrelevant, so to maximise the coverage of our snowballing. We tentatively included 115 studies.
2. Application of selection criteria Once the snowballing could not find additional studies, we applied our inclusion and exclusion criteria (see Sect. 4.2.1) on the found studies (i.e. start set plus tentative included) to take the final decision about their inclusion. Each study was analysed in two steps: first by considering its `title`, `keywords`, and `abstract`; second, if the analysis will not result in a clear decision, by `introduction`, and `conclusions` sections. To select papers objectively, all three researchers actively participated in this phase. More specifically, by following the method proposed in [27], each potentially relevant approach was classified by a selection team, composed of two researchers, as *relevant*, *uncertain*, or *irrelevant* according to the selection criteria described in the next section. Approaches classified as *irrelevant* were immediately excluded, whereas all the other approaches were discussed by the selection team. In those cases where an agreement could not be reached, a third researcher, the mediator, was called in. All three researchers played both roles, selection team member and mediator. Out of the gathered 5104, we eventually selected 102 studies during this phase.

3. Combination If a primary study was published in more than one paper (e.g. if a conference paper is extended to a journal version), only one instance was counted as a primary study. We excluded four papers by selecting their most complete version. After combination, we had 98 included studies.

4.2.1 Selection criteria

As recommended in the guidelines for performing systematic literature reviews by Kitchenham et al. [22], the selection criteria of this study were decided during the protocol definition, so to reduce the likelihood of bias. A study was (i) selected as a primary study if it satisfied *every* inclusion cri-

Table 2 Start set of core studies

ID	Title	Year
P1	Inconsistency handling in multi-perspective specifications	1994
P3	Viewpoint consistency in ODP	2000
P6	Supporting viewpoint-oriented enterprise architecture	2004
P10	Separation of non-orthogonal concerns in software architecture and design	2006
P19	Aspect-oriented multi-view modelling	2009
P23	Detecting inconsistencies in multi-view models with variability	2010
P28	A prototype implementation of an orthographic software modelling environment	2013
P31	Model-driven development of multi-view modelling tools: the MUVIEMOT approach	2014
P33	View-based model-driven software development with ModelJoin	2014
P35	EMF views: a view mechanism for integrating heterogeneous models	2015

terion, or (ii) discarded if it satisfied *any* of the exclusion criteria.

Inclusion criteria

IS1 Studies with a clear focus on some aspects of multi-view modelling.
IS2 Studies providing evidence for assessing the proposed approach.
IS3 Studies subject to peer review [28] (e.g. journal papers, book chapters, papers published as part of conference or workshop proceedings will be considered, whereas white papers will be discarded).
IS4 Studies written in English.
IS5 Studies available as full text.

Exclusion criteria

ES1 Studies that do not provide any conceptual, formal, or technical solution for the proposed approach to multi-view modelling.
ES2 Secondary and tertiary studies (e.g. systematic literature reviews, surveys, etc.).
ES3 Studies in the form of tutorial papers, short papers, poster papers, editorials, manuals, because they do not provide enough information.
ES4 Studies not focusing on modelling for system or software engineering/development/design.

4.3 Data extraction and taxonomy definition

In this phase, we created a data extraction form to collect data extracted from each primary study. The data extraction form was composed of four facets, each of which addressing a specific research question. The facet for RQ1 represents standard information such as title, list of authors, and publication details. For RQ2, RQ3, and RQ4, we performed a systematic *keywording* step for: (i) defining the parameters of each facet, and (ii) extracting data from the primary studies accordingly. Through keywording, we could effectively develop an extraction form for existing studies, and which took their characteristics into consideration [23]. During the data extraction activity, we left the data extraction form open so it could evolve as the extraction processed.

The process to define and use the data extraction form was done by all three researchers together and was composed of the following steps:

(1) *Identify keywords and concepts* We collected keywords and concepts by reading the full text of each pilot study. The extracted keywords and concepts were then combined to identify the context, nature, and contribution of research on multi-view modelling.
(2) *Cluster keywords and define parameters* We performed a clustering operation on collected keywords and concepts to organise them according to the identified characteristics. The output of this stage was the initial data extraction form containing all the identified parameters. The following steps were performed individually for each primary study.
(3) *Extract data from current study* We first extracted information about the current primary study to be analysed and then collected them based on the parameters of the data extraction form. Moreover, we collected additional information that could be relevant but that was not encompassed by the form. Collected information was refined when it did not fit naturally in the data extraction form.
(4) *Refine data extraction form* We reviewed the collected additional information that did not fit into the data extraction form to check whether:
    - collected information was not interpreted correctly; in this case, the collected information was refined;
    - the parameters of the data extraction form were not representative enough for the considered primary study; in this case, the data extraction form was refined and previously analysed primary studies were re-analysed according to the refined data extraction form.

Once we had a complete data extraction form, we analysed the results in depth, to identify relationships between parameters as well as to determine multiplicity and possible values of each parameter. This process led us to the definition of a *taxonomy* for multi-view modelling approaches, shown in Fig. 4. The taxonomy is meant as a tool for researchers and practitioners to classify, compare, and evaluate solutions for multi-view modelling. The taxonomy was defined through a systematically conducted process (i.e. the keywording activity for the data extraction form), and it was applied to all the 40 primary studies (see Table 3). For this reason, we are confident that it provides a generic means for evaluating and comparing approaches for multi-view modelling. Moreover, we provide the taxonomy as an open online form[4] (Google forms) in order to gather information concerning existing multi-view modelling approaches categorised according to our taxonomy.

The taxonomy has a tree-based hierarchical structure with two types of node:

- Parent category, representing concepts that do not assume values but are rather composed of other parent categories and/or parameters;
- Leaf parameter, representing concepts that do assume values. Leaf parameters can assume values in a closed binary set or in a multiple set. Multiple sets can be open, meaning that we see clear need for extendibility, or closed. Both in binary and multiple sets, depending on the specific case, parameters can assume single or multiple values. To avoid researcher bias and to reflect the reality as much as possible, even ranges of values as well as their characterisation have been entirely derived from the extracted data, rather than defined a priori.

In the figure, we also show which categories and parameters are mandatory for minimal categorisation of a multi-view modelling approach and reflect our inclusion criteria. For brevity, we describe the taxonomy's concepts directly when discussing the results of our study in Sects. 6–8.

4.4 Data synthesis

With this activity, we collected and summarised the extracted data with the goal of understanding, analysing, and classifying the current state of the art in the area of multi-view modelling [9, § 6.5]. We designed and executed our data synthesis activity by following the guidelines presented by Cruzes et al. [29]. Practically, we performed vertical and horizontal analysis.

Vertical analysis aims at finding trends and collecting information about each parameter of the data extraction form.

4 https://sites.google.com/view/multi-view-modelling.

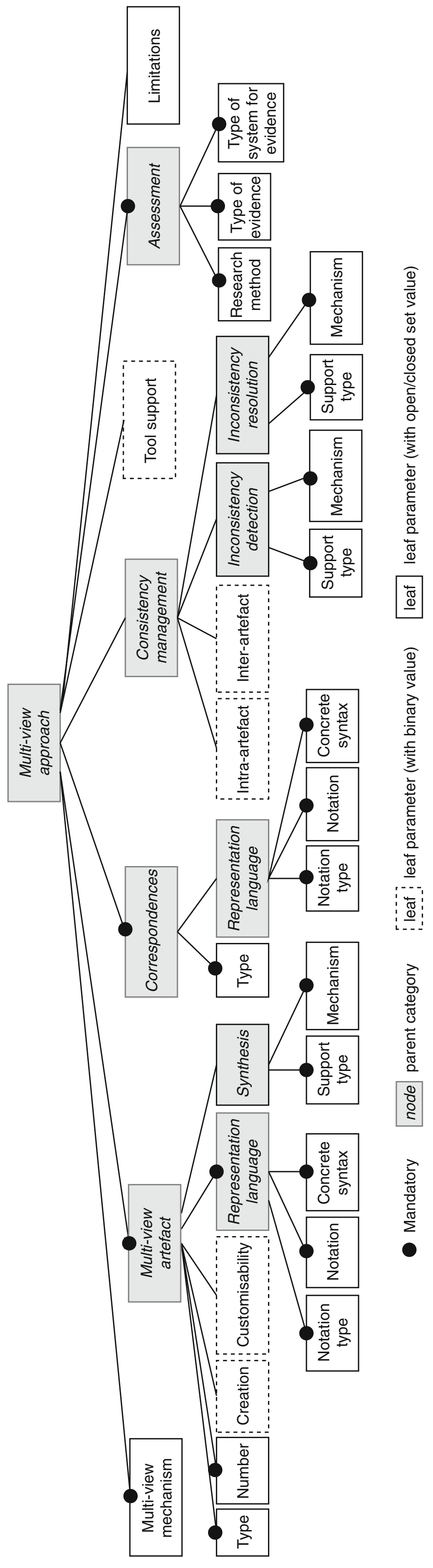


Fig. 4 Taxonomy for multi-view modelling

Table 3 List of primary research studies

ID	Title	Author	Year
P1	Inconsistency handling in multi-perspective specifications	Finkelstein et al.	1994
P2	Inconsistency management for multiple-view software development environments	Grundy et al.	1998
P3	Viewpoint consistency in ODP	Boiten et al.	2000
P4	Formal consistency of models in multi-view modelling	Bhaduri et al.	2002
P5	A methodological framework for viewpoint-oriented conceptual modelling	Andrade et al.	2004
P6	Supporting viewpoint-oriented enterprise architecture	Steen et al.	2004
P7	Generalising consistency checking between software views	Muskens et al.	2005
P8	Using Maude to write and execute ODP information viewpoint specifications	Vallecillo et al.	2005
P9	Modelling ODP correspondences using QVT	Vallecillo et al.	2006
P10	Separation of non-orthogonal concerns in software architecture and design	Giese et al.	2006
P11	Multiple viewpoint contract-based specification and design	Benveniste et al.	2007
P12	Relating software architecture views by using MDA	Cordero et al.	2007
P13	Black cats and coloured birds: what do viewpoint correspondences do?	Linington	2007
P14	Change management in multi-viewpoint systems using ASP	Eramo et al.	2008
P15	Consistency in multi-viewpoint design of enterprise information systems	Dijkman et al.	2008
P16	Well-formed rules for viewpoint correspondences specification	Vallecillo et al.	2008
P17	Consistent integration of models based on views of metamodels	Ehrig et al.	2009
P18	Modelling heterogeneous points of view with ModHel'X	Boulanger et al.	2009
P19	Aspect-oriented multi-view modelling	Kienzle et al.	2009
P20	Viewpoint synchronisation of UWE models	Vallecillo et al.	2009
P21	Realising correspondences in multi-viewpoint specifications	Vallecillo et al.	2009
P22	Multi-view modelling to support embedded systems engineering in SysML	Shah et al.	2010
P23	Detecting inconsistencies in multi-view models with variability	Lopez-Herrejon	2010
P24	View consistency in architectures for cyber-physical systems	Bhave et al.	2011
P25	Supporting incremental synchronisation in hybrid multi-view modelling	Cicchetti et al.	2011
P26	Viewpoint co-evolution through coarse-grained changes and coupled transformations	Wimmer et al.	2012
P27	Multi-perspectives on feature models	Schroeter et al.	2012
P28	A prototype implementation of an orthographic software modelling environment	Atkinson et al.	2013
P29	Synthesis of component and connector models from crosscutting structural views	Maoz et al.	2013
P30	View-centric engineering with synchronised heterogeneous models	Kramer et al.	2013
P31	Model-driven development of multi-view modelling tools: the MUVIEMOT approach	Bork et al.	2014
P32	Formalising correspondence rules for automotive architecture views	Dajsuren et al.	2014
P33	View-based model-driven software development with ModelJoin	Burger et al.	2014
P34	Integrating viewpoints in the development of mechatronic products	Törngren et al.	2014
P35	EMF views: a view mechanism for integrating heterogeneous models	Bruneliere et al.	2015
P36	Advanced local checking of global consistency in heterogeneous multimodelling	König et al.	2016
P37	Leveraging semantic Web technologies for consistency management in multi-viewpoint systems engineering	Steyskal et al.	2016
P38	Multi-view modelling and automated analysis of product line variability in systems engineering	Nesic et al.	2016
P39	Constraint-based consistency checking for multi-view models of cyber-physical system	Gang et al.	2017
P40	Multi-view refactoring of class and activity diagrams using a multi-objective evolutionary algorithm	Mansoor et al.	2017

We applied the line of argument synthesis [28], meaning that we first analysed each primary study individually to classify its main features according to each specific parameter of the data extraction form. Then, we analysed the set of studies as a whole, in order to reason about potential patterns and trends. We present the results of this vertical analysis in Sects. 5–8.

