ORIGINAL ARTICLE (PRE-PRINT)

Journal of Software: Evolution and Process

A systematic mapping of test case generation techniques using UML interaction diagrams

Nasir Mehmood Minhas[1,2] | Sohaib Masood[2] | Kai Petersen[1,4] | Aamer Nadeem[3]

[1]Department of Software Engineering, Blekinge Institute of Technology, Karlskrona, Blekinge, 37179, Sweden

[2]University Institute of Information Technology, PMAS - Arid Agriculture University, Rawalpindi, Punjab, 46300, Pakistan

[3]Faculty of Computing, Capital University of Science and Technology, Islamabad, 45750, Pakistan

[4]University of Applied Sciences Flensburg, Flensburg, 24943, Germany

Correspondence
Nasir Mehmood Minhas, Department of Software Engineering, Blekinge Institute of Technology, Karlskrona, Blekinge, 37179, Sweden
Email: nasir.mehmood.minhas@bth.se

Funding information
The funding agency Vinnova, as well as Sony Mobile Communications, Axis and Softhouse support the work through the EASE Industrial Excellence Center (reference number 2015-03235).

Model-based test case generation techniques provide a mechanism to derive tests systematically. This study provides a systematic mapping of test case generation techniques based on UML interaction diagrams. The study compares the test case generation techniques regarding their capabilities and limitations, and it also assesses the reporting quality of the primary studies. We can conclude that the studies presenting test case generation techniques using UML interaction diagrams were not following guidelines for research methods (e.g., case studies or experiments). Solutions were not empirically evaluated in industrial contexts. Our study revealed that better tool support is needed to introduce the UML interaction diagram based test case generation techniques in the industry.

KEYWORDS
Software testing, Test case generation, Interaction diagrams, Model based testing, Systematic mapping

1 | INTRODUCTION

Testing is an integral part of the software development life cycle to improve the quality and reliability of software [1]. Testing is one of the most expensive and labor intensive tasks. Organizations spend more than 50% of their total budget

Abbreviations: UML, Unified Modeling Language; MBT, Model based testing; SD, Sequence diagram; CD, Collaboration diagram; H, Hybrid.

on software testing [2, 3]. Testing can be automated to various degrees, such as completely manual, semi-automated, or fully automated. Automated testing has benefits in terms of cost reduction, ease of use and better coverage [2, 4].

The main objective of testing is to verify that the software meets the acceptance criteria as documented in requirements specifications [5]. There are two main aspects of this objective: (1) to demonstrate that the requirements specifications are correct, and (2) to demonstrate that the design and code of the software fulfill the requirements [5].

The creation of test cases is an intellectually challenging task [6, 7, 1, 4]. Since the complexity of software is increasing rapidly, effective testing mechanisms are required to identify and eliminate defects. Researchers and practitioners proposed various automated test case generation techniques during the last decade, among others, Model-Based Testing (MBT). In MBT, Unified Modeling Language (UML) models are commonly used as a means to generate test cases. UML models can be classified as static or dynamic(c.f. [8]). Structural models represent static aspects of a system, whereas the behavioral models represent dynamic aspects. Structural models are represented by *class diagrams, component diagrams, deployment diagrams, object diagrams, package diagrams, profile diagrams, and composite diagrams*. Interaction diagrams fall under the category of behavioral models. Other diagrams in this category are *use case diagrams, statechart diagrams, and activity diagrams* [8].

In this study, we focus on dynamic models as testing focuses on the dynamic aspects of a system during its execution. UML interaction diagrams are of high relevance in later testing activities (such as integration testing) as the interaction between different components has to be considered. The first step to test software automatically is to describe its behavior or structure either statically or dynamically by using UML models [9, 4]. By means of UML models, practitioners may perform unit [10, 3], integration [11, 12, 13, 14], system [15, 16, 5] and conformance testing [17].

Existing literature reviews did not put their primary focus on UML interaction diagrams, and did not follow a documented systematic procedure [18, 19, 20]. Practitioners and researchers have proposed various techniques to generate test cases from UML interaction diagrams during the past few years. Classification of existing MBT techniques based on UML interaction diagrams is therefore essential for researchers before proceeding toward proposing new approaches using interaction diagrams. Furthermore, practitioners and researchers currently utilizing UML interaction diagrams are provided with an inventory of results, as well as a documentation of the capabilities and limitations of these approaches.

To complement the existing literature reviews, we investigate the use of UML interaction diagrams making the following contributions:

- C1: Identify the approaches for MBT test case generation based on UML interaction diagrams proposed in the literature.
- C2: Identify the capabilities and limitations of these approaches.
- C3: Characterize the quality of the studies using the rigor and relevance scoring framework by Ivarsson and Gorschek [21].

In this study, we focus on test case generation techniques based on UML interaction diagrams, and not model-based testing in general. As a research method, we utilized the guidelines for systematic mappings by Petersen et al. [22, 23]. Furthermore, for quality assessment, the rigor and relevance scoring framework by Ivarsson and Gorschek [21] was used. Rigor focuses on scientific quality concerning the research process and method used. Relevance is focused on whether evaluations have been conducted in a realistic environment concerning scale, context, and subjects.

The remainder of this paper is structured as follows: Section 2 discusses the related work regarding existing surveys/ reviews in the area of MBT. Section 3 describes the research methodology, and Section 4 presents the findings of this mapping study. The results are summarized and discussed in Section 5, and Section 6 concludes the paper.

2 | RELATED WORK

Several secondary studies related to MBT have been published, see Table 1 for an overview. The table summarizes the aims, research methods, and the number of primary studies included. Several studies were generic without a particular focus on UML interaction diagrams, such as [18, 24, 25, 26]. We identified only a small set of studies that focused on interaction diagrams. For example, the literature review by Shirole and Kumar [20] focused on behavioral models. Dias Neto [24] and Shirole and Kumar [20] identified 21 and 29 studies based on interaction diagrams, respectively.

Concerning the methodologies applied in the existing secondary studies, four conducted a traditional literature review [18, 19, 20, 27]. Three [24, 25, 28] referred to the guidelines by Kitchenham and Charters [29] as their guide for conducting their systematic literature reviews. One mapping study [26] was reported. From a methodological perspective, it is noteworthy that no quality assessments of the primary studies were conducted.

The previous studies did not investigate test case generation techniques based on UML interaction diagrams using a systematic approach for studying the literature (such as systematic mapping or review studies). The topic itself is of interest as the potential of models (e.g., automating the derivation of test cases) is high. However, the studies showed that models are not generally adopted in the industry (see, e.g., Gorschek et al. [30]). Hence, it is interesting to review the literature to communicate and summarize the approaches. To evaluate the readiness for industry adoption, we assess the quality of the studies.

We complement the existing secondary studies with an explicit focus on test case generation techniques based on UML interaction diagrams (finding a total of 54 articles) and hence expanding the findings by Shirole and Kumar [20]. Furthermore, we complement the existing works by analyzing the strength of evidence provided through the means of the quality assessment rubrics proposed by Ivarsson and Gorschek [21]. The rubrics describe rules by which reported studies could be assessed for their scientific rigor and practical relevance. The details of the rubrics are described in the following section.

TABLE 1 Existing secondary studies on MBT

Authors/Ref.	Aim	Method	#Studies
Sing [18]	Present test generation techniques for software testing of object oriented systems	LR	55
Dias Neto [24]	Characteristics, application, and limitations of MBT approaches	SLR	78
Häser et al. [25]	Classification of model-based testing approaches; characterize the type of assessment done to guide the testing process; determine the coverage of non-functional requirements by MBT approaches	SLR	83
Shafique and Labiche [28]	Identification of MBT testing tools; extraction of the capabilities of tools	SLR	12
Utting et al. [19]	Proposal of a taxonomy to classify MBT approaches; classification of four tools to demonstrate the use of the taxonomy	LR	55
Shirole and Kumar [20]	Provide an overview of solutions to generate tests from behavioral models (sequence diagrams, state machines, activity diarams)	LR	29
Mohi-Aldeen et al. [26]	Coverage criteria used to drive automated testing; methods used for test generation; benefits of approaches	SM	85
Khandai et al. [27]	Overview of UML model based test case generation techniques proposed for concurrent and non-concurrent systems	LR	15

3 | RESEARCH METHOD

In this research, we conducted a systematic mapping study (SMS). Systematic mapping studies provide an overview of a research area and classify the research contributions. Like systematic literature reviews, they provide a systematic way to search the literature, to identify the topics that have been reported in the literature and to identify the publication venues for the area of research [23, 22]. We followed the guidelines provided in [23]. In these guidelines, the authors included some essential aspects of the SLR guidelines proposed by Kitchenham and Charters [29].

3.1 | Planning and conducting the mapping

| Research questions

The objective of this paper is to analyze the research on UML interaction diagram based test case generation. Research questions and objectives are highlighted in Table 2. The research questions map directly to the objectives stated in the introduction, namely, to identify approaches for MBT test case generation (RQ1), to extract their capabilities and limitations (RQ2) and to assess the quality of the studies concerning rigor and relevance (RQ3).

TABLE 2 Research questions

ID		**Research question**
RQ1		Which MBT test case generation approaches based on UML interaction diagrams have been proposed in the literature?
	RQ1.1	What are the publication trends of test case generation approaches based on UML interaction diagrams?
	RQ1.2	How can we classify the test case generation techniques based on UML interaction diagrams?
RQ2		What capabilities and limitations are observed in relation to the approaches?
RQ3		What is the rigor and relevance of the included primary studies according to the rigor and relevance scoring rubric proposed by Ivarsson and Gorschek?
	RQ3.1	What is the extent of relatedness among the included primary studies?

| Developing and validating the study protocol

The development and validation of the study protocol are critical. The first and second authors developed the protocol while the fourth author critically reviewed the developed protocol.

