ORIGINAL ARTICLE

Journal Section

A Systematic Mapping of Test Case Generation Techniques Using UML Interaction Diagram

Nasir Mehmood Minhas[1,2] | Sohaib Masood[2] | Kai Petersen[1] | 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

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).

Testing plays a vital role for assuring software quality. Among the activities performed during testing process, test cases generation is a challenging and labor intensive task. Test case generation techniques based on UML models are getting the attention of researchers and practitioners. This study provides a systematic mapping of test case generation techniques based on 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.

It has been revealed that UML interaction diagrams based techniques are mainly used for integration testing. The majority of the techniques are using sequence diagrams as input models, while some are using collaboration. A notable number of techniques are using interaction diagram along with some other UML diagram for test case generation. These techniques are mainly focusing on interaction, scenario, operational, concurrency, synchronization and deadlock related faults.

From the results of this study, we can conclude that the studies presenting test case generation techniques using UML interaction diagrams failed to illustrate the use of rigor-

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

ous methodology, and these techniques did not demonstrate the empirical evaluation in an industrial context. Our study revealed the need for tool support to facilitate the transfer of solutions to 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]. It is one of the most expensive and labour intensive tasks. Organizations spend more than 50% of their total budget to test the software [2, 3]. Testing can be automated to various degrees, such as completely manual, semi-automated or fully automated. Automated testing has a benefit 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 design and code of software fulfills with requirements [5].

Among several testing activities the generation of test cases is an intellectually challenging task [6, 7, 1, 4]. In recent times, the complexity of software is increasing rapidly, which requires sufficient testing mechanisms to identify and eliminate the defects. Researchers and practitioners proposed various automated test case generation techniques during the last decade. Model Based Testing (MBT) is one of testing techniques. In the MBT approach, Unified Modeling Language (UML) models are commonly used as a means to generate test cases. Different system aspects could be tested from different UML models. UML models can be classified according to static or dynamic aspects of a system (c.f. [8]). Static aspects of a system can be represented by the structural models whereas dynamic aspects of the system can be represented by the behavioral models. Interaction diagrams fall under the category of behavioral models. Other diagrams related to this category are *use case diagram*, *stat chart diagram and activity diagram*. Whereas, structural models are represented by *class diagram*, *component diagram*, *deployment diagram*, *object diagram*, *package diagram*, *profile diagram*, *and composite diagram* [8]. In this study we focus on dynamic models as testing focuses on the dynamic aspects of the system during its execution. Interaction diagrams are of high relevance in later testing activities (such as integration testing) as the interaction between different components have 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 using UML models, practitioners may perform unit [10, 3], integration [11, 12, 13, 14], system [15, 16, 5] and conformance testing [17].

Looking at existing literature reviews, the reviews so far did not put their main focus on interaction diagrams, and the reviews 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. We believe that a classification of existing MBT techniques based on UML Interaction diagrams is essential for researchers before proceeding toward proposing new approaches using interaction diagrams. Furthermore, practitioners and researchers currently utilizing interaction diagrams have an inventory of results as well as a documentation of their capabilities and limitations. Furthermore, the quality of the study is assessed. In order to complement the existing literature reviews, we investigate the use of interaction diagrams making the following contributions:

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

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 the scientific quality with regard to the research process and method used. Relevance is focused on whether evaluations have been conducted in a realistic environment with regard to scale, context, and subjects.

The rest of this paper is structured as section: Section 2 discusses the related work regarding the already presented surveys/ reviews in area of model base testing. Section 3 describes the research methodology. The MBT test case generation approaches along with their capabilities and limitations compared and the results of the quality assessment are presented in Section 4. The results are summarized and discussed in Section 5. Section 6 concludes the paper.

2 | RELATED WORK

A number of literature reviews related to model based testing have been published. An overview of related literature reviews is provided in Table 1. The table states the objectives, the research method, as well as the number of primary studies that were included in the literature reviews. Several studies were generic without a particular focus on interaction diagrams, such as [18, 24, 25, 26]. Only a small set of studies included were discovered focusing on interaction diagrams. For example, the 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, retrospectively.

Looking at the methodologies applied, three reviews conducted a traditional literature review [18, 19, 20, 27]. Three studies [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 also noteworthy that no quality assessment has been conducted.

We complement the existing literature reviews with an explicit focus on interaction-based diagrams (finding a total of 55 articles with a focus on interaction diagram) and hence expanding the findings by Shirole and Kumar [20]. Furthermore, we complement the existing work by analyzing the strength of evidence provided through the means of the quality assessment rubric proposed by Ivarsson and Gorschek [21]. The rubric describes rules by which reported studies may be assessed for their scientific rigor and practical relevance. The details of the rubric are described in the following section.

3 | RESEARCH METHOD

In this study, we have adopted the systematic mapping study (SMS) as our research method. Systematic mapping studies are meant to provide an overview of a research area. SMS classify and count the contribution about the categories of that classification. Like systematic literature review, it provides a systematic way to search the literature, to know what are the topics that have been reported in the literature and what are the publication venues for the area of research [23, 22].

We followed the guidelines provided in [23], in these guidelines the authors included some important aspects from the SLR guidelines proposed by Kitchenham and Charters [29].

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 oft 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 diagram, state machine, activity diaram)	LR	29
Mohi-Aldeen et al., [26]	Coverage criteria used to drive automated testing; methods used for test generation; benefits of approaches	SM	85

3.1 | Planning adn conduction the mapping

This section describes the decision making regarding the conduct of systematic mapping study.

| 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), extraction of merits and demerits (RQ2) and assessment of the quality of the studies with respect to rigor and relevance (RQ3).