Horizontal analysis aims at identifying possible relations across different parameters of the data extraction form. We cross-tabulated and grouped extracted data and made comparisons between pairs of parameters of the data extraction form. Through contingency tables, we extracted and evaluated relevant pair-wise relations. The results of the horizontal analysis are presented in Sect. 9.

In both cases, we performed a combination of content analysis [30] (for categorising and coding approaches under broad thematic categories) and narrative synthesis [31] (for detailed explanation and interpretation of the findings coming from the content analysis).

5 Publication trends

In this section, we describe publication trends extracted from the analysed research studies. More specifically, we aim at answering the following research question:

RQ1—What are the publication trends of research studies in the context of multi-view modelling?

To answer RQ1, for each analysed research study we extracted the publication year and venue as well as the application domain of the proposed solution. The results obtained are discussed below.

5.1 Publication year

Figure 5 presents the distribution of publications on multi-view modelling over time. From the collected data, we can observe that the average number of publications is 2 per year, with a growing trend ended in 2009, when five contributions on the topic have been selected. After 2009, the trend has been more see-sawing with a new peak of four contributions in 2014. This result confirms a consistent interest of the modelling community to multi-view problems.

5.2 Publication venues

We classified the analysed research studies to assess their distribution by (i) type of publication (i.e. journal paper, conference paper, book chapter, or workshop paper) and (ii) targeted publication venues. The most common publication type is *conference* (16 of 40), which also shows a stable trend of at least one publication per year since 2009. However, publications can be considered as evenly distributed among conferences (16), workshops (13), and journals (10), while book chapters have only one occurrence.

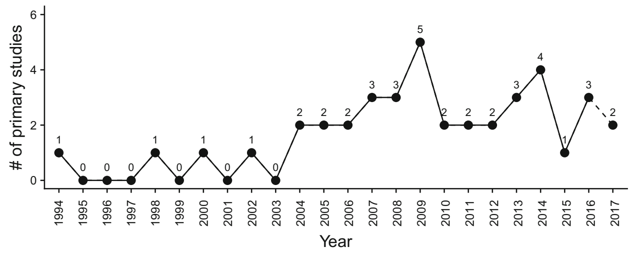


Fig. 5 Publication trend per year

Table 4 shows the publication venues that hosted more than one analysed research study. From the collected data, we can notice that while for journals the selection usually falls on general software engineering venues, for conferences and workshops there exists a prevalence of enterprise-/architecture-related symposia. Moreover, only two contributions pertain to events specifically addressing multi-view modelling research (P28, P30, published at the Workshop on View-Based, Aspect-Oriented and Orthographic Software Modelling).

6 Technical characteristics

In this section, we describe the technical characteristics of the analysed solutions for multi-view modelling as depicted in the taxonomy in Fig. 4. More specifically, it aims at answering the following research question:

RQ2—What are the characteristics of the existing approaches for multi-view modelling?

In order to provide a direct reference to the analysed primary studies and thereby guide the reader through the reported results, we provide a set of summarising tables. In the tables, the analysed technical characteristics are linked to those papers providing them. In this way, the reader can easily track down which paper provides what characteristic throughout the entire taxonomy.

6.1 Multi-view modelling mechanism

This characteristic identifies the type of mechanism for multi-view modelling provided by each of the analysed approaches. In the taxonomy, it is represented by the leaf node `Multi-view mechanism`, which can assume sin-

Table 4 Venues hosting at least two primary studies

Venue	#
Transactions on Software Engineering (TSE)	3
Workshops of Enterprise Computing Conference (EDOC-W)	3
Enterprise Computing Conference (EDOC)	2
Workshop on View-Based, Aspect-Oriented and Orthographic Software Modelling (VAO)	2
European Conference on Modelling Foundations and Applications (ECMFA)	2

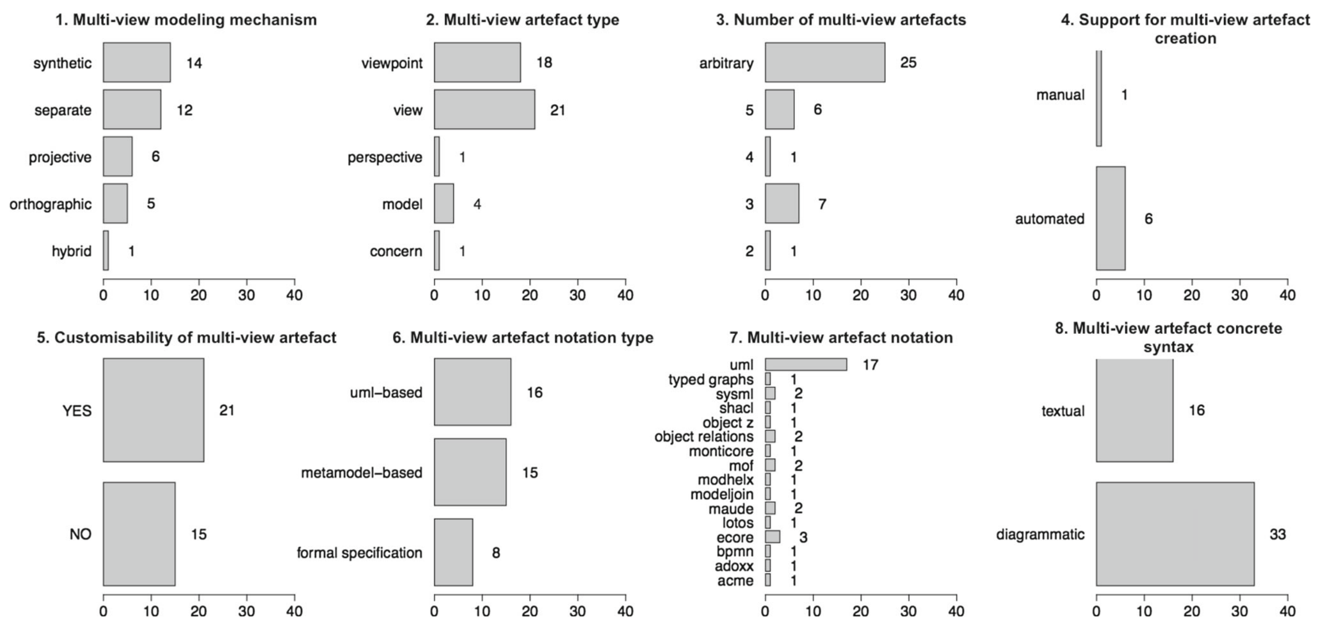


Fig. 6 Vertical analysis results related to multi-view mechanisms and artefacts

gle values in a closed set. The possible mutually exclusive values, identified during the data extraction, are:

- *orthographic* views are orthographic projections of a Single Underlying Model (SUM) [20];
- *synthetic* each view is implemented as a distinct metamodel, and the overall system is obtained as synthesis of the information carried by the different views [32];
- *separate* similarly to synthetic, each view is implemented as a distinct metamodel and represents a stand-alone perspective; differently from synthetic, no synthesis happens, and the views always stay detached [18];
- *projective* end-users are provided with virtual views made up of selected concepts coming from a single base metamodel by hiding details not relevant for the particular viewpoint taken into account [33];
- *hybrid* it is a trade-off between synthetic and projective techniques, where the definition of views is done on top of a common metamodel, and each view is represented by a subportion of the common metamodel [5].

The synthesis of the extracted data from the set of our primary studies gave the results depicted in Fig. 6(1). We can notice that the approaches providing *synthetic* (14 of 40) or *separate* (12 of 40) multi-view modelling outnumber the others, especially considering the fact that *separate* can be seen as a stricter variant (without synthesis) of *synthetic*. This can be explained by the fact that having separated views with dedicated disjoint metamodels can simplify consistency management, especially when synthesis is not the main focal point (Table 5).

Table 5 Multi-view modelling mechanism

Value	Paper ID
Synthetic	P3, P4, P10, P11, P12, P14, P15, P17, P19, P24, P29, P31, P32, P38
Separate	P1, P2, P7, P8, P9, P13, P16, P20, P21, P23, P26, P34
Projective	P6, P22, P27, P35, P37, P40
Orthographic	P28, P30, P33, P36, P39
Hybrid	P25

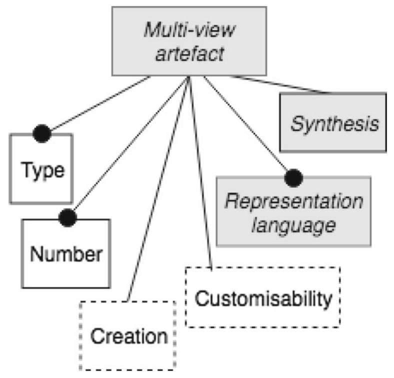


Fig. 7 Related taxonomy elements

6.2 Multi-view artefact

This set of characteristics describes the multi-view artefact that represents the focal point of a multi-view modelling approach. In the taxonomy, it is represented by the parent node `Multi-view artefact` (see Fig. 7).

Table 6 Multi-view artefact type

Value	Paper ID
Viewpoint	P1, P3, P5, P6, P8, P9, P11, P13, P14, P15, P16, P21, P26, P27, P31, P34, P35, P39
View	P2, P4, P7, P12, P15, P17, P18, P19, P20, P22, P24, P25, P27, P28, P29, P30, P31, P32, P33, P35, P38
Perspective	P27
Model	P23, P36, P37, P40
Concern	P10

6.2.1 Multi-view artefact type

This characteristic defines the type of multi-view artefact and is represented in the taxonomy by the leaf node `Multi-view artefact::Type`. It can assume multiple values in an open set. In the analysed approaches, it assumed one or more of the following values: *model*, *viewpoint*, *view*, *perspective*, and *concern*. The results of the extracted data synthesis are shown in Fig. 6(2), and they clearly show a dominance of the concepts *view* and *viewpoint*, with 21 and 18 occurrences, respectively.

