Comparison of Software Design Models: An Extended Systematic Mapping Study

LUCIAN JOSÉ GONÇALES and KLEINNER FARIAS, University of Vale do Rio dos Sinos, Brazil
TOACY CAVALCANTE DE OLIVEIRA, Federal University of Rio de Janeiro, Brazil
MURILO SCHOLL, University of Vale do Rio dos Sinos, Brazil

Model comparison has been widely used to support many tasks in model-driven software development. For this reason, many techniques of comparing them have been proposed in the last few decades. However, academia and industry have overlooked a classification of currently available approaches to the comparison of design models. Hence, a thorough understanding of state-of-the-art techniques remains limited and inconclusive. This article, therefore, focuses on providing a classification and a thematic analysis of studies on the comparison of software design models. We carried out a systematic mapping study following well-established guidelines to answer nine research questions. In total, 56 primary studies (out of 4,132) were selected from 10 widely recognized electronic databases after a careful filtering process. The main results are that a majority of the primary studies (1) provide coarse-grained techniques of the comparison of general-purpose diagrams, (2) adopt graphs as principal data structure and compare software design models considering structural properties only, (3) pinpoint commonalities and differences between software design models rather than assess their similarity, and (4) propose new techniques while neglecting the production of empirical knowledge from experimental studies. Finally, this article highlights some challenges and directions that can be explored in upcoming studies.

CCS Concepts: • **Software and its engineering → System modeling languages**;

Additional Key Words and Phrases: UML, model comparison, model similarity, software design models

ACM Reference format:
Lucian José Gonçales, Kleinner Farias, Toacy Cavalcante de Oliveira, and Murilo Scholl. 2019. Comparison of Software Design Models: An Extended Systematic Mapping Study. *ACM Comput. Surv.* 52, 3, Article 48 (July 2019), 41 pages.
https://doi.org/10.1145/3313801

1 INTRODUCTION

The comparison of software design models plays a pivotal role in several model-centric software development tasks, e.g., identifying commonalities and differences between software design models, or even pinpointing conflicting changes as the relevant models are executed in parallel by

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

Authors' addresses: L. José Gonçales, K. Farias, and M. Scholl, University of Vale do Rio dos Sinos, Graduate Program in Applied Computing, São Leopoldo, RS, Brazil; emails: lucianj@edu.unisinos.br, kleinnerfarias@unisinos.br, mrscholl@edu.unisinos.br; T. Cavalcante de Oliveira, Federal University of Rio de Janeiro, Rio de Janeiro, RJ, Brazil; email: toacy@cos.ufrj.br.

0360-0300/2019/07-ART48 $15.00
https://doi.org/10.1145/3313801

different software development teams. The term "comparison of software design models" can be briefly defined as a set of tasks that should be executed over two input models, M_A and M_B, to analyze the equivalence or similarity between their elements. Many approaches to the comparison have been proposed in the last few decades (e.g., MADMatch [32], GaMMa [21], SiDiff [52], DSMDiff [34], and UMLDiff [64]) to support researchers and practitioners compare models like the UML (Unified Modeling Language), structural and behavioral models [41], and business process models [40]. Previous studies [33, 53, 59] have pointed out that measuring similarity remains an error-prone and time-consuming task [33, 53]. Regardless, researchers and practitioners in the industry still need to choose from among the available approaches in the literature the one that best fits their needs. Unfortunately, this choice is not trivial for two reasons.

The first is that the number of studies in the literature that have systematically classified approaches to the comparison of software models is still small. We noted this after a careful search to find a systematic map of model comparison approaches. Most studies explore issues in the field of model comparison, including comparative analysis and surveys of studies [30, 57], use of design comparison in the versioning of design models [4], brief reviews of approaches to the comparison of UML models [54], use of model comparison to promote artifact reuse [49], introductions to model comparison approaches and their main applications [57], and the elaboration of comparison techniques [2, 32, 64]. The second reason for why the choice of model is non-trivial is that although some studies have aimed to survey the literature, they fail to reflect a thorough understanding of the area. The review protocols used are not as rigorous and detailed as those found in a systematic mapping study (SMS) or a systematic literature review (SLR). Instead, the review procedures do not make clear important points, including research questions, retrieval and filtering procedures, and steps used to synthesize the collected data.

We also think that while many comparison approaches [2, 32, 64] were proposed between 2003 and 2018, academia and the industry have neglected their careful classification. A comprehensive understanding of the literature on these issues is severely lacking, including the software design models that are supported, the data structures that are often used, criteria that are considered to define the similarity between design models, how fine-grained the similarity measures are, the empirical methods used to evaluate the comparison approaches, and the level of automation. In general, current literature reviews use classification criteria based not on rigorous and broad reviews, but on choices based on expert judgment. Hence, a thorough understanding of state-of-the-art approaches remains limited and inconclusive. Even worse, researchers and industry professionals end up lacking the knowledge needed to help them in a judicious choice of comparison approaches, and in setting new directions for their research.

This article, therefore, aims at (1) providing a classification and a thematic analysis of studies on the comparison of software design models (Section 5), and (2) pinpointing gaps and directions of research for further investigation (Section 6). To this end, we carried out an SMS based on well-established guidelines [24, 26, 43]. A robust review protocol is elaborated by combining automatic and heuristic search in 10 widely recognized electronic databases and running a careful filtering process over a sample of 4,132 potentially relevant studies. In total, 56 primary articles were selected to answer nine research questions. We chose SMS as review method instead of SLR for three reasons. First, the goal of SMS is to classify and analyze literature on a particular topic [26, 63]. SLR aims to discuss and contrast related studies to identify gaps and progress. Second, the research questions explored in SMS are generic and often related to research trends, whereas SLR investigates specific ones, usually derived from findings and outcomes of empirical studies. Third, the expected results in SMS are selections of articles on research topics. SMS categorizes the selected articles based on a variety of dimensions and classifies the work into various categories. On

the contrary, SLR seeks to aggregate outcomes of the selected studies to answer specific research questions [26, 63].

This article is a substantially extended version of our previous work [18] in a number of aspects. First, this study includes new research dimensions explored through the nine research questions, instead of two research questions. Second, the entire research protocol was revisited to make its review procedures as more rigorous as possible. The study selection procedures were improved by supporting more refined selection criteria, the number of filtering steps to identify representative studies was increased from four to eight, and the search scope was enhanced to support 10 electronic databases instead of six. Third, we identified 56 primary studies, representing an increase of 16 studies over the previous version. Moreover, this article presents additional discussion, identifies open challenges and trends, and outlines key underlying issues that need to be tackled in future investigation.

The remainder of this article is organized as follows: Section 2 gives a comprehensive resume of the background needed to understand this study and Section 3 introduces our review protocol. Section 4 describes the procedures to filter potentially relevant studies and Section 5 presents the collected results. Section 6 introduces a discussion and a delineation of outstanding challenges, and Section 7 reports actions taken to minimize threats to the validity of our results. Section 8 describes related work and Section 9 presents our final remarks and directions for future work in the area.

2 BACKGROUND

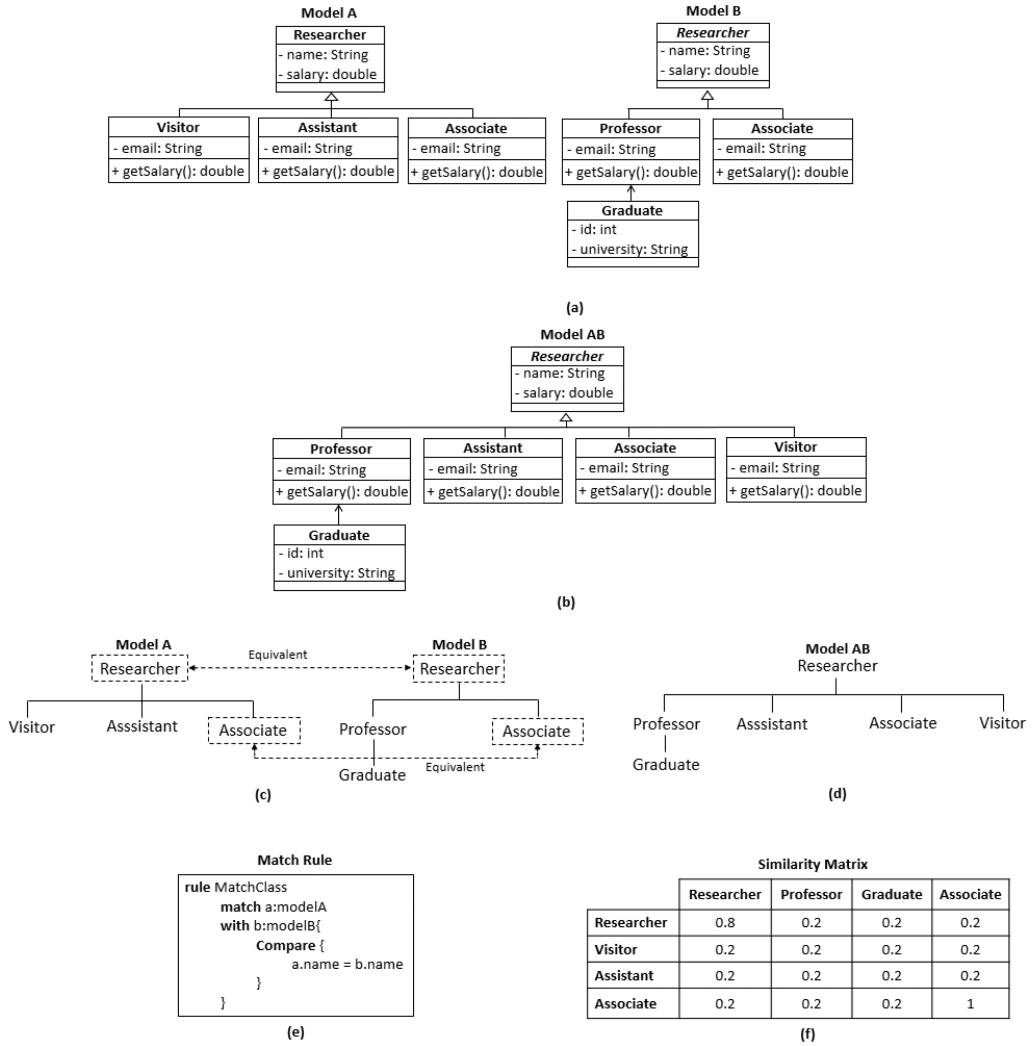
This section discusses the main terms used throughout this article. Section 2.1 presents key concepts by describing a comprehensive example of model comparison. Section 2.2 outlines the use of software design models. Section 2.3 compares SMS and SLR, and discusses some benefits of the former.

2.1 Model Comparison

Figure 1 presents an illustrative example of a UML class diagram [41] to understand model comparison. Figure 1(a) has two models, *Model A* and *Model B*. The first can be seen as a base model while the second represents an evolution of *Model A*. Suppose a developer needs to combine the contents of *Model A* and *Model B* to produce a consolidated view, i.e., *Model AB* as shown in Figure 1(b). For this, before combining *Model A* and *Model B*, the developer needs to identify the equivalence between them. For example, the class *Researcher* in *Model A* is defined as a concrete class (i.e., *Researcher.isAbstract* = false), whereas in *Model B*, it is set as an abstract one (i.e., *Researcher.isAbstract* = true). They have similar names but different values assigned to the meta-attribute *isAbstract*.[1] These contradicting values should be solved. Thus, the developer should answer the following question: What is the proper value of the "isAbstract" property? Based on the desired model in Figure 1(b), the correct answer is that *Researcher* is abstract—i.e., *Researcher.isAbstract* = true. Moreover, the attributes *Researcher.name*, found in *Model A* and *Model B* are equal. However, their attributes *Researcher.salary* have different types. Third, the methods *calcSalary()* and *reportSalary()* have different names but an identical purpose (i.e., computing salary). Although they have different signatures, their semantics are the same.

Moreover, the classes *Associate* in *Model A* and *Model B* have the same attributes and methods. Furthermore, both classes have an inheritance relationship with the class *Researcher*. Thus, they are equivalent, forming a single class *Associate* in *Model AB*. Furthermore, there is no matching pair for the remaining classes, i.e., *Visitor* and *Assistant* in *Model A*, and *Professor* and *Graduate* in *Model B*.

[1] This meta-attribute is defined in the UML metamodel [41].



	Researcher	Professor	Graduate	Associate
Researcher	0.8	0.2	0.2	0.2
Visitor	0.2	0.2	0.2	0.2
Assistant	0.2	0.2	0.2	0.2
Associate	0.2	0.2	0.2	1

Fig. 1. An illustrative example of two UML class diagrams (*Model A* and *Model B*) that need to be analyzed to identify commonalities and differences (a). *Model AB* represents a consolidated view of the contents of *Model A* and *Model B* (b). *Models A* and *B* are illustrated by a tree representation in (c), and *Model AB* in (d). Developers can use matching rules to compare the contents of *Models A* and *B*. *MatchClass* represents an introductory example of a match rule using the Epsilon comparison language (e). The similarity between the elements of *Models A* and *B* can be characterized by assigning a value of 0 to 1 for each element in them (f).

None of them is equivalent to another, and they are inserted into *Model AB*. Figure 1(c) presents the tree representation of *Model A* and *Model B*, and Figure 1(d) shows the tree representation of *Model AB*. The broken arrows show that elements of *Model A* and *Model B* are equivalent.

We now highlight the three main approaches to comparison commonly used to compare design models:

(1) *Rule-Based Technique.* This approach aims to find equivalences between design models by allowing users to manually specify a set of matching rules (e.g., [28] and [29]). Figure 1(e)

presents an introductory example of Epsilon's matching rule. This rule is a script that describes how to process a comparison of model elements. In this case, the matching rule compares elements of *Models A* and *B* based on their signatures (i.e., names). Typically, the matching rules are elaborated by considering a set of properties (like *name* and *isAbstract*) defined in the language metamodel used to specify the design models. Developers can create matching rules according to their needs. A rule library can also be created to promote rule reuse.

(2) *Similarity Technique.* The main purpose of this is to determine how close two software design models are by calculating a similarity value, which often ranges from zero to one. Figure 1(f) presents an illustrative example of a similarity matrix. For example, the classes *ModelA.Researcher* and *ModelB.Researcher* are the most similar, with similarity equal to 0.8. This technique can be used for clone detection where a high similarity between fragments of design models might indicate the presence of clones. It can also be used in the context of model merging to calculate similarity between model elements. For this purpose, a similarity matrix might be used to store values from zero to one to indicate the similarity between model elements. Similar model elements might be combined to form a merged model. Al-Khiaty et al. [3] applied a greedy algorithm to compute similarity between UML class diagrams.

(3) *Matching Technique.* The main purpose of this technique is to locate equivalent model elements (e.g., [60]). Current modeling tools assign an identifier (i.e., *ID*) to each model element created. That is, if some model elements are created, new *IDs* are assigned to them. By contrast, as model elements are removed, their *IDs* are removed as well. In Figure 1(a), for example, the class Researcher, found in *Model A* and *Model B*, has the same *ID*, as *Model B* can be seen as an evolution of *Model A*. These *IDs* are used to identify the equivalence between model elements. Model elements with equal *IDs* are considered equivalent. Thus, matching design models is intended to find model elements with the same *IDs*. These approaches focus on detecting and reporting differences between models, typically consecutive versions of design models. Matching techniques are often used in the context of model evolution by "tracking" the model elements added, removed, or changed from one version to another.

2.2 Software Design Models

Software design models (e.g., UML models) capture important aspects and concepts of software design represented from a certain point of view, thereby simplifying or abstracting the rest [48]. Such models are elaborated using a modeling language (e.g., UML [41]), and thus have well-formed syntax, semantics, and notations [48]. These issues need to be considered by comparison approaches. Software design models can be used for several purposes [48]. First, developers can capture and state requirements and domain knowledge so that all stakeholders may understand and agree on them. Second, developers can use them to think about the design of a system, getting the overall architecture right before detailed design begins. At low cost, developers can be innovative and creative due to the simplicity of creating and modifying small, editable architecture abstractions. Using design models, developers can explore several design solutions and architectures more easily before writing code. Third, developers may represent design decisions required for implementing a set of software requirements. For example, they might use UML class diagrams to define internal classes and operations that implement external behaviors expected by stakeholders. Given that there are many ways to implement these behaviors, developers can make use of UML sequence diagrams to express how the internal classes should collaborate with one another. Fourth, developers can master complex systems and generate usable work products. For example, the UML

component diagram transforms complex design decisions of object-oriented software systems into human-readable abstractions. Thus, it becomes possible to deal with complexity without getting lost in comprehensibility.