TABLE 2 Research questions

ID		**Research question**
RQ1		What are various proposed MBT test case generation approaches based on UML interaction diagrams?
	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 relevance among the included primary studies?

| Developing and validating the study protocol

The development and validation of study protocol is critical. The developed protocol is critically reviewed by the fourth author, who is an experienced researcher. He has more than 80 research publications, mainly from the testing

domain. He has published 15 survey/ review papers. He has 28 years of industrial, teaching and research experience. He is serving as an associate professor at Faculty of Computing Capital University of Science and Technology (CUST), Islamabad, Pakistan. Modifications were suggested in research question, publication selection, inclusion/exclusion criteria and data extraction phase. After modifying this suggestion, the same expert again critically reviewed 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 term were identified in 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 [30, 31]. 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
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 to identify 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 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. To collect the required literature 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 was performed on the identified publications. If a publication's title is related to testing (test case generation) and UML models, then this publication is selected further for abstract screening. During abstract screening we determined whether the publication is relevant to model based test case generation and it is using interaction diagrams. A total of 55 primary studies were selected (See Table 5). Among these 55 studies, 25 studies were found directly from the publishing databases (i.e. 17 from IEEE, 3 from Elsevier, & 5 from ACM). While other 30 primary studies were found through Google Scholar. Out of these 30 studies, 10 were from Springer, 8 from Scopus indexed journals, and 12 studies were selected from other sources. The following inclusion criteria were applied:

- IC1: Publication must be related to model based testing.
- IC2: Studies that proposed model based testing by the means of interaction diagrams.
- IC3: Studies that describe the capability of model based testing with respect to interaction diagrams
- IC4: Surveys (e.g. questionnaires or interviews) performed on MBT test case generation techniques
- IC5: Select only those studies written in English language

The following studies have been excluded:

- EC1: Studies which do not use interaction diagram although they belong to MBT technique
- EC2: Studies that are duplicated (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(title screening)	Selected(abstract screening)
IEEE	250	65	17
Science Direct (Elsevier)	110	24	3
ACM	347	54	5
Google Scholar	656	107	30
Total	1363	240	55

To reduce the threats of biases and potential mistakes in the selection of studies the study selection was carried out by the first and second authors jointly, studies included or excluded by the first author were crosschecked by the second author, while studies included or excluded by first author were crosschecked 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 rubric 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 four criteria (subject, context, scale and research method). The criteria for rigor were scored on an ordinal scale as suggested by [21], namely strong, medium and weak description. The first author scored the studies, which were all reviewed by the third author. In case of corrections proposed, these were discussed and agreed upon

TABLE 5 List of Primary Studies

Reference	Study ID	Reference	Study ID
[32]	S1	[33]	S29
[34]	S2	[35]	S30
[36]	S3	[4]	S31
[37]	S4	[38]	S32
[39]	S5	[6]	S33
[7]	S6	[40]	S34
[41]	S7	[42]	S35
[43]	S8	[44]	S36
[45]	S9	[46]	S37
[14]	S10	[47]	S38
[48]	S12	[49]	S39
[50]	S13	[15]	S40
[51]	S14	[5]	S41
[17]	S15	[52]	S42
[53]	S16	[54]	S43
[55]	S17	[56]	S44
[57]	S18	[16]	S45
[58]	S19	[59]	S46
[60]	S20	[61]	S47
[12]	S21	[62]	S48
[2]	S22	[1]	S49
[63]	S23	[64]	S50
[13]	S24	[3]	S51
[65]	S25	[66]	S52
[57]	S26	[10]	S53
[67]	S27	[68]	S54
[11]	S28	[69]	S55

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 context. We emphasize the subjects type (academic, industrial), development methodology and experience of 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 these factors are completely ignored then C evaluates to 0.

Study design (D):

- *Strong description:* 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 C evaluates to 0.5.
- *Weak description:* 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:* In case of 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, scale, and using research methods that facilitate investigations in realistic environments.

Subjects (U):

- Contribution to relevance: Subjects or users are delegate of real user such as for industry professional, 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: The study is conducted in a real industrial setting, then C evaluates to 1.
- No contribution: The study is not conducted in a real industrial setting, then C evaluates to 0.

Scale (S):

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

| Data extraction and analysis

The selected studies are saved for the purpose of data extraction. The data extraction was performed jointly by authors. The second and third author randomly selected included primary studies and reviewed the results produced by the first author. The data extraction comprised of the following elements:

- *Publication data:* Publication ID, publication title, year of publication, type of publication, source of publication, author name, keywords
- *Test scope:* Type of testing, test objective
- *Input to tests:* UML model used, intermediate models constructed for test data generation
- *Data generation approach:* Description of test data generation process, algorithms used, tool support
- *Assessment:* Coverage criteria, research method, rigour and relevance scores
- *Outcome:* Merits and demerits, 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 models for test data generation). This shows which areas are covered by the literature, and is similar to the intent of a mapping study where an overview of an area is 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 rubric used in this study assesses 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 rubric favors 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 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 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:

- Missing relevant studies: Our search focused on interaction diagram types 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 much larger set of studies on the topic when compared to previous 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 very specific, there is a risk of still missing studies from the population
- Biases in study selection, data extraction and analysis: The selection, data extraction and analysis activities are prone to biases and mistakes. Thus, multiple researchers have been involved and cross-checking of the studies has been done among the authors of the paper. 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 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: What are various proposed MBT test case generation approaches based on UML interaction diagrams?

RQ1 was divided into two sub-questions (RQ1.1 & RQ1.2). The purpose of the first sub-question was to see the trends of test case generation techniques which are using interaction diagrams as input. Whereas the second sub-question was designed to classify these test case generation techniques by different factors, for instance, by the input model, intermediate model, and the outcome of the techniques.