It is worth noting that while most approaches lay their focus on one specific artefact type, there are cases in which they focus on multiple types, as in P35, where *viewpoints* are considered metamodels for run-time *views* (each view conforms to a specific viewpoint only). In P27, authors address views and viewpoints, but they also add the notion of *perspective* (only case among the primary studies) as "virtual view resulting from the aggregation of multiple views, where each view is dedicated to a stakeholder's concern". Moreover, *viewpoint* is defined as the "set of views to build a valid perspective", and *view* as a "model specifies relationships among stakeholder's concerns by hierarchical view refinements and multiple inheritance". The notions of *concern* and *perspective* are used in less strict manner (Table 6).

As it can already be seen from the aforementioned examples, after analysing the primary studies we could conclude that there is no uniformity nor agreement in the terminology when it comes to multi-view artefact types. For this reason, we advocate the need of a focused effort of the interested research community in standardising concepts and terminology.

6.2.2 Number of multi-view artefacts

This characteristic describes the number of multi-view artefacts. It is represented in the taxonomy by the leaf node `Multi-view artefact::Number` and can assume single integer values > 0 in case a fixed number is specified, or 'arbitrary' in case the number is flexible. In Fig. 6(3), we can see that this number can be fixed (15 of 40) or arbitrary (25 of 40). In the former case, we have encountered approaches defining 2, 3, 4, or 5 fixed multi-view artefacts. Approaches defining five multi-view artefacts are those reflecting the RM-ODP standard [18], while the others are not correlated in any particular way (Table 7).

Table 7 Number of multi-view artefacts

Value	Paper ID
Arbitrary	P1, P2, P5, P6, P7, P10, P13, P15, P17, P18, P22, P23, P25, P26, P27, P29, P30, P31, P33, P34, P35, P36, P37, P38, P40
5	P3, P8, P9, P14, P16, P21
4	P24
3	P4, P11, P12, P19, P20, P28, P39
2	P32

Table 8 Support for multi-view artefact creation

Value	Paper ID
Manual	P17
Automated	P1, P6, P25, P30, P33, P35

6.2.3 Support for multi-view artefact creation

It is represented in the taxonomy by the leaf node `Multi-view artefact::Creation`, and it assumes a binary value (*YES* or *NO*). Few approaches provide the possibility to create multi-view artefacts, 7 of 40 [see Fig. 6(4)]. Out of the few approaches provided, the majority provides automated support for the creation activity in terms of, for example, wizards in the modelling environment (P25) (Table 8).

6.2.4 Customisability of multi-view artefacts

With this characteristic, we describe whether an approach provides support for customising the set of multi-view artefacts in terms of its composition (e.g. add/remove a view) and the single artefacts in terms of their specification (e.g. change a view).

In general, these are desirable features from a modelling perspective, since they allow to tailor the set of views to the problem at hand. However, artefacts customisation introduces new problems to be managed, as, for example, co-adaptation of consistency management and synthesis mechanisms. This characteristic is represented in the taxonomy by the leaf node `Multi-view artefact::Customisability` and can assume a binary value (*YES* or *NO*). We were not able to extract values for all primary studies since some (4 of 40) did not explicitly report on this specific characteristics. In 21 cases, approaches

Table 9 Customisability of multi-view artefacts

Value	Paper ID
Yes	P1, P2, P6, P7, P10, P11, P13, P15, P17, P18, P22, P25, P27, P29, P30, P31, P33, P35, P36, P37, P38
No	P3, P4, P5, P8, P9, P12, P14, P19, P20, P21, P24, P28, P32, P39, P40

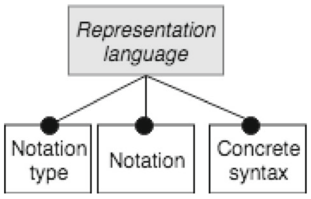


Fig. 8 Related taxonomy elements

provide the possibility to customise the set of multi-view artefacts [see Fig. 6(5)] (Table 9).

6.2.5 Multi-view artefact representation language

Multi-view artefacts are described via a specific *Representation language*, which in turn is composed of: *Notation type*, *Notation*, and *Concrete syntax*. In this section, we describe the values identified for each of those characteristics. This category is represented in the taxonomy by the parent node `Multi-view artefact::Representation language` (see Fig. 8).

Multi-view artefact notation type
It represents the notation types used for defining multi-view artefacts of any of the aforementioned types, both implicit and explicit, and the corresponding instances found in the analysis [see Fig. 6(6)]. It is represented in the taxonomy by the leaf node `Multi-view artefact::Representation language::Notation type` and can assume multiple values in the following closed set:

- *uml-based* multi-view artefacts like views are defined through standard UML diagrams and/or specific profiles [34]. An example is orthographic modelling in P28 where multiple UML diagrams represent different projections of a common SUM.
- *metamodel-based* a specific metamodel is defined to represent multi-view artefacts. An example is the MUVIEMOT approach in P31 which is based on the Adoxx metamodel.[5]
- *formal specification* a formal language is used to define multi-view artefacts. For instance in P3, authors exploit formal languages like Object Z [35] and Lotos [36].

[5] https://www.adoxx.org/live/web/home.

Table 10 Multi-view artefact notation

Type	Language	Paper ID
UML-based	UML	P4, P6, P7, P9, P10, P13, P14, P16, P19, P20, P21, P23, P28, P34, P36, P40
	Maude	P26
Formal specification	Typed graphs	P36
	Maude	P8, P26
	LOTOS	P3
	Object Z	P3
	N/A	P11, P17, P27, P38, P39
N/A	Object relations	P2
Metamodel-based	Object relations	P1
	SysML	P22, P32
	SHACL	P37
	Monticore	P29
	MOF	P12, P15
	ModHel'X	P18
	ModelJoin	P33
	Ecore	P23, P25, P35
	BPMN	P6
	ADOxx	P31
	ACME	P24

It is worth noting that *uml-based* and *metamodel-based* are the most common values, with 16 and 15 occurrences, respectively. Moreover, values are not mutually exclusive and can be used in combination. An example is P6 where UML (*uml-based*) and BPMN [37] (*metamodel-based*) are used in combination.

Multi-view artefact notation
Apart from *uml-based* notation type, which includes any UML-based notation (including profiles), the other two types can be materialised through multiple notations [see Fig. 6(7)]. It is represented in the taxonomy by the leaf node `Multi-view artefact::Representation language::Notation` and can assume multiple values in an open set. Examples are *Shacl*, *Monticore* or *Ecore*, for *metamodel-based*, *Lotos* or *Object Z*, for *formal specification*. The UML standard notation dominates the category with 17 occurrences, while the remaining 23 are scattered across 15 different notations (Table 10).

Multi-view artefact concrete syntax
Another important aspect of a *Representation language* is the concrete syntax(es) used for modelling purposes [see Fig. 6(8)]. It is represented in the taxonomy by the leaf node `Multi-view artefact::Representation language::Concrete syntax` and can assume multiple values in the following open set:

Table 11 Multi-view artefact concrete syntax

Value	Paper ID
Textual	P3, P7, P8, P9, P16, P17, P20, P21, P26, P28, P29, P30, P33, P34, P39, P40
Diagrammatic	P1, P2, P4, P6, P7, P9, P10, P12, P13, P14, P15, P1, P17, P18, P19, P20, P21, P22, P23, P24, P25, P26, P27, P28, P29, P30, P31, P32, P34, P35, P36, P37, P38, P40

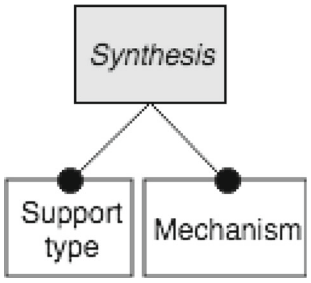


Fig. 9 Related taxonomy elements

- *textual* artefacts are specified and/or represented in a textual format;
- *diagrammatic* artefacts are visualised and/or manipulated in a graphical format.

In order to better explain these categories, we can mention that, for example, by using UML the artefacts have a diagrammatic concrete syntax; on the contrary, Maude artefacts are based on a textual concrete syntax. In some cases, concrete syntax combines graphics and text to improve readability and scalability of the notations: notably, the Query/View/Transformation-Relation [38] standard combines graphics and Object Constraint Language (OCL) [39]-like clauses to define transformation rules. The value *diagrammatic* has more than twice the occurrences (33 of 40) of *textual*. It is interesting to notice that in nine cases, approaches use both (Table 11).

6.2.6 Support for synthesis of multi-view artefacts

This category describes whether an approach provides support for the *Synthesis* of information coming from different multi-view artefacts and categorises the synthesis approach, when available. It is represented in the taxonomy by the parent node `Multi-view artefact::Synthesis` (see Fig. 9).

Support type

The type of synthesis support is characterised by this category and is represented in the taxonomy by the leaf node `Multi-view artefact::Synthesis::Support type` and can assume a value in the following binary set:

- *manual* the synthesis is (partially) performed by hand, based on the correspondences between multi-view artefacts. User intervention is required to reconcile pending inconsistencies and/or resolve complex synthesis scenarios.
- *automatic* the multi-view framework provides algorithms able to automatically synthesise the information coming from different multi-view artefacts.

Synthesis mechanism

Automated synthesis support is distinguished further by means of the exploited implementation mechanism. It is represented in the taxonomy by the leaf node `Multi-view artefact::Synthesis::Mechanism` and can assume a single value in the following open set:

- *operational* the approach provides algorithms that based on the correspondences derive the synthesis of two or more multi-view artefacts;
- *constraint solver* the synthesis is given in terms of solutions of a constraint-solving problem over multi-view artefacts;
- *aspect weaving* the approach relies on the weaving operation as defined in the aspect-oriented community.

The results collected from the data extraction for these characteristics are shown in Fig. 10. In most cases, there is not support for synthesis (23 of 40). With respect to the 12 providing synthesis support, ten of them are automated, and the more common way of implementing synthesis mechanisms is by means of operational approaches (6 of 10). It is worth noting that in two cases it was not possible to characterise the automated approaches in one of the identified synthesis mechanisms (Table 12).

6.3 Correspondences between multi-view artefacts

This category describes whether the analysed approaches define correspondences between multi-view artefacts. It is represented in the taxonomy by the parent node `Correspondences` (see Fig. 11).