Today, many design model notations graphically represent static and dynamic views of such object-oriented software systems, such as UML. Chaudron et al. [7] noted that UML can be seen as a standard modeling language. Five reasons explain its widespread use. First, UML provides 14 diagrams [41], of which UML sequence and class diagrams are the most used [10]. Second, most modeling tools are dedicated to create and manage UML extensions and its models, such as the IBM Rational Software Architect Designer (RSAD) [20]. Third, the UML is a general-purpose modeling language that can represent many aspects of a software system [41]. Finally, as the UML is the basis of most modeling languages today, its results are transferable to other modeling languages based on it. Finally, our work is not limited to looking for approaches to UML model comparison, but instead conducts a broader search. The following section contrasts SMS and SLR, and highlights the benefits of each.

2.3 Systematic Mapping Study (SMS) and Systematic Literature Review (SLR)

SMS aims at exploring broad research areas by classifying the most representative studies in a particular subject and investigating generic research questions. Examples of these questions are: research issues investigated; methodologies used; and research gaps identified with sufficient background for supporting future investigation [26]. SLR in turn focuses on research questions that can be investigated by further empirical research. It seeks to investigate, for example, whether a particular technique is better than another. The selection process of the most representative studies is narrowly influenced by the research questions formulated, as specific techniques need to be identified so that the findings can be properly produced. Thus, the SMS is valuable and beneficial for practitioners as it provides a general view of the literature in a research area [26, 63].

Broader Scope of SMS. In terms of scope, SMS tends to have a broader scope while SLR is more focused. Even though a wide range of articles related to a topic are considered in SMS, classification data concerning such studies are typically produced. By contrast, SLR contains a narrow range of studies related to a research question on a specific issue, and looks to extract specific research outcomes from each study. It also seeks to aggregate the most representative articles selected in relation to the research results, at same time analyzing the results for validation [26]. In SMS, the studies are classified and aggregated based on a predefined classification scheme formed by a set of categories. Researchers usually elaborate these categories regarding the information available in articles, including the type of techniques, research venue, publication type, and research method used. Consequently, SMS tends to often explore a larger number of research questions than SLR.

Relevance of This Study. In performing an SMS, this article becomes relevant to model comparison for two reasons. First, the SMS protocol ensures that the review is rigorous and reproducible. This protocol enables us to systematically search and analyze past work and collect all relevant articles on the comparison of software design models. Second, it provides a general view of a broad research issue and identifies more detailed issues. Each issue can generate an SLR if there are sufficient fundamentals for it. Moreover, this study provides a classification of the literature, and identifies gaps and research directions for further investigation. With the results of this study at hand, researchers can explore the reported gaps (Section 6).

Educational Benefits of Our SMS. We also point out that researchers and students can benefit from our study in a number of ways. Based on [26], we list some educational benefits of SMS.

First, students can learn how to search literature and organize the results of their searches. For Ph.D students, for example, SMS provides an important starting point for research as it provides reusable research skills. Second, students can replicate our SMS study, or even use the proposed review protocol to run similar studies in other software engineering contexts. Third, the learning curve and bias related to performing literature reviews can be reduced as our SMS protocol (Section 3) can serve as a guide for future studies. Finally, a list of gaps in the literature can be easily accessed, thereby saving effort to locate them. Budgen et al. [6] corroborated these educational benefits and emphasized that "mapping studies serve as a starting point for Ph.D students who need to organize and understand existing research works in a specific domain." Further, they also noted that "SMS can provide beginning researchers with a body of knowledge as starting point for their research rather than forcing every researcher to start from scratch."

Main Benefits for the Industry. Practitioners can benefit from our results typically when performing software modeling tasks, like creating, evolving, or even changing UML design models. By listing potentially relevant comparison approaches, developers can spend their development efforts on selecting approaches more appropriate to their needs, rather than having to scour the literature. Using appropriate approaches, they can pinpoint equivalences between design models more appropriately, boosting the effectiveness of comparison. Our study also provides some educational benefits for practitioners. The reported findings can server as a starting point to develop innovative comparison techniques while avoiding unnecessary effort to identify the gaps in and characteristics of current comparison approaches.

In the industry, teams of software analysts often need to co-evolve and eventually converge design models created in parallel, eliminating redundancies and bringing complementary parts together. For this, analysts need first to compare such models to identify commonalities and differences, and combine the parts co-evolved concurrently so that a consolidated view can be formed. To date, comparing and merging design models is considered a tedious, time-consuming, and error-prone task [33, 53, 66]. In this context, our work adds value for analysts by presenting a multidimensional classification scheme of comparison approaches, including type of model, data structure, and algorithm employed.

3 SMS PLANNING

3.1 Objective and Research Questions

The objective of this work is twofold: (1) to provide a classification of the literature on model comparison and (2) to identify gaps and promising research directions for further investigation. To explore these objectives better, we formulate nine research questions (RQ) to carefully scrutinize different facets of our objectives. Table 1 summarizes the investigated RQs. According to Petersen, RQs in SMS should be generic for uncovering research trends over time and topics already explored in the literature [43]. The motivations to inquire into these RQs are as below.

RQ1: What are the diagrams supported by comparison techniques? Several contemporaneous modeling languages, such as UML [41], have been proposed in the last few decades. Likewise, many comparison approaches have been developed to support comparisons between diagrams of such modeling languages. However, few studies have considered mapping and determining the types of diagram that have been most investigated and explored in the context of model comparison. This also helps understand the diagrams that have been prioritized.

RQ2: What are the commonly used data structures in current techniques? Some studies have reported that comparing models is an NP-hard problem [34, 47]. If data structures are improperly chosen, the comparison of large-scale software design models may encounter severe performance

Table 1. Research Questions Investigated

Research Question	Motivation	Variable
RQ1: What are the diagrams supported by comparison techniques?	Determine the diagram types that have been most investigated and explored.	Supported diagram
RQ2: What are the commonly used data structures in current techniques?	Discover and understand the data structures that are most used in the literature.	Data structures
RQ3: What aspects are considered for comparing design models?	Understand the different aspects considered for comparing software design models.	Comparison aspects
RQ4: How fine-grained is the measurement of similarity?	Grasp how fine-grained the similarity measures are.	Granularity
RQ5: What are the types of model comparison?	Reveal and quantify how the comparison types spread over work already published.	Comparison types
RQ6: What are the empirical methods used to evaluate comparison techniques?	Uncover the research methods used to evaluate model comparison approaches.	Research methods
RQ7: What is the level of automation of comparison techniques?	Investigate how the comparison process is conducted, e.g., manual, semi-automatic, or automatic.	Degree of automation
RQ8: What are the most commonly used comparison techniques?	Reveal the most commonly used comparison techniques.	Comparison techniques
RQ9: Where have the studies been published?	Elicit the target vehicles used to disclose the results.	Research venue

problems. Unfortunately, knowledge on how design models are represented and manipulated is still scarce. To account for this, RQ2 seeks to discover data structures most used in the literature.

RQ3: What aspects are considered for comparing design models? Multiview software modeling tends to be applied to a wide range of contexts (e.g., software factory) and application domains (e.g., health, finance, aerospace, and e-commerce). Comparing models that represent small parts of an overall architecture is not a trivial task, as many aspects can be considered, including syntactical, semantic, structural, and layout related. RQ3 aims at uncovering the different aspects considered for comparing software design models.

RQ4: How fine-grained is the measurement of similarity? The granularity of comparison refers to the number of model properties considered to compare different aspects of design models. Examples of these aspects are syntactical, structural, layout related, and semantic. Little is known about the extent to which current comparison approaches are fine or coarse grained. Exploring this issue is important because modeling languages (e.g., UML [41]) are used for documenting, specifying, and constructing the abstractions of a software system under development. Such languages are general purpose, and can represent concepts from several programming paradigms (e.g., object orientation (OO)), and can be utilized for diverse application domains (e.g., oil, health, education, and finance) and implementation platforms (e.g., Java Enterprise Edition (JEE), .NET, and NodeJS).

RQ5: What are the types of model comparison? In the last few decades, academia and industry have proposed a wide variety of comparison techniques. While some techniques are based on *ID* or signature comparison—even focusing on proposing a measurement of similarity that considers

the commonalities and differences between models—others piece together the syntax and semantics of models. Unfortunately, a comprehensive analysis of the available comparison techniques is still lacking. We are thus looking to scrutinize contemporary comparison techniques to recognize, differentiate, and understand their rationales.

RQ6: What are the empirical methods used to evaluate comparison techniques? Little is known about the kinds of methods that have been applied to the comparison of software design models. We want to understand how these methods have been used to evaluate model comparison techniques. This can reveal, for example, how practical knowledge, findings, and insights have been generated about a particular technique.

RQ7: What is the level of automation of comparison techniques? A comparison technique can be used to reduce development effort by automating tasks or more precisely executing manual tasks. Unfortunately, comparison techniques have not been classified based on degree of automation. Without this knowledge, it is particularly challenging, especially for developers, to choose comparison techniques given a particular constraint.

RQ8: What are the most commonly used comparison techniques? Based on knowledge generated from RQ5, we determine the types of comparison techniques that have been used or adopted most often. Despite widespread interest in comparing diagrams in several fields of software engineering, there is little quantitative evidence on how types of comparison techniques have been adopted. Moreover, we seek to identify possible trends of use.

RQ9: Where have the studies been published? The main objective is to scrutinize recent trends in publication as the vehicle representing work on model comparison. That is, we look to understand the main research venue for the publication and dissemination of work related to the comparison of software design models.

3.2 Search Strategy

This section describes the strategy used to search the literature in this study. We specified an unbiased and iterative search strategy by considering well-known guidelines [24, 43] related to the definition of the search scope and the construction of search strings. We systematized the selection of a list of potentially relevant studies strictly related to the research questions defined in Section 3.1. The search strategy was iteratively elaborated so that we could pinpoint and evaluate literature reviews (i.e., SLR, SMS, and surveys) quantitatively and qualitatively.

3.2.1 Construction of Search Strings. The results depended on how well the search strings (SS) had been formalized. To systematically identify the terms of the search strings, we adopted the PIO (populations, interventions, and outcomes) criteria. We chose this method because it has been applied to a wide range of empirical studies [27, 45]. Kitchenham et al. [27] advocate breaking down research questions into facets. Thus, a list of synonyms, abbreviations, and alternative spellings can be drawn up [27]. Innovative search strings can be thus constructed using Boolean ANDs and ORs.

The populations refer to the terms related to the main object being investigated, such as diagram, design model, and structure. Although UML is widely used as a modeling language of software systems [7], there is a wide range of specific notations to represent the features of Domain Specific Languages (DSL). To the best of our knowledge, to list all these variants is not a viable task since all of them are not known. Therefore, the inclusion of specific terms in the search string such as UML, and the notations of DSL variants was not considered. This may limit the search to just a few design notations. The intervention is related to the terms considering a specific process or

Table 2. A Description of the Major Terms and Their Synonyms

Major Terms	Synonym Terms
Diagram	Model OR Design OR Structure
Comparison	Similarity OR Match OR Differencing

context involving the population, such as comparison and matching. The outcomes concern factors important to the practitioners, for example, reduced effort, improved precision, and reduced time to market [24, 43]. In the context of our work, it can refer to the precision of model comparison and the level of automation for comparing software design models. Given that we are not concerned with restricting our search units considering such factors, the outcomes were not included in the search terms. To sum up, the following steps were executed to define our search terms:

- *Step 1: Define Candidate Keywords.* For this, we read two articles [4, 57] to pinpoint the main terms. These articles were chosen because they are extensive literature reviews closely related to the subject explored in our SMS.
- *Step 2: Identify Related Words and Alternative Terms or Synonyms Related to the Initial Keywords.* To this end, we scrutinized the literature based on keywords defined in Step 1. An initial sample of articles was found and the candidate keywords refined.
- *Step 3: Verify if the Major Keywords Are in Articles in the Research Area, i.e., Comparison of Design Models.* In this sense, we manually checked if the selected terms were commonly found in our initial sample of retrieved articles. This analysis was done in an interactive and incremental way by the authors. The result of this step was a set of terms commonly used in our initial sample.
- *Step 4: Associate Synonyms, Alternative Words or Terms Related to the Main Keywords using the Boolean Operator "OR."* The terms carefully identified in Step 3 were brought together using the operator "OR."
- *Step 5: Relate the Major Terms with Boolean Operator "AND."*

Table 2 shows two major terms and their synonyms. Even though many combinations of the search terms could be formulated, we verified that many of them returned similar results. Thus, the majority of important results returned by search engines (Table 3) were obtained using the following search string:

((diagram OR model OR design OR structure) AND (comparison OR similarity OR match OR differencing))

3.2.2 Source of Information. Table 3 shows 10 electronic databases (our search scope) used to retrieve potentially relevant articles. Our search string was applied to each database listed in Table 3. This extensive number of electronic databases was considered to cover the main conferences, journals, and workshops in computer science. The search looked at studies published in conferences proceedings, educational institutions, and electronic digital libraries of journals. We also considered papers written in English and published before April 2018.

3.3 Exclusion and Inclusion Criteria

This section establishes the exclusion and inclusion criteria used to filter the studies retrieved from the selected electronic databases. The following list specifies the Exclusion Criteria (EC) used in this article. EC excluded studies where:

Table 3. List of Electronic Databases

Search Engines	Link
ACM Digital Library	http://dl.acm.org/
CiteSeerX Library	http://citeseerx.ist.psu.edu/
Google Scholar	https://scholar.google.com.br/
IEEE Explore	http://ieeexplore.ieee.org/
Inspec	http://digital-library.theiet.org/
Scopus	http://www.scopus.com/
Science Direct	http://www.sciencedirect.com/
Springer Link	http://link.springer.com/
Web of Science	http://apps.webofknowledge.com/
Wiley Online Library	http://onlinelibrary.wiley.com/

- *EC1.* The title, abstract or any other part of their content was closely related to the search keywords (described in Section 3.2.1), however, the study clearly belongs to another research domain, such as, biology, and finance.
- *EC2.* A patent had been registered, or the study was not published in English (the default language considered in our study), or was in an initial stage, typically presenting an abstract and summary of future steps;
- *EC3.* The title did not have any term defined in the search string, or the meaning of the title does not address the issues in the research questions;
- *EC4.* The abstract did not address any aspect of the research questions;
- *EC5.* The study was duplicate; and
- *EC6.* The full text did not address issues about model comparison techniques.

Four Inclusion Criteria (IC) were used to add studies to our sample. The IC inserted studies that:

- *IC1.* were articles, master dissertations, doctoral theses, books, or book chapters on comparisons of software design models;
- *IC2.* were published or disseminated in English;
- *IC3.* were published in scientific journals, conferences or workshops; and
- *IC4.* were published before April 2018.

3.4 Data Extraction Strategy

This section defines the data extraction strategy used to collect data on our RQs. A classification scheme (Table 4) was produced to guide data extraction related our RQs from primary studies. This taxonomy was created based on the consensus of the authors after reading and peer reviewing the articles, and categorizing the collected information related to our RQs. To this end, two meeting cycles were required. An extraction form (Figure 2) was also elaborated based on the classification scheme to mitigate error during data extraction. Our form was inspired by previous work, such as Fernandez-Saez et al. [14] and Smite et al. [56]. The authors filled out the form considering the taxonomy values of each work explored. We used spreadsheets to store and generate statistics from the collected data.