Modifications were suggested in research questions, publication selection, inclusion/exclusion criteria, and data extraction. After modifying the protocol, according to these suggestions, the same expert (author four) again critically reviewed the developed protocol. Thus, this was a means of using observer triangulation to increase the quality of the research.

| Identifying the related literature

To identify the relevant literature, the main and alternative search terms were identified (see Table 3) by specifying the population and intervention. Even though comparison and outcome were proposed to consider during the search, additional search terms are likely to reduce the set of identified studies, as the comparison and outcome variables may not be mentioned in the abstract and title [31, 32]. In our case the population is *"UML diagrams"* and the intervention is *"test case generation"*. Major terms used to search the literature include *"test case generation"*, *"sequence diagram"*, *"collaboration diagram"*, *"model based testing"* and *"UML diagrams"*.

TABLE 3 Search terms

Main search terms	Alternative search terms
Test case generation	Automated test case generation, Test path generation
Sequence Diagram	UML sequence diagram, Interaction diagrams, UML interaction diagrams
Collaboration diagram	UML collaboration diagram, Communication diagram, UML interaction diagrams
Model based testing	Testing using UML models
UML diagrams	UML models, UML behavioural diagram

| Search string construction

Three different search strings have been constructed. One search string focuses on sequence diagrams (Category *"SD"*), one on collaboration diagrams ("CD"), and one on identifying testing for UML diagrams in general, combining sequence and collaboration diagrams and other UML models ("Combination/ Hybrid"). The search strings were as follows:

- *Sequence diagrams (SD):* ("Test case generation" OR "Automated test generation" OR "Test path generation") AND "Model based testing" AND ("Sequence diagram" OR "UML sequence diagram" OR "UML interaction diagrams")
- *Collaboration diagrams (CD):* ("Test case generation" OR "Automated test generation" OR "Test path generation") AND "Model based testing" AND ("Collaboration diagram" OR "UML collaboration diagram" OR "Communication diagram" OR "UML Communication diagram" OR "UML interaction diagrams")
- *Hybrids (H):* ("Test case generation" OR "Automated test generation" OR "Test path generation") AND "Model based testing" AND ("UML diagrams" OR "UML models" OR "UML behavioural models")

How the individual databases have been used for the search is specified in Appendix A. The search strings were executed on various online resources. Table 4 summarizes the selection procedure adopted as well as the number of papers identified and selected. Title and abstract screening were performed. If a publication's title was related to testing (test case generation) and UML models, then this publication was selected for screening the abstract. During the abstract screening, we determined whether the publication was relevant to model-based test case generation and whether it used UML interaction diagrams. A total of 54 primary studies were selected. Among these 54 studies, 24 studies were found directly in the publisher databases (i.e., 17 from IEEE, 3 from Elsevier, & 4 from ACM). Thirty primary studies were found through Google Scholar. Out of these 30 studies, 10 were from Springer, 8 from other journals indexed by Scopus, and 12 studies were selected from other sources.

The following inclusion criteria were applied:

- IC1: Studies related to MBT.
- IC2: Studies that propose model-based test case generation techniques by means of interaction diagrams.

- IC3: Studies that describe the capabilities of model-based test case generation techniques with respect to UML interaction diagrams.
- IC4: Surveys (e.g., questionnaires or interviews) performed on model-based test case generation techniques.
- IC5: Studies written in the English language.

The studies were excluded on the basis of the following criteria:

- EC1: Studies which do not use UML interaction diagram although they belong to the MBT technique.
- EC2: Duplicates (i.e., if the same study found from more than one search engine).
- EC3: Studies with missing full-text.
- EC4: Secondary studies (e.g., systematic reviews or mapping studies).

TABLE 4 Database results

Database	Search results	Selected for title screening	Selected for abstract screening
IEEE	250	65	17
Science Direct (Elsevier)	110	24	3
ACM	347	54	4
Google Scholar	656	107	30
Total	1363	240	54

The study selection was carried out by the first and second authors jointly to reduce bias and potential mistakes. Studies included or excluded by the first author were crosschecked by the second author, while the first author cross-checked studies included or excluded by the second author. Duplicate studies were eliminated during the crosscheck. Finally, a random crosscheck was applied by the fourth author.

| Quality assessment

To assess the quality of the included primary studies, we utilized the scoring rubrics proposed by Ivarsson and Gorschek [21]. Rigor was evaluated based on three criteria: the description of context, study design, and validity. Relevance was evaluated based on the criteria subject, context, and scale. The criteria for rigor were scored on an ordinal scale, as suggested by [21]: Strong, medium, and weak description. The first author scored the studies, which were all reviewed by the third author. In case of issues in the scoring, these were discussed and resolved with the first author. In the following, the quality evaluation criteria are explained.

Rigor: The research rigor criteria check whether the methodology is described in a way so that the rigor of the research may be judged [21].

Context (C):

- *Strong description:* Context is strongly described and comparable with other contexts. We emphasize the subjects' type (academic, industrial), the development methodology and the experience of the subjects. If these factors are presented, then C evaluates to 1.
- *Medium description:* In absence of any factor (see strong description), C evaluates to 0.5.

- *Weak description:* If the factors are not present, then C evaluates to 0.

Study design (D):

- *Strong description:* If the research design is clear to readers by documenting expected outcomes, measurement criteria, data selection and collection, etc., then D evaluates to 1.
- *Medium description:* In absence of an essential factor in the research design, D evaluates to 0.5.
- *Weak description:* If no description of the research design is provided, then D evaluates to 0.

Validity threats (V):

- *Strong description:* If internal, external, construct, and conclusion validity threats are highlighted, then V evaluates to 1.
- *Medium description:* If a subset of these threats are provided, then V evaluates to 0.5.
- *Weak description:* If validity threats are not discussed, then V evaluates to 0.

Relevance: Several measures are used to evaluate academic relevance in research. The impact of industrial relevance is hard to measure. The assessment criteria proposed by Ivarsson and Gorschek [21] assume that the relevance is higher if the study is conducted with realistic subjects, realistic scale, and using research methods that facilitate investigations in realistic environments.

Subjects (U):

- Contribution to relevance: If subjects or users are representative of a real user, such as for example industry professionals, then U evaluates to 1.
- Partially contribute to relevance: If subjects are researchers or students, then U evaluates to 0.5.
- No contribution: If subjects are not mentioned, then U evaluates to 0.

Context (C):

- Contribution to relevance: If the study is conducted in a real industrial setting, then C evaluates to 1.
- No contribution: If the study is not conducted in a real industrial setting, then C evaluates to 0.

Scale (S):

- Contribution to relevance: If the used application is representative of real (industry) systems in terms of scale, then S evaluates to 1.
- No contribution: If the used application is a toy example or down-scaled, then S evaluates to 0.

| Data extraction and analysis

The selected studies were stored for data extraction. Data extraction was performed jointly by the first and second authors. The third and fourth authors randomly selected included primary studies and reviewed the results produced by the first and second authors. Issues were discussed and resolved jointly. Data extraction comprised of the following elements:

- *Publication data:* Publication ID, publication title, year of publication, type of publication, source of publication, author names, keywords.
- *Test scope:* Type of testing, test objective.
- *Input to tests:* UML model used, intermediate forms constructed for test data generation.
- *Data generation approach:* Description of test data generation process, algorithms used, tool support.
- *Assessment:* Coverage criteria, research method, rigor, and relevance scores.
- *Outcome:* Capabilities and limitations, proposed future works, and identified gaps.

We utilized content analysis to group and assign studies to categories (e.g., categorizing the studies with regard to testing type and type of intermediate forms for test data generation). This shows which areas are covered by the literature and is consistent with the intent of a mapping study where an overview of an area should be provided. Furthermore, narrative summaries for the different types of test generation approaches are provided.

3.2 | Validity threats

The limitations of the study are discussed with regard to different types of validity threats, namely, theoretical (internal) validity, generalizability, objectivity, and interpretive validity.

Theoretical validity: The scoring rubrics used in this study assess the reporting of studies, while the actual study design may not be described completely. Thus, the actual rigor and relevance may not be determined, but rather what is possible to judge from the presentation in the paper. The rubrics favor results reported from practical applications. This is not to say that the results are not of value for practice, though given the current presentation, evidence of a wide-spread technology transfer to the industry was not found.

Generalizability: Generalizability is concerned with the ability to generalize the findings from this study to the population of studies. Given that searches are limited and may not always find all relevant studies, the threat is relevant to this study. However, the results obtained from the papers are consistent and reveal a pattern of how test case generation approaches based on UML interaction diagrams are designed, consisting of the main steps of creating an intermediate form and applying an algorithm to generate test paths, and in some cases test cases. We also did not limit our search with regard to the evaluation done (e.g., only including case studies).

Objectivity: Objectivity determines the degree to which we can report our observations accurately reflecting the reality of the field. Multiple threats are of relevance here, which are shortly discussed below.

- Missing relevant studies: Our search focused on UML interaction diagrams and not model-based testing in general. Thus, some studies from the population might have been missed. However, the search returned a high number of relevant studies with little noise. We also included a larger set of studies on the topic when compared to previous literature reviews (see Section 2). This indicates that the systematic search helped to cover a larger portion of the population. However, given that the search terms were precise, there is a risk of missing studies from the population.
- Biases in study selection, data extraction, and analysis: The selection, data extraction, and analysis activities are prone to bias and mistakes. Thus, multiple researchers have been involved, and cross-checking of the studies has been done among the authors of the paper. Two of the authors were involved in the selection of primary studies,

and the other two were responsible for cross-checking of the selected studies. The use of a research protocol is also a means of reducing bias. Though, the research protocol itself was only evaluated by one researcher, hence posing a threat to objectivity.

Interpretative validity: From the findings of the study, conclusions are drawn. We utilized observer triangulation and discussed the implications and key findings among the authors. This reduces the threat of drawing the wrong conclusions from the data.

4 | RESULTS

This section presents the findings of our study. We have organized the presentation of the results according to the research questions listed in Table 2.

4.1 | RQ1: Which MBT test case generation approaches based on UML interaction diagrams have been proposed in the literature?

RQ1 was divided into two sub-questions (RQ1.1 & RQ1.2). The purpose of the first sub-question was to identify the trends of test case generation techniques using UML interaction diagrams as input. The second sub-question focuses on classification, for example, by the input model, intermediate form, and the outcome of the techniques.

| RQ1.1: What are the publication trends of test case generation approaches based on UML interaction diagrams?