6.3.1 Type

This characteristic defines the instances of correspondence supertypes that is the concrete technical solutions to specify correspondences between multi-view artefacts. It is represented in the taxonomy by the leaf node

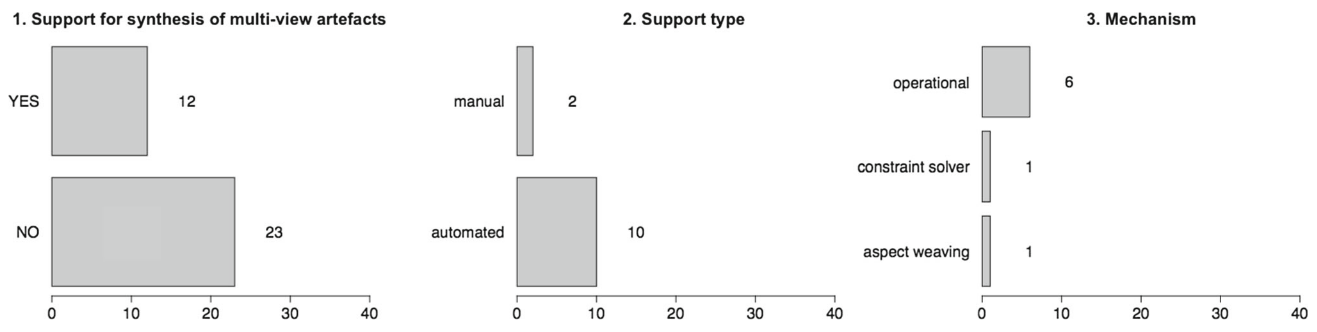


Fig. 10 Vertical analysis results related to synthesis of multi-view artefacts

Table 12 Synthesis type and mechanism

Type	Mechanism	Paper ID
Automated	Operational	P10, P12, P22, P25, P28, P30
	Constraint solver	P29
	Aspect weaving	P19
	N/A	P3, P31
Manual	–	P5, P11

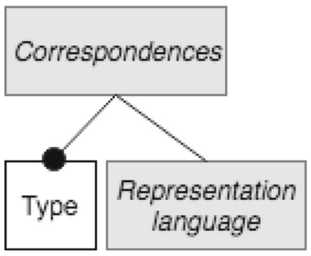


Fig. 11 Related taxonomy elements

`Correspondences::Type` and can assume a single value in the following open set:

- *relation* the correspondences between the multi-view artefacts are given in terms of relations that must hold between the corresponding elements;
- *model* correspondences are defined by means of dedicated artefacts, possibly based on a different language with respect to the one used for the views;
- *implicit* end-users do not provide correspondences explicitly. The correspondences are indirectly established through the use of conventions like naming, identifiers, and so forth;
- *constraint* this case is a kind of hybrid correspondence definition technique. In fact, it is based on the specification of mutual constraints defined over viewpoints that indirectly introduce correspondences between the corresponding elements.

The values obtained in the analysis are synthesised in Fig. 12(1): the majority of the approaches (31 of 40) rely on explicit correspondences, while 7 of 40 are based on implicit correspondences. It is worth noting that for the remaining two contributions, it was impossible to clearly establish whether they explicitly defined correspondences. The values extracted from the analysis show an even distribution of the different correspondence definition techniques, with a little prevalence of relations (13 of the 31 explicit cases). Note that in two cases, it was not possible to distinguish between relation/constraint and constraint/model categories, and as a consequence the total number of contributions involving explicit types is greater than the corresponding value for their supertype. More in general, at this level the distinction between relation, model, and constraint might appear blurry, since all correspondences are a kind of relation. However, the distinction becomes clearer when considering also the impact of the adopted correspondence type with respect to related consistency support (see Sect. 6.4) (Table 13).

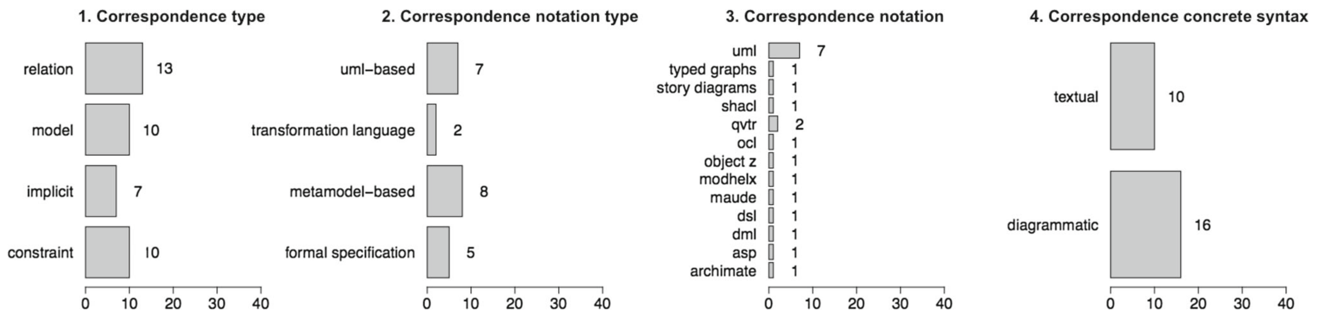


Fig. 12 Vertical analysis results related to correspondences definition

Table 13 Correspondences supertype and type

Supertype	Type	Paper ID
Explicit	Relation	P1, P2, P7, P9, P12, P15, P18, P21, P22, P25, P28, P32, P33
	Model	P6, P13, P20, P24, P26, P31, P34, P35, P36, P37
	Constraint	P3, P7, P10, P11, P14, P16, P29, P30, P34, P39
Implicit	–	P4, P17, P19, P23, P27, P38, P40

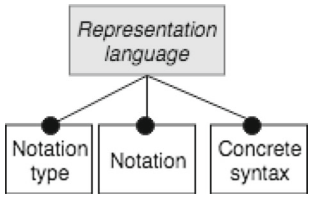


Fig. 13 Related taxonomy elements

Table 14 Correspondences representation language

Type	Notation	Paper ID
Formal specification	N/A	P11, P17
	OCL	P15
	Object Z	P3
	Typed graphs	P36
Transf. language	QVTr	P9, P12
UML-based	UML	P13, P15, P16, P19, P20, P21, P32
Metamodel-based	ASP	P14
	ArchiMate	P6
	ModHel'x	P18
	story diagrams	P22
	Maude	P26
	DSL	P30
	DML	P34
	SHACL	P37

6.3.2 Representation language

Explicit correspondences are described via a specific *Representation language*, which in turn is composed of: *Notation type*, *Notation*, and *Concrete syntax*. In this section, we describe the values identified for each of those characteristics. This category is represented in the taxonomy by the parent node `Correspondences::Representation language` (see Fig. 13).

Correspondences notation type
This category identifies the notation type used for explicit correspondences [see Fig. 12(2)]. It is represented in the taxonomy by the leaf node `Correspondences:: Representation language:: Notation type` and can assume multiple values in the following closed set:

- *uml-based* correspondences are defined by exploiting standard UML diagrams and/or specific profiles. Typical instances are represented by the implicit consistency relationships embedded in UML diagrams and explicit consistency links represented in UML;
- *transformation language* correspondences are given in terms of transformation language rules. Rules conforming to the standard for QVT-R [38] for defining explicit relations between viewpoints are instances of this type;
- *metamodel-based* a specific metamodel is defined to represent the correspondences between viewpoints, relations, or constraints. Notably, Archimate and Shacl define correspondences as models, Story Diagrams and Modhel'X as relations, ASP and *DML* as constraints;
- *formal specification* a formal approach is exploited to define correspondences as models, relations, or constraints. Instances of this type are Typed Graphs, OCL, and Object Z, respectively.

One noticeable result of the extracted values is that nearly half of the considered works do not provide enough details about the notation exploited for the correspondences (18 of 40). When specified, the most used notations rely on *uml-based* or *metamodel-based* notations (8 and 7 of 40, respectively). The remaining works are based on *formal specification* (5 of 40) or on a *transformation language* (2 of 40). Interestingly, the latter seems to be rare; this is surprising when considering the opportunity of having an executable specification of correspondences that could be used for automatically resolving inconsistencies.

Correspondences notation
Apart from *uml-based* notation type, which conceives any UML-based notation (including profiles), the other three types can be materialised through multiple notations [see Fig. 12(3)]. This is represented in the taxonomy by the leaf node `Multi-view artefact::Representation language::Notation` and can assume multiple values in an open set. An example of *transformation language* is *QVT-R*, where transformation rules are used to define explicit relations between multi-view artefacts. For the *metamodel-based* category, examples are *Archimate* and *Shacl*, which define correspondences as models, *Story Diagrams* and *Modhel'X* as relations, *ASP* and *DML* as constraints. For *formal specification*, examples are *Typed Graphs*, *OCL*, and *Object Z* (Table 14).

Correspondences concrete syntax
This characteristic defines how the correspondences are rendered to the end-users that is their concrete syntax. It is represented in the taxonomy by the leaf node `Correspond`

ences::Representation language::Concrete syntax and can assume multiple values in the following open set:

- *textual* correspondences are specified and/or represented in a textual format;
- *diagrammatic* correspondences are visualised and/or manipulated in a graphical format.

The results from the data extraction are shown in Fig. 12(4): the diagrammatic form is used more often than the textual one (16 vs 10 of 40, respectively). It is worth noting that in several cases the concrete syntax is a mixture of diagrams and text (e.g. UML in combination with OCL), so there exist publications that pertain to both the categories.

6.4 Support for consistency management

Section 3 introduces the relevance of monitoring consistency each time different viewpoints framed overlapping concerns. Such a relevance is confirmed also for multi-view: our analysis in Fig. 12(1) shows that the majority of the approaches provide explicit ways of defining correspondences between views. However, by recalling the goals of consistency rules described in the software architecture standard [15], while these results can help in understanding if it is possible to *express* and *record* consistency, the existence of correspondences does not provide any details about the capabilities to *analyse* and *enforce* consistency. In this respect, automated consistency management can be considered as a feature suggested by the pragmatics of model-driven development [40]. Indeed, correspondences are increasingly expressed by means of executable notations, including transformations of graph grammars, that can automate the detection and resolution of inconsistencies. This trend is confirmed by the included papers: as shown in Fig. 16(1), the large majority of approaches provide some form of consistency management (37 of 40). Therefore, in the following we present a set of characteristics to better describe the kind of support provided by the included works. This is represented in the taxonomy by the parent node Consistency management (see Fig. 14).

6.4.1 Intra-artefact consistency

This category specifies whether the considered approach provides support for consistency for each multi-view artefact considered in isolation. It is represented in the taxonomy by the leaf node Consistency management::Intra-artefact, and it assumes a binary value (*YES* or *NO*). It is worth noting that even if this feature is typically not specific to multi-view support, it is relevant due to potential side effects of the changes triggered by multi-view artefact correspondences. In fact, some concerns might require an appropriate propagation inside a multi-view artefact when they are subject to modifications. In this respect, it is interesting to notice that only one-fourth of the approaches provides this feature [11 of 40, see Fig. 16(2)]. Nevertheless, in the analysed papers it was not explicitly described whether and how they manage the potential impact of intra-artefact changes on the consistency across multi-view artefacts (e.g. views).

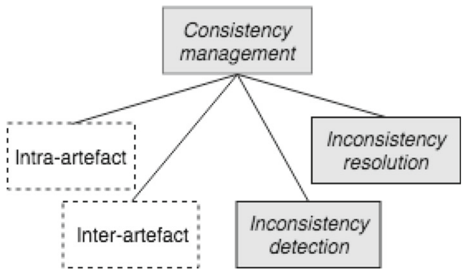


Fig. 14 Related taxonomy elements

Table 15 Type of consistency management

Intra-artefact	Inter-artefact	Paper ID
Yes	Yes	P1, P2, P4, P7, P19, P20, P23, P35, P36, P39, P40
No	Yes	P3, P5, P9, P10, P11, P12, P13, P14, P15, P16, P17, P21, P22, P24, P25, P26, P27, P28, P29, P30, P31, P32, P34, P37, P38
Yes	No	P33
No	No	P6, P8, P18

6.4.2 Inter-artefact consistency

This characteristic identifies whether or not there exists support for consistency between multi-view artefacts. It is represented in the taxonomy by the leaf node Consistency management::Inter-artefact, and it assumes a binary value (*YES* or *NO*). As depicted in Fig. 16(3), the considered approaches predominantly manage inter-artefact consistency (36 of 40). The following set of characteristics provides more details about the support of inter-artefact consistency, and in particular about detection and resolution mechanisms (Table 15).