Three review cycles were performed in the data extraction process. This process was conducted collaboratively by the authors to avoid false negatives and false positives, and to cover important open issues. At least two authors reviewed and classified each study using the data extraction form and the classification scheme. The spreadsheets generated were consolidated into one spreadsheet. To this end, the authors met in person to discuss and resolve conflicting data in spreadsheets

Table 4. Classification Scheme Used to Extract Data From the Studies

Question	Variable	Answers
RQ1	Supported diagram	Generic, Metamodels, Business Process Models, Use Case Diagram,
		Class Diagram, Sequence Diagram, Activity Diagram, Statechart Diagram
RQ2	Data structures	Graph, Tree
RQ3	Comparison aspects	Structure, Syntactic, Semantics, Layout, Lexical, Multicriteria
RQ4	Level of granularity	Coarse-grained, Partial, Fine-grained
RQ5	Comparison types	Similarity, Matching, Rule-based
RQ6	Research methods	Evaluation Research, Solution Proposal, Validation Research, Philosophical Papers,
		Opinion Papers, Experience Papers
RQ7	Degree of automation	Automatic, Semi-automatic, Manual
RQ8	Comparison techniques	UUID, Heuristic, Search-based, Rule-based
		Signature-based, None
RQ9	Research venue	Publication year, Name of venue, Kind of venue

Article's Data						
Title						
First Author						
Source (name of journal/conference)						
Year of Publication						
Type of Publication		Conference		Journal		Workshop
Research Questions						
Diagram Type						
Data Structure						
Aspects						
Granularity		Coarse-Grained		Partial		Fine-Grained
Comparison Approach						
Research Method						
Automation Degree		Automatic		Semi-Automatic		Manual

Fig. 2. Data extraction form (inspired by [14] and [56]).

generated in parallel. Finally, we performed a third parallel review cycle so that improper classifications could be uncovered. We discuss the classification scheme created in more detail as follows:

RQ1: Supported Diagram. We counted the types of diagrams supported by the selected studies considering the following diagrams: (1) generic (GD), (2) metamodels (MM), (3) business process models (BPM), (4) use case diagrams (UC), (5) class diagrams (CD), (6) sequence diagrams (SD), (7) activity diagrams (AD), and (8) statechart diagrams (SCD). Even though some diagrams may be based on specific UML notations, we did not consider the UML versions in question.

RQ2: Data Structure. We list below the data structures considered as follows: (1) *Graph*: The algorithms used the structure of a graph to compute similarity between diagrams; and (2) *Tree*: The approaches used tree structure to compare elements.

RQ3: Comparison Aspects. Even though there is no predefined set of aspects of comparison for model evaluation in the literature, we identified six commonly used ones in recent approaches: (1) *Structure*: It considers the structure of model elements to compute how similar they are; (2) *Syntactic*: It takes into account the constructs of the language for comparing design models; (3) *Semantics*: The meaning of the model elements is considered to compare input models;

(4) *Layout*: The approach considers visual aspects, e.g., color and position, directly related to comprehensibility issues; (5) *Lexical*: These name-based comparison approaches consider the value of properties of model elements to infer their similarities; and (6) *Multicriteria*: This aspect considers approaches that use two or more of the previous aspects to compare models.

RQ4: Granularity. We considered three degrees of granularity: (1) *Coarse-grained*: Only one attribute is analyzed to compute differences between the models to be compared, e.g., the names of the elements only; (2) *Partial*: A quantity of partial attributes is evaluated for a comparison of elements, i.e., more than one element; and (3) *Fine-grained*: It uses all possible attributes for comparing model elements.

RQ5: Comparison Types. We analyzed three types of comparison: (1) *Similarity*: The goal is to identify similarity, returning values indicating how similar elements are; (2) *Matching*: The mechanism produces a set of elements considered equivalent. In some cases, these sets are separated considering the types of comparison of the model elements, i.e., deleted, inserted, or updated elements; and (3) *Rule-based*: This is matching based on pre-defined rules.

RQ6: Research Methods. This question provides an overview of the direction of current studies, i.e., the kinds of studies that academia has produced. Wieringa et al. [62] presented a list of empirical strategies as follows: (1) *Evaluation Research*: It evaluates commonly used approaches in the industry; (2) *Solution Proposal*: It proposes a solution based on a new or a previous approach; (3) *Validation Research*: These studies focus on evaluating techniques not yet known in the industry; (4) *Philosophical Papers*: These discuss a new and revolutionary manner to conduct comparison techniques; (5) *Opinion Papers*: These studies report the experience of authors related to a particular subject; and (6) *Experience Papers*: They argue what must be done to resolve the addressed problem based on personal experience.

RQ7: Degree of Automation. The goal of this item is to classify studies considering their levels of automation: (1) *Automatic*: It requires no human involvement; (2) *Semi-automatic*: It requires that users specify configuration parameters before differentiation. Approaches that require manual user intervention for correcting evaluation procedures are also classified as semi-automatic; and (3) *Manual*: The comparison of models is conducted by hand.

RQ8: Comparison Techniques. This examines the types of techniques used to compare model elements: (1) *UUID*: These techniques present match-by-identifier approach, i.e., ID-based ones; (2) *Heuristic*: Elements are equivalent through combinations of metrics; (3) *UUID-Heuristics*: Some studies provide functionality for comparing models using UUID or heuristics; (4) *Search-based*: These algorithms use metrics as scores to restrict search-space to find equivalences; (5) *Rule-based*: Here, elements are compared according to pre-defined semantic or logical rules; (6) *Signature-based*: Similarity is based on signatures of model elements; and (7) *None*: Study does not specify the matching technique used.

RQ9: Research Venue. It scrutinizes recent research trends in the context of model comparison. We extracted the following information: publication year, name of venue, and kind of venue (i.e., conference, journal, or workshop).

3.5 Selection Procedures

This section presents the filtering process used to select the most relevant studies. Petersen et al. [43] have noted that "a larger number of articles may not be better than fewer, if the fewer are a better representation of the population of articles for the targeted topic." With this in mind, the following filtering process aimed to present the procedures used to select the representative studies:

- *Step 1: Initial Search.* The process collected the search results after submitting a search string to digital libraries.
- *Step 2: Remove Impurities.* The process removed discrepancies obtained in the search results. For this, the exclusion criteria EC1 and EC2 were applied. Calls for papers to conferences, special issues of journals, patent specifications, research reports, and non-peer-reviewed materials are examples of work retrieved improperly.
- *Step 3: Filter by Title and Abstract.* The process filtered studies by applying the exclusion criteria, EC3 and EC4. Thus, we discarded any study whose title and abstract had no term from the search string, and those not related to the main issues addressed in our research questions (Table 1).
- *Step 4: Combination.* In the previous steps, the primary studies were in their respective directories of the search engines. Therefore, all filtered studies from the last step were then brought together.
- *Step 5: Duplicate Removal.* Usually, a study can be found in two or more digital libraries. Thus, we applied EC5 to remove all duplicates, thereby ensuring the uniqueness of each study.
- *Step 6: Study Addition by Heuristic.* Although the search mechanisms of the digital libraries are widely recognized, some works may not be found in them sometimes. Thus, we added certain studies manually to our sample of primary studies according to expert opinions.
- *Step 7: Filter by Full Text.* The process filtered studies by applying EC6 to the full text, excluding studies not relevant to model comparison or matching.
- *Step 8: Representative Work Selection.* The process defined the final list of primary studies.

3.6 Quality Assessment

We defined a quality assessment questionnaire to provide insight into the quality of the retrieved studies and help us filter them based on qualitative issues. In total, nine questions were formulated. They were chosen because they had already used in previous studies [14, 27, 45]. This qualitative analysis helped us grasp the literature in terms of quality, and generated insights more appropriately. It also helped reduce threats to the replication of this study. The nine questions (Q) of the quality assessment questionnaire are as follows:

- *Q1*: Is there a clear statement of the goals investigated?
- *Q2*: Is the context well defined?
- *Q3*: Does the work carefully describe related work?
- *Q4*: Has the work provided a clear explanation of the comparison technique?
- *Q5*: Have the authors evaluated the approach?
- *Q6*: Do the researchers discuss threats to validity?
- *Q7*: Do the authors write clear conclusions?
- *Q8*: Do the conclusions comply with the objectives or research questions?
- *Q9*: Do the authors present suggestions for further research?

3.7 Data Synthesis

The data synthesis in this work is based on concepts of *Grounded Theory* [17] and descriptive analysis. We also followed lessons learned concerning data synthesis [24]. First, literature reviews in software engineering are mostly qualitative (i.e., descriptive) in nature. Second, meta-analysis is challenging even when quantitative information is collected, as the reported protocols tend to vary between studies [15]. Third, data tabulation is a crucial means to producing results by aggregating

the extracted data, but it is pivotal to scrutinize how the aggregated data answer the research questions.

Grounded Theory. The classical Grounded Theory is formed by three general coding phases [65]: open coding, selective coding, and theoretical coding. The first aims at generating codes for perceptions that may be clustered into concepts and categories (i.e., coding phase). The second focuses on pinpointing core categories that can better explain variations in the extracted data [65] (i.e., categorizing phase). In this study, we used the first two to help us synthesize the data obtained. We clustered the works considering the presented classification scheme (summarized in Table 4), so that the quantity and percentage of works by category could be produced. The third coding phase of grounded theory was not used because the main purpose of this SMS was not to generate theories. Sbaraini et al. [51] reported that data analysis might be conducted based on a process of (1) splitting data up into smaller issues, labeling and classifying those issues (i.e., coding), and (2) comparing the classified data to explain and grasp variations in our perceptions (i.e., categorizing). The produced classifications are more abstract. With this in mind, after reading the filtered articles, the first three authors separately created a short observation-based memo recording their perceptions of the issues addressed in each research question. After that, their notes were compared, organized, and compiled according to the classification scheme shown in Table 4. Note that Table 4 was created to present the values of each question as a classification scheme. The results are shown in Table B.1.

Descriptive Analysis. Spreadsheets were used to integrate the results obtained from data extraction (described in Section 3.4). All descriptive information was calculated and organized using MS Excel™. In Appendix B, Table B.1 presents a general classification of the primary studies. The collected data are used to generate graphs and tables. While the columns address the research questions summarized in Table 1, the rows store values based on the classification scheme introduced in Table 4. For example, as the third column addresses results related to RQ2, its rows store the values assigned to RQ2, i.e., graph or tree. In particular, the descriptive analysis was performed on the tree steps. First, the total number of answers to each question was computed. Second, the median, average, and standard deviation were computed for each answer. We noted that too many granular values near one usually were related terms that could be combined into more general groups, i.e., the multiple criteria in RQ3, and heuristic and search-based categories in RQ8. Third, the percentage of correspondent answers was calculated. Finally, the values and related primary studies formed each table in the results' section. At least two researchers read each primary study and coded their contents according to Table 4. The results collected are visually illustrated using charts and tables.

4 STUDY FILTERING

This section explains how the study filtering process and quality assessment of the selected studies were executed.

4.1 Execution of Study Filtering

The execution of the study filtering process consisted of applying the eight steps defined in the selection procedures (described in Section 3.5) of potentially relevant works. These selection steps were performed sequentially, i.e., starting at the first step and ending at the eighth. Figure 3 shows the results obtained from the execution of each step of the filtering process. The results are available as additional material on the study website.[2]

[2] https://luciangoncales.github.io/studies/acmcs2018/.

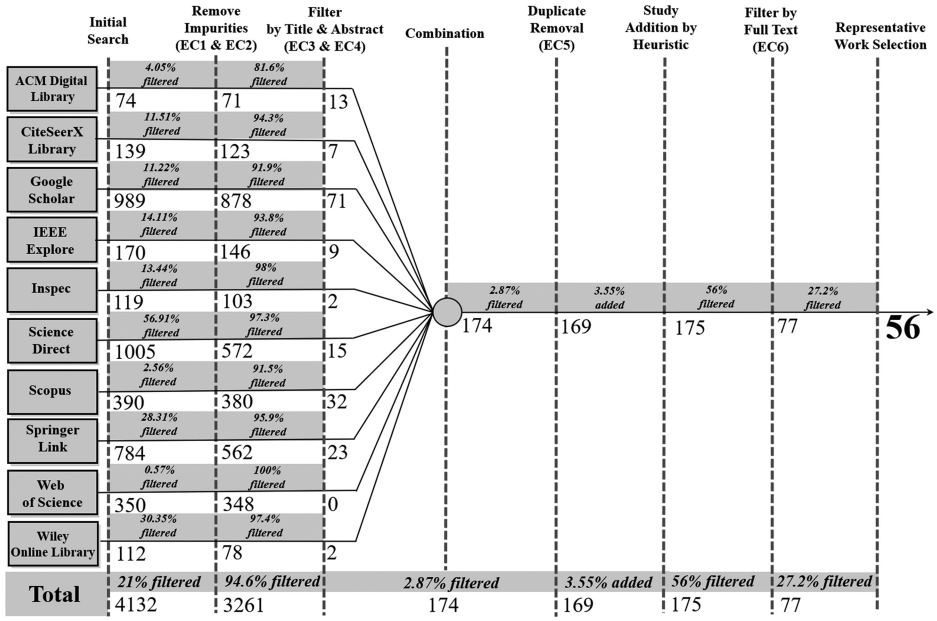


Fig. 3. The obtained results from the execution of the study filtering process.

Our initial search retrieved 4,132 articles. We then applied two exclusion criteria, EC1 and EC2 (described in Section 3.3), to remove impurities, and then applied EC3 and EC4 to the remaining 3,261 studies, 99% (3,087/3,261) of the articles having been filtered through title and abstract review. The remaining studies were brought together to produce a sample of 174 studies. The next step was to remove duplicate studies by applying EC5: 2.87% (5/174) were discarded. We added six studies by applying heuristics, producing a sample of 175 studies, an increase of 3.55%. C6 was then applied to the 175 studies, and 56.3% (98/175) of the articles were discarded. By examining the remaining 77 studies, we observed that some were technically similar. That is, works were produced based on previous ones, and their contributions were closely related. Thus, 27.2% (21/77) were excluded. Finally, 56 works were selected as the most representative, hereinafter called primary studies (Appendix A).

4.2 Execution of Quality Assessment

Before exploring the primary studies, we assessed their quality to check the studies that could be used to respond to the research questions. For this, we applied the nine questions of our quality assessment questionnaire (Section 3.6). Our focus was to prioritize high-quality studies to respond to our research questions, generate insight, and delineate future challenges. In Appendix B, Table 17 presents the quality assessment applied to the primary studies. If a primary study satisfies a question, the symbol ✓is used to represent "yes." ; otherwise, the symbol ✗ is applied, meaning "no." For each satisfied question, a point was added for a study and a question. The last column (i.e., points) represents the number of points obtained for each study. The last line (i.e., total) shows the total points for each question.

All 56 primary studies fulfilled questions Q1, Q2, Q7, and Q8. By contrast, only one study satisfied the sixth question (i.e., Q6). Other questions–Q3, Q4, Q5, Q6, and Q9–were partially satisfied. The results indicate that 13 studies (23.2%, 13/56) did not mention any related work (Q3), eight studies (14.3%, 8/56) did not evaluate the proposed approach (Q5), and 13 studies (23.2%, 13/56) did

Table 5. Classification of the Primary Studies Based on Their Supported Diagrams (RQ1)

Diagram types	#Studies	Percentage	List of primary studies
Class diagram	22	39%	[S7], [S10], [S22], [S24], [S25], [S26], [S28], [S32], [S33], [S35], [S36], [S37], [S38], [S40], [S41], [S42], [S45], [S47], [S48], [S52], [S53], [S56]
Generic diagram	17	30%	[S2], [S3], [S4], [S8], [S9], [S11], [S14], [S19], [S20], [S21], [S23], [S34], [S39], [S44], [S46], [S50], [S51]
Metamodels	3	5%	[S12], [S29], [S49]
Sequence diagram	4	7%	[S30], [S31], [S35], [S55]
Statechart	3	5%	[S15], [S16], [S43]
All UML diagrams	3	5%	[S18], [S27], [S54]
Activity diagram	2	4%	[S13], [S56]
Business process model	1	2%	[S6]
Use case	1	2%	[S5]
Component and connector	1	2%	[S1]
UML profile	1	2%	[S17]
Total	58*		

*[S35] appears in two categories: Class diagram and Sequence diagram.
*[S56] appears in two categories: Activity diagram and Class diagram.

not discuss future work (Q9). We noted that S46 [52] fulfilled the lower number of questions (only five questions). On the contrary, 28 studies answered eight questions. As all articles answered at least five questions, none was discarded. Therefore, the articles selected had the quality required to respond to the research questions.

5 SMS RESULTS

This section presents the results for the research questions formulated (Table 1) after analyzing the 56 primary studies (Appendix A). Our findings are based on numerical processing and graphical representation of interesting aspects of our results.

5.1 RQ1: What Are the Diagrams Supported by Comparison Techniques?

Table 5 introduces the diagrams supported by the primary studies. These results reveal two interesting findings. First, there may be a relation between diagrams supported by comparison techniques and their adoption. We learned from previous empirical studies [10, 44] that the UML class diagram and generic diagrams are the most commonly used in practice. They were also the most frequently supported diagrams in our primary studies. UML class diagrams and generic diagrams were supported by 39% (22/56) and 30% (17/56) of the studies, respectively. The least adopted diagrams, UML profile, were also the least supported by the comparison approaches.