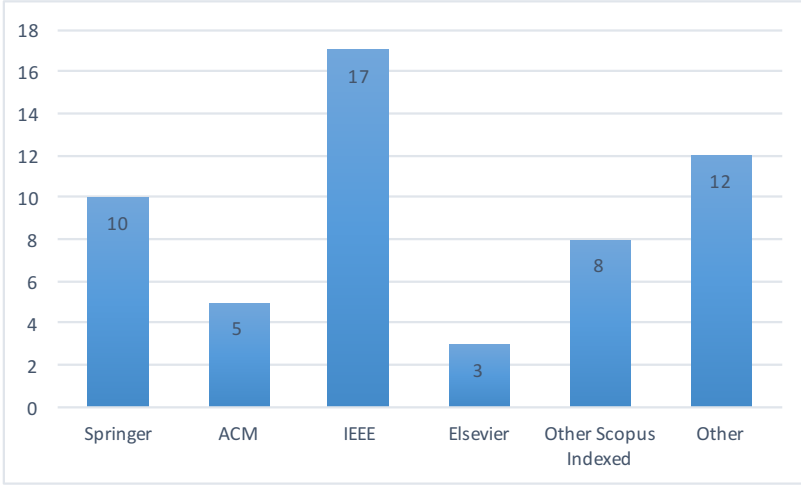
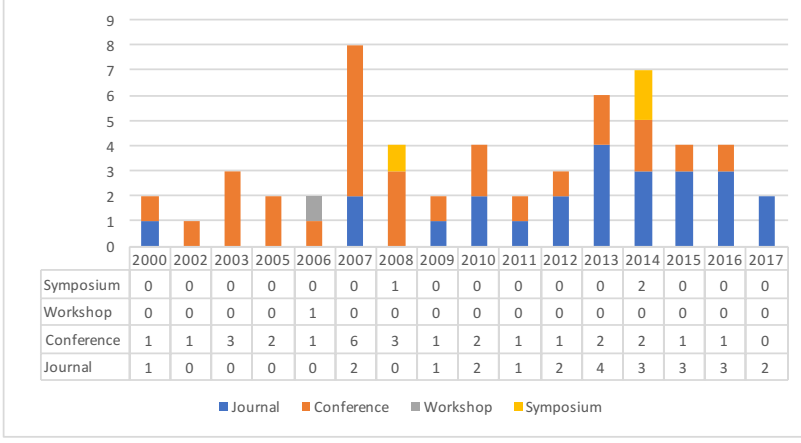


FIGURE 1 Publishing Venues of Primary Studies



	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	1	0	0	0	0	2	0	1	2	1	2	4	3	3	3	2

FIGURE 2 Year and Venue Wise Distribution of Primary Studies

| 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 (65%) included in this study are from the four publishing venues (IEEE, Springer, ACM, and Elsevier). Figure 1 presents the distribution of primary studies with respect to publishing venues. An increasing trend in the number of publications is seen with peaks in 2007 and 2014 (Figure 2). The publication trend shows that the research area is growing gradually. A complete list of primary studies and the corresponding publication venues is presented in Table 13 (See Appendix B). It is apparent that only a few well known journals in the field of software engineering have been published in (e.g. Information and Software Technology and ACM SIGSOFT Software Engineering Notes). In addition, many different publication forums have been targeted. Hence, no clear recommendation could be given where to start to look for papers on the topic.

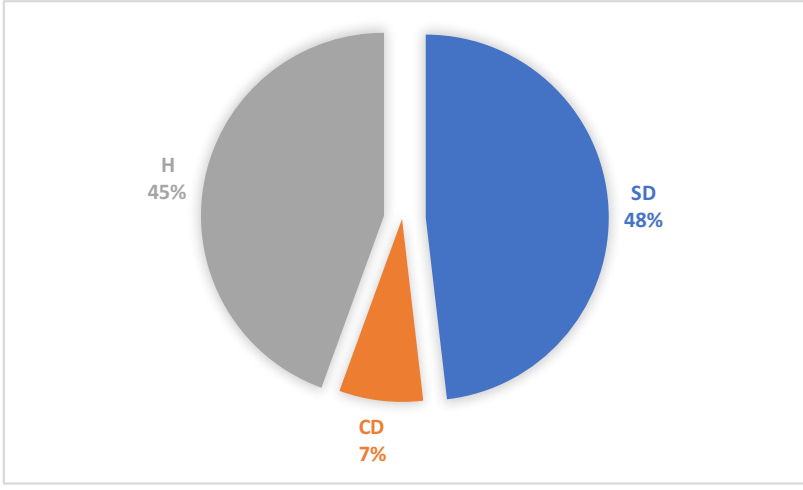


FIGURE 3 Distribution of studies according to input model used

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

The selected UML interaction diagram based testing techniques are reviewed regarding their similarities and differences. The majority of the techniques reported in the literature use UML sequence diagram as input for test case generation and are marked as category *"SD"*, in the selected primary studies 48% techniques are based on sequence diagram. UML collaboration diagram based approaches are also identified and marked as category *"CD"*, only 7% of the selected primary studies belong to this category. 45% of the selected studies are using interaction diagrams along with some other UML model as input for test case generation, we refer these techniques as hybrid solutions and mark as category *"H"*. Figure 3 shows the distribution of the primary studies according to the type of test case generation techniques.

Regarding the testing types, 84% of the SD-based studies are proposed for the integration testing, while 8% for the

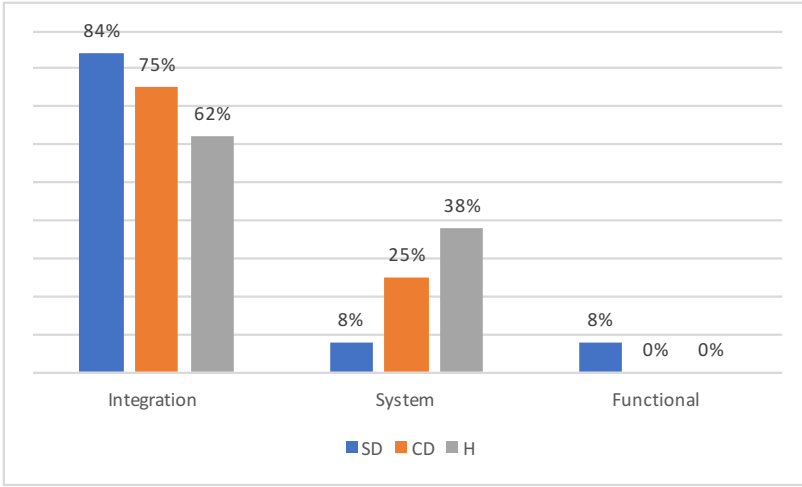


FIGURE 4 Distribution of studies according to testing types covered

functional, and 8% studies are proposed for the system testing. Similarly, 75% of the CD-based studies are proposed for integration testing, and 25% for system testing. Finally, 62% of the hybrid solution based studies are proposed for integration testing, and 38% of these solution work for system testing. Figure 4 presents the classification of proposed techniques according to the testing types covered. A complete overview of the categories of techniques with regard to their testing types and intermediate models generated is presented in Table 6. The models form the basis to generate the test paths. Some techniques only determine the test paths that should be covered. However, in order to execute the paths the test data still needs to be generated (e.g. in the case of a decision node, the two input values for it to either evaluate to true or false 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.