6.4.3 Inconsistency detection

This set of characteristics identifies more details about inter-artefact inconsistency detection. A specific detection mechanism is usually provided in all those cases where correspondences do not imperatively prescribe or entail a resolution strategy. It is represented in the taxonomy by the parent node

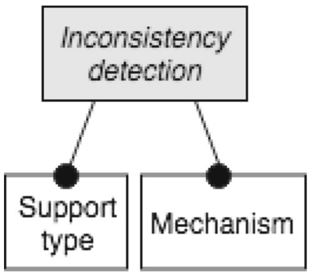


Fig. 15 Related taxonomy elements

`Consistency management::Inconsistency detection` (see Fig. 15).

Inconsistency detection type
This category specifies the type of detection mechanism exploited to discover inconsistencies. It is represented in the taxonomy by the leaf node `Consistency management ::Inconsistency detection::Support type`, and it assumes a binary value as:

- *manual* correspondences between viewpoints are analysed by hand to check for inconsistencies;
- *automated* there exists an automated mechanism able to interpret correspondences and detect inconsistencies.

Figure 16(4) illustrates the results from data extraction: there exists only one approach proposing manual detection of inconsistencies, while the remaining ones providing inconsistency detection do it in an automated manner. Note that, in eight cases, no information was given on whether the detection process was manual or automated.

Inconsistency detection mechanism
This category identifies further the mechanisms exploited in automated detection of inconsistencies. It is represented in the taxonomy by the leaf node `Consistency management::Inconsistency detection:: Mechanism`, and it assumes a single value in the following open set:

- *operational* operational semantics (e.g. algorithms) are exploited to detect inconsistencies;
- *operation recorder* artefact modifications are logged by means of an operation recorder. Inconsistencies are detected by analysing the changes that have been registered in the logs.
- *formal methods* the detection of inconsistencies is performed by executing some form of formal verification (e.g. interpreting expressions over artefacts or performing deductive proofs);
- *constraint solver* inconsistencies are an indirect result of a constraint-solving problem. Notably, a problem with no solutions is a typical indicator of inconsistencies among artefacts.

Table 16 Inconsistency detection support

Type	Mechanism	Paper ID
Automated	Operational	P4, P9, P15, P20, P21, P26, P27, P32, P34, P36, P37, P40
	Operation recorder	P2, P31, P33, P35
	Formal methods	P1, P3, P7, P10, P17, P24, P30, P38, P39
	Constraint solver	P14, P29
	N/A	P19, P22, P23
Manual	–	P5

Please note that constraint solvers are normally underpinned by solid theoretical foundations and as such they may be considered among formal methods. However, there are considerable differences in the way these two families of techniques are employed. In more details, (a) formal methods normally assume that artefacts are written in a formal notation, while solvers can be employed with any syntactical format (e.g. Ecore) provided that there exists a mapping targeting the solver notation as, for instance, happens with JTL [7] that translates models and correspondences into logic programs in answer set programming [41]; (b) the adoption of solvers implies that correspondences can be resolved (often non-deterministically) in a fully automated way, while formal methods normally provide the modellers with the mathematical means for performing, for instance, deductive proofs as it can happen like when verifying if a correspondence between two specifications written in Object Z [35] exists. Because of such differences in their pragmatics, we preferred to keep them separated.

The results collected from the data extraction are shown in Fig. 16(5): the majority of the contributions propose operational inconsistency detection mechanisms (12 of 40) followed by formal methods (7 of 40) (Table 16).

6.4.4 Inconsistency resolution

This set of characteristics identifies further details about inter-artefact inconsistency resolution. It is represented in the taxonomy by the parent node `Consistency management::Inconsistency resolution` (see Fig. 17).

It is worth noting that in general the selected approaches conceive inconsistency resolution as a synonym of change propagation/synchronisation among views; therefore, the categories represented in the taxonomy and discussed in the following should be read by taking into account such specific task. Further discussion about open issues related to inconsistency management in a broader perspective is provided in Sect. 10.

Inconsistency resolution type
This category specifies the type of inconsistency resolution mechanism proposed in the papers. It is represented in the taxonomy by the leaf node `Consistency management:: Inconsistency resolution:: Support type`, and it assumes a multiple value in the following closed set:

- *manual* the resolution is delegated to the end-user;
- *automated* the approach provides automation support to reconcile the artefacts.

Figure 16(6) depicts the data extracted from the analysis: most of the approaches supporting inconsistency resolution provide forms of automation (16 of 19). It is worth noting that these categories are not mutually exclusive; therefore, some approaches might provide automated resolution skeletons embedding requests for end-user inputs to fine-tune reconciliation strategies.

Inconsistency resolution mechanism

This category details further the mechanisms exploited for the resolution of inconsistencies. It is represented in the taxonomy by the leaf node `Consistency management:: Inconsistency resolution::Mechanism`, and it assumes a single value in the following open set:

- *proxy* modifications coming from multi-view artefacts are collected in a common place and projected towards the existing views. In this respect, the resolution mechanism acts as a proxy that broadcasts modifications from the commonplace;
- *operational* the resolution strategy is defined in an operational way;
- *constraint solver* the inconsistencies are reconciled as the solution of a constraint-solving problem.

The results collected from the data extraction are illustrated in Fig. 16(7): most of the resolution approaches rely on operational mechanisms, that is, on strategies defined in an imperative way (12 of 40). Techniques based on constraint solving and proxies are less (2 and 1 of 40, respectively).

It is interesting to notice that in general the *time* of resolution is given less relevance: in particular, a limited number of approaches do specify that inconsistency resolution is immediately triggered after each change, while for others it is clear that a completion of the editing phase is needed in order to be able to run inconsistency analysis and resolution (e.g. those exploiting a translation towards another semantic domain in which it is possible to realise inconsistency management). Then, there exist approaches that could work in both the previously mentioned ways (e.g. those based on synchronisation algorithms or constraint solving), but that do not specify further when the synchronisation mechanism is supposed to be run (Table 17).

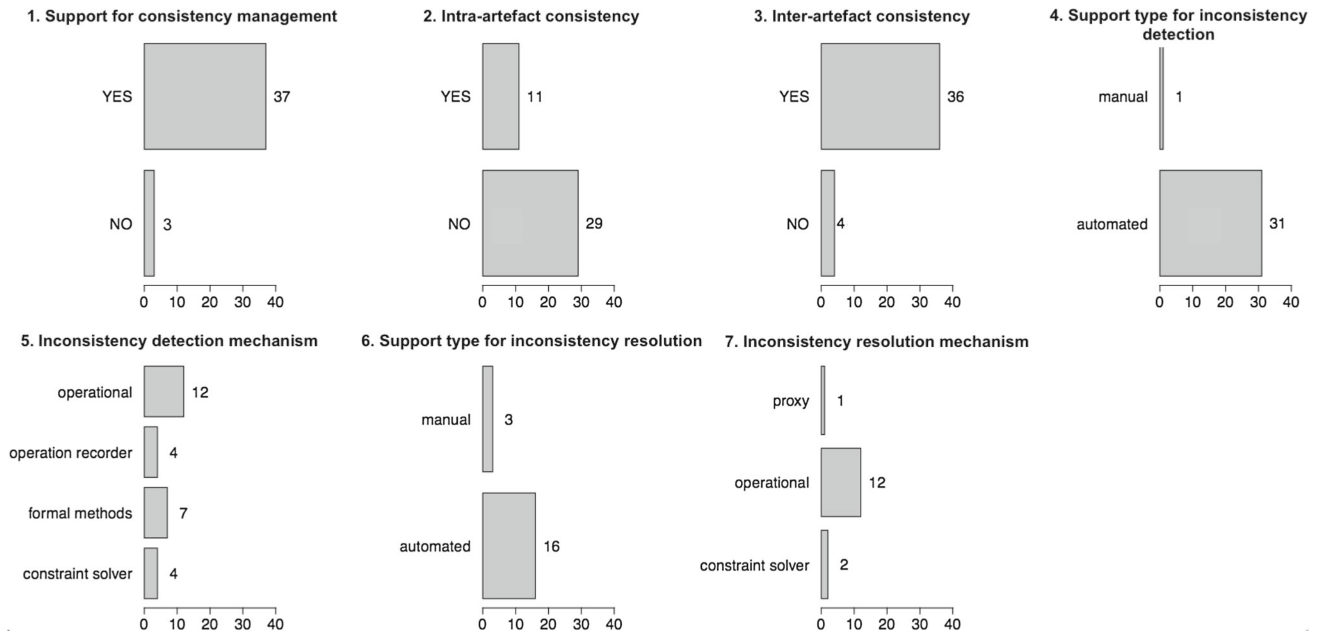


Fig. 16 Vertical analysis results related to consistency management

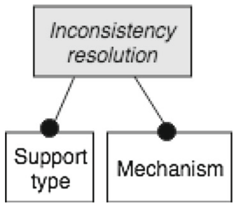


Fig. 17 Related taxonomy elements

Table 17 Inconsistency resolution support

Type	Mechanism	Paper ID
Automated	Proxy	P35
	Operational	P12, P20, P21, P25, P26, P28, P30, P31, P33, P34, P40
	Constraint solver	P14, P23
	N/A	P2
Manual	–	P1, P2, P5

6.5 Tool support

This characteristic shows whether the analysed approaches provide support in terms of tooling. It is represented in the taxonomy by the leaf node `Tool support`, and it assumes a binary value (*YES* or *NO*). Our analysis shows that the majority (28 of 40) of the approaches provide such a support (Table 18).

Highlights

RQ2. What are the characteristics of the existing approaches for multi-view modelling?

- There is no uniformity nor agreement in the terminology when it comes to multi-view artefact types;
- Only very few approaches provide the possibility to create multi-view artefacts;
- UML-based and metamodel-based are the most common multi-view artefact notation types;
- Multi-view artefacts synthesis is not provided in more than 50% of the analysed approaches;
- Only two approaches rely on transformation languages for describing correspondences;
- Most inconsistency resolution approaches rely on operational strategies defined imperatively.

Table 18 Tool support

Value	Paper ID
Yes	P1, P2, P3, P6, P7, P8, P10, P12, P14, P15, P18, P19, P20, P21, P22, P23, P24, P25, P26, P27, P28, P29, P30, P31, P32, P33, P35, P37
No	P4, P5, P11, P13, P17, P36, P38, P39

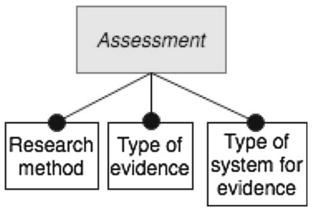


Fig. 18 Related taxonomy elements

7 Provided assessment

In this section, we describe how the analysed research studies have been assessed and thereby which the evidence motivating the potential adoption of the specific solutions is. More specifically, we answer the following research question:

RQ3—What is the evidence that motivates the adoption of existing solutions for multi-view modelling?