In response to this research question, we present the demographics of the studies, including the number of publications over time, and the publication forums targeted.

A majority of the primary studies (63%) included in this study are from the four publishing venues IEEE, Springer, ACM, and Elsevier, see Figure 1. An increasing trend in the number of publications is seen with peaks in 2007, 2013, and 2014 (Figure 2). The publication trend shows that the research area is growing gradually. The complete list of primary studies with corresponding publication venues is presented in Table 15 in Appendix B. Only a few well-known journals in the field of software engineering have published research in the area. Also, many different publication forums have been targeted. Hence, no clear recommendation can be given where to start looking for papers on the topic.

| RQ1.2: How can we classify the test case generation techniques based on UML interaction diagrams?

The selected testing techniques based on UML interaction diagrams are reviewed regarding their similarities and differences. The majority of the techniques reported in the literature use UML sequence diagrams as input for test case generation and are marked as category *"SD"*, in the selected primary studies 48% (26 of 54) the techniques are based on sequence diagrams. Approaches based on UML collaboration diagrams are marked as category *"CD"*; only 7% (4 of 54) of the selected primary studies belong to this category. 45% (24 of 54) of the selected studies are using interaction diagrams along with some other UML model (including use case, statechart, interaction overview, activity, or class diagrams) as input for test case generation, we refer these techniques as hybrid solutions (category *"H"*). Figure 3

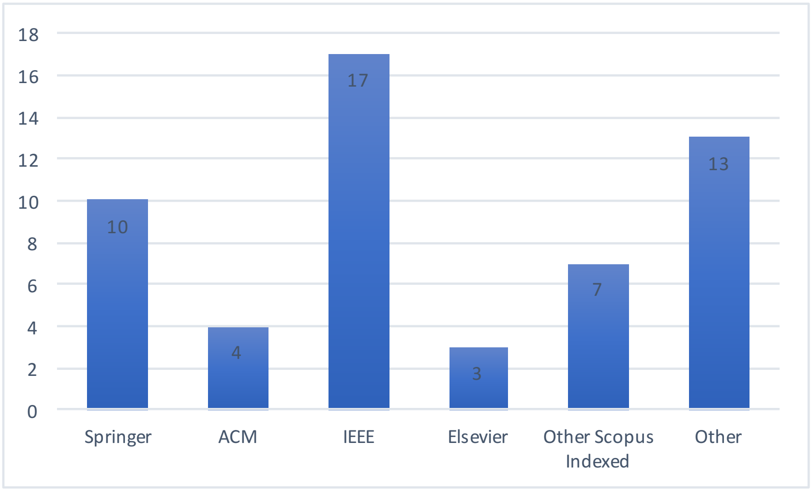
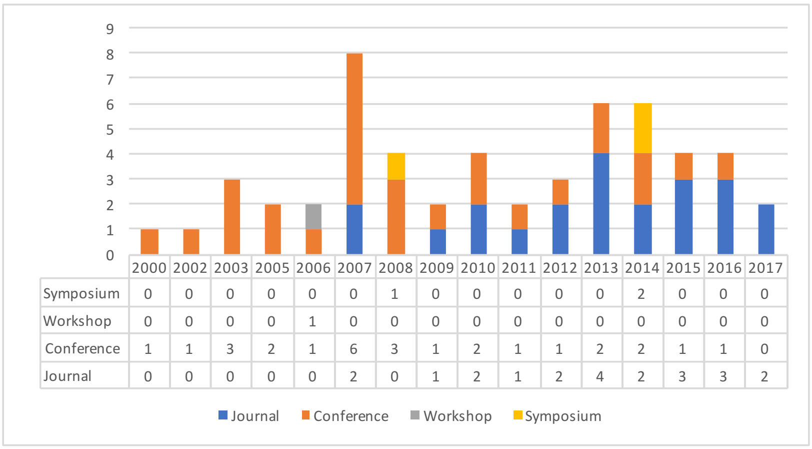


FIGURE 1 Publishing venues of primary studies



	2000	2002	2003	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017
Symposium	0	0	0	0	0	0	1	0	0	0	0	0	2	0	0	0
Workshop	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
Conference	1	1	3	2	1	6	3	1	2	1	1	2	2	1	1	0
Journal	0	0	0	0	0	2	0	1	2	1	2	4	2	3	3	2

FIGURE 2 Year and venue wise distribution of primary studies

shows the distribution of the primary studies according to the type of test case generation techniques.

Regarding the testing types, 81% of the SD-based studies are proposed for integration testing, while 8% for functional testing, and 11% for system testing. All of the CD-based studies are proposed for integration testing. Finally, 62.5% of the hybrid solutions are proposed for integration testing, and 37.5% for system testing. Figure 4 presents the classification of proposed techniques according to the testing types covered. Table 5 presents a complete overview of the categories of techniques based on target testing types (system, integration, or unit) and intermediate forms used to generate the test paths. The models form the basis for generating the test paths. Some techniques only determine the test paths that should be covered. However, to execute the paths, the test data still needs to be generated. For example, in the case of a decision node, the two input values that evaluate to true or false, respectively, need to be determined. The input values are also referred to as test data, i.e., concrete inputs to be given to the system under test.

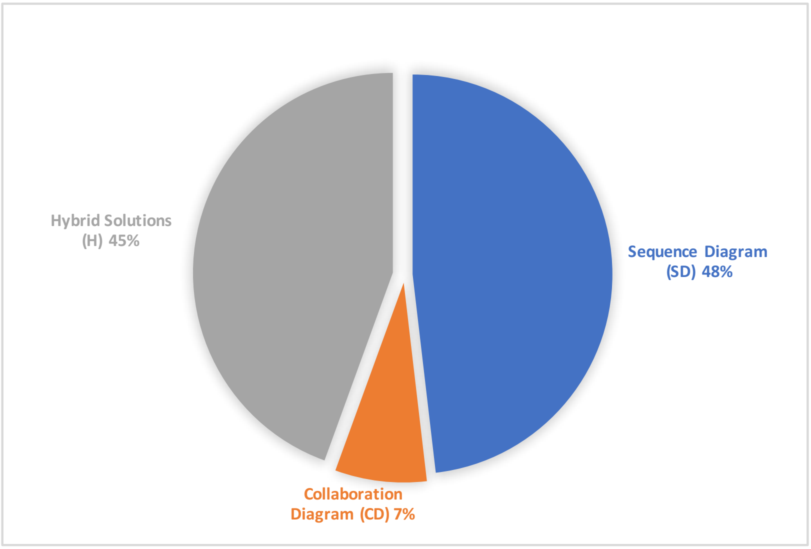


FIGURE 3 Distribution of studies according to input model used

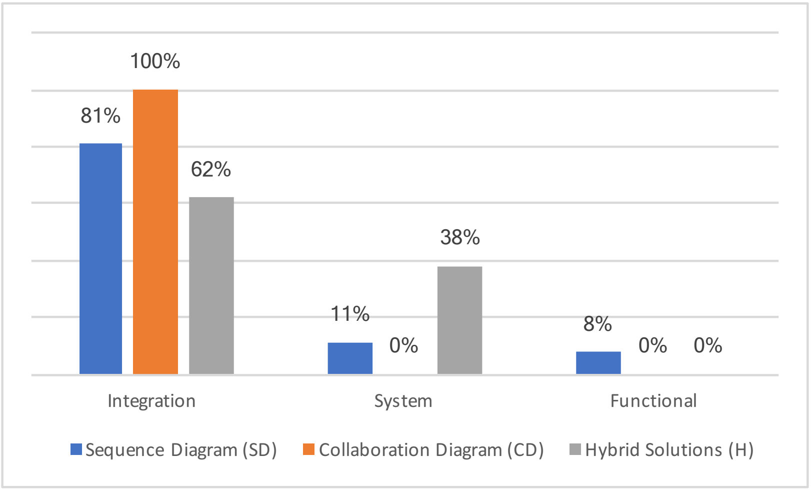


FIGURE 4 Distribution of studies according to testing types covered

In the following sub-sections, we provide a synthesis of how the approaches proposed in the literature (SD, CD, and H) are structured.

| Solutions using SD

We found a total of 26 studies that use sequence diagrams as an input model. The activity diagram in Figure 5 gives an overview of the approaches described in the literature. The references in the diagram show how the techniques

TABLE 5 Overview of solutions

Category	Testing type	Intermediate form	References
SD	Integration	Graph	[7, 12, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43]
		XMI	[44]
		Non-deterministic	[11, 13, 17]
		Dependency table	[45]
		Tree	[2]
		SLT	[46]
		EDFA	[47]
	System	Graph	[48]
		Activity diagram	[14, 49]
	Functional	Graph	[50]
		LTS	[51]
CD	Integration	Graph	[52, 53, 54]
		Tree	[4]
H	Integration	Graph	[1, 6, 55, 56, 57]
		XMI	[58, 59, 60]
		Tree	[61],
		Petal files	[62]
		SCOTEM	[63]
		Non-deterministic	[3, 64, 65, 10]
	System	Graph	[5, 15, 16, 66, 67, 68, 69]
		Activity diagram	[70, 71]

proposed in the papers are structured. The first step towards test case generation is that sequence diagrams are transformed into intermediate forms, which are then used as a basis for test case generation.

A majority of the proposed techniques [7, 12, 33, 34, 35, 36, 38, 39, 40, 41, 42, 43, 37, 48, 50] use different forms of graphs as an intermediate form. Other intermediate forms used in the proposed techniques are labelled transition systems (LTS) [51], XML metadata interchange format (XMI) [44], sequence dependency tables [45], trees [2], activity diagrams [14, 49], stimulus link tables (SLT)[46], and event deterministic finite automata (EDFA) [47].

The techniques presented in [11, 13, 17] do not use an intermediate form and have been listed as non-deterministic in Table 5 and Figure 5. The proposed techniques use a variety of algorithms for the traversal of graphs, generation of tests, and/or minimization of tests, see Table 6 for a summary.