TABLE 6 Overview of Solutions

Category	Testing type	Intermediate model	References
SD	Integration	Graph	[12, 14, 17, 34, 45, 55, 65, 50, 63, 60, 53, 57, 36]
		XMI	[11, 67]
		Dependency table	[51, 48]
		Tree	[2]
		SLT	[32]
		ETDA	[37]
	System	Graph	[58]
		Activity	[39]
	Functional	Graph	[43, 41]
	Conformance	Graph	[17]
CD	Integration	Graph	[33, 35]
		Tree	[4]
	System	Graph	[38]
H	Integration	Graph	[1, 3, 6, 62, 47, 59, 56]
		XMI	[42, 66, 61, 52]
		Tree	[46],
		Petal files	[54]
		SCOTEM	[40]
	System	Graph	[5, 15, 16, 44, 64, 49, 70]
		Activity Diagram	[69, 68]

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 are using the sequence diagram as an input model, Figure 5 presents an overview of these. The activity diagram shows the different possibilities described in the literature. The references in the diagram show how the techniques in the papers are structured. The first step towards the test case generation is that sequence diagram is transformed into intermediate forms, which are then used as a basis for generating the tests.

Common intermediate forms are LTS (Labelled Transition Systems) [41], XML Metadata Interchange (XMI) [11, 67], sequence dependency table [51, 48] and graphs [2, 7, 12, 34, 45, 55, 65, 50, 63, 60, 53, 57, 36, 51, 58, 43]. Two studies [14, 39] transform the sequence diagram into activity diagram. One study [32] generates the stimulus link table (SLT) from the sequence diagram. The techniques presented in [17, 13, 57] do not use an intermediate form. While the technique presented in [37] make use of event deterministic finite automata (ETDA).

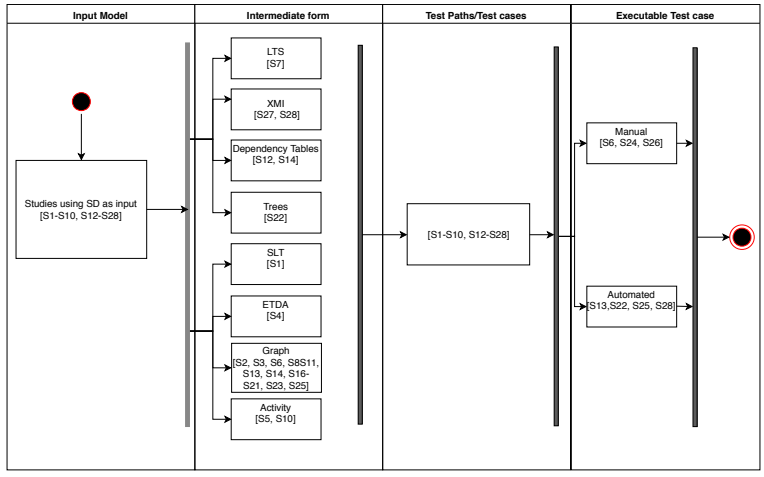


FIGURE 5 Solutions using sequence diagram (SD) as input

SD based Solutions Using Graph and its variants as intermediate form: Graphs are the most commonly used intermediate form, 65% of the SD based solutions use the graph as intermediate form, though the type of graph differs, as is shown in Figure 5.

Bao-Lin et al. [2] converted the sequence diagram into a tree representation. The tree is traversed to select the conditional predicates. Bao-Lin et al. use of OCL to describe the pre- and post-conditions. Finally, function minimization is used to generate test data from conditional predicates.

A wait-for graph was used by Patnaik et al. as well as Fraikin and Leonhardt [53, 11] to identify deadlock points. The primary objective of their techniques is to identify concurrency issues. Both techniques [53, 11] only generate test paths.

Hoseini and Jalili [63] generated a control flow graph and thereafter used a prime path extraction algorithm to generate the test paths. A message control flow graph is used as an intermediate form by Jena et al. as well as Lei

and Lin [7, 60]. Thereafter the graph is traversed using the depth first search algorithm. Lei and Lin [60] 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 [63, 7] use a genetic algorithm to calculate the fitness value for generated test paths to optimize them. A sequence graph was used by Samulatah et al. [12] as an intermediate form. The sequence graph was traversed using genetic algorithm to generate the test paths. A fitness function was defined calculating the fitness value of the generated test paths. Chandnani et al. [34] presented a model based test case generation technique to test the interaction of objects. A sequence dependency table form the sequence diagrams is created and then a control flow graph is generated. The intermediate graph is traversed into depth first manner to obtain test paths. Finally test cases are generated using particle a swarm optimization algorithm (PSO). Rhmann and Saxena [36] 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 model. Test scenarios are generated by using concurrent queue search (CQS). Test scenarios generated here are useful for detecting data safety errors. Seo et al. [39] presented a test case generation technique based on sequence diagrams. The sequence diagram is transformed into an activity diagram, which represents the scenario of sequence diagram. In the next step, the activity diagram was fragmented into sub activities. UML testing-profile based test case were designed, on the basis of the principal that each individual activity represents a test scenario. A concurrent composite graph was used by Khandai et al. [55] who generated test paths by traversing the graph with the help of a message sequence graph algorithm. Khandai et al. also make use of the Object Constraint Language (OCL). Both techniques [55, 14] are meant to detect concurrency errors.