This category is represented in the taxonomy by the parent node `Assessment` (see Fig. 18).

7.1 Applied research method

Through this parameter, we categorise analysed solutions according to the type of applied research method they used for assessment. It is represented in the taxonomy by the leaf node `Assessment::Type`, and it assumes a binary value in the set defined in [42, fig. 19]: *validation* and *evaluation*. *Validation* is performed in laboratory, whereas *evaluation* is run in an industrial context. The latter usually provides stronger evidence about the applicability of the solution in practice. The results of our analysis undoubtedly show that evidence in the field is fairly shallow, since none of the analysed solutions is evaluated in an industrial context.

7.2 Type of evidence

This characteristic describes the type of evidence (e.g. empirical study) used to validate/evaluate the analysed solutions [see Fig. 19(1)]. It is represented in the taxonomy by the leaf node `Assessment::Evidence type`, and it assumes a single value in the following closed set:

- *example* 1 in-house example;
- *set of examples* several in-house (not industrial) examples, several runs and comparisons of one example,

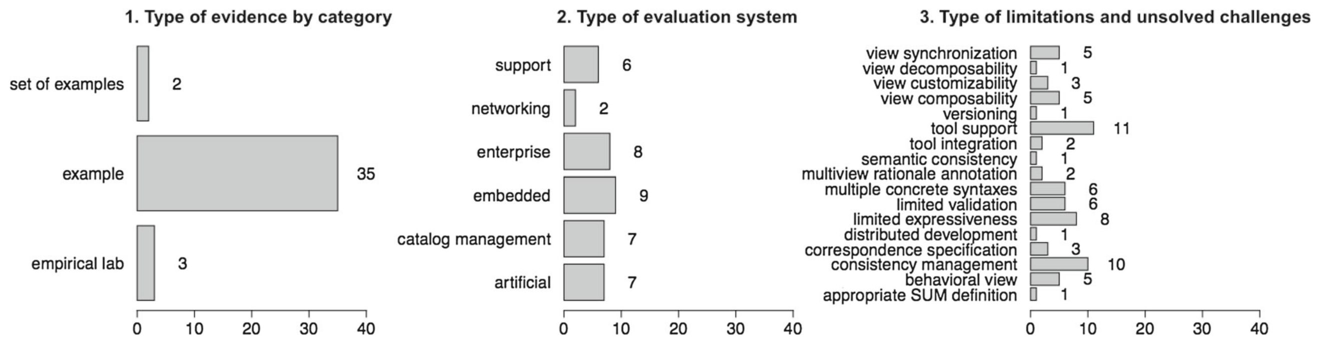


Fig. 19 Vertical analysis results related to evaluation and limitations

several models of different sizes for the same application;

- *empirical lab* case studies, controlled experiments, and empirical evaluations in laboratory;
- *industrial example* example coming from industry and performed in laboratory;
- *set of industrial examples* several industrial examples (similarly to *set of examples*, both industrial);
- *empirical industry* case studies, controlled experiments, and empirical evaluations either in industry or in real-world scenarios;
- *industrial evaluation* evaluation performed by industrial actors, solution used in industry.

The most common means to provide evidence is through the application of the solution to an *example*, thus showing the fact that documented approaches for multi-view modelling have not been evaluated in industrial settings yet.

7.3 Type of system for evidence

This parameter describes the type of system by the analysed solutions for providing evidence [see Fig. 19(2)]. It is represented in the taxonomy by the leaf node `Assessment::System type`, and it assumes a single value in the following open set:

- *embedded* embedded application;
- *enterprise* enterprise application;
- *data management* application for management of data, e.g. online booking system, library;
- *artificial* application artificially built for validation purposes and not reflecting any specific domain;
- *support* application providing some kind of support, for example, development tool, test bed.

While there is a pretty even distribution across the following types, the most common ones are *embedded* and *enterprise* applications. This is not surprising, given the heterogenous nature of certain embedded systems, such as cyber-physical systems, and the efforts in standardising models for multi-view enterprise applications, such as RM-ODP.

Highlights

RQ3. What is the evidence that motivates the adoption of existing solutions for multi-view modelling?

- No approaches have been evaluated in industrial settings, although some of them have been applied to use cases coming from industry.
- No approaches exploit industrial applications to provide evidence; the most used means is in-house examples;
- Embedded and enterprise applications are the most common types of validation system.

8 Identified limitations

In this section, we describe the limitations (and thereby need for improvement) of the solutions as they were reported in the analysed research studies. More specifically, we aim at answering the following research question:

RQ4—What are the limitations of existing solutions for multi-view modelling?

Among the various types of limitations and unsolved challenges [see Fig. 19(3)], the most common is *tool support*, for automated mechanisms related to multi-view artefact creation and synthesis as well as consistency management. Other relevant identified limitations are the provided features for *consistency management* and the *limited expressiveness* in terms of, for example, multi-view artefacts aggregation operations (EMF Views in P35). It is surprising that very few approaches mention limitations related to lack of *versioning* support for multi-view artefacts and *distributed development*, since they represent crucial aspects of industrial development. At least as surprising is the fact that only one approach mentions the need for *semantic consistency*, especially since several approaches only tackle syntactical consistency.

Highlights

RQ4. What are the limitations of existing solutions for multi-view modelling?

- Most common limitations are related to tool support, consistency management, and limited expressiveness to describe multi-view artefacts and the way they interact;
- Lack of support for (meta)model versioning and distributed development related to multi-view modelling is only mentioned in two approaches;
- Lack of support for semantic consistency management is not regarded as a priority by the community.

9 Horizontal analysis findings

The horizontal analysis aims at investigating possible correlations between taxonomy parameters. We produced a contingency table [43] for all possible pairs of parameters and analysed them to identify those showing interesting or unexpected results. In the following, we describe those of interest; however, all generated contingency tables can be found in the replication package.

9.1 Multi-view modelling mechanism versus multi-view artefact customisability

Looking at the correlation between these two characteristics, we can highlight one fact regarding the three most common mechanisms (i.e. *synthetic*, *separate*, *projective*). While for *synthetic* and *separate* approaches exactly 50% of the analysed approaches provide customisability of the set of multi-view artefacts, in the case of *projective*, over 80% of the analysed approaches provide customisability. This is probably due to the fact that the effort needed for tailoring the set of projections can be substantially lower than, for instance, *separate*, where metamodels need to be created or modified.

9.2 Multi-view modelling mechanism versus multi-view artefact notation type

It is interesting to highlight the fact that while *synthetic* approaches rely more often on *formal specification* (28%) or ad hoc *metamodel-based* notations (57%) to describe multi-view artefacts, *separate* approaches strongly rely on *UML-based* notations (73%). Also interesting is that *projective* and *orthographic* approaches mostly rely on *metamodel-based* notations (57% and 75%, respectively). The predominance of *UML-based* notations in *separate* approaches can be justified by the fact that this type of multi-view modelling mechanism aims often at bridging the gap between existing correlated notations, such as UML diagrams describing various aspects of the same problem.

9.3 Multi-view modelling mechanism versus multi-view artefact synthesis

Another interesting aspect is the support for synthesis in relation to the multi-view modelling mechanism. Investigating their correlation, we noticed that none of the analysed *separate* approaches provides such a support, while most *synthetic* approaches (77%) do provide it. This confirmed the two following facts. First, that a *synthetic* approach is by definition a set of distinct views implemented as distinct metamodels where the overall system is obtained as a synthesis of the information carried by the views. Second, that *separate* approaches focus on keeping views detached and no synthesis is supposed to happen. In three cases (P12, P19, and P29), the authors provide synthesis of different models for a specific multi-view artefact. Two of them (P12 and P29) synthesise component and connector models from different structural view representations.

9.4 Multi-view modelling mechanism versus intra-artefact consistency

In our taxonomy, we describe two types of multi-view artefact consistency: *intra-artefact*, representing consistency of modelled concepts within one single multi-view artefacts (e.g. models conforming to one view), and *inter-artefact*, representing consistency of concepts modelled across multiple multi-view artefacts (e.g. models in different views). While the latter is typically entailed in multi-view modelling artefacts, the correlation between *intra-artefact* consistency type and modelling mechanism is less obvious. For *synthetic* approaches, the focus is not on *intra-artefact* consistency, since it is often already there by construction. When looking at *separate* approaches, half of them consider *intra-artefact* consistency as part of the definition of the multi-view artefact notation (e.g. ad hoc *metamodel-based*). While in *projective* approach *intra-artefact* consistency is most often not a priority (67% of the analysed approaches does not provide it), for *orthographic* approaches we have the opposite situation (75% of the analysed approaches provides it).

9.5 Multi-view artefact concrete syntax versus multi-view artefact synthesis

While intuition may suggest that approaches expressing multi-view artefacts using *textual* notations are more amenable for supporting synthesis of such artefacts, due to a potentially higher level of details carried by textual artefacts, our analysis gave a different result. In fact, it appears that *diagrammatic* notations are furnished with synthesis support in 32% of the cases (out of which one-third leverages *textual* notations too). Only one approach (P3) provides synthesis leveraging *textual* notations only.

9.6 Correspondences notation type versus correspondences concrete syntax

Metamodel-based and *uml-based* approaches rely on exactly the same proportion on *textual* (37%) and *diagrammatic* (63%) notations. Interestingly, one approach based on *transformation language* (P12) uses a mix of both notations to describe correspondences conforming to the QVT relational specification. *Diagrammatic* notations are used by 66% of the approaches based on a *formal specification*. Although this may seem surprising, it can actually be explained by the fact that those approaches are based on graph-based (diagrammatic) notations.

9.7 Correspondences notation type versus multi-view artefact synthesis

Since correspondences are key to the synthesis, it is interesting to investigate the correlation between the notation type (used for describing correspondences) and the support for synthesis of multi-view artefacts. It is worth noting that all approaches describing correspondences through a *formal specification* provide synthesis too; this was somehow expected, since formally specified correspondences can simplify the definition of well-defined and deterministic synthesis. Approaches defining correspondences via *uml-based* or *metamodel-based* notations provide synthesis in 19% and 34% of the cases, respectively.

9.8 Inconsistency detection support type versus inconsistency resolution support type

When considering the correlation between the type of support (*manual* or *automated*) for inconsistency detection and resolution, it emerges that only 30% provides automation for both tasks. 42% provides automation for inconsistency detection, but no resolution at all.

9.9 Inconsistency detection mechanism versus inconsistency resolution mechanism

Looking at the correlation between the mechanisms for inconsistency detection and resolution, we discovered that *operational* mechanisms are used for both tasks in 50% of the analysed approaches. In one case, *constraint solver* mechanisms are used for both tasks too. In the remaining cases, the mechanisms used for the two tasks differ.

9.10 Inconsistency detection mechanism versus multi-view artefact modelling

The correlation between modelling mechanism and the way inconsistencies are detected is useful for understanding how multi-view artefact modelling determines which detection mechanism is typically used. In particular, in *synthetic* approaches, *formal methods* represent the most common detection mechanism (36%), followed by *constraint solver* and *operational* mechanisms (with 27% each); however, for *separate* approaches, *operational* mechanisms dominate the others with 63%. The same goes for *projective*, where *operational* prevails with 75%.