TABLE 6 Algorithms used in SD based solutions

Studies	Algorithm
[12, 13, 39]	Genetic algorithm (GA)
[33, 38, 40, 45, 51]	Depth first search (DFS)
[7]	DFS and GA
[35, 37]	Message sequence path algorithm based on DFS and BFS
[14]	Concurrent queue search algorithm (CQS)
[36]	Edge marking algorithm
[34]	Tarjan's cycle detection algorithm
[2, 17, 41, 43, 44, 46, 47, 48, 49, 50]	Other traversing algorithms
[11, 42]	No algorithm mentioned

SD based solutions using a graph and its variants as intermediate form: In their test case generation technique [2],

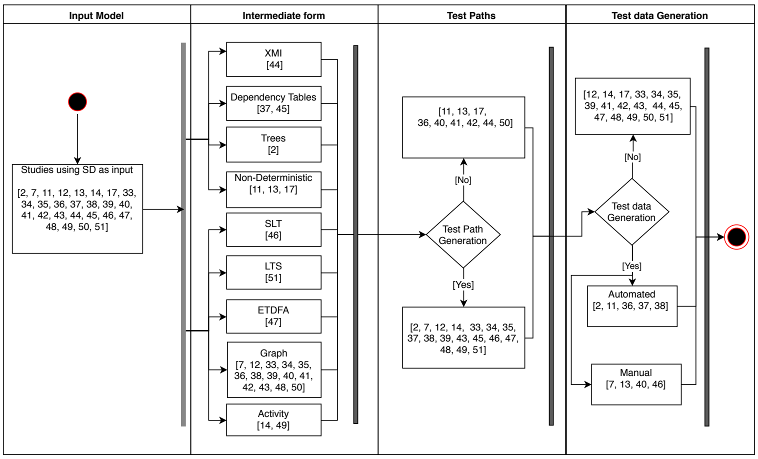


FIGURE 5 Solutions using sequence diagram (SD) as input

Bao-Lin et al. convert the sequence diagram into a tree representation, which is then traversed to select the conditional predicates. The authors use the object constraint language (OCL) for the description of pre- and post-conditions, and for the generation of test cases, they use a function minimization technique.

A wait-for graph was used by Patnaik et al. [41] to identify deadlock points. The primary objective of their technique is to identify concurrency issues. Hoseini and Jalili [39] generated a control flow graph and thereafter used a prime path extraction algorithm to generate the test paths and to minimize the tests, the authors used a genetic algorithm. A message control flow graph is used as an intermediate form by Jena et al. as well as Lei and Lin [7, 40]. After that the graph is traversed using the depth-first search algorithm. Lei and Lin [40] also provide an automated tool that can transform sequence diagrams into control flow graphs and then generate the test paths automatically. Their technique also works for test case generation. The techniques presented in [39, 7] use a genetic algorithm to calculate the fitness value for generated test paths to optimize them. Sumalatha et al. [12] used a sequence graph as an intermediate form. The sequence graph was traversed using a genetic algorithm to generate the test paths. A fitness function was defined by calculating the fitness value of the generated test paths. Chandnani et al. [33] presented a model-based test case generation technique to test the interaction of objects. A sequence dependency table from the sequence diagrams is created, and then a control flow graph is generated. The intermediate graph is traversed into a depth-first manner to obtain test paths. Finally, test cases are generated using a particle swarm optimization algorithm (PSO). Rhmann and Saxena [43] converted sequence diagram into message flow graphs. They used a depth-first search algorithm to traverse the message flow graph and generated the test paths.

The technique presented by Shirole and Kumar [14] uses activity diagrams as an intermediate form. Test scenarios are generated by using a concurrent queue search (CQS). Test scenarios generated here are useful for detecting data safety errors. Seo et al. [49] presented a test case generation technique based on sequence diagrams. The sequence diagram is transformed into an activity diagram. In the next step, the activity diagram was fragmented into sub-activities. UML testing-profile based test cases were designed, based on the principle that each activity represents a test scenario. A concurrent composite graph was used by Khandai et al. [35] who generated test paths by traversing the graph with the help of a message sequence graph algorithm. Khandai et al. also make use of OCL. Both techniques [35, 14] are meant to detect concurrency errors.

A sequence dependency graph was used as an intermediate form in multiple studies [50, 38, 45, 48, 36] . Dhineshkumar et al. [50] define pre-and-post conditions using OCL. Test paths are generated by traversing the sequence dependency graph with the help of an iterative depth-first search algorithm. Panthi and Mohapatra [38] also use the depth-first search algorithm to traverse the message sequence graph, and predicates are selected. By using these predicates, the source code is produced, and then test cases are generated. The algorithm traverses the sequence dependency graph and test sequences are generated in [48]. Sequence dependency graph was also used as an intermediate form by Samuel et al. [36]. They use the concept of dynamic slicing to generate test cases automatically. Priya and Malarchelvi [45] generate a sequence dependency table from the sequence diagram, which is further transformed into a sequence dependency graph. A depth-first search algorithm is used to generate test paths from the sequence dependency graph. Shanti and Khumar [37] also used a sequence dependency table to generate the test paths from the dependency table, using a genetic algorithm.

SD based solutions using other intermediate forms: XMI (even though not a model, it was used as an intermediate form and is hence reported here) is used as intermediate form by Beyer et al. [44]. The authors proposed a test case generation tool called “MaTeLo”, the tool makes use of Markov chains as an intermediate form. It converts the sequence diagram into XMI format, thereafter using the Markov chain markup language (MCML) to generate a Markov chain usage model (MCUM). The main objective is to derive test cases compatible with the testing and test control notation version 3 (TTCN-3). This approach mainly focuses on statistical usage testing. Cartaxo et al. [51] introduced the functional (feature) test case generation method. In this method, the authors converted sequence diagram into a labeled transition system (LTS). In LTS transitions are represented as states, it also supports the modeling of loops and alternative flows. Cartaxo et al. used a depth-first search (DFS) to generate test paths from the LTS.

SD based solutions without any intermediate forms: Furthermore, solutions were available that did not use any intermediate form. Frain and Leonhardt [11] have presented a test tool for object-oriented applications. Their “SeDiteC” tool extracts the necessary information from SDs by using XML. The tool enables early testing and provides facilities such as automatic test stub generation and initialization of objects. The technique presented in [13] does not use any intermediate form, it makes use of a genetic algorithm to generate valid and invalid test paths and related test cases. Cho et al. [42] suggested a method to test web applications. The proposed method generates test cases for single, mutual, and integrated web pages. For single web pages, test cases are generated from the self-call message in the sequence model. For multiple web pages, test cases are extracted from messages passed between web pages. Integrated test cases are designed from messages, which are passed to the system. Cho et al. also proposed a tool, though it is not available online. Lund and Stolen [17] proposed an algorithm to derive test cases from UML sequence diagrams. The sequence diagram is specified using neg (unary operator to specify negative behavior) and assert (operator used to specify universal behavioral) operators. Zhang et al. [47] presented a test case generation approach based on sequence diagrams and event deterministic finite automata (EDFA). The objects participating in the sequence diagram are transformed into EDFA. Propositional projection temporal logic (PPTL) was used for model checking, and the EDFA is verified. Finally, they get the composite automata by synthesis rules and generated the test cases. Mani and Prasanna [46] proposed a test case generation approach using sequence diagrams. A stimulus liking table (SLT) is generated from the sequence diagram. Using SLT, the stimulus paths are derived, and from the stimulus paths test cases were generated. Finally, they used boundary value analysis for test case reduction.

| Solutions using CD

Test case generation techniques using collaboration diagrams as input models are presented in [4, 52, 53, 54] (see Figure 6). All the techniques presented in [4, 52, 53] use different variants of graphs as an intermediate form. Samuel et al. [4]

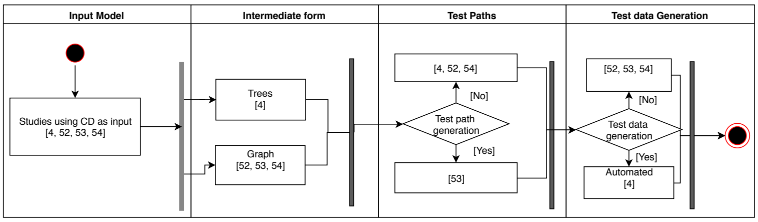


FIGURE 6 Solutions using collaboration diagram (CD) as input

use trees while Ahmed et al. and Prasanna et al. [52, 53] use a weighted graph. The weighted graph is then traversed using Prims and Dijkstra algorithms. An advantage of [53] over [52] is that mutation analysis is performed to evaluate the effectiveness of the proposed technique. Samuel et al. [4] convert a collaboration diagram into a tree format. The predicates are selected from the tree structure in an iterative manner. In the end, a function minimization technique is applied to generate test data for the selected predicates. Abdurazik and Offutt [54] identified four items that could be used for static checking. The items that have been identified are classifier roles, collaborating pairs, message or stimulus, and local variable definition – usage link pairs. For dynamic testing, they make use of message sequence paths. It is stated that message sequence path includes all variable def-use link pairs, object def-use link pairs, object creation-usage link pairs, and object usage-destruction link pairs. The techniques proposed by [54, 52, 53] only generate test paths, the only technique which generates test data and test cases are proposed by Samuel et al. [4].
Algorithms used in collaboration diagram based solutions are listed in Table 7.

TABLE 7 Algorithms used in CD based solutions

Studies	Algorithm
[52, 53]	Prim's and Dijkstra's
[4]	Post order traversal
[54]	Other (Self defined)

| Hybrid solutions

In this study, we found 24 hybrid solutions (combining interaction diagram with some other UML model) for integration testing [63, 60, 56, 57, 70, 6, 64, 69, 15, 61, 65, 59, 16, 66, 67, 10, 5, 1, 68, 62, 3, 71, 55, 58]. An overview of the solutions is shown in Figure 7. Hybrid solutions utlize two or more input models. Interaction diagrams (sequence or collaboration) are the main input models along with other UML model types. The models which are being used as inputs for integration testing along with the interaction models are: class diagrams, state charts, activity diagrams, use case diagrams, and interaction overview diagrams. Among the investigated techniques 20 [60, 56, 70, 6, 64, 15, 61, 65, 59, 16, 66, 67, 10, 5, 1, 68, 62, 55, 68, 69] use sequence diagrams along with some other UML diagram as input models. Four solutions [63, 57, 3, 71] use collaboration diagrams along with some other UML model. Figure 7 shows the combinations of UML diagrams used as input (e.g. sequence diagrams are combined with class diagrams, state charts, etc.).