The second finding is that the primary studies tended to support the main UML behavioral and structural diagrams, such as sequence and class diagrams. This was the case in 46% (26/56) of the primary studies. Chaudron et al. [7] also emphasized that these diagram types are the most commonly used. Dobing and Parsons [10] reported that they were the most commonly used in mainstream software projects. By contrast, the primary studies overlooked diagrams used for specifying business processes (e.g., business process model), software requirements (e.g., UML use case), representing architectural design (e.g., component and connector), describing workflows of stepwise activities and actions supporting iteration and concurrency (e.g., UML activity diagram), and

Table 6. Primary Studies Classified in Relation to Types of Data Structures Used (RQ2)

Data structures	#Studies	Percentage	List of primary studies
Graph	37	66%	[S4], [S8], [S9], [S10], [S12], [S13], [S14], [S18], [S20], [S21], [S22], [S23], [S25], [S26], [S27], [S29], [S30], [S31], [S33], [S34], [S35], [S36], [S39], [S40], [S41], [S42], [S43], [S44], [S46], [S47], [S48], [S49], [S50], [S53], [S54], [S55], [S56]
Tree	6	11%	[S1], [S2], [S3], [S19], [S32], [S38]
Other	13	23%	[S5], [S6], [S7], [S11], [S15], [S16], [S17], [S24], [S28], [S37], [S45], [S51], [S52]
Total	56		

extending the UML metamodel (e.g., UML profiles). These diagrams were supported by only one primary study.

Third, generic diagrams had representative support (30%, 17/56). They were used to represent design decisions more abstractly, like node-link diagrams. The nodes were typically used as entities (i.e., concepts in a particular domain) while links (i.e., relationship between such concepts) were represented as lines. This diagram fitted well in the industry environment, where developers make selective (or informal) use of UML diagrams. Petre [44] reinforced this observation. She interviewed 50 practitioners and concluded that 22% tended to use generic UML diagrams in a personal, selective, and informal way to generically represent software abstractions for as long as they were considered useful. Moreover, little was done to support all diagrams (5%, 3/56), or even a comparison of metamodels (5%, 3/56). Kessentini et al. [21] developed a tool supporting the comparison of metamodels. Some other works [22, 34, 57, 61] have associated the problem of comparing metamodels with the widely known graph isomorphism problems, a well-known class of NP-hard problems.

Finally, while most comparison approaches were developed and assessed to support generic diagrams and UML class diagram (39%, 22/56), not much was done to support such widely used design diagrams as UML sequence, activity diagrams, and business process models. Even worse, knowledge of how to port these techniques to cope with more semantically enriched software design models is still lacking. In particular, the comparison of semantically enriched models (e.g., business process models) is an ever-present problem [46] because semantic information concerning business rules is seldom included in any formal way.

5.2 RQ2: What Are the Commonly Used Data Structures in Current Techniques?

Table 6 presents the commonly used data structures in the primary studies. Grasping a data structure is pivotal to understanding how the software design models are represented to address, for example, performance issues [4]. The collected data highlight a strong trend toward the adoption of graph structures (66%, 37/56). This superior adoption might be explained by the ease of mapping of several software design models into graph. Given that model comparison is strictly related to the problem of graph isomorphism [4], researchers and practitioners end up applying graph matching solutions to comparison problems. We think that the graph is widely recognized as an important data structure, as many software design models can be represented like a graph. Examples vary from UML class diagrams to business process models that are used to carefully represent business rules.

In addition to the use of graphs, the primary studies also adopted trees as data structures in 11% (6/56) of cases. They used trees to represent hierarchical structures, such as component-and-connector diagrams [1]). Van den Brand et al. [60] applied trees to represent metamodels (i.e., a

Table 7. Classification of Primary Studies Based on Their Aspects of Comparison (RQ3)

Comparison aspects	#Studies	Percentage	List of primary studies
Structure	28	50%	[S1], [S2], [S3], [S4], [S5], [S8], [S10], [S14], [S18], [S20], [S21], [S22], [S25], [S28], [S29], [S30], [S31], [S32], [S35], [S36], [S40], [S43], [S44], [S47], [S48], [S49], [S51], [S55]
Multi-criteria	18	32%	[S6], [S12], [S15], [S16], [S17], [S26], [S27], [S33], [S37], [S38], [S39], [S41], [S42], [S46], [S50], [S53], [S54], [S56]
Semantic	5	9%	[S7], [S11], [S13], [S45], [S52]
Lexical	3	5%	[S19], [S24], [S33]
Syntactic	1	2%	[S23]
Layout	1	2%	[S9]
Total	56		

non-hierarchical diagram) in their comparison technique. They argued that prevalent algorithms dealt with comparison in polynomial time, as a structure is imposed on a tree [8, 67]. We note that the choice of data structure is related to performance and precision. These issues are narrowly related to the capability to cope with complex comparisons, e.g., ones caused by severe architecture-level changes. Farias et al. [11, 12] have reported that heuristics to match and combine software design models are inefficient for implementing wide changes in architectural scope, such as the refinement of architecture based on the MVC (model-view-controller) pattern, layout changes that modify the position of model elements, or even the renaming of global variables.

This problem can be explained by the fact that name-based model comparison (sometimes known as signature-based comparison) cannot identify more tangled relationships of equivalence among model elements. We observed that such name-based comparison approaches tended to be limited and error prone whenever there was a 1:N or an N:N correspondence among elements of two or more software design models. These approaches pinpoint equivalences between model elements by comparing the values of their metaproperties, such as *isAbstract* found in the UML class diagram. However, more global correspondences cannot be computed because they compute at most one correspondence for each model element property that has been changed (e.g., *isAbstract*). Finally, graphs and trees are promising data structures for supporting evolving software design models. However, little is known about the extent to which graph- or tree-based comparison techniques may scale up to large software design models with many changes made in parallel.

5.3 RQ3: What Aspects Are Considered for Comparing Design Models?

Table 7 presents the results concerning aspects used to compare software design models. The collected data show that most works (50%, 28/56) relied on structural aspects to differentiate models. Multicriteria comparison (32%, 18/56) was in second place, where two or more characteristics were used for comparison. Semantic aspects (9%, 5/56), in turn, compare models by considering the meanings of their elements. Lexical comparison studies (5%, 3/56) evaluated the distance between labels of model elements. Studies that compared by taking into account syntactic (2%, 1/56) and layout-related aspects (2%, 1/56) were in the minority. We noted the use of several characteristics of models to level up the precision of comparison.

We also noted that syntactic, semantic, and lexical aspects were related to multicriteria comparison. Table 8 specifies each aspect considered in primary studies classified as multicriteria. Therefore, considering only lexical or semantic comparison is not enough to produce accurate

Table 8. The Classification of Primary Studies Based on Their Multicriteria Aspects

Multi-criteria category	List of primary studies
Semantic and Syntactic	[S06], [S17], [S33], [S38], [S39], [S46], [S56]
Lexical and Structural	[S12]
Syntactic and Structural	[S15], [S16], [S42], [S50], [S53], [S54]
Syntactic and Layout	[S27]
Syntactic, Semantic, and Structural	[S37], [S41]

Table 9. Classification of Primary Studies Based on Their Granularity (RQ4)

Granularity	#Studies	Percentage	List of primary studies
Coarse-grained	43	77%	[S1], [S2], [S3], [S4], [S5], [S6], [S7], [S8], [S11], [S13], [S14], [S18], [S19], [S20], [S21], [S22], [S23], [S24], [S25], [S27], [S28], [S29], [S30], [S31], [S32], [S34], [S35], [S36], [S39], [S41], [S42], [S43], [S44], [S45], [S46], [S47], [S48], [S50], [S52], [S53], [S54], [S55], [S56]
Partial	8	14%	[S9], [S10], [S12], [S15], [S16], [S26], [S33], [S40]
Fine-grained	5	9%	[S17], [S36], [S37], [S49], [S51]
Total	56		

comparisons. Moreover, no work has been done on issues related to the comprehensibility of design models. For example, no study has explored layout-related issues. In some situations, the colors and positions of elements helped the relevant approach achieve a more precise result.

5.4 RQ4: How Fine-Grained Is the Measurement of Similarity?

Table 9 presents data collected concerning the granularity of the primary studies. Altmanninger et al. [4] noted that granularity refers to comparison units used to identify the commonalities and differences between input models. The authors also emphasized that granularity depends on the modeling language as each language is defined by its abstract syntax expressed in its metamodel. Thus, comparing two UML models means that the metaproperties defined in the UML metamodel should be considered. Likewise, metaproperties defined in the BPMN 2.0 metamodel should be taken into account to compare two business process models.

The collected data suggest that most primary studies proposed approaches with coarse-grained (77%, 43/56) or partial-grained (14%, 8/56) granularity, rather than fine-grained one (9%, 5/56). This means that the greater part of primary studies considered a set of properties of model elements, rather than taking into account all properties. Still, most current studies did not consider properties of the model elements defined in the metamodel of the relevant modeling language. This reduced their capacity to pinpoint differences among the software design models to be compared. For example, these studies were unable to carefully pinpoint differences among the model elements of semantically enriched and structurally complex software design models.

A persistent challenge is (1) to define the level of granularity of the comparison techniques and (2) to understand their impact on the accuracy and scalability of the comparison of large-scale software design models used in practice. If the level of detail is high, the technique should consider a large number of properties of model elements. With coarse-grained comparison techniques (e.g., match by name), developers can detect correspondences (or even differences) between the names of common model elements as well as those that have been removed, added, changed, or moved.

Table 10. Classification of Primary Studies Based on Their Comparison Types (RQ5)

Comparison types	#Studies	Percentage	List of primary studies
Matching	34	61%	[S1], [S2], [S3], [S4], [S5], [S6], [S7], [S8], [S9], [S10], [S12], [S13], [S14], [S19], [S21], [S22], [S23], [S24], [S25], [S27], [S29], [S30], [S32], [S33], [S34], [S37], [S39], [S43], [S46], [S47], [S48], [S49], [S50], [S56]
Similarity	19	34%	[S15], [S16], [S17], [S18], [S20], [S28], [S29], [S31], [S35], [S36], [S38], [S40], [S41], [S42], [S44], [S52], [S53], [S54], [S55]
Rule-based	3	5%	[S11], [S45], [S51]
Total	56		

Because of their very coarse granularity, however, name-based comparison techniques fail to precisely determine equivalences, for example, wide-scoped equivalences (1:N or N:M), where one model element can be equivalent to "n" model elements.

Finally, we also noted that some works had proposed a rigid number of model properties to be considered. Future techniques should be flexible enough to allow developers to define how fine-grained the comparison technique should be. Thus, the number of model properties to be considered during the comparison step will also be defined based on the purpose of the comparison. For example, developers will be able to choose a smaller number of model properties to compare more abstract software design models (e.g., domain models), whereas a larger number will be considered to compare more detailed software design models (e.g., UML class diagrams used for implementations where the relation between model and code is almost 1:1). Still, this approach might also avoid jeopardizing the flexibility and accuracy of the comparison techniques.

5.5 RQ5: What Are the Types of Model Comparison?

Table 10 introduces the collected data related to the types of model comparison. The majority of primary studies (61%, 34/56) proposed matching approaches. In general, these techniques are used to detect structural, semantic, and syntactic differences (or correspondences). Usually, these techniques produce an output with a set of categorized elements considering the operations performed, such as the removal of, changes to, and addition of model elements. These operations may overlap with each other or occur separately. For example, matching approaches are used to identify changes in software design models to support evolution.

Second, similarity approaches have also been largely covered (34%, 19/56) in academia. This wide adoption can explain the existence of many kinds of metrics that might be the number of nodes [58], neighbors [3], string distance [21], and many others. Third, rule-based approaches were in the minority (5%, 3/56). The small number of rule-based approaches can be explained as follows: First, they make use of uncommon techniques to find correspondences. For example, Maoz et al. [35, 36] proposed a technique based on formal rules in which the results are neither user-friendly nor reproducible. Hence, this reduces the chance of other tools using these techniques. Another example is the Epsilon approach [30], which demands additional effort to specify the comparison rules manually, thereby making the comparison more error prone [13].

Finally, there is a lack of a comprehensive taxonomy of the types of correspondences between software design models. This taxonomy can be supported by more flexible, multicriteria comparison techniques so that more effective correspondence relationships can be produced. This will allow us to better grasp the commonalities and differences between software design models in practical comparison scenarios.

Table 11. Classification of Primary Studies in Relation to Their Research Methods (RQ6)

Research method	#Studies	Percentage	List of primary studies
Proposal of solution	45	80%	[S1], [S2], [S3], [S5], [S6], [S7], [S8], [S9], [S10], [S11], [S12], [S13], [S14], [S15], [S17], [S18], [S19], [S20], [S21], [S22], [S26], [S28], [S29], [S30], [S31], [S32], [S33], [S36], [S38], [S39], [S40], [S41], [S42], [S43], [S44], [S45], [S48], [S49], [S50], [S51], [S52], [S53], [S54], [S55], [S56]
Philosophical paper	5	9%	[S4], [S23], [S27], [S35], [S46]
Evaluation research	4	7%	[S16], [S24], [S25], [S34]
Experience papers	1	2%	[S37]
Validation research	1	2%	[S47]
Opinion papers	0	0%	
Total	56		

5.6 RQ6: What Are the Empirical Methods Used to Evaluate Comparison Techniques?

Table 11 presents the relation between the primary studies and six empirical methods [62]. We discuss these relations as follows: Most studies (80%, 45/56) focused on proposing new solutions. This result suggests that the primary studies were chiefly concerned with closing research gaps by proposing innovative comparison techniques. Second, some studies (9%, 5/56) were classified as philosophical papers. These articles addressed frameworks defining broad concepts and studies explaining future research venues. In this group, three frameworks were proposed [5, 42, 52]. Moreover, Kolovos [31] presented a rule-based approach as a comparison solution, while Ohst et al. [39] analyzed the quality issues related to model comparison tools.

Third, few evaluation researches (7%, 4/56) were proposed in the primary studies. These articles were limited to evaluating their own proposed approaches, often measuring precision and recall under some evaluation scenarios. This means that empirical investigations of the effects of comparison tools have not been explored in a systematic way. Hence, their benefits and drawbacks are still unknown. The presence of few studies reporting evaluation research can be explained by a few reasons. Even though the comparison of design models is an important activity, no single tool has been widely adopted in the industry for it. This can affect the execution of empirical studies, such as case studies and controlled experiments, because they often require stable and easy-to-use tools. This even discourages academia from designing and running empirical studies to evaluate such techniques under different circumstances following a rigorous protocol. The lack of robust and handy techniques makes it challenging for both researchers and practitioners to properly elaborate them in more realistic scenarios or determine the best option under the given circumstances. Another reason is that running careful comparative studies requires at least comparable techniques so that they can be evaluated. Based on the results presented in Section 5.1, for example, the comparison of some diagrams was supported by only one approach. Hence, it is controversial to compare techniques without a range of counterpart approaches. Even worse, little is known about the software design models that offer proper tool support.

Moreover, only one primary study (2%, 1/56) was a validation paper. This means that the literature is only slightly concerned with validating comparison techniques. Studies to test comparison techniques in enterprise environments have been rarely explored [9, 19]. For example, the work of Girschick and Darmstadt [16] was classified as a validation paper because it reported a case study designed to validate the proposed approach to the differentiation of class diagrams. In practice, this validation consisted of a case study run to determine the differences between different versions

Table 12. Classification of Primary Studies Based on Their Degree of Automation (RQ7)

Automation degree	#Studies	Percentage	List of primary studies
Automatic	36	64%	[S2], [S4], [S5], [S6], [S7], [S9], [S10], [S11], [S12], [S13], [S14], [S15], [S16], [S17], [S18], [S21], [S22], [S25], [S26], [S28], [S29], [S31], [S33], [S34], [S35], [S39], [S40], [S44], [S45], [S47], [S49], [S52], [S53], [S54], [S55], [S56]
Semi-automatic	19	34%	[S1], [S3], [S8], [S19], [S23], [S24], [S27], [S30], [S32], [S36], [S38], [S41], [S42], [S43], [S46], [S48], [S50], [S51]
Manual	1	2%	[S17]
Total	56		

of an academic system. However, this validation was limited to only analyzing the results of the evaluated tool. Therefore, this work neither considers a rigorous experimental design, supported by statistical methods, to analyze the results derived from the tools, nor does it explore the gains in productivity from the perspective of the developers. Moreover, the validation should be more rigorous and performed in practical, large-scale production environments so that important issues like reusability, precision, accuracy, and the scalability of comparison approaches can be properly assessed.