A sequence dependency graph was used as an intermediate form in multiple studies [43, 50, 51, 58, 65] . Dhineshkumar et al [43] 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 [50] also use the depth first search algorithm to traverse the message sequence graph and predicate are selected. Using the selected predicates the source code is produced and then test cases are generated. The sequence dependency graph is traversed by the algorithm and test sequences are generated in [58]. Sequence dependency graph was also used as intermediate form by Samuel et al. [65]. They use the concept of dynamic slicing to generate test cases automatically. Priya and Malarchelvi [51] generate a sequence dependency table from the sequence diagram, which is further transformed into a sequence dependency graph. The depth first search algorithm is used to generate test paths from the sequence dependency graph. A sequence dependency table is also used by Shanti and Khumar [48] who use a genetic algorithm to generate test paths from the dependency table.

SD based Solutions Using other intermediate forms: XMI is used as intermediate form by Beyer et al. and Frain and Leonhardt [67, 11]. A test case generation tool called "MaTeLo" has been proposed by Beyer et al. [67]. The tool makes use of Markov chains as an intermediate model. 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. A test tool for object oriented applications has been presented by Frain and Leonhardt [11]. They transform a Sequence diagram into XMI format. Their "SeDiteC" tool extracts the necessary information from XMI. The tool enables early testing and provide facilities such as automatic test stub generation and initialization of objects. Cartaxo et al. [41] introduced the functional (feature) test case generation method.In this method sequence diagrams are converted into a labeled transition system (LTS). In a LTS transitions are represented as states. Loops and alternative flows are also modeled in LTS. 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. The technique presented in [13] does not use any intermediate form, instead it makes use of a genetic algorithm to generate valid and invalid test paths and related test cases. To test the web applications a method has been suggested by Cho et al. [57]. Test cases are generated for single, mutual and integrated web pages. For single web pages, test cases are generated from self-call message in the sequence model. For mutual 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. Moreover, a tool has been proposed, though it is not available online. Lund and Stolen [17] proposed an algorithm to derive test cases from UML sequence diagram. The sequence diagram is specified using neg (unary operator to specify negative behavior) and assert (operator used to specify universal behavioral) operators. Lund and Stolen propose the STAIRS semantic, which is used for conformance testing. Zhang et al. [37] presented a test case generation approach based on sequence diagrams and event deterministic finite atuomata (ETDA). The objects participating in the sequence diagram are transformed into ETDA. Propositional projection temporal logic (PPTL) was used for model checking and the ETDA is verified. Finally, they get the composite automata by synthesis rules and generated the test cases. Mani and Prasanna [32] 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.

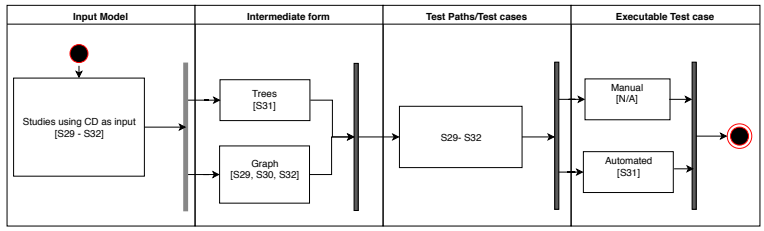


FIGURE 6 Solutions using collaboration diagram (CD) as input

| Solutions Using CD

Test case generation techniques using collaboration diagrams as input models are presented in [4, 33, 35, 38] (see Figure 6). All the techniques presented in [4, 33, 35] use different variants of graphs as an intermediate form. Trees are used by Samuel et al. [4] while a weighted graph is used by Ahmed et al. and Prasanna et al. [33, 35]. The weighted graph is then traversed using Prims and Dijkstra algorithms. An advantage of [35] over [33] is that mutation analysis is performed to evaluate the effectiveness of the proposed technique. Samuel et al. [4] convert a collaboration diagram into tree format. The predicates are selected from the tree structure in an iterative manner. At the end, a function minimization technique is applied to generate test data for the selected predicates. Abdurazik and Offutt [38] 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 paths 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 [38, 33, 35] only generate test paths, the only technique which generates test data and test cases is proposed by Samuel et al. [4].

| Hybrid Solutions

In this study, we found 24 techniques which are presenting hybrid solutions (combining interaction diagram with some other UML model) for integration testing [40, 52, 59, 56, 69, 6, 62, 70, 15, 46, 66, 61, 16, 44, 64, 10, 5, 1, 49, 54, 3, 68, 47, 42]. An overview of the solutions is shown in Figure 7. Hybrid solution refer to techniques utilizing two or more input models. Interaction diagrams (sequence or collaboration) are the main input model along with another UML model type. The models which are being used as input 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 there are 19 techniques [52, 59, 69, 6, 62, 15, 46, 66, 61, 16, 44, 64, 10, 5, 1, 49, 54, 47] which are using sequence diagram along with some other UML diagram as input model. Four solutions [40, 56, 3, 68] 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.).

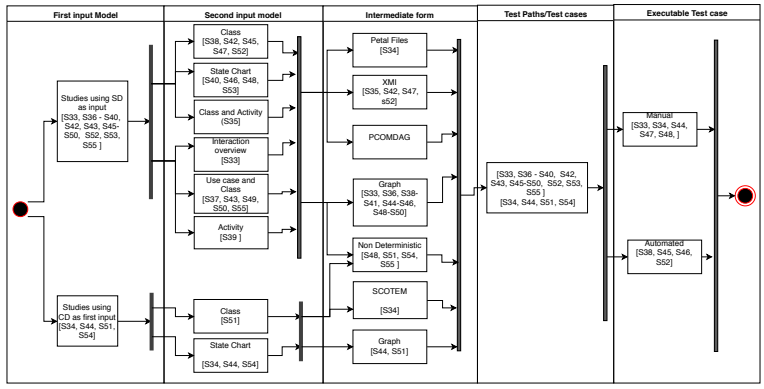


FIGURE 7 Solutions using interaction diagram and some other UML diagram (H) as input

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 used are XMI [52, 42, 66, 10], Petal files [54], "state collaboration test model" (SCOTEM) [40], "polymorphisms class object method acyclic graph" (PECOMDAG) [47], "system testing graph" [15, 64, 49], sequence diagram graph [44], activity-sequence graph [5], sequence interaction graph [6], test model grap [56], concurrent control flow [59], Extended Variable Assignment Graph (EVAG) [59], structured composite graph [16], and tree [46].