All the hybrid solutions convert each primary input model into some intermediate forms and then convert these intermediate forms into final intermediate form, from which test paths are generated. The final intermediate forms

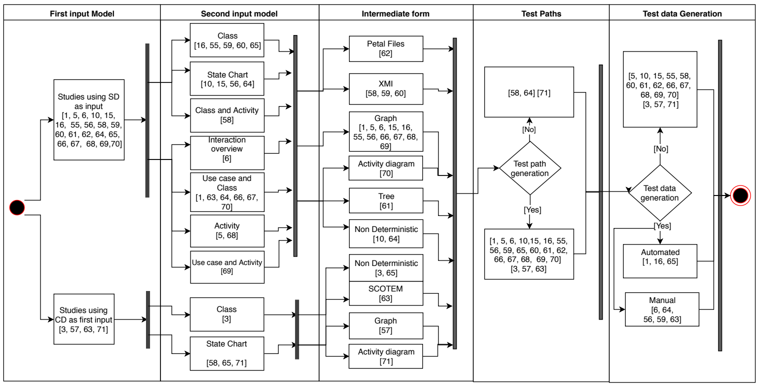


FIGURE 7 Solutions using interaction diagram and some other UML diagram (H) as input

used are different variants of graphs [1, 6, 55, 56, 57, 5, 15, 16, 66, 67, 68, 69], XMI [58, 59, 60], Tree [61], Petal files [62], "state collaboration test model" (SCOTEM) [63], and activity diagram [70, 71]. The techniques presented in [3, 64, 65, 10] do not use any intermediate form. The variants of graphs used in different techniques are polymorphism class object method acyclic graph (PECOMDAG) [55], system testing graph [15, 67, 68], sequence diagram graph [66], activity-sequence graph [5], sequence interaction graph [6], test model graph [57], concurrent control flow graph [56], extended variable assignment graph (EVAG) [56], and structured composite graph [16].

Twenty-one out of twenty-four techniques generate test paths. One technique presented by Wu et al. [71] does not generate test paths. Instead, it derives a test adequacy criteria based on input diagrams, that is collaboration and state chart diagrams. Algorithms used in hybrid solutions are listed in Table 8.

TABLE 8 Algorithms used in hybrid solutions

Studies	Algorithm
[15, 69]	Genetic algorithm (GA)
[1, 6, 68]	Depth first search algorithm (DFS)
[5, 16, 66]	Breadth first search algorithm (BFS)
[55, 56, 57, 61, 63, 67, 70]	Other traversing algorithms
[3, 10, 64, 65, 59, 60, 62, 58, 71]	Not mentioned

The techniques function in a similar way to the approaches described in Section 4.1. Given the similarity they are not explained in the same detail as the SD-based solutions.

Nine techniques [63, 56, 57, 6, 64, 65, 59, 16, 55] provide solutions for generating test data. Four techniques [56, 65, 16, 55] generate test data automatically while five techniques [63, 57, 6, 64, 59] rely on manual test data generation. An overview of the techniques based on hybrid solutions is presented in Figure 7.

4.2 | RQ2: What capabilities and limitations are observed in relation to the approaches?

The capabilities and limitations of the approaches are highlighted to answer RQ2. Table 9 provides an overview of the capabilities. A subset of approaches (57%) supports the generation of test paths.

The types of faults possible to detect are interaction, scenario, and operational faults. Only for SD-based solutions the ability to detect concurrency problems has been highlighted. While test scenarios are generated, only a subset of solutions also provide support to generate test input data, either manually (9 of 54) or automatically (7 of 54).

An important feature of the approaches proposed in the literature is the degree of evaluation. The most commonly stated methods used for the evaluation are case studies and experiments. Furthermore, to demonstrate the merits of the approaches, proof of concepts have been used, though they represent the weakest form of evaluation. Overall, 29 papers stated that they present evaluations either using case studies or experiments. In the following section presenting RQ3, the completeness of reporting with regard to rigor and relevance is assessed using the rubric by Ivarsson and Gorschek [21].

Coverage criteria were defined for the approaches, and the most commonly targeted criterion was path coverage, 46% of the selected studies are using this criterion. Among the other criteria, condition coverage was used by 9%, message coverage was used by 20%, and model coverage was used by 11% of the selected primary studies; 14% of the studies use other criteria, or they do not mention them.

TABLE 9 Capabilities in relation to MBT approaches

Capabilities		SD	CD	H
Testing	Test paths (scenario) generation	[2, 7, 12, 14, 35, 45, 37, 38, 39, 48, 33, 43, 49, 47, 46, 51]	[53]	[1, 3, 6, 69, 61, 15, 5, 55, 10, 56, 60, 68, 57, 63]
	Test input generation (Manual)	[7, 46, 40, 13]		[6, 64, 56, 59, 63]
	Test input generation (Automated)	[36, 2, 11]	[4]	[16, 65, 1]
	Tool support	[37, 38, 40, 42, 11, 44, 13, 66, 2]	[4]	[1, 64, 56, 65, 59, 62, 63, 66, 70]
Detection of faults	Interaction, scenario, operational	[7, 50, 45, 37, 51, 38, 12, 40, 13, 33, 43]	[53, 4]	[6, 15, 64, 66, 65, 1, 67, 59, 60, 63, 69]
	Concurrency (e.g. deadlock)	[34, 35, 41, 14]		
Coverage criteria	Path	[12, 14, 34, 39, 40, 33, 43, 45, 66, 2, 46, 49, 47, 50, 51]	[4]	[6, 56, 57, 63, 5, 15, 66, 68, 69]
	Predicates/ conditions	[36, 38]	[4]	[1, 64]
	Messages	[7, 35]	[53, 54]	[60, 61, 10, 16, 67, 70, 71]
	Transitions	[48]		
	Model	[11, 13, 17]		[58, 62, 3]
Stated evaluation method	Experiment	[39, 40, 13, 66, 2]		[64, 66]
	Case study	[7, 45, 37, 51, 38, 12, 41, 33, 43, 47]	[52, 53]	[6, 15, 5, 62, 1, 55, 59, 60, 63, 3, 69]
	Proof of concept (Examples)	[34, 35, 36, 50, 48, 11, 44, 17, 42, 14, 46, 49]	[4, 54]	[58, 61, 16, 65, 70, 67, 10, 68, 57]

The limitations are summarized in Table 10. A majority of the proposed techniques do not provide any support for test automation (tool support), test data generation, and test case execution. A large number of techniques are categorized as complex because these are the techniques which are not only manual but also involve multiple complicated steps. Some of the techniques have issues in the transformation of the primary model to the intermediate form. For example, the algorithms used for transformation are of complex nature [48]. Some of the techniques have restrictions in the proposed methods, for example, working with a specific format [58]. A few techniques lack optimization capabilities, these techniques generate test cases for all possible paths, and thus, there is a possibility to generate redundant test cases.

TABLE 10 Limitations

Limitations	SD	CD	H
No tool support	[7, 12, 14, 34, 35, 36, 39, 41, 43, 17, 45, 46, 47, 48, 49, 50, 51]	[52, 53, 54]	[3, 6, 55, 57, 58, 60, 61, 10, 5, 15, 16, 67, 68, 69, 71]
No test data generation	[14, 34, 35, 36, 39, 40, 41, 42, 43, 44, 13, 17, 45, 66, 47, 48, 49, 50, 51]	[52, 53, 54]	[3, 55, 57, 58, 61, 62, 10, 5, 15, 66, 67, 68, 69, 70, 71]
Complexity	[7, 12, 14, 34, 35, 36, 41, 17, 45, 46, 47, 48, 49, 50, 51]	[52, 53, 54]	[6, 55, 57, 60, 61, 10, 5, 15, 16, 67, 68, 69, 71]
No test case execution	[7, 12, 14, 34, 35, 36, 38, 39, 40, 41, 42, 43, 11, 44, 13, 17, 45, 66, 2, 46, 47, 48, 49, 50, 51]	[52, 53, 4, 54]	[1, 3, 6, 64, 55, 56, 57, 58, 60, 61, 62, 10, 5, 15, 16, 66, 67, 68, 69, 70, 71]
Redundant test cases	[35, 45]		[55]
Issues in transforming model into intermediate form	[7, 50, 51, 39, 48, 41]		[61, 66, 16]
Restriction in proposed methods	[11]	[4, 54]	[55, 56, 58]

4.3 | RQ3: What is the rigor and relevance of the included primary studies according to the rigor and relevance scoring rubric proposed by Ivarsson and Gorschek?

The main objective of this section is to provide the rigor and relevance scores for the primary studies. The rigor and relevance scores of each publication containing an empirical study (i.e., case study or experiment) are summarized in Tables 11 to 13. The tables show the studies that either stated that a case study or experiment, i.e. an empirical study has been conducted. A large number of studies in all categories scored only the rating "0" according to the rubric by Ivarsson and Gorschek. The reason for the assessment can be found in the article by Wohlin [72], who highlights that *"case study is often a misused research method. In many papers, the authors mean example or illustration and not a case study in the way it is defined"*. In particular, Wohlin highlights the references by Yin [73] and Runeson and Höst [74] as a guide to follow. Similarly, the experiment was frequently used as a means to present results from operations on an example system. Though many important parts of experimental research have not been documented. In particular the guidelines by Basili [75], Wohlin et al. [76] and Juristo [77] are to be highlighted here. None of the case studies referred to a guide for the research method. Hence, a key area for improvement is to utilize guidelines and to label the research method used correctly. In many cases, the correct label (example or proof of concept) was used. In Table 13, we can see some good scores in the rigor column. Similarly, the relevance score is comparatively better for the studies presented during recent years. In Table 11, we can see the presence of rigor for the years 2005 and 2016. This is an indication for 1) quality of the recent studies regarding rigor score is slightly better, 2) hybrid solutions have better relevance score. However, regarding the overall results, it is visible that no clear trend of scores concerning years can be seen in Tables 11 to 13 as we found papers with high and low scores for both new and old papers.

| RQ3.1: What is the extent of relatedness among the included primary studies?