Finally, the collected data did not feature any opinion paper. Only one personal experience paper was identified, in which the authors described their experience of conducting manual merge activities. We note that flexible and generic model comparison frameworks are required. At the same time, this area requires more research effort to overcome technical limitations [9].

5.7 RQ7: What Is the Level of Automation of the Comparison Techniques?

Table 12 classifies the primary studies in terms of their degree of automation, including automatic, semi-automatic, or manual. The collected data suggest a strong tendency toward automatic approaches. Most primary studies (64%, 36/56) compared models in an automatic way while a smaller number (34%, 19/56) compared them in a semi-automatic way, where users needed to make adjustments during the comparison. Only one study (2%, 1/56 studies) analyzed how people manually compare UML diagrams.

Usually, automatic approaches allow users to configure how the comparison should be performed. This configuration allows them to explore specific characteristics of the models. To exemplify, we revisit the example shown in Figure 1. We saw there an example of matching rules in which specific features of the models were considered. Formulating rules considering specific characteristics of the models can increase the accuracy of the comparisons, as they consider particularities of the models. However, using such rules does not allow us to customize the comparison at run time. Semi-automatic approaches allow the user to act as the comparison is performed, which enhances the quality of comparison. This is due to the possibility of users customizing the parameters for comparison based on the level of abstraction of software design models, thus improving comparison accuracy. Manual approaches were scarce. Only one study was identified, in which comparison was used to support model integration.

5.8 RQ8: What Are the Most Used Comparison Techniques?

Table 13 presents the most commonly used comparison techniques. Figure 4 shows how the primary studies were published over the years. The collected data revealed some trends. First, most studies (57%, 32/56) used heuristics as the main comparison technique. Tables 14 and 15 specify the

Table 13. Classification of Primary Studies Based on Type of Comparison Technique Used (RQ8)

Comparison technique	#Studies	Percentage	List of primary studies
Heuristic	32	57%	[S1],[S4], [S5], [S9], [S12], [S15], [S16], [S17], [S18], [S19], [S20], [S21], [S22], [S25], [S30], [S32], [S33], [S34], [S35], [S36], [S38], [S39], [S40], [S42], [S43], [S44], [S48], [S50], [S52], [S54], [S55], [S56]
Search-based	8	14%	[S8], [S10], [S26], [S28], [S31], [S41], [S49], [S53]
Rule-based	5	9%	[S11], [S23], [S29], [S45], [S51]
None	4	7%	[S7], [S13], [S37], [S46]
Signature-based	3	5%	[S24], [S27], [S47]
UUID-Heuristics	2	4%	[S3], [S6]
UUID	2	4%	[S2],[S14]
Total	56		

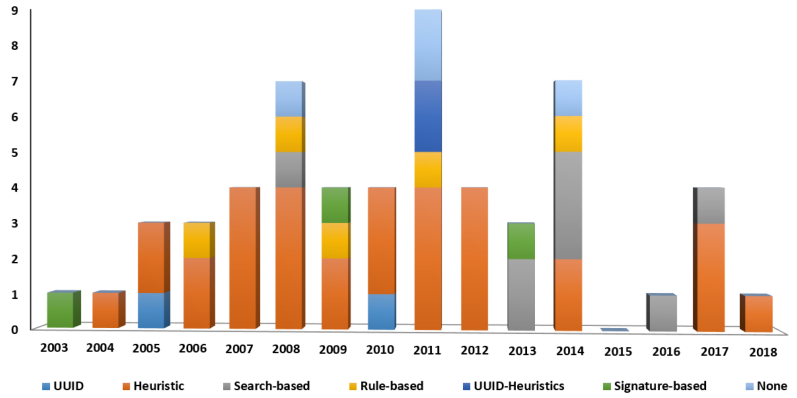


Fig. 4. Number of comparison techniques proposed over the years.

Table 14. Classification of Primary Studies Based on the Heuristics Used to Compare Design Models

Heuristic-based technique	List of primary studies
Static	[S1], [S3], [S4], [S6], [S12], [S17], [S18], [S19], [S20], [S21], [S22], [S25], [S30], [S32], [S33], [S34], [S35], [S36], [S38], [S39], [S40], [S43], [S44], [S48], [S50], [S52], [S54], [S55], [S56]
Static and Behavioral	[S15], [S16]
Tree-to-tree correction optimization	[S1]
Bayesian inference	[S9]

Table 15. Classification of Primary Studies Based on the Search-based Technique Used to Compare Design Models

Search-based technique	List of primary studies
Tabu search	[S8],[S10]
Genetic algorithm	[S28],[S31]
Greedy algorithm	[S41],[S42]
Hybridized Greedy-Genetic algorithm	[S53]
Simulated Annealing	[S26]

heuristics and search-based techniques, respectively, used in the primary studies. This work also categorized heuristics according to Nejati et al. [38], i.e., into on static and behavioral heuristics. A smaller number (14%, 8/56) explored search-based approaches to find correspondences between model elements. The numbers of studies based on rules and signatures registered were small, 9% (5/56) and 5% (3/56), respectively. Moreover, two studies (4%, 2/56) proposed an approach based on UUID and heuristics (classified into both categories). A few studies (7%, 4/56) did not mention the comparison technique used.

The focus on heuristic-based approaches decreased as that on search-based ones increased. In Figure 4, the growth in the number of published works began in 2005, reaching its peak in 2011. Note that in addition to being the most adopted type, the number of publications did not focus on a specific year; on the contrary, studies were published evenly from 2005 to 2012. The number of publications decreased from 2013 to 2016, returning to its average of three papers published per year in 2017, and one paper was published in 2018. Finally, academia aimed to produce heuristic-based model comparison techniques while modeling tools were proposed based on IDs. Furthermore, current commercial tools still do not use a search-based comparison. On the contrary, some articles, e.g., Kpodjedo et al. [32] and Kessentini et al. [21], had applied this technique in 2013 and in 2014, respectively.

5.9 RQ9: Where Were the Studies Published?

This section examines when and where the primary studies were published over the last few years to uncover publication trends. Figure 5 provides a chronology of our primary studies, groups them by publication type, and shows the number of studies per year.

Number of Publications. To reveal the year in which the largest number of articles were published, we assigned one point for each published article. The dashed blue line in Figure 5 summarizes the number of published articles per year. The largest numbers of publications were obtained in 2008, 2011, and 2014. This helps us understand how the studies were distributed and where they were published. The search strategy included studies published until 2018, but no work published before 2003 was considered in the selection procedures (Section 3.5), likely because the MDA (model-driven architecture) and MDE (model-driven engineering) paradigms emerged at the beginning of 2000 and prompted research on model comparison. Note also that between 2003 and 2018, research on model comparison witnessed a strong increase.

Venue of Publication. Figure 5 also shows the percentage in relation to the venue of publication of the primary studies. The results show that 70% (39/56) of the primary studies originated in conferences. Six primary studies (S01, S14, S21, S22, S45, and S28) were published in ASE, one (S11) in MODELS, and three studies (S44, S15, S46) were published in ICSE. Two studies each were published in ICSESS (S26 and S42), and in ICECC (S38 and S47). The other studies were presented at distinct conferences. A total of 25% (14/56) were obtained from journals. Of these, three (S08, S09, and S16) were issued in *IEEE Transactions on Software Engineering (TSE)*, two (S24 and S56) in *Software and Systems Modeling*, one (S30) in *IEEE Software*, one (S35) in *Information and Software Technology (IST)*, and an article (S49) was published in the *Journal of System and Software*. Finally, 5% (3/56) were published in workshops.

Trends. The number of studies grew until 2008. Following this, the frequency of publications began to oscillate, i.e., reaching a minimum number in 2015 without any publication. However, after 2008 the area produced more work. The number of articles published was larger than the range before 2008. Moreover, 2011 witnessed a higher number of articles produced than another years. A considerable number of primary studies was published in high-impact conferences and journals.

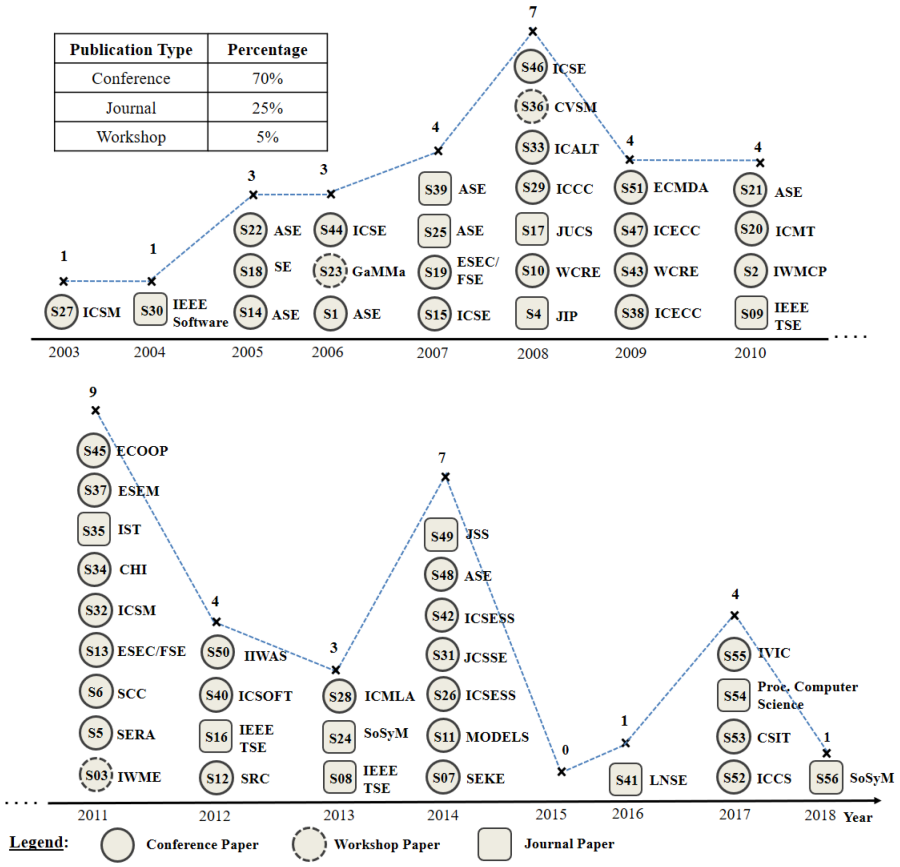


Fig. 5. The research venues that primary studies were published over the last few years.

Five studies (S08, S09, S24, S35, S49) were published in premier journals[3] (*IEEE TSE*, *IST*, *JSS*, and *SoSym*), while 10 papers (S01, S11, S14, S15, S21, S22, S45, S28, S44, S46) were published in *MODELS*, *ICSE*, and *ASE*. Although a considered number of studies were published, there remain outstanding challenges and open questions to be resolved, and these are discussed in the following section.

6 DISCUSSION AND CHALLENGES FOR FUTURE RESEARCH

Figure 6 shows a bubble chart that organizes primary studies in three dimensions (d_1, d_2, d_3), where d_1 represents the research method used (Figure 6(a)) or the supported diagram (Figure 6(b)), d_2 is the publication year, and d_3 is the number of primary studies. Each bubble has values assigned to d_1, d_2, and d_3. This bubble chart helps understand relations among the supported diagrams (RQ1) or research methods used (RQ6), year, and the number of studies. That is, it shows how primary studies were distributed over the past 15 years, considering the research methods used and design models explored. Some insights are as follows:

(1) *Lack of Empirical Studies.* There is a pattern of distribution in the use of research methods over the years. Studies proposing solutions (80%, 45/56) were the most common research method used. The other methods together registered a small number (20%, 11/56), i.e., evaluation research

[3]Journals with h5-median higher than 40 according to the Google Scholar (https://scholar.google.com/).

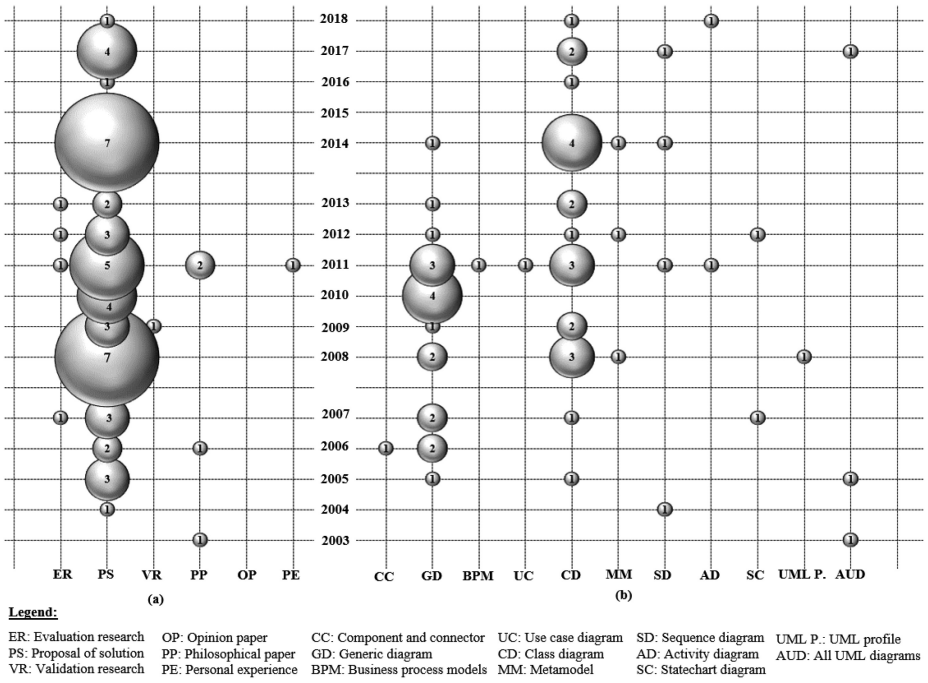


Fig. 6. Bubble chart that shows the relationship among three variables. The *x*-axis shows the research method used (a) and the supported diagrams (b). The *y*-axis represents year. The size of the disks consists of the number of primary studies classified according to the criteria along the *x*- and *y*-axes.

(7%, 4/56), philosophical papers (9%, 5/56), personal experience (2%, 1/56), validation research (2%, 1/56), and opinion papers (no case). That is, the number of studies proposing solutions was almost four times higher. This may mean that empirical studies are needed to evaluate methods that have already been proposed in the literature.

Another interesting feature is that the primary studies mostly focused on proposing comparison techniques for class (39%, 22/56) and generic (30%, 17/56) diagrams over the years, representing 69% (39/56) of the total. On the contrary, other diagrams were investigated at most in three primary studies. In 2008, seven proposals covered four types of diagrams. Specifically, three studies focused on class diagrams, two on generic diagrams, one on metamodels, and one on UML profiles. Moreover, generic diagrams received strong attention for approximately three years in academia from 2008 to 2011. In practical terms, this means that the literature is still in its infancy. Only early proof-of-concept techniques have been proposed, rather than user-friendly, handy techniques with verified effectiveness in practical scenarios. Further, few empirical studies have been performed to evaluate whether techniques in the literature are reliable for application to software projects in the industry, where there are limitations of cost and time.

(2) *Approaches Aware of Architecturally Relevant Changes.* In general, several techniques have been proposed in the literature to compare software design models by considering several aspects, including structural, syntactical, semantic, lexical, layout-related, and even multicriteria approaches (Table 7). However, at a closer look at the available approaches, most tend to follow a primitive approach of structural comparison. This allows them to resolve a set of elementary comparison cases. Nevertheless, these approaches are prone to error in determining correspondences between semantically enriched design models. They are often unable to pinpoint structural

similarities, or differences, caused by restructuring changes, for example. This type of modification is typically found in changes that are wide in scope in software design models owing to refactoring tasks or the refinement of architectural styles, such as client–server, component-based, MVC, three-tier, or service-oriented styles). Therefore, an interesting direction of research is to create comparison techniques that are aware of coarse-grained structural changes (N: N), rather fine-grained ones (1:1). Thus, a relevant question is: To what extent can current techniques produce appropriate correspondences between elements of the design model in the context of architecturally relevant changes?

