Testing on Dynamically Adaptive Systems: Challenges and Trends

Isabely do Nascimento Costa[1][a], Ismayle S. Santos[2][b] and Rossana M. C. Andrade[1][c]
[1]*Federal University of Ceará (UFC), Fortaleza, Brazil*
[2]*Ceará State University (UECE), Fortaleza, Brazil*

Keywords: Adaptive Systems, Software Testing, Systematic Review.

Abstract: Dynamically Adaptive Systems (DAS) are systems capable of modifying themselves automatically according to the surrounding environment. Traditional testing approaches are ineffective for these systems due to their dynamic aspects, making fault detection complex. Although various testing approaches have been proposed for DASs, there is no up-to-date overview of the approaches, challenges, and trends. This research therefore presents the results of a systematic literature review to identify the challenges, approaches and trends in testing dynamically adaptable systems. For this objective, 25 articles between 2020 and 2023 were analyzed to answer our research questions. As a result, approaches and their characteristics were identified, such as what type of system they can be applied to, what activity is included in the testing process, and at what level of testing. We also highlighted challenges that are still being faced and trends in testing dynamically adaptive systems. For a more in-depth analysis of the results related to the challenges, grounded theory procedures were applied to organize them and encourage future research that seeks to overcome and mitigate them.

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS

1 INTRODUCTION

Modern information systems are becoming increasingly complex due to the growing use of mobile devices and, consequently, the need for them to work uninterruptedly in any environment. The software industry has had to adapt to these demands by using highly distributed systems to meet this need. These systems must integrate all available, highly specialized, and heterogeneous devices and data flows operating in a constantly changing environment with fluctuating network availability and resources. Developing, configuring, and maintaining these systems is challenging, error-prone, and costly. One solution to this problem is self-adaptation, and a dynamically adaptive system (DAS) is capable of automatically modifying itself in response to changes in its operating environment (Krupitzer et al., 2015).

However, these dynamic adaptations of systems, while they are already in operation, can lead to unsafe changes at runtime and can then lead to new risks of bugs, unexpected interactions, performance degradation, and unwanted modes of operation (Lahami and Krichen, 2021). In the context of DASs, traditional testing approaches are ineffective due to the inherent characteristics of these systems, and therefore, detecting faults effectively is not a trivial task (Siqueira et al., 2016).

Based on this challenge, various approaches have already been proposed to solve the most diverse challenges listed in the literature, as we can find in the following works (de Sousa Santos et al., 2017), (Matalonga et al., 2022), (Priya and Rajalakshmi, 2022) and (Lahami and Krichen, 2021). However, there was a lack of studies presenting an up-to-date overview of the challenges of testing DASs and their approaches. To contribute to future research related to testing in DASs that seeks to solve these challenges, this research proposes a systematic literature review with the application of *Grounded theory* procedures. To this end, two research questions were defined, and 25 articles from the last three years (2020, 2021, 2022, and early 2023) were analyzed.

The rest of this paper is organized into five sections: Section 2 describes the methodology adopted in this research; Section 3 presents the results obtained by the systematic review; Section 4 analyzes the results obtained and presents the threats to validity; Section 5 presents the related work; and finally, Section 6 summarizes the results and indicates future work.

[a] https://orcid.org/0009-0008-5879-7469
[b] https://orcid.org/0000-0001-5580-643X
[c] https://orcid.org/0000-0002-0186-2994

Costa, I., Santos, I. and Andrade, R.
Testing on Dynamically Adaptive Systems: Challenges and Trends.
DOI: 10.5220/0012555900003690
Paper published under CC license (CC BY-NC-ND 4.0)
In *Proceedings of the 26th International Conference on Enterprise Information Systems (ICEIS 2024) - Volume 2*, pages 129-140
ISBN: 978-989-758-692-7; ISSN: 2184-4992
Proceedings Copyright © 2024 by SCITEPRESS – Science and Technology Publications, Lda.

2 METHODOLOGY

2.1 Background

This paper aims to present the challenges of testing adaptive systems and identify current testing approaches and possible ways of testing DAS through a systematic literature review. To this purpose, the guideline of (Kitchenham et al., 2016) for Systematic Literature Review (SLR) was followed. In addition, procedures from *Grounded Theory* (GT) (Strauss and Corbin, 1990) were applied to part of the data obtained from the synthesis stage of the SLR.