Twenty-two out of twenty-four techniques generate test paths. One technique presented by Wu et al. [68] does not generate test paths. Instead it derives a test adequacy criteria based on input diagrams, that is collaboration and state chart diagrams. Only a subset of techniques used algorithms for the generation of test paths:

- Depth first search algorithm (DFS) [69, 6, 7]
- Breadth first search algorithm (BFS) [16, 44, 5]
- Traversing algorithms [52, 64, 47]
- Genetic Algorithm (GA) [15]
- Parsers [61, 54]

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 [40, 59, 56, 6, 62, 66, 61, 16, 47] provide solutions for generating test data. Four techniques [59, 66, 16, 47] generate test data automatically while five techniques [40, 56, 6, 62, 61] 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?

To respond RQ2 the capabilities and limitations of selected approaches are highlighted. Table 7 provides an overview of the capabilities. A subset of approaches (39%) supports the generation of test paths. Only very few studies (6%) also support the automated execution in connection to the generation.

The types of faults possible to detect are interaction, scenario and operational faults. Only for SD the ability of detecting 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 or automatically.

An important feature of the approaches proposed in the literature is the degree to which they were evaluated. 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, 26 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, 43% of the selected studies are using this criterion. Among the other criteria, condition coverage was used by 13%, message coverage was used by 19%, and model coverage was used by 7% of the selected primary studies. While 18% of the studies are either using other criteria or they did not mention any.

TABLE 7 Capabilities in relation to MBT approaches

Capabilities		SD	CD	H
Testing	Static, dynamic testing		[38]	[42]
	Test paths (scenario) generation	[7, 55, 63, 34, 36, 39, 37]	[35]	[70, 15, 16, 5, 54, 1, 69, 64, 52, 49, 56, 40, 3]
	Test case execution			[61, 56, 40]
Detection of faults	Interaction, scenario, operational	[7, 43, 51, 48, 41, 50, 12, 60, 13, 34, 36]	[35, 4]	[6, 15, 44, 66, 1, 64, 61, 52, 40, 70]
	Concurrency (e.g. deadlock)	[45, 55, 53, 14]		
Test input generation	Manual	[7, 60, 13]		[6, 59, 61, 56, 40]
	Automated (Efficient)	[65, 2, 50, 11]	[4]	[16], [66], [1]
Stated method	Experiment	[2, 48, 63, 13, 14]	[35]	[44]
	Case Study	[7, 51, 48, 41, 50, 12, 34, 36]	[33]	[6, 15, 5, 54, 1, 47, 59, 61, 52, 40, 3, 70]
	Proof of Concept	[45, 55, 65, 43, 58, 11, 60, 53, 32, 39]	[4, 38]	[42, 46, 16, 66, 69, 64, 49, 56]
Coverage criteria	Path	[43, 47, 48, 41, 63, 12, 60, 34, 36, 39, 37]	[33, 4]	[6, 15, 44, 5, 69, 49, 56, 40, 70]
	Predicates/ conditions	[65, 50]	[33]	[46, 16, 1, 47]
	Messages	[7, 13, 14]	[38]	[66, 64, 10, 61, 52, 68]
	Transitions	[58]		
	Model	[11]		[42, 54, 3]

The limitations are summarized in Table 8. The lack of tool support, test data generation, and test case execution are commonly highlighted. Furthermore, high complexity has been stressed as an issue. Every technique has to follow some specified step to generate test cases. A technique has high complexity if it is manual, semi-automated (with a

number of different steps) and no tool support.

TABLE 8 Limitations

Limitations	SD	CD	H
No tool support	[7, 45, 55, 43, 51, 48, 63, 58, 12, 17, 53, 13, 14, 34, 36, 39, 37, 32]	[33, 4, 38]	[6, 46, 15, 44, 16, 5, 69, 64, 10, 56, 3, 68, 70]
No test data generation	[45, 55, 43, 51, 48, 41, 63, 58, 12, 53, 14]	[33, 35, 38]	[42, 46, 15, 44, 5, 54, 47, 64, 10, 52, 49, 3, 68]
High complexity	[7, 45, 55, 65, 48, 50, 63, 58, 17, 53, 34, 36, 37]	[33, 35]	[46, 15, 16, 69, 64, 49, 56]
No test case execution	[7, 45, 55, 43, 51, 48, 63, 58, 12, 17, 53, 13, 14, 34, 36, 39]	[33, 4, 38]	[6, 42, 46, 15, 5, 54, 1, 69, 64, 10, 52, 49, 70]
More test case generation	[55, 51]		[47]
Issues in transforming model into intermediate form	[7, 43, 41, 63, 58, 53]		[46, 44, 16]
Restriction in proposed methods	[11]	[4, 38]	[47, 59]

4.3 | RQ3: How strong the evidence with respect to rigor and relevance to support the outcome of various test case generation techniques?

The main objective of this section is to provide the score of individual selected publication according to rigor and relevance. The complete results of each publication containing an empirical study (i.e. case study or experiment) with respect to rigor and relevance are summarized in Tables 9 to 11. The tables show the studies that either stated that a case study or experiment, i.e. an empirical study has been conducted. It is evident that a high 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 [71], who highlights that *"case study is often a misused research method. In many papers, the authors actually mean example or illustration and not a case study in the way it is defined"*. In particular, Wohlin highlights the references by Yin and Runeson and Höst [72] as a guide to follow. Similarly, 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 [73], Wohlin et al. [74] and Juristo [75] 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 correctly label the research method used. In many cases, the correct label (example or proof of concept) was used. In Table 11 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 9 we can see the presence of rigor score for the years 2016 and 2017. So we can infer two facts, 1) quality of the recent studies regarding rigor score is slightly better, 2) hybrid solutions have better relevance score. These facts indicate that from an industry perspective hybrid solution (i.e., test case generation techniques using a combination of interaction diagrams along with some other model) are more relevant. However, regarding the overall results, it is visible that no clear trend of scores with respect to years can be seen in Tables 9 to 11 as we found papers with high and low scores for both new and old papers (see Figure 8).