(3) *Hybrid Techniques to Boost Quality of Comparison.* Another avenue of research is to build hybrid comparison techniques based on features of prevalent methods. For example, it would be helpful for the seamless integration of aspects of comparison (e.g., structural, syntactical, semantic, lexical, and layout-related) into a flexible multicriteria approach so that more precise and adaptable techniques can be produced. For example, developers can improve precision if they can tailor the level of detail of the comparison process according to an appropriate level of abstraction of the diagrams. In model-based software development, developers can derive a UML class diagram from a UML component diagram. Thus, this is relevant to understand the extent to which a UML class diagram (more detailed) is comparable (or close to) to a particular UML component diagram (more abstract).

(4) *Effects of Comparison Approaches on Quality Attributes.* Another interesting direction of research is the generation of empirical evidence concerning the quality of current techniques, e.g., granularity, accuracy, and scalability. Evaluations to date have been based on the opinions and intuitions of experts rather than practical evidence. We also found evidence that the adoption of comparison techniques in projects has not been supported by findings obtained from experimental studies, but these findings are supported by mere opinions of experts [11, 13].

Most comparison techniques have been evaluated using simple software design models, usually represented in generic modeling languages, rather than using semantically enriched software design models, e.g., business process models. For example, BPMN 2.0 [40] presents a standard for business process modeling by providing a graphical notation for specifying business processes. It is inspired by a flowcharting technique analogous to activity diagrams found in the UML. Although they are based on flowcharts, their constructs are semantically different. Thus, it is questionable whether findings derived from an empirical study using simple models are generalizable to models that are semantically enriched. The practical reality of software projects today is that modelers have been defining the "goodness" of correspondences between model elements supported based on feedback from experts. Therefore, researchers and practitioners should focus on running more experimental studies to address practical issues, which not only consider the accuracy of the correspondence identified, but are also concerned with the effort and time needed to identify the desired correspondences between software design models. In the following section, we describe some implications for future studies in greater detail.

7 THREATS TO VALIDITY

This section discusses measures adopted to minimize threats to the validity of our results. Many aspects threaten the validity of this study, such as the validity of the construct, internal validity, and the validity of the statistical conclusions [63]. External validity was not considered because our work cannot be generalized to the industry.

Construct Validity. This discusses the statement validity in this article. It consists of the degree to which our SMS explores what it claims to be exploring. The mismatch between our search string and the keywords of the study may cause some studies to be excluded. To avoid this problem, we

used rigorous procedures for retrieving and filtering the potential studies. Keywords and their synonyms were defined according to well-established methods presented in previous studies [24, 43]. We also included a long list of search engines (Table 3) and added studies by heuristics. Moreover, we avoided grey literature (non-peer-reviewed material), such as, technical reports, or documents describing that a patent has been registered, where the review process is more superficial than in traditional journals, or conferences. Finally, to ensure the removal of duplicated studies, we verified the title and article's full-text in a separate directory.

Internal Validity. This determines whether the conclusions derived from the data are internally valid [55]. In this sense, we categorized the primary studies by considering the hierarchical structure proposed by Altmanninger et al. [4], and Mens [37], to avoid incorrect classification. Nonetheless, understanding and mapping each technique can itself be seen as a threat to validity. We thus sought to mitigate this risk by discussing and analyzing all papers. Moreover, an ever-present concern throughout the study was to ensure that the selected primary studies were consistently identified and analyzed. For this, we invested considerable effort in screening the primary studies.

Conclusion Validity. This threat is strictly related to problems that can affect the reliability of our conclusions. To mitigate this, we followed the steps provided by well-established SMS protocols [23–26, 43]. We also extensively described all steps of the SMS. Finally, all the conclusions in this article were made after collecting the results, thus avoiding the fishing problem [63].

8 RELATED WORK

The analysis in this study contributes to (a) providing an in-depth understanding of state-of-the-art model comparison in a broader context (Section 5), and (b) pinpointing open questions and future opportunities, and outlining important issues that need to be tackled (Section 6). Table 16 presents an overview of six related works (RWs) that were explored, and their main features are highlighted.

RW1 [57] presented a state-of-the-art survey of model comparison. It covered 31 articles published until 2013, considered proprietary tools, but did not follow a research protocol. It examined techniques that support two branches of applications: model evolution and similarity. The authors considered four types of comparison strategies: static identity, similarity, signature, and custom language specific. On the contrary, we considered structural, lexical, syntactic, layout-related, and multicriteria strategies. Finally, the authors of RW1 measured the "work required for comparison" by evaluating the usability of the comparison tools. However, they did not explore the literature using a rigorous research protocol.

RW2 [4] examined contemporary techniques of the version control system (VCS) for model artifacts. Ten works published until 2009 were analyzed through a survey. This work emphasized the key role of model comparison in supporting model versioning, and described the main aspects of model comparison, such as level of granularity, techniques used to compare models, and the data structure used to represent them. Although VCS encompasses several comparison activities, and is representative of broader issues, no claim was made about how generalizable such activities or the results are. The processes of study selection and filtering used are also not clear.

RW3 [30] conducted a comparative study of 11 approaches to model comparison published till 2009. The main aim was to compare the state-of-the-art model matching approaches based on the type of comparison (i.e., signature, static identity, or similarity based) and flexibility (i.e., independent or specific language). Following this, RW3 carried out a case study investigating how four comparison approaches behaved with the same input models, i.e., a UML class diagram. The effort for the comparison and the required set-up to perform the comparison were discussed. The authors reported that the more configurable the precision and the more manageable the effort to be invested, the better the comparison technique.

Table 16. Main Features about the Related Works Explored

ID	Research Method	Sample	Search Protocol	Goals	Search Period	Research questions*
RW1 [57]	Survey	31	No	Presents the current state of model comparison research and discusses future directions of research.	Until 2013	What is the state of the art in model comparison research?
RW2 [4]	Survey	10	No	Describes the functionalities of techniques on model versioning systems. Identifies challenges to model versioning.	Until 2009	What is the current literature in model versioning approaches?
RW3 [30]	Comparative Study	11	No	Investigates contemporary approaches to model matching, which were evaluated using accuracy and precision as measures.	Until 2009	What are the approaches to identifying equivalent model elements?
RW4 [50]	Systematic Literature Review	17	Yes	Points out and reviews current works on traceability management in MDE (model-driven engineering).	Until 2011	"RQ1: What is the level of automation suggested by methodological proposals for the generation of trace links? RQ2: How do methodological proposals suggest that traceability be managed and analyzed? RQ3: Are there tools or frameworks that provide technological support for the management of traceability in the context of MDE? RQ4: What are the limitations of the state of the art in traceability management in the context of MDE? RQ5: Are there forums (e.g., journals or conferences) that explore traceability management in MDE?"
RW5 [49]	Survey	19	No	Investigates current approaches to UML artifacts' reuse. Investigates the retrieval of design models, tool and artifact support, and the controlled experiments produced.	Until 2013	What is the current literature on UML artifacts' reuse?
RW6 [54]	Literature review	5	No	Presents a synthesis of key characteristics of currently available UML-based model comparison approaches, and compares commonalities and differences.	Until 2007	What are UML-based model comparison approaches?

*Research questions identified in the studies

RW4 [50] analyzed literature on the traceability management of design models for MDE approaches. The authors ran an (SLR systematic literature review) [24]. Five research questions were used to examine 29 primary studies published until 2011, and specific issues, such as the level of automation, limitations in current studies, and common places where traceability management studies were published, were addressed. RW4 aimed at specific research questions in traceability management in MDE. By contrast, our mapping study aimed at broader topics concerning model comparison.

RW5 [49] analyzed literature on UML diagram reuse published until 2012. It examined 15 works based on four dimensions: artifact support, retrieval method, experiments performed, and tool support. The study revealed that the UML class, sequence diagrams, and use case diagrams are the UML artefacts most commonly supported by tool. The authors also recommended future directions of research on artifacts reuse. Similar to Stephan and Cordy [57] and Altmanninger et al. [4], this analysis did not follow a rigorous research protocol to survey the literature.

RW6 [54] explored five studies published until 2007 to synthesize the key characteristics of UML-based approaches to model comparison. The study explored potential comparison usage

scenarios and motivated research by citing a lack of clear understanding of how corresponding model elements are identified. RW6 did not follow a rigorous research protocol for selecting studies, and extracting and synthesizing the collected data as presented in our study (Section 3).

9 CONCLUSION AND FUTURE WORK

This article reported an SMS on model comparison. We performed an in-depth literature review using 10 widely used electronic databases. In total, 56 primary papers were selected after a careful filtering process applied to a sample of 4,132 potentially relevant studies. The work was motivated by the idea that a lack of a comprehensive overview and understanding of past results in the field of model comparison hampers surveys focusing on research gaps, challenges, and trends. Researchers and developers may benefit from our findings typically when starting a new research, adopting a methodology to run a study considering a broader software engineering topic than was considered in an initial similar study, developing new techniques of comparison by identifying gaps in the literature, and choosing reusable research skills, like the proposed SMS planning.

Moreover, some research opportunities reported are as follows: (1) Research on how to compare software design models based on quality attributes is still lacking; (2) Further studies are needed to develop comparison techniques and tools that can address more robust comparisons of realistic software design models of open-source projects, including cloud-based and service-oriented projects with rapid development cycles; and (3) A refined classification scheme can be designed based on the findings presented here. Such a scheme, derived from the concepts presented in Section 3.4, will allow for the re-categorization of primary studies, and can provide a firm ground for pinpointing related research in sub-fields of model comparison.

Further empirical studies are required to generate practical knowledge and advice for researchers and practitioners concerning the circumstances in which a particular comparison technique (e.g., rule-based technique) would be acceptable, those in which another (e.g., heuristic-based technique) would be the best option. Despite all research efforts, the literature is lacking studies on the large-scale collaborative development of software models by focusing on scalability and applicability. More empirical evidence on the reusability, precision, accuracy, and scalability of comparison approaches is necessary, and the amount of time and effort needed to detect and resolve equivalences between software design models need to be investigated. We also need easier ways to specify which parts of software design models that must be compared, and efficient ways of managing similar parts as models on an enterprise software system grow in size.

Finally, other investigations concerning scalability may examine how comparison techniques perform with multiple fragments of software design models created or changed in parallel by different software-development teams. Usually, developers work on changes in parallel to the same software artefact. Then, prevalent techniques should be able to simultaneously compare more than two parallel versions of software design models. However, most techniques support only a comparison of two software design models at a time. We also expect that the results and findings outlined here can motivate researchers to examine them further. This work can be considered an initial step on improving surveys of state-of-the-art techniques of comparison.

APPENDIXES

A LIST OF SELECTED PRIMARY STUDIES

S01 M. Abi-Antoun, J. Aldrich, N. Nahas, B. Schmerl, & D. Garlan. 2008. Differencing and merging of architectural views. In Int. Conf. on Automated Software Engineering, v 15, n 1, pp 35-74.

S02 M. Brand, Z. Protic, & T. Verhoeff. 2010. Fine-grained Metamodel-assisted Model Comparison. In Int. Workshop on Model Comparison in Practice, pp. 11-20, July, Malaga, Spain.

S03 M. Brand, Z. Protic, & T. Verhoeff. 2010. RCVDiff-a stand-alone tool for representation, calculation and visualization of model differences, In Int. Workshop on Models and Evolution-ME.

S04 C. Brun & A. Pierantonio. 2008. Model differences in the eclipse modeling framework. Upgrade Journal, v 9, n 2, pp. 29-34.

S05 M. El-Attar. 2011. UseCaseDiff: An Algorithm for Differencing Use Case Models. Int. Conf. on Soft. Eng. Res. Man. and App, pp. 148-152.

S06 C. Gerth, M. Luckey, J. M. Kuster, & G. Engels. 2011. Precise Mappings between Business Process Models in Versioning Scenarios. In IEEE International Conference on Services Computing, pp. 218-225.

S07 V. Costa, R. Monteiro, & L. Murta. 2014. Detecting Semantic Equivalence in UML Class Diagrams. In International Conference on Software Engineering and Knowledge Engineering, pp. 318-323.

S08 S. Kpodjedo, F. Ricca, P. Galinier, G. Antoniol, & Y. Gueheneuc. 2013. MADMatch: Many-to-Many Approximate Diagram Matching for Design Comparison. IEEE Transactions on Software Engineering, v 39, n 8, pp. 1090-1111.

S09 D. Kimelman, M. Kimelman, D. Mandelin, & D. Yellin. 2010. Bayesian Approaches to Matching Architectural Diagrams. IEEE Transactions on Software Engineering, v. 36, n. 2, pp. 248-274.

S10 S. Kpodjedo, F. Ricca, P. Galinier, & G. Antoniol. 2008. Error Correcting Graph Matching Application to Software Evolution. In Working Conference on Reverse Engineering, pp. 289-293.

S11 P. Langer, T. Mayerhofer, & G. Kappel. 2014. Semantic Model Differencing Utilizing Behavioral Semantics Specifications, In Model-Driven Engineering Languages and Systems, v. 8767, pp. 116-132.

S12 Q. Liu, M. Mernik, & B. Bryant. 2012. MMDiff: A Modeling Tool for Metamodel Comparison, In Annual South. Reg. Conf., pp. 118-123.

S13 S. Maoz, J. Ringert, & B. Rumpe. 2011. ADDiff: Semantic Differencing for Activity Diagrams, ACM Symp. and European Conference on Foundations of Software Engineering, pp. 179-189, Sept., Szeged, Hungary.

S14 A. Mehra, J. Grundy, & J. Hosking. 2005. A Generic Approach to Supporting Diagram Differencing and Merging for Collaborative Design, Int. Conf. on Automated Software Engineering, pp. 204-213, Nov., Long Beach, CA, USA.

S15 S. Nejati, M. Sabetzadeh, M. Chechik, S. Easterbrook, & P. Zave. 2007. Matching and Merging of Statecharts Specifications, International Conference on Software Engineering, pp. 54-64.

S16 S. Nejati, M. Sabetzadeh, M. Chechik, S. Easterbrook, & P. Zave. 2012. Matching and merging of variant feature specifications. IEEE Transactions on Software Engineering, v 38, n 6, pp. 1355-1375.

S17 K. Oliveira, K. Breitman, & T. Oliveira. 2009. A Flexible Strategy-Based Model Comparison Approach: Bridging the Syntactic and Semantic Gap. Journal of Universal Computer Science, v 15, n 11, pp. 2225-2253.

S18 U. Kelter, J. Wehren, & J. Niere. 2005. A Generic Difference Algorithm for UML Models. Software Engineering, 64, pp. 4-9.

S19 C. Treude, S. Berlik, S. Wenzel, & U. Kelter. 2007. Difference Computation of Large Models, Joint Meeting of the Eur. Soft. Eng. Conf. and the ACM SIGSOFT Symp. on the Foundations of Software Engineering, pp. 295-304, Croatia.