10 Prospects and challenges

This section discusses some technical and epidemiological challenges for multi-view modelling research and outlines potential paths for future research. However, it is arguably difficult to outline a convincing and coherent research strategy due to the depth and diversity of the considered approaches. Based on the lessons learned in the course of this study, we give here our interpretation of the previously described findings by discussing prospects for the technical advancement of multi-view modelling.

10.1 Consistency in the *small* versus in the *large*

To the best of our knowledge, whenever the multi-view mechanism is synthetic or separate, the restoration process is typically pair-wise, (e.g. [44,45]); i.e. it is based on techniques or resolution mechanisms specifically defined on pairs of artefacts as for instance with bidirectional model transformations that propagate changes from one source to the other and vice versa. The intricacy of the solution is represented by the necessity of defining a large number of consistency relations and corresponding restoring procedures; for instance in the case of a multi-view specification with n artefacts to be kept consistent, it would require to restore the consistency among

$$\frac{n!}{2(n-2)!}$$

pairs of views, not to mention the ripple effect that might lead to a non-confluent process. The problem at the moment remains largely open although two approaches have been presented over the last months. In particular, Stevens in [46] considers multi-ary consistency relations to be defined in terms of binary consistency relations, and how consistency restoration may be carried out on a network of models and relationships between them specified by means of an underlying mega-model. However, the conclusions are mainly negative as the paper suggests how in practice it remains resolutive to prescribe a resolution path, rather than relying on confluence. Another approach [47] consists in defining an intermediate metamodel for a declarative notation where to express the modelling (meta) elements to be kept consistent and the related bidirectional expressions to be used for the mappings. Despite these new approaches, the problem of multi-ary consistency relations remains unresolved as it poses severe difficulties on both the theoretical and practical side.

10.2 Non-determinism

Bidirectional transformations [48] are among the most important mechanisms used for consistency restoration and change propagation among models, especially in separate and synthetic approaches.

One of the main assumptions in most bidirectional approaches, for example, QVT-R [38], is that transformations are deterministic procedures, i.e. that they always deliver one and only one solution regardless of the source model and the subsequent round tripping. Unfortunately, if on the one side determinism is a highly desirable property that permits to have complete control on the transformation behaviour, on the other it requires to formalise at design-time a general restoration procedure capable of unambiguously restoring the consistency; i.e. it must be able to select the *right* solution out of the many possible. As described in [49], this is a difficult problem as it is not possible to demonstrate that a program is deterministic.[6]

In a broader perspective, a degree of non-determinism is also added by the *time* at which consistency restoration is performed. As already mentioned in Sect. 6.4.4, inconsistency resolution mechanisms might be triggered instantaneously at each change (named as *live* in the model transformation community), or could act at the end of the manipulation process, through an explicit invocation (typically named as *batch*). In some cases, it is clear that an approach works live, batch, or even provides both the modes. However, in the general case, the approaches do not analyse at a sufficient level of details what are the consequences of such modes of consistency restoration. Notably, a live restoration mechanism would intrinsically order the modifications on the views by the time each change is propagated; on the contrary, batch restoration would need explicit policies to manage changes operated on overlapping sets of elements; otherwise, the restoration could introduce corruptions in the views or undesired consistency restoration modifications.

These represent, in our opinion, another difficulty in realising multi-view modelling environments that rely on synthetic or separate mechanisms.

10.3 Terminology

The absence of a recognised consensus on many concepts and terminology of multi-view modelling represents a very challenging task for learning, analysing, classifying, and assessing existing approaches. Despite the existence of a well-established standard, i.e. the ISO/IEC/IEEE 42010:2011 [15] standard, vocabulary conflicts and inconsistencies can be frequently found among the many sources and references commonly used by researchers and practitioners. The basic unit of modelling in a multi-view specification (see Sect. 6.2), for instance, is often called view; however, a number of terms, such as model, viewpoint, perspective, and concern, to denote the same ontological concept are also used depending on the modeller's background and application domain. A similar issue is also observable when dealing with (in-)consistency management: in most of the approaches, inconsistency resolution is implicitly intended to be consistency restoration that is change propagation or synchronisation among the views. However, the problem can be potentially more complex: notably, there exist scenarios where change propagation can corrupt the target view, both in syntax and semantics terms, because the modifications in the source view are incompatible with the existing targets. These cases are typically referred to as merge conflicts in the version management world, and there exists corresponding literature also in the MDE research field. However, they usually do not find a careful handling in consistency management. In [51], the authors propose to exploit the term *discrepancy* whenever multiple views become out of sync, and then adopt the terms *inconsistency* and *conflict* to distinguish between change propagation and conflict resolution scenarios, respectively. This distinction is critical, since, for example, an imperative inconsistency resolution might be able to deal with any discrepancy problem by simply over-

[6] Confluence tests defined through critical pair analysis [50] are defined for graph transformations but not directly generalisable to arbitrary transformation languages (e.g. QVT-R).

writing the target view during the synchronisation; however, this might not be the desired way of resolving merge conflicts.

These findings should incite a stronger effort in disambiguating the terminology and the concepts of multi-view modelling. However, this must be a collaborative endeavour in the medium/long term that amalgamates reflections coming from practitioners and researchers working in areas that are adjacent but yet different.

11 Threats to validity

We are confident in the quality of our study thanks to two factors: (i) we defined and meticulously followed a detailed research protocol document, which was subject to external reviews by independent researchers and (ii) we conducted our study by following well-accepted guidelines for systematic studies. Anyhow, it is important to discuss the main threats to validity to our work as well as the means we undertook to mitigate them.

External validity This deals with the generalisability of results and findings [28]. Here, one of the most severe threats consists in that the selected set of primary studies is not representative of the state of the art of multi-view modelling. Since we employed a search strategy based on exhaustive backward and forward snowballing (that is to say iterated until no papers were left to scrutinise), we are confident in the coverage of our search. We strengthened external validity by applying a set of well-defined and validated inclusion and exclusion criteria, which were refined iteratively by considering the pilot studies of our review. As it often happens with secondary studies, a typical threat is represented by those (commercial or open source) tools that while providing some kind of support to multi-view modelling, do not have any significant research paper describing them.

Internal validity This refers to extraneous variables that may have had a negative impact on the design of the study. We mitigated it through the definition of a protocol and a data extraction form by following well-established guidelines. Concerning the validity of the data synthesis, threats were minimal since we used descriptive statistics. Moreover, we cross-analysed the different data extraction form parameters to make a sanity check of the extracted data. This task made us identify and solve potential issues on the consistency of the extracted data.

Construct validity This deals with the representativeness of the selected primary studies in the context of the research questions. Thanks to our exhaustive snowballing search, we are reasonably confident that we did not miss any significant relevant research study. Once primary studies were selected, we rigorously explored them according to well-documented inclusion and exclusion criteria. All phases were performed by all three researchers together, thus minimising threats to construct validity.

Conclusion validity This refers to the relationship between extracted data and obtained findings [28]. We mitigated potential threats by systematically applying and documenting well-defined processes; we provided a freely accessible package which allows to replicate each step of the processes too. The data extraction form definition can be a big threat to conclusion validity if based on own experience or other informal sources. We mitigated this by (i) letting the parameters and their facets emerge from the pilot studies and refining them throughout the entire data extraction activity, and (ii) making all three researchers to be actively involved in both the definition of the form and the extraction itself.

12 Conclusion

The scientific body of knowledge related to multi-view software and system modelling is variegated and rich. In order to identify, classify, and evaluate existing solutions for multi-view modelling, we conducted a systematic literature review of the existing literature. We selected and analysed 40 research studies among over 8,000 entries, defined a taxonomy for characterising solutions for multi-view modelling, and applied it to the selected studies. We analysed the studies both vertically (i.e. by single technical characteristic) and horizontally (i.e. pair-wise on related technical characteristics) and highlighted a set of main findings for each research question. In addition, based on those findings and our own interpretation of some of the reasons behind them, we depicted a set of highly prioritised challenges to be solved and future prospects for the technical advancement of multi-view modelling.

The taxonomy defined in this study is not meant to be a disposable artefact. On the contrary, we aim at making it available for the community as an interactive classification and persistent collection tool. Providing it as an open online form (Google Forms) linked to a browsable repository of categorised approaches, anyone can access it and use it to:

(i) categorise her own multi-view modelling approach to self-assess it and make it part of the common repository,
(ii) compare her own approach to others already categorised in the repository,
(iii) browse through the existing approaches in the repository and compare them to identify what best fits her purpose.

Currently, we provide the possibility to enter the characteristics of multi-view modelling approaches according to the taxonomy to make it part of the repository (i.e. (i)). The remaining features are currently underdevelopment.

This study is complementary to the survey on modelling views [10] mentioned previously, both in the way they achieved a taxonomy/feature model for describing characteristics of multi-view modelling (bottom-up vs top-down), and more importantly as they investigate the multi-view modelling problem at two different granularity levels (characteristics of views in isolation vs inter-views characteristics). We plan to collaborate with the authors of that survey to merge our results and thereby orthogonally cover a wider range of aspects of multi-view modelling at multiple granularity levels.

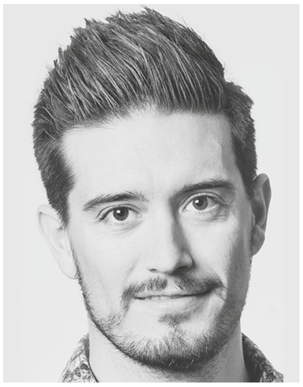
Acknowledgements This work has been supported by the Knowledge Foundation through the projects MOMENTUM (Ref. 20160153) and Strategic Knowledge Enhancement (Ref. 20150056).