2.2 Research Questions

The research questions were initially defined based on the objective of this study and are as follows:

- **RQ1) What Are the Characteristics of Current Approaches to Testing DAS?** *This research question aims to present testing approaches for DASs by categorizing the type of test the approach performs, the level of testing and type of activity the approach is inserted in, the type and domain of the system under test the approach is applied to, and when the approach is applied.*
- **RQ2) What Are the Current Challenges Related to Testing DAS?** *This research question presents the challenges of testing DASs by categorizing them using Grounded Theory procedures.*

2.2.1 Extraction Questions

The following extraction questions were defined to obtain the information to answer the research questions. The correlation with the research questions can be visualized through the IDs, where questions RQ1.1 to RQ1.3 support the search for the answer to question RQ1, and RQ2.1 supports the search for the answer to question RQ2.

- RQ1.1) What are the approaches to testing adaptive systems?
- RQ1.2) What are the System Under Test (SUT) types of the approach?
- RQ1.3) Are the testing approaches applied at runtime or design time?
- RQ2.1) What are the challenges of DAS testing?

2.3 Search Strategy

The strategy used to search for articles was an automated search. A search string defined by (Siqueira et al., 2021) was used. After all, it was the most up-to-date systematic review (with papers up to 2019), which had the characteristic of having a comprehensive string because it used more generic keywords. The guidelines of (Kitchenham et al., 2016) were followed to validate the search strategy, starting with identifying relevant electronic resources. Then, some articles found by the string in the IEEE database were analyzed, and based on the inclusion and exclusion criteria, it was confirmed that the string was providing satisfactory results. Finally, an automated search was performed.

The motivation for a new systematic review rather than an update of the review by (Siqueira et al., 2021) was based on the 3PDF checklist (Mendes et al., 2020) for defining when to update a systematic review. (Mendes et al., 2020) define an update of a systematic review only when the same methodology as the previous review is followed. Neither is it possible to compare the results of reviews that followed different protocols. From this definition, differences were noted compared to this review and the (Siqueira et al., 2021), such as different databases, this work did not use snowballing, how the systems were categorized, different inclusion and exclusion criteria, the previous review did not use a checklist to assess the quality of the studies or grounded theory procedures to analyze the results.

2.3.1 Databases

The databases selected for this work were IEEE[1], ACM[2], Scopus[3] and ScienceDirect[4]. These databases were chosen because of their widespread use by the academic community. In addition, four databases were chosen to cover a greater diversity of work.

2.3.2 Search String

The following search string was used:

(”Testing”) AND (”adaptive systems” OR ”adaptive system” OR ”context aware” OR ”context-aware” OR ”context awareness” OR ”context-awareness” OR ”adaptive software” OR ”autonomic”)

2.3.3 Inclusion and Exclusion Criteria

The following inclusion and exclusion criteria were used to select the articles:

[1]https://ieeexplore.ieee.org/Xplore/home.jsp
[2]https://dl.acm.org/
[3]https://www.scopus.com/
[4]https://www.sciencedirect.com/

Inclusion Criteria: The article deals with testing DASs and is a primary work.
Exclusion Criteria: Article in a language different from English, an article published before 2020, and an article dealing with network testing

2.4 Study Selection

In the conduction stage, 312 articles were found using the search string. A filter for possible duplicates was carried out using the Parsifal [5] tool, which found 109 duplicates between the databases. The title and abstract of the articles were then read using the inclusion and exclusion criteria, giving a total of 25 articles for analysis. The 25 articles are listed in Table 1.

2.5 Quality Assessment

To assess the quality of the 25 articles selected for the study, they were analyzed according to a checklist for assessing the quality of studies. This checklist [6] was adapted to identify better information relevant to this research, such as the description of the approach and how it is presented, based on the checklist and evaluation scale suggested by (Kitchenham et al., 2010).