S20 K. Voigt & T. Heinze. 2010. Metamodel Matching Based on Planar Graph Edit Distance, International Conference on Theory and Practice of Model Transformations, pp. 245-259.
S21 Z. Xing. 2010. Model Comparison with GenericDiff, IEEE/ACM Int. Conf. on Automated Software Engineering, pp. 135-138, Sept., Antwerp, Belgium.
S22 Z. Xing & E. Stroulia. 2005. UMLDiff: An Algorithm for Object-oriented Design Differencing, Automated Soft. Eng., pp. 54-65.
S23 D. Kolovos, R. Paige, & F. Polack. 2006. Model Comparison: A Foundation for Model Composition and Model Transformation Testing, Int. Workshop on Global Integrated Model Management, pp. 13-20, Shanghai, China.
S24 V. Milovanovic & D. Milicev. 2015. An interactive tool for UML class model evolution in database applications, Software & Systems Modeling, n 14, v 3, pp. 1273-1295.
S25 Z. Xing & E. Stroulia. 2007. Differencing logical UML models, Int. Conf. on Automated Software Engineering, v. 14, n. 2, pp 215-259.
S26 M. Al-Khiaty & M. Ahmed. 2014. Similarity assessment of UML class diagrams using a greedy algorithm, International Computer Science and Engineering Conference, pp. 228-233.
S27 D. Ohst, M. Welle, & U. Kelter. 2003. Difference tools for analysis and design documents, In Proc. of Int. Conf. on Soft. Maint., pp. 13-22.
S28 H. Salami & M. Ahmed. 2013. Class Diagram Retrieval Using Genetic Algorithm, Int. Conf. on Machine Learning. and App., pp. 96-101.
S29 M. Asztalos & L. Lengyel. 2008. A Metamodel-Based Matching Algorithm for Model Transformations, In IEEE International Conference on Computational Cybernetics, pp. 151-155.
S30 W. Robinson & H. Woo. 2004. Finding reusable UML sequence diagrams automatically. IEEE Software, v. 21, n. 5, pp. 60-67.
S31 H. Salami & M. Ahmed. 2014. Retrieving sequence diagrams using genetic algorithm, International Joint Conference on Computer Science and Software Engineering, pp. 324-330.
S32 N. Tsantalis, N. Negara & E. Stroulia. 2011. Webdiff: A generic differencing service for software artifacts, International Conference on Software Maintenance, pp. 586-589.
S33 L. Auxepaules, D. Py & T. Lemeunier. 2008. A Diagnosis Method that Matches Class Diagrams in a Learning Environment for Object-Oriented Modeling, International Conference on Advanced Learning Technologies, pp. 26-30.
S34 L. Zaman, A. Kalra, & W. Stuerzlinger. 2011. DARLS: Differencing and Merging Diagrams Using Dual View, Animation, Re-layout, Layers and a Storyboard, In Extended Abstracts on Human Factors in Computing Systems, pp. 1657-1662.
S35 W. Park & D. Bae. 2011. A two-stage framework for UML specification matching, Inf. and Software Technology, v. 53, n. 3 pp. 230-244.
S36 S. Uhrig. 2008. Matching Class Diagrams: With Estimated Costs Towards the Exact Solution? International Workshop on Comparison and Versioning of Software Models, pp. 7-12.
S37 R. Lutz, D. Wurfel & S. Diehl. 2011. How Humans Merge UML-Models, Int. Symp. on Emp. Soft. Eng. and Measurement, pp. 177-186.
S38 K. Oliveira, K. Breitman & T. Oliveira. 2009. Ontology Aided Model Comparison, Int. Conf. on Engineering of Complex Computer Systems, pp. 78-83.
S39 Y. Lin, J. Gray, & F. Jouault. 2007. DSMDiff: a differentiation tool for domain-specific models, European Journal of Information Systems, v. 16, n. 4, pp. 349-361.
S40 J. Su & J. Bao. 2012. Measuring UML Model Similarity, Int. Conference on Software Paradigm Trends, pp. 319-323.
S41 M. Al-Khiaty & M. Ahmed. 2016. UML Class Diagrams: Similarity Aspects and Matching. Lec. Notes on Soft. Engineering, v. 4, pp. 41-47.

S42 M. Al-Khiaty & M. Ahmed. 2014. Similarity assessment of UML class diagrams using a greedy algorithm, International Computer Science and Engineering Conference, pp. 228-233.
S43 K. Bogdanov & N. Walkinshaw. 2009. Computing the structural difference between state-based models, In Working Conference on Reverse Engineering, pp. 177-186.
S44 D. Mandelin, D. Kimelman, & D. Yellin. 2006. A Bayesian Approach to Diagram Matching with Application to Architectural Models, International Conference on Software Engineering, pp. 222-231, Shanghai, China.
S45 S. Maoz, J. Ringert, & B. Rumpe. 2011. CDDiff: Semantic Differencing for Class Diagrams, European Conf. on Object-Oriented Programming, v. 6813, pp. 230-254.
S46 M. Schmidt & T. Gloetzner. 2008. Constructing Difference Tools for Models Using the SiDiff Framework, Conf. Soft. Eng., pp. 947-948.
S47 M. Girschick & T. Darmstadt. 2006. Difference detection and visualization in UML class diagrams, Technical University of Darmstadt, Technical Report, pp. 1-15.
S48 R. Lutz & S. Diehl. 2014. Using Visual Dataflow Programming for Interactive Model Comparison. International Conference on Automated Software Engineering, pp. 653-664, Sept., Sweden.
S49 M. Kessentini, A. Ouni, P. Langer, M. Wimmer, & S. Bechikh. 2014. Search-based Metamodel Matching with Structural and Syntactic Measures, Journal of Systems and Software, v 97, pp. 1-14.
S50 G. M. Kapitsaki & A. P. Achilleos. 2012. Model Matching for Web Services on Context Dependencies, In International Conference on Information Integration and Web-based Applications Services, pp. 45-53, Dec., Bali, Indonesia.
S51 D. Kolovos. 2009. Establishing Correspondences between Models with the Epsilon Comparison Language, Model Driven Architecture: Foundations and Applications, pp. 146-157, June, Netherlands.
S52 E. Kaundal & E. Kaur. 2017. CoMSS: Context based Measure for Semantic Similarity between Conceptual models. International Conference on Intelligent Computing and Control Systems, pp. 1080-1088, June, India.
S53 M. AL-Khiaty & M. Ahmed. 2017. Matching UML Class Diagrams using a Hybridized Greedy-Genetic Algorithm. International Scientific and Technical Conference on Computer Sciences and Information Technologies, pp. 161-166, Sept., Lviv, Ukraine.
S54 A. Adamua & W. Zainon. 2017. Multiview Similarity Assessment Technique of UML Diagrams. Proc. Comp. Science, v. 124, pp. 311-318.
S55 A. Adamua & W. Zainon. 2017. Similarity Assessment of UML Sequence Diagrams Using Dynamic Programming, Int. Visual Inform. Conf., pp. 270-278.
S56 S. Maoz, & J. O. Ringert. 2018. A framework for relating syntactic and semantic model differences. Software & Systems Modeling, v. 17, n.3, pp 753-777.

B GENERAL CLASSIFICATION AND QUALITY ASSESSMENT OF PRIMARY STUDIES

Table B.1. General Classification and Quality Assessment of Primary Studies

	General classification									Quality Assessment									
ID	RQ1	RQ2	RQ3	RQ4	RQ5	RQ6	RQ7	RQ8	RQ9	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Points
S1	Component and Connector	Tree	Structure	Coarse-grained	Matching	Proposal of Solution	Semi-automatic	Heuristic	Conference	✓	✓	✓	✓	✓	✗	✓	✓	✗	7
S2	Generic Diagram	Tree	Structure	Coarse-grained	Matching	Proposal of Solution	Automatic	UUID	Workshop	✓	✓	✓	✓	✗	✗	✓	✓	✓	7
S3	Generic Diagram	Tree	Structure	Coarse-grained	Matching	Proposal of Solution	Semi-automatic	UUID-Heuristics	Workshop	✓	✓	✓	✓	✓	✗	✓	✓	✗	7
S4	Generic Diagram	Graph	Structure	Coarse-grained	Matching	Philosophical Paper	Automatic	Heuristic	Journal	✓	✓	✗	✓	✓	✗	✓	✓	✓	7
S5	Use case	Other	Structure	Coarse-grained	Matching	Proposal of Solution	Automatic	Heuristic	Conference	✓	✓	✗	✓	✓	✗	✓	✓	✓	7
S6	Business Process Model	Other	Multi-criteria	Coarse-grained	Matching	Proposal of Solution	Automatic	UUID-Heuristics	Conference	✓	✓	✓	✓	✓	✗	✓	✓	✓	8
S7	Class Diagram	Other	Semantic	Coarse-grained	Matching	Proposal of Solution	Automatic	None	Conference	✓	✓	✓	✓	✓	✗	✓	✓	✓	8
S8	Generic Diagram	Graph	Structure	Coarse-grained	Matching	Proposal of Solution	Semi-automatic	Search-based	Journal	✓	✓	✓	✓	✓	✗	✓	✓	✓	8
S9	Generic Diagram	Graph	Layout	Partial	Matching	Proposal of Solution	Automatic	Heuristic	Journal	✓	✓	✓	✓	✓	✗	✓	✓	✓	8
S10	Class Diagram	Graph	Structure	Partial	Matching	Proposal of Solution	Automatic	Search-based	Conference	✓	✓	✓	✓	✓	✗	✓	✓	✓	8
S11	Generic Diagram	Other	Semantic	Coarse-grained	Rule-based	Proposal of Solution	Automatic	Rule-based	Conference	✓	✓	✓	✓	✓	✗	✓	✓	✓	8
S12	Metamodels	Graph	Multi-criteria	Partial	Matching	Proposal of Solution	Automatic	Heuristic	Conference	✓	✓	✓	✓	✓	✗	✓	✓	✓	8
S13	Activity Diagram	Graph	Semantic	Coarse-grained	Matching	Proposal of Solution	Automatic	None	Conference	✓	✓	✓	✓	✓	✗	✓	✓	✗	7
S14	Generic Diagram	Graph	Structure	Coarse-grained	Matching	Proposal of Solution	Automatic	UUID	Conference	✓	✓	✓	✓	✓	✗	✓	✓	✗	7
S15	Statechart	Other	Multi-criteria	Partial	Similarity	Proposal of Solution	Automatic	Heuristic	Conference	✓	✓	✓	✓	✓	✗	✓	✓	✓	7
S16	Statechart	Other	Multi-criteria	Partial	Similarity	Evaluation Research	Automatic	Heuristic	Journal	✓	✓	✗	✓	✓	✗	✓	✓	✓	7
S17	UML Profile	Other	Multi-criteria	Fine-grained	Similarity	Proposal of Solution	Automatic	Heuristic	Journal	✓	✓	✓	✓	✓	✗	✓	✓	✓	8
S18	All UML Diagrams	Graph	Structure	Coarse-grained	Similarity	Proposal of Solution	Automatic	Heuristic	Conference	✓	✓	✓	✓	✓	✗	✓	✓	✗	7

(Continued)

Table B.1. Continued

	General classification									Quality Assessment									
ID	RQ1	RQ2	RQ3	RQ4	RQ5	RQ6	RQ7	RQ8	RQ9	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Points
S19	Generic Diagram	Tree	Lexical	Coarse-grained	Matching	Proposal of Solution	Semi-automatic	Heuristic	Conference	✓	✓	✓	✓	✓	✗	✓	✓	✗	7
S20	Generic Diagram	Graph	Structure	Coarse-grained	Similarity	Proposal of Solution	Semi-automatic	Heuristic	Conference	✓	✓	✓	✓	✓	✗	✓	✓	✓	8
S21	Generic Diagram	Graph	Structure	Coarse-grained	Matching	Proposal of Solution	Automatic	Heuristic	Conference	✓	✓	✓	✓	✓	✗	✓	✓	✓	8
S22	Class Diagram	Graph	Structure	Coarse-grained	Matching	Proposal of Solution	Automatic	Heuristic	Conference	✓	✓	✓	✓	✓	✗	✓	✓	✓	8
S23	Generic Diagram	Graph	Syntactic	Coarse-grained	Matching	Philosophical Paper	Semi-automatic	Rule-based	Workshop	✓	✓	✓	✓	✓	✗	✓	✓	✓	8
S24	Class Diagram	Other	Lexical	Coarse-grained	Matching	Evaluation Research	Semi-automatic	Signature-based	Journal	✓	✓	✓	✓	✓	✗	✓	✓	✓	8
S25	Class Diagram	Graph	Structure	Coarse-grained	Matching	Evaluation Research	Automatic	Heuristic	Journal	✓	✓	✓	✓	✓	✗	✓	✓	✓	8
S26	Class Diagram	Graph	Multi-criteria	Partial	Similarity	Proposal of Solution	Automatic	Search-based	Conference	✓	✓	✓	✓	✓	✗	✓	✓	✓	8
S27	All UML Diagrams	Graph	Multi-criteria	Coarse-grained	Matching	Philosophical Paper	Semi-automatic	Signature-based	Conference	✓	✓	✓	✓	✓	✗	✓	✓	✓	8
S28	Class Diagram	Other	Structure	Coarse-grained	Similarity	Proposal of Solution	Automatic	Search-based	Conference	✓	✓	✓	✓	✓	✗	✓	✓	✓	8
S29	Metamodels	Graph	Structure	Coarse-grained	Matching	Proposal of Solution	Automatic	Rule-based	Conference	✓	✓	✓	✓	✗	✗	✓	✓	✓	8
S30	Sequence Diagram	Graph	Structure	Coarse-grained	Matching	Proposal of Solution	Semi-automatic	Heuristic	Journal	✓	✓	✗	✓	✓	✗	✓	✓	✗	6
S31	Sequence Diagram	Graph	Structure	Coarse-grained	Similarity	Proposal of Solution	Automatic	Search-based	Conference	✓	✓	✓	✓	✓	✗	✓	✓	✗	7
S32	Class Diagram	Tree	Structure	Coarse-grained	Matching	Proposal of Solution	Semi-automatic	Heuristic	Conference	✓	✓	✗	✓	✗	✗	✓	✓	✓	6
S33	Class Diagram	Graph	Multi-criteria	Partial	Matching	Proposal of Solution	Automatic	Heuristic	Conference	✓	✓	✗	✓	✓	✗	✓	✓	✓	7
S34	Generic Diagram	Graph	Lexical	Coarse-grained	Matching	Evaluation Research	Automatic	Heuristic	Conference	✓	✓	✓	✓	✗	✗	✓	✓	✓	7
S35	Class and Sequence Diagram	Graph	Structure	Coarse-grained	Similarity	Philosophical Paper	Automatic	Heuristic	Journal	✓	✓	✓	✓	✓	✗	✓	✓	✓	8

(Continued)

Table B.1. Continued

	General classification									Quality Assessment									
ID	RQ1	RQ2	RQ3	RQ4	RQ5	RQ6	RQ7	RQ8	RQ9	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Points
S36	Class Diagram	Graph	Structure	Coarse-grained	Similarity	Proposal of Solution	Semi-automatic	Heuristic	Workshop	✓	✓	✗	✓	✗	✗	✓	✓	✓	6
S37	Class Diagram	Other	Multi-criteria	Fine-grained	Matching	Experience Papers	Manual	None	Conference	✓	✓	✓	✓	✓	✓	✓	✓	✗	8
S38	Class Diagram	Tree	Multi-criteria	Fine-grained	Similarity	Proposal of Solution	Semi-automatic	Heuristic	Conference	✓	✓	✗	✓	✗	✗	✓	✓	✓	6
S39	Generic Diagram	Graph	Multi-criteria	Coarse-grained	Matching	Proposal of Solution	Automatic	Heuristic	Journal	✓	✓	✓	✓	✓	✗	✓	✓	✗	7
S40	Class Diagram	Graph	Structure	Partial	Similarity	Proposal of Solution	Automatic	Heuristic	Conference	✓	✓	✗	✓	✓	✗	✓	✓	✓	7
S41	Class Diagram	Graph	Multi-criteria	Coarse-grained	Similarity	Proposal of Solution	Semi-automatic	Search-based	Journal	✓	✓	✗	✓	✓	✗	✓	✓	✓	7
S42	Class Diagram	Graph	Multi-criteria	Coarse-grained	Similarity	Proposal of Solution	Semi-automatic	Heuristic	Conference	✓	✓	✗	✓	✓	✗	✓	✓	✓	7
S43	Statechart	Graph	Structure	Coarse-grained	Matching	Proposal of Solution	Semi-automatic	Heuristic	Conference	✓	✓	✓	✓	✓	✗	✓	✓	✓	8
S44	Generic Diagram	Graph	Structure	Coarse-grained	Similarity	Proposal of Solution	Automatic	Heuristic	Conference	✓	✓	✗	✓	✓	✗	✓	✓	✓	7
S45	Class Diagram	Other	Semantic	Coarse-grained	Rule-based	Proposal of Solution	Automatic	Rule-based	Conference	✓	✓	✓	✓	✓	✗	✓	✓	✓	8
S46	Generic Diagram	Graph	Multi-criteria	Coarse-grained	Matching	Philosophical Paper	Semi-automatic	None	Conference	✓	✓	✗	✓	✗	✗	✓	✓	✗	5
S47	Class Diagram	Graph	Structure	Coarse-grained	Matching	Validation Research	Automatic	Signature-based	Conference	✓	✓	✓	✓	✓	✗	✓	✓	✓	8
S48	Class Diagram	Graph	Structure	Coarse-grained	Matching	Proposal of Solution	Semi-automatic	Heuristic	Conference	✓	✓	✓	✓	✓	✗	✓	✓	✗	7
S49	Metamodels	Graph	Structure	Fine-grained	Matching	Proposal of Solution	Automatic	Search-based	Journal	✓	✓	✓	✓	✓	✗	✓	✓	✓	8
S50	Generic Diagram	Graph	Multi-criteria	Coarse-grained	Matching	Proposal of Solution	Semi-automatic	Heuristic	Conference	✓	✓	✓	✓	✓	✗	✓	✓	✓	8
S51	Generic Diagram	Other	Structure	Fine-grained	Rule-based	Proposal of Solution	Semi-automatic	Rule-based	Conference	✓	✓	✓	✓	✗	✗	✓	✓	✓	7

(Continued)

Table B.1. Continued

	General classification									Quality Assessment									
ID	**RQ1**	**RQ2**	**RQ3**	**RQ4**	**RQ5**	**RQ6**	**RQ7**	**RQ8**	**RQ9**	**Q1**	**Q2**	**Q3**	**Q4**	**Q5**	**Q6**	**Q7**	**Q8**	**Q9**	**Points**
S52	Class Diagram	Other	Semantic	Coarse-grained	Similarity	Proposal of Solution	Automatic	Heuristic	Conference	✓	✓	✓	✓	✓	✗	✓	✓	✗	7
S53	Class Diagram	Graph	Multi-criteria	Coarse-grained	Similarity	Proposal of Solution	Automatic	Search-based	Conference	✓	✓	✓	✓	✓	✗	✓	✓	✓	8
S54	All UML Diagram	Graph	Multi-criteria	Coarse-grained	Similarity	Proposal of Solution	Automatic	Heuristic	Journal	✓	✓	✓	✓	✓	✗	✓	✓	✓	8
S55	Sequence Diagram	Graph	Structure	Coarse-grained	Similarity	Proposal of Solution	Automatic	Heuristic	Conference	✓	✓	✓	✓	✓	✗	✓	✓	✓	8
S56	Activity, Class, and Feature Diagrams	Graph	Multi-criteria	Coarse-grained	Matching	Proposal of Solution	Automatic	Heuristic	Journal	✓	✓	✓	✓	✓	✗	✓	✓	✓	8
									Total	56	56	43	56	48	1	56	56	43	415

Legend:
ID: identifier of the primary study, RQ1: supported diagram, RQ2: data structures, RQ3: comparison aspects, RQ4: granularity.
RQ5: comparison types, RQ6: research methods, RQ7: automation degree, RQ8: comparison techniques, RQ9: publication type.