Another interesting factor is to see how the studies are related to each other. Of the 54 primary studies, only 25 studies are referred by the other included primary studies. Generally, studies are said to be related to each other if authors are referring the other studies while defining their methodology. We found only two studies ([55] is citing [11], and [52] is citing [53]) in their methodology directly and explicitly drawing ideas from them. Other studies are referring to each other, mainly in the related work section or in the introduction section. Furthermore, it is of interest to see whether studies conduct a comparative analysis. Again, we found only two studies in which authors compared their work with

TABLE 11 Rigor and relevance scores: Sequence diagrams

Ref	Rigor				Relevance				Year
	C	D	V	C+D+V	U	C	S	U+C+S	
[43]	0.5	0.5	0	1	0	0	0	0	2016
[47]	0.5	0.5	0	1	0	0	0	0	2016
[7]	0	0	0	0	0	0	0	0	2015
[51]	0	0	0	0	0	0	0	0	2014
[37]	0	0	0	0	0	0	0	0	2013
[38]	0	0	0	0	0	0	0	0	2013
[45]	0	0	0	0	0	0	0	0	2012
[41]	0	0	0	0	0	0	0	0	2011
[40]	0	0	0	0	0	0	0	0	2008
[12]	0	0	0	0	0	0	0	0	2007
[2]	0	0	0	0	0	0	0	0	2007
[39]	0	0	0	0	0	0	0	0	2006
[13]	1	1	0.5	2.5	0.5	0	0	0.5	2005

Rigor– C: Context, D: Study design, V: Validity threats
Relevance– U: User/Subjects, C: Context (industry), S: Scale

TABLE 12 Rigor and relevance scores: Collaboration diagrams

Ref	Rigor				Relevance				Year
	C	D	V	C+D+V	U	C	S	U+C+S	
[52]	0	0	0	0	0	0	0	0	2013
[53]	0.5	1	0	1.5	0.5	0	0	0.5	2011

Rigor– C: Context, D: Study design, V: Validity threats
Relevance– U: User/Subjects, C: Context (industry), S: Scale

already existing studies ([60] compared with [10] and [1] compared with [16]). A complete mapping of the references of the primary studies with each other is presented in Table 14.

5 | DISCUSSION

This section presents the analysis of the results that are synthesized during the review of the literature with respect to the research questions specified. Furthermore, relationships between the included papers are explored.

5.1 | Analysis of research questions

About **RQ1** (*Which MBT test case generation approaches based on UML interaction diagrams have been proposed in the literature?*), we were interested to see the publication trends and classification of the techniques. Regarding publication trends, we found that the authors who published their articles in the journals, few of them targeted venues with a high scientific impact (ISI and SCOPUS indexed journals). On the other hand, regarding conferences, we have found that the majority of papers are from the conferences who have published their proceedings in IEEE, ACM, and Springer. One reason for not considering high impact journals could be that the studies do not contain a strong empirical evaluation (see results of RQ3).

Regarding the classification of the techniques, we found that the majority (48%) of the proposed test case generation approaches belong to category "SD", transforming UML sequence diagrams into intermediate forms such as control flow graph, sequence dependency graph, or sequence diagram graph. The integration level testing has been the main

TABLE 13 Rigor and relevance scores: Hybrid

Ref	Rigor				Relevance				Year
	C	D	V	C+D+V	U	C	S	U+C+S	
[69]	0	1	0.5	1.5	0	0	0	0	2016
[6]	1	1	0	2	0.5	0	0	0.5	2015
[63]	1	1	0.5	2.5	0.5	0	1	1.5	2015
[66]	0	0	0	0	0	0	0	0	2014
[55]	0.5	0	0	0.5	0	0	0	0	2014
[15]	0	0	0	0	0	0	0	0	2012
[5]	0	0	0	0	0	0	0	0	2010
[60]	1	1	0.5	2.5	0.5	0	0	0.5	2010
[62]	0	0	0	0	0	0	0	0	2009
[56]	0.5	0.5	0	1	0	0	0	0	2008
[59]	0	0	0	0	0	0	0	0	2008
[64]	1	1	1	3	0.5	0	0	0.5	2007
[1]	0	0.5	0	0.5	0	0	0	0	2007
[3]	0	0	0	0	0	0	0	0	2007

Rigor– C: Context, D: Study design, V: Validity threats
Relevance– U: User/Subjects, C: Context (industry), S: Scale

TABLE 14 Primary studies referring to each other

Referred to	**Referring Primary Studies**			
Primary Study	Introduction	Related Work	Methodology	Comparison
[54]	[52, 57, 6, 2]	[63, 56, 50, 38, 4, 66, 12]		
[63]	[57, 7],	[53]		
[60]		[39, 35, 12, 1]		
[56]	[45, 13]			
[44]		[51, 48]		
[70]	[10]	[63, 56, 50, 1]		
[51]	[53, 1]	[35, 13, 14]		
[11]	[38]	[63, 51, 17, 48, 4, 66, 15, 13, 12, 1, 40, 55]	[55]	
[64]	[45, 13]			
[35]	[39]	[14]		
[2]		[35, 53, 12, 62]		
[17]		[48]		
[34]		[58]		
[16]				[1]
[38]	[39]	[62]		
[53]	[45]		[52]	
[45]	[58]			
[48]		[39, 35]		
[4]		[40]		
[67]		[52, 6, 35, 53, 1]		
[37]	[52, 45]	[62]		
[10]		[60, 45, 48]		[60]
[5]	[45]			
[1]		[50, 6, 35, 53]		
[68]		[15]		

focus of the approaches. The main objectives included: the detection of interaction, operational, or scenario faults [7, 50, 45, 37, 51, 38, 12, 40, 13, 33, 43]; to address concurrency issues such as deadlocks and synchronization problems [34, 35, 41, 14]; to generate test paths with high coverage [12, 14, 34, 39, 40, 33, 43, 45, 66, 2, 46, 49, 47, 50, 51]; to generate a Markov chain model [44]; to test web pages [42]; to test object oriented applications [36, 11].

Commonly used algorithms are depth first search [33, 38, 40, 45, 51], genetic algorithm (GA) [12, 13, 39] , and other traversing algorithms that are proposed [2, 17, 41, 43, 44, 46, 47, 48, 49, 50].

During review of literature, it has been noted that a few techniques (only 7%) have been identified, in which the input model is a UML collaboration diagram. The main objective of these techniques is to increase the reliability and detection of faults (interaction). The intermediate forms that are generated include trees and weighted graphs. These intermediate forms are traversed to generate test paths. The traversing algorithms are Prim's and Dijkstra for the weighted graph.

The hybrid techniques for test case generation are also proposed, and 45% of the selected primary studies belong to this category. The UML interaction models (sequence, collaboration) are used with a combination of other diagrams (class, activity, statechart, and use case). The main objectives of these proposed technique is to detect interaction, scenario and operational faults [6, 15, 66, 65, 1, 67, 59, 60, 63, 69], and test cases generation from analysis artefacts [65, 1]. These input models are transformed into intermediate forms. The most common intermediate forms include sequence interaction graph [7], structured composite graph [16], system testing graph [67], activity sequence graph [5] and system graph [68]. The intermediate forms are traversed to formulate the test paths. These test paths are then used for test case generation.

With regard to **RQ2** (*"What capabilities and limitations are observed in relation to the approaches?"*) the most common merits that are achieved include detection of specific fault types, test input generation (few techniques generate automated test input), and coverage criteria achieved (paths, messages, transition). The most commonly highlighted drawbacks were the lack of support for test data generation, no tool support, complexity of test generation steps, no support for test case execution. Furthermore, issues while converting input models into intermediate representations have been highlighted. In particular, high complexity and lack of tool support have been highlighted as hindrances for a transfer of research results to industry practice [78]. Thus, this is an important focus of future research to address.

The rigor and relevance of studies has been assessed to answer **RQ3** (*"What is the rigor and relevance of the included primary studies according to the rigor and relevance scoring rubric proposed by Ivarsson and Gorschek?"*). With regard to rigor of the primary studies only few studies received scores higher than "0". The main reason may be that no research guideline has been followed, and the research method has been wrongly labeled. As mentioned earlier, Wohlin [72] pointed out that in many cases, illustrations or examples are presented instead of studies following the methodological guidelines. Hence, the clear need for empirical studies following the guidelines for experimentation [75, 77, 76] and case studies [73, 74] should be highlighted. In particular, empirical comparisons between different approaches presented here are of interest. For future work, we propose to compare approaches within the categories shown in Table 5. With regard to relevance, given that examples have been used, no real users or industrial systems have been used as the system under test (SUT). Hence, seeking collaboration with industry is an important future endeavor.

Furthermore, our results reveal that the studies do not make explicit enough, which techniques are used as inputs to create a new technique. Thus, we define relatedness as follows: Authors of papers reference other techniques in the presentation of their approach, or they are comparing their technique with existing techniques. We found only four studies that are referenced by other studies included in our map according to the definition above. The weak relatedness is defined if the authors are citing to other techniques in their related work section. In this regard, we only found 19 studies included in our map that are referenced by other studies in our systematic map. This shows that test case generation using interaction diagrams is not a well researched and mature area.

6 | CONCLUSION

This paper presents a systematic mapping study of model-based test case generation techniques based on UML interaction diagrams. It has been explored that these diagrams are being used mostly for integration testing. We discovered that UML interaction diagrams are being used as a standalone model to generate test cases, at the same time there are the techniques which combine interaction models with some other UML model (class, state chart, etc.). The proposed solutions have various capabilities. Approaches from all investigated categories (SD, CD, and H) are capable of generating test paths, though only a few approaches generate the needed inputs (test data). The techniques also support detecting different types of defects, such as interaction and concurrency problems. It is also important to point out that solutions from sequence diagrams are applicable to collaboration diagrams if the collaboration diagrams include temporal information (numbering of messages).