References

1. France, R., Rumpe, B.: Model-based development. Softw. Syst. Model. **7**(1), 1–2 (2008)
2. Shah, A.A., Kerzhner, A.A., Schaefer, D., Paredis, C.J.J.: Multi-view modeling to support embedded systems engineering in SysML. In: MoDELS Workshops, vol. 5765 LNCS, no. 2, pp. 580–601 (2010)
3. France, R., Rumpe, B.: Model-driven development of complex software: a research roadmap. In: Future of Software Engineering (FOSE'07)
4. Schmidt, D.C.: Guest editor's introduction: model-driven engineering. Computer **39**(2), 25–31 (2006)
5. Cicchetti, A., Ciccozzi, F., Leveque, T.: Supporting incremental synchronization in hybrid multi-view modelling. In: Procs. of the 5th International Workshop on Multi-paradigm Modeling at MoDELS. Springer (2011)
6. Stevens, P.: Bidirectional model transformations in QVT: semantic issues and open questions. Softw. Syst. Model. **9**(1), 7–20 (2010)
7. Cicchetti, A., Di Ruscio, D., Eramo, R., Pierantonio, A.: JTL: a bidirectional and change propagating transformation language. In: International Conference on Software Language Engineering, pp. 183–202. Springer (2010)
8. Bhave, A., Krogh, B.H., Garlan, D., Schmerl, B.: View consistency in architectures for cyber-physical systems. In: Proceedings—2011 IEEE/ACM 2nd International Conference on Cyber-Physical Systems, ICCPS 2011, pp. 151–160. Carnegie Mellon University, Pittsburgh, United States (2011)
9. Kitchenham, B.A., Charters, S.: Guidelines for performing systematic literature reviews in software engineering. Keele University and University of Durham, Tech. Rep. EBSE-2007-01 (2007)
10. Bruneliere, H., Burger, E., Cabot, J., Wimmer, M.: A feature-based survey of model view approaches. Softw. Syst. Model. **8**, 1–22 (2017)
11. Spanoudakis, G., Zisman, A.: Inconsistency Management in Software Engineering: Survey and Open Research Issues, pp. 329–380. World Scientific, Singapore (2001)
12. Lucas, F.J., Molina, F., Toval, A.: A systematic review of UML model consistency management. Inf. Softw. Technol. **51**(12), 1631–1645 (2009)
13. Awadid, A., Nurcan, S.: Consistency requirements in business process modeling: a thorough overview. Softw. Syst. Model. (2017). https://doi.org/10.1007/s10270-017-0629-2
14. Dijkstra, E .W.: On the Role of Scientific Thought, pp. 60–66. Springer, New York (1982)
15. ISO/IEC/IEEE 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 (2011)
16. Leff, A., Rayfield, J.T.: Web-application development using the model/view/controller design pattern. In: Proceedings of the 5th IEEE International Conference on Enterprise Distributed Object Computing, ser. EDOC '01. Washington, DC, USA. IEEE Computer Society, p. 118 (2001). http://dl.acm.org/citation.cfm?id=645344.650161. Accessed 10 Oct 2018
17. Kruchten, P.: The 4+1 view model of architecture. IEEE Softw. **6**, 42–50 (1995). https://doi.org/10.1109/52.469759
18. Vallecillo, A., et al.: RM-ODP: the ISO reference model for open distributed processing. DINTEL Ed. Softw. Eng. **3**, 66–69 (2001)
19. Zachman, J.A.: A framework for information systems architecture. IBM Syst. J. **26**(3), 276–292 (1987). https://doi.org/10.1147/sj.263.0276
20. Atkinson, C., Stoll, D., Bostan, P: Supporting view-based development through orthographic software modeling. In: ENASE, pp. 71–86 (2009)
21. Finkelstein, A.C.W., Gabbay, D., Hunter, A., Kramer, J., Nuseibeh, B.: Inconsistency handling in multiperspective specifications. IEEE Trans. Softw. Eng. **20**(8), 569–578 (1994). https://doi.org/10.1109/32.310667
22. Kitchenham, B., Brereton, P.: A systematic review of systematic review process research in software engineering. Inf. Softw. Technol. **55**(12), 2049–2075 (2013)
23. Petersen, K., Feldt, R., Mujtaba, S., Mattsson, M.: Systematic mapping studies in software engineering. In: Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering, ser. EASE'08. Swinton, UK, UK: British Computer Society, pp. 68–77 (2008). http://dl.acm.org/citation.cfm?id=2227115.2227123. Accessed 10 Oct 2018
24. Brereton, P., Kitchenham, B.A., Budgen, D., Turner, M., Khalil, M.: Lessons from applying the systematic literature review process within the software engineering domain. J. Syst. Softw. **80**(4), 571–583 (2007)
25. Basili, V.R., Caldiera, G., Rombach, H.D.: The goal question metric approach. In: Encyclopedia of Software Engineering, vol. 2, pp. 528–532. Wiley (1994)
26. Wohlin, C.: Guidelines for snowballing in systematic literature studies and a replication in software engineering. In: Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering, ser. EASE '14. New York, NY, USA: ACM, pp. 38:1–38:10 (2014). https://doi.org/10.1145/2601248.2601268
27. Ali, N.B., Petersen, K.: Evaluating strategies for study selection in systematic literature studies. In: International Symposium on Empirical Software Engineering and Measurement. ACM (2014)
28. Wohlin, C., Runeson, P., Höst, M., Ohlsson, M., Regnell, B., Wesslén, A.: Experimentation in Software Engineering. Springer, Berlin (2012)
29. Cruzes, D.S., Dybå, T.: Research synthesis in software engineering: a tertiary study. Inf. Softw. Technol. **53**(5), 440–455 (2011)

30. Franzosi, R.: Quantitative Narrative Analysis, vol. 162. Sage, Thousand Oaks (2010)
31. Rodgers, M., Sowden, A., Petticrew, M., Arai, L., Roberts, H., Britten, N., Popay, J.: 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 (2009)
32. Bork, D., Karagiannis, D.: Model-driven development of multi-view modelling tools the muviemot approach. In: 2014 9th International Conference on Software Paradigm Trends (ICSOFT-PT). IEEE, pp. IS–11 (2014)
33. Bruneliere, H., Perez, J.G., Wimmer, M., Cabot, J.: Emf views: A view mechanism for integrating heterogeneous models. In: International Conference on Conceptual Modeling, pp. 317–325. Springer (2015)
34. Object Management Group (OMG): The unified modeling language specification version 2.5.1 (2017). https://www.omg.org/spec/UML. Accessed 10 Oct 2018
35. Smith, G.: The Object-Z Specification Language. Kluwer Academic Publishers, Norwell (2000)
36. Information Processing Systems—Open Systems Interconnection—LOTOS: A Formal Description Technique based on the Temporal Ordering of Observational Behaviour. ISO/IEC International standard 8807:1989, pp. 1–142 (1989)
37. Object Management Group (OMG): The business process model and notation specification version 2.0 (2011). https://www.omg.org/spec/BPMN/2.0/. Accessed 10 Oct 2018
38. OMG: MOF 2.0 Query/View/Transformation specification (QVT), version 1.3 (2016). http://www.omg.org/spec/QVT/1.3/. Accessed 10 Oct 2018
39. Object Management Group (OMG): The object constraint language specification version 2.4 (2014). https://www.omg.org/spec/OCL/. Accessed 10 Oct 2018
40. Selic, B.: The pragmatics of model-driven development. IEEE Softw. **20**(5), 19–25 (2003). https://doi.org/10.1109/MS.2003.1231146
41. Gelfond, M., Lifschitz, V.: The stable model semantics for logic programming. In: ICLP/SLP, vol. 88, pp. 1070–1080 (1988)
42. Petersen, K., Vakkalanka, S., Kuzniarz, L.: Guidelines for conducting systematic mapping studies in software engineering: an update. Inf. Softw. Technol. **64**, 1–18 (2015)
43. Agresti, A., Kateri, M.: Categorical Data Analysis. Springer, Berlin (2011)
44. Persson, M., Törngren, M., Qamar, A., Westman, J., Biehl, M., Tripakis, S., Vangheluwe, H., Denil, J.: A characterization of integrated multi-view modeling in the context of embedded and cyber-physical systems. In: Proceedings of the 11th ACM International Conference on Embedded Software, ser. EMSOFT '13, pp. 10:1–10:10. IEEE Press, Piscataway, NJ, USA (2013). http://dl.acm.org/citation.cfm?id=2555754.2555764. Accessed 10 Oct 2018
45. Xiong, Y., Song, H., Hu, Z., Takeichi, M.: Synchronizing concurrent model updates based on bidirectional transformation. Softw. Syst. Model. **12**(1), 89–104 (2013)
46. Stevens, P.: Bidirectional transformations in the large. In: 2017 ACM/IEEE 20th International Conference on Model Driven Engineering Languages and Systems (MODELS), pp. 1–11 (2017)
47. Gleitze, J.: A declarative language for preserving consistency of multiple models. Karlsruher Institut für Technologie (KIT), Karlsruhe (2017)
48. Czarnecki, K., Foster, J.N., Hu, Z., Lämmel, R., Schürr, A., Terwilliger, J.F.: Bidirectional transformations: a cross-discipline perspective. In: International Conference on Theory and Practice of Model Transformations, pp. 260–283. Springer (2009)
49. Eramo, R., Pierantonio, A., Rosa, G.: Managing uncertainty in bidirectional model transformations. In: Proceedings of the 2015 ACM SIGPLAN International Conference on Software Language Engineering, pp. 49–58. ACM (2015)
50. Mens, T., Taentzer, G., Runge, O.: Detecting structural refactoring conflicts using critical pair analysis. Electron. Notes Theor. Comput. Sci. **127**(3), 113–128 (2005)
51. Andrade, J., Ares, J., García, R., Pazos, J., Rodríguez, S., Silva, A.: A methodological framework for viewpoint-oriented conceptual modeling. IEEE Trans. Softw. Eng. **30**(5), 282–294 (2004). https://doi.org/10.1109/TSE.2004.1

Antonio Cicchetti is an Associate Professor at the IDT department, Mälardalen University, Sweden. He got his Ph.D. in Computer Science in 2008 at the University of L'Aquila with the thesis entitled "Difference Representation and Conflict Management in Model-Driven Engineering". His research interests include the interplay of model-driven and component-based engineering techniques and their application in the development of industrial systems. Moreover, he investigates the general problems related to the design of modelling languages, mutiview systems, and model transformations, in the context of both academic research and industrial application. Further, he is interested in the concerns related to the management of evolution of both language and models. He can be reached at antonio.cicchetti@mdh.se. For more information, see also http://www.es.mdh.se/staff/198-Antonio_Cicchetti.



Federico Ciccozzi is Associate Professor at Mälardalen University, Sweden, School of Innovation, Design and Engineering where he received his Ph.D. degree in 2014. His research focuses on the definition of metamodels and model transformations for several automation aspects in the model-driven development of component-based embedded real-time systems, such as code generation, preservation of system properties, back-propagation, to mention a few. Moreover, he carries out research in the area of multi-paradigm modelling, model versioning, (co)evolution and synchronisation, as well as application of model-driven and component-based techniques to (multi-)robot systems. He has co-authored more than 75 publications in journals and international conferences and workshops in these areas. He has been serving the community as conference track and workshop organiser, expert panellist, program committee member, and reviewer for conferences, workshops, and international journals. In his research activity, he has collaborated with several companies and research institutions such as Ericsson, ABB, Alten, Thales, Saab, and CEA list. He is a

member of the IEEE. More information is available at http://www.es.mdh.se/staff/266-Federico_Ciccozzi.

Alfonso Pierantonio is Associate Professor at the University of L'Aquila, Italy. He has published more than 120 papers in international journals, conferences, and workshops. He has been guest editor of a number of special issues on prominent journals (e.g. Journal of Software and Systems Modeling, Journal of Systems and Software, Science of Computer Programming). Alfonso is editor-in-chief of the Journal of Object Technology, and he is in the editorial board of the Journal of Software and Systems Modeling and Science of Computer Programming (advisory board). He has been PC chair of ECMFA 2018 and general chair of STAF 2015. Alfonso is chair of the steering committee of the International Conference on Model Transformation (ICMT) and of the Workshop on Models and Evolution (ME). In the last 5 years, he (co-)organised about 20 conferences and workshops. He has been member of the Program Board of the 21th International Conference on Model Driven Engineering Languages and Systems (MODELS'17). He has been co-director of the 12th International School on Formal Methods for the Design of Computer, Communication and Software Systems: Model-Driven Engineering held in Bertinoro (Italy), 2012.