ACKNOWLEDGMENTS

We thank Dr. Jon Whittle for reviewing the manuscript and providing insightful recommendations.

REFERENCES

[1] Marwan Abi-Antoun, Jonathan Aldrich, Nagi Nahas, Bradley Schmerl, and David Garlan. 2008. Differencing and merging of architectural views. *Automated Software Engineering* 15, 1 (2008), 35–74.

[2] Mojeeb Al-Rhmam Al-Khiaty and Moataz Ahmed. 2016. UML class diagrams: Similarity aspects and matching. *Lecture Notes on Software Engineering* 4, 1 (2016), 41–47.

[3] Mojeeb Al-Rhman Al-Khiaty and Mariwan Ahmed. 2014. Similarity assessment of UML class diagrams using a greedy algorithm. In *International Computer Science and Engineering Conference*. IEEE, Thailand, 228–233.

[4] Kerstin Altmanninger, Martina Seidl, and Manuel Wimmer. 2009. A survey on model versioning approaches. *International Journal of Web Information Systems* 5, 3 (2009), 271–304.

[5] Cédric Brun and Alfonso Pierantonio. 2008. Model differences in the eclipse modeling framework. *The European Journal for the Informatics Professional* 9, 2 (2008), 29–34.

[6] David Budgen, Mark Turner, Pearl Brereton, and Barbara Kitchenham. 2008. Using mapping studies in software engineering. In *Proceedings of PPIG*, Vol. 8. 195–204.

[7] Michel Chaudron, Werner Heijstek, and Ariadi Nugroho. 2012. How effective is UML modeling? *Software & Systems Modeling* 11, 4 (2012), 571–580.

[8] Sudarshan S. Chawathe, Anand Rajaraman, Hector Garcia-Molina, and Jennifer Widom. 1996. Change detection in hierarchically structured information. In *ACM SIGMOD Record*, Vol. 25. ACM, 493–504.

[9] Jesus S. Cuadrado, Javier L. C. Izquierdo, and Jesus G. Molina. 2014. Applying model-driven engineering in small software enterprises. *Science of Computer Programming* 89, Part B (2014), 176–198.

[10] Brian Dobing and Jeffrey Parsons. 2006. How UML is used. *Communications of ACM* 49, 5 (May 2006), 109–113.

[11] Kleinner Farias. 2012. *Empirical Evaluation of Effort on Composing Design Models*. Ph.D. Dissertation. Department of Informatics, PUC-Rio. Rio de Janeiro, Brazil.

[12] Kleinner Farias, Alessandro Garcia, and Jon Whittle. 2010. Assessing the impact of aspects on model composition effort. In *Proceedings of the 9th International Conference on Aspect-Oriented Software Development*. 73–84.

[13] Kleinner Farias, Alessandro Garcia, Jon Whittle, Christina Flacha Garcia Chavez, and Carlos Lucena. 2014. Evaluating the effort of composing design models: A controlled experiment. *Software & Systems Modeling* 14, 4 (2014), 1349–1365.

[14] Ana M. Fernandez-Saez, Marcela Genero, and Michel R.V. Chaudron. 2013. Empirical studies concerning the maintenance of UML diagrams and their use in the maintenance of code: A systematic mapping study. *Information and Software Technology* 55, 7 (2013), 1119–1142.

[15] Matthias Galster, Danny Weyns, Dan Tofan, Bartosz Michalik, and Paris Avgeriou. 2014. Variability in software systems: A systematic literature review. *IEEE Transactions on Software Engineering* 40, 3 (March 2014), 282–306.

[16] Martin Girschick and T. Darmstadt. 2006. *Difference Detection and Visualization in UML Class Diagrams*. Technical University of Darmstadt Technical Report TUD-CS-2006-5 (2006), 1–15.

[17] Barney Glaser and Anselm Strauss. 1999. *The Discovery of Grounded Theory: Strategies for Qualitative Research*. Aldine Transaction.

[18] Lucian Gonçales, Kleinner Farias, Murilo Scholl, Maurício Veronez, and Toacy de Oliveira. 2015. Comparison of design models: A systematic mapping study. *International Journal of Software Engineering and Knowledge Engineering* 25 (2015), 1765–1769.

[19] John Hutchinson, Mark Rouncefield, and Jon Whittle. 2011. Model-driven engineering practices in industry. In *Proceedings of the 33rd International Conference on Software Engineering (ICSE'11)*. 633–642.

[20] IBM. 2017. Rational Software Architect Designer (RSAD). http://www-03.ibm.com/software/products/en/ratsadesigner

[21] Marouane Kessentini, Ali Ouni, Philip Langer, Manuel Wimmer, and Slim Bechikh. 2014. Search-based metamodel matching with structural and syntactic measures. *Journal of Systems and Software* 97, C (Oct 2014), 1–14.

[22] Samir Khuller and B. Raghavachari. 1996. Graph and network algorithms. *Comput. Surveys* 28, 1 (March 1996), 43–45.

[23] Barbara Kitchenham, Pearl Brereton, and David Budgen. 2010. The educational value of mapping studies of software engineering literature. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 1*. 589–598.

[24] Barbara Kitchenham and Stuart Charters. 2007. *Guidelines for Performing Systematic Literature Reviews in Software Engineering*. Technical Report. Keele University and University of Durham. EBSE-2007-01, Version 2.3.

[25] Barbara A. Kitchenham, David Budgen, and O. Pearl Brereton. 2010. The value of mapping studies - A participant-observer case study. In *Proceedings of the 14th International Conference on Evaluation and Assessment in Software Engineering*. 25–33.

[26] Barbara A. Kitchenham, David Budgen, and O. Pearl Brereton. 2011. Using mapping studies as the basis for further research - A participant-observer case study. *Information Software Technology* 53, 6 (June 2011), 638–651.
[27] Barbara A. Kitchenham, Emilia Mendes, and Guilherme H. Travassos. 2007. Cross versus within-company cost estimation studies: A systematic review. *IEEE Transactions on Software Engineering* 33, 5 (May 2007), 316–329.
[28] Dimitrios Kolovos. 2009. Establishing correspondences between models with the epsilon comparison language. In *5th European Conference on Model Driven Architecture - Foundations and Applications*. 146–157.
[29] Dimitris Kolovos, Louis Rose, Antonio Garcia-Dominguez, and Richard Paige. 2017. *The Epsilon Book*. Eclipse, http://www.eclipse.org/epsilon.
[30] Dimitrios S. Kolovos, Davide Di Ruscio, Alfonso Pierantonio, and Richard F. Paige. 2009. Different models for model matching: An analysis of approaches to support model differencing. In *Workshop on Comparison and Versioning of Software Models, ICSE*. 1–6.
[31] Dimitrios S. Kolovos, Richard F. Paige, and Fiona A. C. Polack. 2006. Model comparison: A foundation for model composition and model transformation testing. In *Internat. Workshop on Global Integrated Model Management*. 13–20.
[32] Segla Kpodjedo, Filippo Ricca, Philippe Galinier, Giuliano Antoniol, and Yann-Gael Gueheneuc. 2013. MADMatch: Many-to-many approximate diagram matching for design comparison. *IEEE Transactions on Software Engineering* 39, 8 (Aug 2013), 1090–1111.
[33] Marcello La Rosa, Marlon Dumas, Reina Uba, and Remco Dijkman. 2013. Business process model merging: An approach to business process consolidation. *ACM Transactions on Software Engineering and Methodology* 22, 2 (2013), 11.
[34] Yuehua Lin, Jeff Gray, and Frédéric Jouault. 2007. DSMDiff: A differentiation tool for domain-specific models. *European Journal of Information Systems* 16, 4 (2007), 349–361.
[35] Shahar Maoz, Jan Oliver Ringert, and Bernhard Rumpe. 2011. ADDiff: Semantic differencing for activity diagrams. In *Proceedings of the ESEC/FSE (ESEC/FSE'11)*. 179–189.
[36] Shahar Maoz, Jan Oliver Ringert, and Bernhard Rumpe. 2011. CDDiff: Semantic differencing for class diagrams. In *European Conference on Object-Oriented Programming*. Lecture Notes in Computer Science (LNCS), vol. 6813. Springer, Berlin, 230–254.
[37] Tom Mens. 2002. A state-of-the-art survey on software merging. *IEEE Transactions on Software Engineering* 28, 5 (May 2002), 449–462.
[38] Shiva Nejati, Mehrdad Sabetzadeh, Marsha Chechik, Steve Easterbrook, and Pamela Zave. 2012. Matching and merging of variant feature specifications. *IEEE Transactions on Software Engineering* 38, 6 (2012), 1355–1375.
[39] Dirk Ohst, Michael Welle, and Udo Kelter. 2003. Difference tools for analysis and design documents. In *Proceedings of the International Conference on Software Maintenance*. 13–22.
[40] OMG. 2014. *Business Process Model and Notation, Version 2.0.2*. Technical Report. http://www.omg.org/spec/BPMN.
[41] OMG. 2017. *Unified Modeling Language - Infrastructure, Version 2.5.1*. Technical Report. https://www.omg.org/spec/UML/2.5.1/.
[42] Wei-Jin Park and Doo-Hwan Bae. 2011. A two-stage framework for UML specification matching. *Information and Software Technology* 53, 3 (2011), 230–244.
[43] Kai Petersen, Sairam Vakkalanka, and Ludwik Kuzniarz. 2015. Guidelines for conducting systematic mapping studies in software engineering: An update. *Information and Software Technology* 64 (2015), 1–18.
[44] Marian Petre. 2014. "No shit" or "Oh, shit!": Responses to observations on the use of UML in professional practice. *Software & Systems Modeling* 13, 4 (2014), 1225–1235.
[45] Dong Qiu, Bixin Li, Shunhui Ji, and Hareton Leung. 2014. Regression testing of Web service: A systematic mapping study. *Computing Surveys* 47, 2, Article 21 (Aug. 2014), 46 pages.
[46] Marcello La Rosa, Marlon Dumas, Reina Uba, and Remco M. Dijkman. 2013. Business process model merging: An approach to business process consolidation. *ACM Transactions on Software Engineering Methodology* 22, 2 (2013), 11:1–11:42.
[47] Julia Rubin and Marsha Chechik. 2013. N-way model merging. In *Proceedings of the 9th Joint Meeting on Foundations of Software Engineering (ESEC/FSE 2013)*. 301–311.
[48] James Rumbaugh, Ivar Jacobson, and Grady Booch. 1999. *The Unified Modeling Language Reference Manual*. Addison-Wesley.
[49] Hamza Salami and Moataz Ahmed. 2014. UML artifacts reuse: State of the art. *arXiv:1402.0157* (2014).
[50] Iván Santiago, Alvaro Jiménez, Juan Manuel Vara, Valeria De Castro, VeróNica A Bollati, and Esperanza Marcos. 2012. Model-driven engineering as a new landscape for traceability management: A systematic literature review. *Information and Software Technology* 54, 12 (2012), 1340–1356.
[51] Alexandra Sbaraini, Stacy M. Carter, R. Wendell Evans, and Anthony Blinkhorn. 2011. How to do a grounded theory study: A worked example of a study of dental practices. *BMC Medical Research Methodology* 11, 1 (2011), 1–10.

[52] Maik Schmidt and Tilman Gloetzner. 2008. Constructing difference tools for models using the SiDiff framework. In *Companion of the 30th International Conference on Software Engineering*. ACM, New York, NY, USA, 947–948.
[53] Andreas Schoknecht, Tom Thaler, Peter Fettke, Andreas Oberweis, and Ralf Laue. 2017. Similarity of business process models — A state-of-the-art analysis. *ACM Computing Surveys (CSUR)* 50, 4 (2017), 52.
[54] Petri Selonen. 2007. A review of UML model comparison approaches. In *Nordic Workshop on MDE*. 37.
[55] William R. Shadish, Thomas D. Cook, and Donald Thomas Campbell. 2002. *Experimental and Quasi-experimental Designs for Generalized Causal Inference*. Wadsworth Cengage Learning.
[56] Darja Smite, Claes Wohlin, Tony Gorschek, and Robert Feldt. 2010. Empirical evidence in global software engineering: A systematic review. *Empirical Software Engineering* 15, 1 (2010), 91–118.
[57] Matthew Stephan and James R. Cordy. 2013. A survey of model comparison approaches and applications. In *Modelsward*. 265–277.
[58] Jie Su and Junpeng Bao. 2012. Measuring UML model similarity. In *Proceedings of the 7th International Conference on Software Paradigm Trends*. 319–323.
[59] Tom Thaler, Andreas Schoknecht, Peter Fettke, Andreas Oberweis, and Ralf Laue. 2016. A comparative analysis of business process model similarity measures. In *Proceedings of the International Conference on Business Process Management*. 310–322.
[60] Mark Van den Brand, Zvezdan Protić, and Tom Verhoeff. 2010. Fine-grained metamodel-assisted model comparison. In *Proceedings of the 1st International Workshop on Model Comparison in Practice*. 11–20.
[61] Konrad Voigt. 2011. *Structural Graph-B ased Metamodel Matching*. Ph.D. Dissertation. University of Desden.
[62] Roel Wieringa, Neil Maiden, Nancy Mead, and Colette Rolland. 2005. Requirements engineering paper classification and evaluation criteria: A proposal and a discussion. *Requirements Engineering* 11, 1 (December 2005), 102–107.
[63] Claes Wohlin, Per Runeson, Martin Höst, Magnus C. Ohlsson, Björn Regnell, and Anders Wesslén. 2012. *Experimentation in Software Engineering*. Springer Science & Business Media.
[64] Zhenchang Xing and Eleni Stroulia. 2005. UMLDiff: An algorithm for object-oriented design differencing. In *Proceedings of the 20th IEEE/ACM International Conference on Automated Software Engineering*. 54–65.
[65] Chen Yang, Peng Liang, and Paris Avgeriou. 2016. A systematic mapping study on the combination of software architecture and agile development. *Journal on Systems and Software* 111, C (January 2016), 157–184.
[66] Jae young Bang, Yuriy Brun, and Nenad Medvidovic. 2018. Collaborative design conflicts: Costs and solutions. *IEEE Software* (2018).
[67] Kaizhong Zhang, Rick Statman, and Dennis Shasha. 1992. On the editing distance between unordered labeled trees. *Information Processing Letters* 42, 3 (1992), 133–139.

Received May 2018; revised December 2018; accepted February 2019