The study shows that the practical usefulness of the proposed solutions still needs to be demonstrated. Key issues highlighted by the authors were the lack of tool support and the complexity of the approaches proposed. Thus, the transfer of the approaches to the industry would be facilitated through developing tools that can be integrated into existing development frameworks. Furthermore, studies did not follow guidelines when conducting and reporting case studies or experiments. The systematic use of guidelines will aid in further improving the rigor of studies.

In summary, the following work should be focused on in the future:

- Provide tool support to enable practitioners to integrate the solutions into their development processes and development tools.
- Within an academic environment, empirically compare different types of solutions (such as different intermediate forms) with respect to ease of use, efficiency (number of tests run in a specific time frame) and effectiveness (defect detection ability) through experimentation. The results provide preliminary insights of which solutions have the potential for practical use.
- Empirically compare and study the solutions in real industrial contexts through case studies and action research.

With these steps taken, different forums may be targeted, such as Empirical Software Engineering, primarily focusing on empirical studies.

APPENDIX

A | TREATMENT OF DATABASES

Databases often provide different settings and possibilities for searches. In the following, we specify how the different databases have been used to retrieve the articles. Google Scholar has been used as an index database.

- *IEEE Search Criteria:* The default search option is used to search the related publications. No advanced options such as search only title, keywords and abstract, etc. were used.
- *Science Direct:* The advanced search option was used to identify the related publications. The database did not include options for conferences, but rather only for books and journals. Reference works (secondary studies) have been excluded.
- *ACM:* The advanced search option was used to identify the related publications from ACM digital library. The filter is also applied executing the search string, and only title is selected to match with the search string. Moreover, we

searched for the publications from “ACM full-text selection”.

- *Google scholar:* The advanced search option was used to identify publications from Google scholar. The first filter that is applied is search only “in the title of the article”. From the “find article” option “with the exact phrase” and “with at least one word” has been selected. More specifically, the search string is (“test case generation”) AND (“Model based testing”), and after that “test case generation” were searched with exact phrase “and model based testing” with at least one word in the phrase.

B | PUBLICATION VENUES

TABLE 15 Publication types (J = Journal, C = Conference, S = Symposium, W = Workshop) and publication venues of the primary studies

Ref#	Type	Venue	Year
[33]	J	International Refereed Journal of Engineering & Technology	2017
[46]	J	Journal of Engineering Science and Technology	2017
[43]	J	British Journal of Mathematics & Computer Science	2016
[47]	J	Chinese Journal of Electronics	2016
[69]	J	Journal of Software	2016
[6]	J	Computational Intelligence in Data Mining	2015
[15]	J	Procedia Computer Science	2015
[58]	J	International Journal for Innovative Research in Science and Technology	2015
[34]	J	International Journal of Computer Science and Engineering	2014
[62]	J	International Journal of Innovations in Engineering and Technology	2014
[52]	J	World Journal of Science and Technology	2013
[45]	J	International Journal of Advanced Research in Computer Science and Software Engineering	2013
[12]	J	International Journal of Computer Applications	2013
[5]	J	The International Journal of Computer Science & Applications (TIJCSA)	2012
[14]	J	ACM SIGSOFT Software Engineering Notes	2012
[53]	J	IETE Journal of research	2011
[41]	J	International Journal of Computer Science and Information Technologies	2011
[16]	J	Journal of Object Technology	2010
[1]	J	International Journal of Software Engineering	2010
[57]	J	Information Technology and Control	2009
[4]	J	Information and Software Technology	2007
[63]	J	Information and Software Technology	2007
[49]	C	International Conference on Computer and Information Science	2016
[7]	C	Intelligent Computing, Communication and Devices	2015
[50]	C	International Conference on Intelligent Computing Applications (ICICA)	2014
[61]	C	International Conference on Computer Science & Education	2014
[38, 68]	C	International Conference on Advances in Computing	2013
[37]	C	International Conference on Software and Computer Applications	2012
[35]	C	International Conference on Electronics Computer Technology (ICECT)	2011
[60, 13]	C	International Conference on Contemporary Computing	2010
[56]	C	International Conference on Software Testing Verification and Validation	2009
[55]	C	International Conference for Young Computer Scientists	2008
[48]	C	International Conference on Software Engineering, Articial Intelligence, Networking, and Parallel/Distributed Computing	2008
[64]	C	Ninth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing	2008
[51]	C	International Conference on Systems, Man and Cybernetics	2007
[2]	C	International Conference on Computational Intelligence and Security	2007
[65]	C	International Conference on Quality Software	2007
[59]	C	Conference of the center for advanced studies on Collaborative research	2007
[66]	C	International Conference on Advanced Computing and Communications	2007
[67]	C	International Conference on Information Technology	2007
[10]	C	GI Jahrestagung	2006
[42]	C	International Conference on Computational Science and Its Applications	2005
[36]	C	Annual IEEE India Conference-Indicon	2005
[44]	C	Asian Test Symposium, ATS 2003	2003
[71]	C	International Conference on COTS-Based Software Systems	2003
[70]	C	International Conference on the Unified modeling Language	2003
[11]	C	International Conference on Automated Software Engineering, ASE	2002
[54]	C	International Conference on the Unified modeling Language	2000
[39]	S	International Symposium on Telecommunications (IST)	2014
[3]	S	Annual Midwest Instruction and Computing Symposium	2014
[40]	S	International Computer Symposium	2008
[17]	W	International workshop on software test	2006

REFERENCES

[1] Swain SK, Mohapatra DP, Mall R. Test case generation based on use case and sequence diagram. International Journal of Software Engineering 2010;3(2):21–52.

[2] Li BL, Li Zs, Qing L, Chen YH. Test case automate generation from UML sequence diagram and OCL expression. In: International Conference onComputational Intelligence and Security, 2007 IEEE; 2007. p. 1048–1052.

[3] Wang Y, Zheng M. Test case generation from uml models. In: 45th Annual Midwest Instruction and Computing Symposium, Cedar Falls, Iowa, vol. 4; 2012. .

[4] Samuel P, Mall R, Kanth P. Automatic test case generation from UML communication diagrams. Information and software technology 2007;49(2):158–171.

[5] Sumalatha VM, Raju G. UML based Automated Test Case Generation technique using Activity-Sequence diagram. The International Journal of Computer Science & Applications (TIJCSA) 2012;1(9).

[6] Jena AK, Swain SK, Mohapatra DP. Model Based Test Case Generation from UML Sequence and Interaction Overview Diagrams. In: Computational Intelligence in Data Mining-Volume 2 Springer; 2015.p. 247–257.

[7] Jena AK, Swain SK, Mohapatra DP. Test case creation from UML sequence diagram: a soft computing approach. In: Intelligent Computing, Communication and Devices Springer; 2015.p. 117–126.

[8] Files A. OMG Unified Modeling Language TM (OMG UML) 2013;.

[9] Hartmann J, Imoberdorf C, Meisinger M. UML-based integration testing. In: ACM SIGSOFT Software Engineering Notes, vol. 25 ACM; 2000. p. 60–70.

[10] Sokenou D. Generating Test Sequences from UML Sequence Diagrams and State Diagrams. In: GI Jahrestagung (2) Citeseer; 2006. p. 236–240.

[11] Fraikin F, Leonhardt T. SeDiTeC-testing based on sequence diagrams. In: 17th IEEE International Conference on Automated Software Engineering, 2002. ASE 2002. IEEE; 2002. p. 261–266.

[12] Sumalatha VM, Raju GSVP. Object Oriented Test Case Generation Technique using Genetic Algorithms. International Journal of Computer Applications 2013;61(20).

[13] Shirole M, Kumar R. A hybrid genetic algorithm based test case generation using sequence diagrams. In: International Conference on Contemporary Computing Springer; 2010. p. 53–63.

[14] Shirole M, Kumar R. Testing for concurrency in UML diagrams. ACM SIGSOFT Software Engineering Notes 2012;37(5):1–8.

[15] Khurana N, Chillar R. Test Case Generation and Optimization using UML Models and Genetic Algorithm. Procedia Computer Science 2015;57:996–1004.

[16] Nayak A, Samanta D. Automatic test data synthesis using uml sequence diagrams. journal of Object Technology 2010;9(2):75–104.

[17] Lund MS, Stølen K. Deriving tests from UML 2.0 sequence diagrams with neg and assert. In: Proceedings of the 2006 international workshop on Automation of software test ACM; 2006. p. 22–28.

[18] Singh R. Test case generation for object-oriented systems: A review. In: Fourth International Conference onCommunication Systems and Network Technologies (CSNT), 2014 IEEE; 2014. p. 981–989.

[19] Utting M, Pretschner A, Legeard B. A taxonomy of model-based testing approaches. Software Testing, Verification and Reliability 2012;22(5):297–312.

[20] Shirole M, Kumar R. UML behavioral model based test case generation: a survey. ACM SIGSOFT Software Engineering Notes 2013;38(4):1–13.

[21] Ivarsson M, Gorschek T. A method for evaluating rigor and industrial relevance of technology evaluations. Empirical Software Engineering 2011;16(3):365–395.

[22] Petersen K, Vakkalanka S, Kuzniarz L. Guidelines for conducting systematic mapping studies in software engineering: An update. Information and Software Technology 2015;64:1–18.

[23] Petersen K, Feldt R, Mujtaba S, Mattsson M. Systematic Mapping Studies in Software Engineering. In: EASE, vol. 8; 2008. p. 68–77.

[24] Dias Neto AC, Subramanyan R, Vieira M, Travassos GH. A survey on model-based testing approaches: a systematic review. In: Proceedings of the 1st ACM international workshop on Empirical assessment of software engineering languages and technologies: held in conjunction with the 22nd IEEE/ACM International Conference on Automated Software Engineering (ASE) 2007 ACM; 2007. p. 31–36.