TABLE 9 Rigor and relevance scores: Sequence Diagrams

Ref	Rigor				Relevance				Year
	C	D	V	C+D+V	U	C	S	U+C+S	
[32]	0.5	0	0	0.5	0	0	0	0	2017
[34]	0.5	0	0	0.5	0	0	0	0	2017
[36]	0.5	0.5	0	1	0	0	0	0	2016
[37]	0.5	0.5	0	1	0	0	0	0	2016
[39]	0.5	0	0	0.5	0	0	0	0	2016
[7]	0	0	0	0	0	0	0	0	2015
[41]	0	0	0	0	0	0.5	0	0.5	2014
[43]	0	0	0	0	0	0	0	0	2014
[45]	0	0.5	0	0.5	0	0	0	0	2014
[14]	0	0	0	0	0	0	0	0	2013
[48]	0	0	0	0	0	0	0	0	2013
[50]	0	0	0	0	0	0	0	0	2013
[51]	0	0	0	0	0	0	0	0	2012
[17]	0	0	0	0	0	0	0	0	2012
[53]	0	0	0	0	0	0	0	0	2011
[55]	0.5	0.5	0	1	0	0	0	0	2011
[57]	1	1	0	2	0.5	0	0.5	1	2010
[58]	0.5	0.5	0	1	0	0	0	0	2008
[60]	0	0	0	0	0	0	0	0	2008
[12]	0	0	0	0	0	0	0	0	2007
[2]	0	0	0	2	0	0	0	0	2007
[63]	0	0	0	0	0	0	0	0	2006
[13]	1	1	0.5	2.5	0.5	0	0	0.5	2005
[65]	0	0	0	0	0	0	0	0	2005
[11]	1	0.5	0	1.5	0	0	0	0	2002

Rigor– C: Context, D: Study Design, V: Validity threats

Relevance– U: User/Subjects, C: Context (industry), S: Scale

| RQ3.1: What is the extent of relevance 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 26 studies are referred by the other primary studies. Generally, studies are said to be related with each other if authors are referring the other studies while defining their methodology. We found only two studies ([47] is citing [11], and [33] is citing [35]) in their methodology directly and explicitly drawing ideas from them. Other studies are referring 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 already existing studies ([52] compared with [10] and [1] compared with [16]). A complete mapping of the references of the primary studies with each other are presented in Table 12.

TABLE 10 Rigor and relevance scores: Collaboration Diagrams

Ref	Rigor				Relevance				Year
	C	D	V	C+D+V	U	C	S	U+C+S	
[33]	0	0	0	0	0	0	0	0	2013
[35]	0.5	1	0	1.5	0.5	0	0	0.5	2011
[4]	1	1	0	2	0	0	0	0	2006
[38]	0.5	0	0	0.5	0	0	0	0	2000

Rigor– C: Context, D: Study Design, V: Validity threats

Relevance– U: User/Subjects, C: Context (industry), S: Scale

TABLE 11 Rigor and relevance scores: Hybrid

Ref	Rigor				Relevance				Year
	C	D	V	C+D+V	U	C	S	U+C+S	
[70]	0	1	0.5	1.5	0	0	0	0	2016
[6]	1	1	0	2	0.5	0.5	0	2	2015
[40]	1	1	0.5	2.5	0.5	0	1	1.5	2015
[42]	0	0	0	0	0	0	0	0	2015
[44]	0	0	0	0	0	0	0	0	2014
[46]	0	0	0	0	0	0	0	0	2014
[47]	0.5	0	0	0.5	0	0	0.5	0.5	2014
[49]	0	0.5	0	0.5	0	0	0	0	2013
[15]	0	0	0	0	0	0	0	0	2012
[5]	0	0	0	0	0	0	0	0	2010
[52]	1	1	0.5	2.5	0.5	0	0	0.5	2010
[54]	0	0	0	0	0	0	0	0	2009
[56]	0.5	0.5	0	1	0	0	0	0	2009
[16]	05	1	0	1.5	0	0	0	0	2009
[59]	0.5	0.5	0	1	0	0	0	0	2008
[61]	0	0	0	0	0	0	0	0	2008
[62]	1	1	1	3	0.5	0	0	0.5	2007
[1]	0	0.5	0	0.5	0	0	0	0	2007
[64]	0	0.5	0	0.5	0	0	0	0	2007
[3]	0	0	0	0	0	0	0	0	2007
[66]	0.5	0	0	0.5	0	0	0	0	2007
[10]	0	0	0	0	0	0	0	0	2006
[69]	0.5	1	0	1.5	0	0	0	0	2001

Rigor– C: Context, D: Study Design, V: Validity threats

Relevance– U: User/Subjects, C: Context (industry), S: Scale

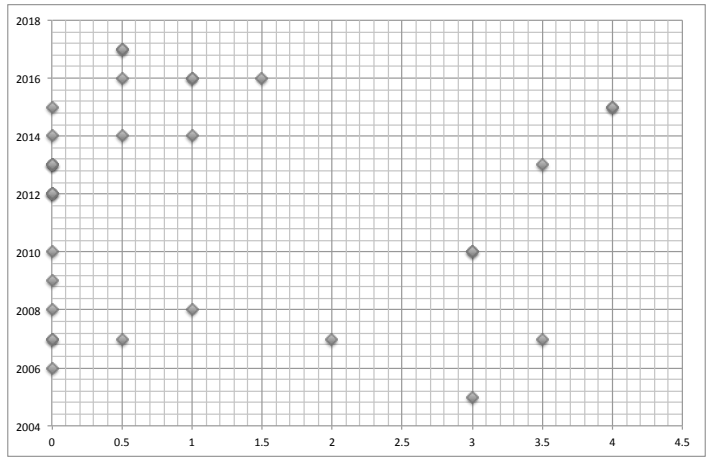


FIGURE 8 Sum of Rigor and Relevance Scores (X-Axis) and Year Published (Y-Axis)