[25] Häser F, Felderer M, Breu R. Software paradigms, assessment types and non-functional requirements in model-based integration testing: a systematic literature review. In: Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering ACM; 2014. p. 29.

[26] Mohi-Aldeen SM, Deris S, Mohamad R. Systematic Mapping Study in Automatic Test Case Generation. In: SoMeT; 2014. p. 703–720.

[27] Khandai M, Acharya AA, Mohapatra DP. A survey on test case generation from uml model. International Journal of Computer Science and Information Technologies 2011;2(3):1164–1171.

[28] Shafique M, Labiche Y. A systematic review of model based testing tool support. Carleton University, Canada, Tech Rep Technical Report SCE-10-04 2010;.

[29] Kitchenham B, Charters S, Guidelines for performing Systematic Literature Reviews in Software Engineering; 2007.

[30] Gorschek T, Tempero E, Angelis L. On the use of software design models in software development practice: An empirical investigation. Journal of Systems and Software 2014;95:176–193.

[31] Brereton P, Kitchenham BA, Budgen D, Turner M, Khalil M. Lessons from applying the systematic literature review process within the software engineering domain. Journal of systems and software 2007;80(4):571–583.

[32] Kitchenham B, Pretorius R, Budgen D, Brereton OP, Turner M, Niazi M, et al. Systematic literature reviews in software engineering–a tertiary study. Information and Software Technology 2010;52(8):792–805.

[33] Chandnani K, Patidar CP, Sharma M. Automatic And Optimized Test Case Generation Using Model Based Testing Based On Sequence Diagram And Discrete Particle Swarm Optimization Algorithm. International Refereed Journal of Engineering & Technology (IRJET) 2017;1(2):1–6.

[34] Mallick A, Panda N, Acharya AA. Generation of Test Cases from UML Sequence Diagram and Detecting Deadlocks using Loop Detection Algorithm. International Journal of Computer Science and Engineering 2014;2:199–203.

[35] Khandai M, Acharya AA, Mohapatra DP. A novel approach of test case generation for concurrent systems using UML Sequence Diagram. In: 3rd International Conference on Electronics Computer Technology (ICECT), 2011, vol. 1 IEEE; 2011. p. 157–161.

[36] Samuel P, Mall R, Sahoo S. Uml sequence diagram based testing using slicing. In: 2005 Annual IEEE India Conference-Indicon IEEE; 2005. p. 176–178.

[37] Shanthi A, Kumar GM. Automated test cases generation from uml sequence diagram. In: International Conference on Software and Computer Applications; 2012. .

[38] Panthi V, Mohapatra DP. Automatic test case generation using sequence diagram. In: Proceedings of International Conference on Advances in Computing Springer; 2013. p. 277–284.

[39] Hoseini B, Jalili S. Automatic test path generation from sequence diagram using genetic algorithm. In: 7th International Symposium on Telecommunications (IST), 2014 IEEE; 2014. p. 106–111.

[40] Lei YC, Lin NW. Test Case Generation Based on Sequence Diagrams. In: International Computer Symposium, ICS2008; 2008. p. 349–355.

[41] DebashreePatnaik AAA, Mohapatra DP. GENERATION OF TEST CASES USING UML SEQUENCE DIAGRAM IN A SYSTEM WITH COMMUNICATION DEADLOCK. (IJCSIT) International Journal of Computer Science and Information Technologies 2011;3:1187–1190.

[42] Cho Y, Lee W, Chong K. The technique of test case design based on the UML sequence diagram for the development of Web applications. In: International Conference on Computational Science and Its Applications Springer; 2005. p. 1–10.

[43] Rhmann W, Saxena V. Generation of Test Cases from UML Sequence Diagram Based on Extenics Theory. British Journal of Mathematics & Computer Science 2016;16(1).

[44] Beyer M, Dulz W, Zhen F. Automated TTCN-3 test case generation by means of UML sequence diagrams and Markov chains. In: 12th Asian Test Symposium, 2003. ATS 2003. IEEE; 2003. p. 102–105.

[45] Priya SS, Malarchelvi PSK. Test Path Generation Using Uml Sequence Diagram. International Journal of Advanced Research in Computer Science and Software Engineering 2013;3(4).

[46] Mani P, Prasanna M. TEST CASE GENERATION FOR EMBEDDED SYSTEM SOFTWARE USING UML INTERACTION DIAGRAM. Journal of Engineering Science and Technology 2017;12(4):860–874.

[47] Zhang C, Duan Z, Yu B, Tian C, Ding M. A Test Case Generation Approach Based on Sequence Diagram and Automata Models. Chinese Journal of Electronics 2016;25(2):234–240.

[48] Samuel P, Joseph AT. Test sequence generation from UML sequence diagrams. In: Ninth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, 2008. SNPD'08. IEEE; 2008. p. 879–887.

[49] Seo Y, Cheon EY, Kim JA, Kim HS. Techniques to generate UTP-based test cases from sequence diagrams using M2M (Model-to-Model) transformation. In: IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS), 2016 IEEE; 2016. p. 1–6.

[50] Dhineshkumar M, et al. An Approach to Generate Test Cases from Sequence Diagram. In: 2014 International Conference on Intelligent Computing Applications (ICICA) IEEE; 2014. p. 345–349.

[51] Cartaxo EG, Neto FG, Machado PD. Test case generation by means of UML sequence diagrams and labeled transition systems. In: 2007 IEEE International Conference on Systems, Man and Cybernetics IEEE; 2007. p. 1292–1297.

[52] Ahmed SU, Sahare SA, Ahmed A. Automatic test case generation using collaboration UML diagrams. World Journal of Science and Technology 2013;2.

[53] Prasanna M, Chandran KR, Thiruvenkadam K. Automatic test case generation for uml collaboration diagrams. IETE Journal of research 2011;57(1):77–81.

[54] Abdurazik A, Offutt J. Using UML collaboration diagrams for static checking and test generation. In: International Conference on the Unified Modeling Language Springer; 2000. p. 383–395.

[55] Zhou H, Huang Z, Zhu Y. Polymorphism sequence diagrams test data automatic generation based on OCL. In: The 9th International Conference for Young Computer Scientists, 2008. ICYCS 2008. IEEE; 2008. p. 1235–1240.

[56] Bandyopadhyay A, Ghosh S. Test input generation using UML sequence and state machines models. In: International Conference on Software Testing Verification and Validation IEEE; 2009. p. 121–130.

[57] Barisas D, Bareiša E. A software testing approach based on behavioral UML models. Information Technology and Control 2015;38(2).

[58] Kumar B, Singh K. Testing UML Designs Using Class, Sequence and Activity Diagrams. International Journal for Innovative Research in Science and Technology 2015;2(3):71–81.

[59] Maibaum T, Li ZJ. A test framework for integration testing of object-oriented programs. In: Proceedings of the 2007 conference of the center for advanced studies on Collaborative research IBM Corp., ACM; 2007. p. 252–255.

[60] Asthana S, Tripathi S, Singh SK. A novel approach to generate test cases using class and sequence diagrams. In: International Conference on Contemporary Computing Springer; 2010. p. 155–167.

[61] Li Y, Jiang L. The research on test case generation technology of UML sequence diagram. In: 2014 9th International Conference on Computer Science & Education; 2014. .

[62] Verma A, Dutta M. Automated Test case generation using UML diagrams based on behavior. International Journal of Innovations in Engineering and Technology (IJIET) 2014;4(1):31–39.

[63] Ali S, Briand LC, Rehman MJu, Asghar H, Iqbal MZZ, Nadeem A. A state-based approach to integration testing based on UML models. Information and Software Technology 2007;49(11):1087–1106.

[64] Kansomkeat S, Offutt J, Abdurazik A, Baldini A. A comparative evaluation of tests generated from different UML diagrams. In: Ninth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, 2008. SNPD'08. IEEE; 2008. p. 867–872.

[65] Li Z, Maibaum T. An approach to integration testing of object-oriented programs. In: Seventh International Conference on Quality Software (QSIC 2007) IEEE; 2007. p. 268–273.

[66] Sarma M, Kundu D, Mall R. Automatic test case generation from UML sequence diagram. In: International Conference on Advanced Computing and Communications, 2007. ADCOM 2007. IEEE; 2007. p. 60–67.

[67] Sarma M, Mall R. Automatic test case generation from UML models. In: 10th International Conference on Information Technology,(ICIT 2007). IEEE; 2007. p. 196–201.

[68] Tripathy A, Mitra A. Test case generation using activity diagram and sequence diagram. In: Proceedings of International Conference on Advances in Computing Springer; 2013. p. 121–129.

[69] Khurana N, Chhillar RS, Chhillar U. A Novel Technique for Generation and Optimization of Test Cases Using Use Case, Sequence, Activity Diagram and Genetic Algorithm. Journal of Software (JSW) 2016;11(3):242–250.

[70] Briand L, Labiche Y. A UML-based approach to system testing. In: International Conference on the Unified Modeling Language Springer; 2001. p. 194–208.

[71] Wu Y, Chen MH, Offutt J. UML-based integration testing for component-based software. In: International Conference on COTS-Based Software Systems Springer; 2003. p. 251–260.

[72] Wohlin C. Writing for synthesis of evidence in empirical software engineering. In: Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement ACM; 2014. p. 46.

[73] Yin RK. Case study research: Design and methods. Sage publications; 2013.

[74] Runeson P, Höst M. Guidelines for conducting and reporting case study research in software engineering. Empirical software engineering 2009;14(2):131–164.

[75] Basili VR, Selby RW, Hutchens DH. Experimentation in software engineering. IEEE Transactions on software engineering 1986;(7):733–743.

[76] Wohlin C, Runeson P, Höst M, Ohlsson MC, Regnell B, Wesslén A. Experimentation in software engineering. Springer Science & Business Media; 2012.

[77] Juristo N, Moreno AM. Basics of software engineering experimentation. Springer Science & Business Media; 2013.

[78] Garousi V, Petersen K, Ozkan B. Challenges and best practices in industry-academia collaborations in software engineering: A systematic literature review. Information and Software Technology 2016;79:106–127.