5 | DISCUSSION

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

TABLE 12 Primary Studies referring to each other

Referred to	Referring Primary Studies			
Primary Study	Introduction	Related Work	Methodology	Comparison
[38]	[33, 56, 6, 2]	[40, 59, 43, 50, 4, 44, 12]		
[40]	[56, 7],	[35]		
[52]		[63, 55, 12, 1]		
[59]	[51, 13]			
[67]		[41, 58]		
[69]	[10]	[40, 59, 43, 1]		
[41]	[35, 1]	[55, 13, 14]		
[11]	[50]	[40, 41, 17, 58, 4, 44, 15, 13, 12, 1, 60, 47]	[47]	
[9]	[40]	[66, 58, 4]		
[62]	[51, 13]			
[55]	[63]	[14]		
[2]		[55, 35, 12, 54]		
[17]		[58]		
[45]		[42]		
[16]				[1]
[50]	[63]	[54]		
[35]	[51]		[33]	
[51]	[42]			
[58]		[63, 55]		
[4]		[60]		
[64]		[33, 6, 55, 35, 1]		
[48]	[33, 51]	[54]		
[10]		[52, 51, 58]		[52]
[5]	[51]			
[1]		[43, 6, 55, 35]		
[49]		[15]		

5.1 | Analysis of research questions

With regard to **RQ1** (*What are various proposed MBT test case generation approaches based on UML interaction diagrams?*), we were interested to see to 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 the good publication venues. On the other hand regarding conferences, we have found the majority of papers are from the conferences who have published their proceedings in IEEE, ACM, and Springer. One reason for not considering the good publication venues could be that the studies are not conducted rigorously, results of RQ3 support this analogy.

Regarding the classification of the techniques, we found that 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 focus of the approaches. The main objective for which techniques have been proposed includes, detection of faults such as interaction, operational, scenario [7, 65, 43, 51, 48] to address concurrency issues such as deadlock detection and synchronization [45, 55, 53, 14], and test paths generation with high coverage [50, 63, 58]. To generate a Markov chain model [67], the testing of web pages [57], to the test object oriented application [65, 11] were also achieved through these approaches. Commonly used traversing algorithms are depth first search [7, 55, 43, 51, 41, 50, 60] heuristic algorithm used in [63, 12, 63, 12, 48], breadth first search [55] and other algorithms that are proposed [45, 65, 58, 17, 53, 14].

During Review of literature, it has been noted that a few techniques (only 7%) have been identified in which input model is a UML collaboration model. The main objective of these techniques is to increase reliability and detection of faults (interaction). The intermediate representations 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 weighted graph.

The hybrid techniques for test case generation are also proposed, 45% of the selected primary studies belong to this category. The UML interaction models (sequence, collaboration) are used with combination of other diagram (Class, Activity, State chart and use case). The main objectives of these proposed technique is to detect interaction, scenario and operational faults [15, 44, 64], test cases generation from analysis artefacts [66, 1]. These inputs models are transformed into intermediate representations. The most common intermediate representation includes sequence interaction graph [7], structured composite graph [16], system testing graphs [64], activity sequence graph [5] and system graph [49],. This intermediate representation has been traversed to formulate the test paths. These test paths are then leads towards test cases generation.

With regard to **RQ2** (*"What capabilities and limitations are observed in relation to the approaches?"*) the most common merits that are achieved in the proposed techniques includes detection of specific fault types, test input generation (few technique generates automated test input), and coverage criteria achieved (paths, messages, transition). The most commonly highlighted drawback was 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 hinders for a transfer of research results to industry practice [76]. 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 [71] 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 [73, 75, 74] and case studies [77, 72] should be highlighted. In particular, empirical comparisons between different approaches presented here are of interest. For future work we propose to compare studies within the categories shown in Table 6. 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 encouraged for future work on MBT with collaboration diagrams.

Furthermore, our results reveal that the proposed techniques are not related to each other. We define the relatedness as if the authors are referring to the other techniques in their methodology or they are comparing their technique(s) with the existing techniques. We found only four studies that are referred by the other authors in this regard. The least relatedness is considered if the authors are referring to other techniques in their related work section. In this regard 14 studies are there that are referred by the other authors in the related work of their studies. This shows that test case generation using interaction diagrams is not a well researched and mature area.

6 | CONCLUSION

This paper presents a mapping of model based testing techniques based on UML interaction diagrams. It has been explored that UML interaction diagrams are being used mostly for integration testing. It is also discovered that interaction diagrams are being used as standalone model to generate test cases, at the same time there are some techniques which are combining interaction models with some other UML model (class, state chart, etc.). The proposed

solutions have various capabilities claimed in the papers. Approaches from all investigated categories (SD, CD and H) are capable of generating test paths, though the needed inputs (test data) are only generated by a few approaches. 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 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 models) 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 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 ans 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 are used such as the researcher do not set at specific criteria to search only title, key words and abstract etc.
- *Science Direct:* The advance search option is 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 advance search option is used to identify the related publications from ACM digital library. The filter is also applied during executing the search string and only title is selected to match with search sting. Moreover, we search the publications from "ACM full text selection".
- *Google scholar:* The advance search option is used to identify the related 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 thereafter "test case generation" were searched with exact phrase "and model based testing" with at least one word in the phrase.

B | PUBLICATION VENUES

TABLE 13 Publication types (J = Journal, C = Conference, S = Symposium, W = Workshop) and publication venues of the primary studies

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

[31] 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.

[32] 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.

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

[34] 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.

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

[36] 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).

[37] 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.

[38] 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.

[39] 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.

[40] 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.

[41] 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.

[42] 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.

[43] 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.

[44] 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.

[45] 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.

[46] 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. .

[47] 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.

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

[49] 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.

[50] 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.

[51] 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).

[52] 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.

[53] 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.

[54] 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.

[55] 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.

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

[57] 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.

[58] 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.

[59] 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.

[60] Y L, N L. Test Case Generation Based on Sequence Diagrams. In: International Computer Symposium, ICS2008; 2008. p. 349–355.

[61] 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.

[62] 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.

[63] 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.

[64] 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.

[65] 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.

[66] 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.

[67] 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.

[68] 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.

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

[70] 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.

[71] 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.

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

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

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

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

[76] 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.

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