Following the evaluation scale suggested by (Kitchenham et al., 2010), ten articles answered "yes" to all the questions, and 15 articles answered most questions (but not all) with "yes". The lowest percentage of questions answered with "yes" was 81.25%, and the closer to 100%, the better the quality of the work. This percentage was calculated by the number of questions in the checklist divided by the number of questions answered with "yes". Thus, the articles selected have an acceptable degree of quality.

2.6 Data Extraction

Three researchers were involved in this review, so the selection activity was distributed between the two of them, and the review and synthesis of the results were shared between all those involved. In addition, alignment meetings were held to ensure that the authors agreed when extracting the information, and the agreement index was calculated from the Kappa test (Cohen, 1960) using the Jamovi tool (Jamovi, 2022) to check whether the reviewers agreed. We obtained a coefficient of 0.8, which, within the scale of interpretation indicated by (Kitchenham et al., 2010), fits "Substantial" and is considered a good coefficient of agreement between authors. The extraction stage was then carried out using the questions mentioned in the 2.2.1 section.

[5]https://parsif.al/

[6]The completed checklist can be found at https://github.com/isabelycosta/Testing-DAS-Challenges-and-Trends

2.7 Data Synthesis and Aggregation Strategy

The data collected was synthesized in two forms: RQ1.1, RQ1.2, and RQ1.3, which were summarized and analyzed to answer research question RQ1. Meanwhile, the data collected by question RQ2 was analyzed and synthesized using Grounded Theory processes described in Section 2.7.1.

2.7.1 Grounded Theory

For the data collected in the extraction stage related to research question RQ2, Grounded Theory procedures were applied to consolidate the results obtained and systematize the analysis of this qualitative data. The procedures carried out were:
Open Coding: This stage involved defining the codes and code identifiers, shown in Table 4, and associating them with the translated citations. The citations would be excerpts from the article collected in the phase described in 2.7. The codes were drawn up from the citations themselves.
Axial Coding: After open coding, the categories and sub-categories were defined at this stage based on the previously defined codes. In addition, the categories and sub-categories were related using propositions of causes and effects, intervening conditions, and action strategies. The propositions used were: *is associated with*, *is the cause of*, and *is part of the*.
Selective Coding: Finally, in this stage, the central category was defined, and the relationships between categories and sub-categories were reviewed. A network view was created to improve data visualization.

2.8 Reporting

Finally, in the final stage of presenting the results, the research and extraction questions were used as a reference for organizing the sections. At the same time, the discussion section was structured following the SLR objective mentioned in Section 2.1.

3 RESULTS

After applying the process described in Section 2, 25 articles were selected to answer the research questions. These articles are listed in Table 1. In addition, each paper is accompanied by its reference and the type of system dealt with in the paper's testing approach (i.e., Android, Web, Embedded Software, Internet of Things (IoT), Cyber-Physical Systems (CPS), and Undefined. A discussion of system types is provided in subsection 3.2.2.

Table 1: Selected papers.

Ref	Title	SUT
(Michaels et al., 2022)	Data Driven Testing for Context Aware Apps	Android
(Dadeau et al., 2022)	Online Testing of Dynamic Reconfigurations w.r.t. Adaptation Policies	Undefined
(Fanitabasi et al., 2020)	A self-integration testbed for decentralized socio-technical systems	IoT
(dos Santos et al., 2021)	Runtime testing of context-aware variability in adaptive systems	Android
(Piparia et al., 2021)	Combinatorial Testing of Context Aware Android Applications	Android
(DeVries et al., 2021)	Analysis and Monitoring of Cyber-Physical Systems via Environmental Domain Knowledge & Modeling	CPS
(Chen et al., 2021b)	Context-Aware Regression Test Selection	Web
(Mandrioli and Maggio, 2022)	Testing Self-Adaptive Software With Probabilistic Guarantees on Performance Metrics: Extended and Comparative Results	Undefined
(Shafiei and Rafsanjani, 2020)	A Test Case Design Method for Context Aware Android Applications	Android
(Mirza et al., 2021)	ContextDrive: Towards a Functional Scenario-Based Testing Framework for Context-Aware Applications	Undefined
(Yigitbas, 2020)	Model-Driven Engineering and Usability Evaluation of Self-Adaptive User Interfaces	Undefined
(Mandrioli and Maggio, 2020)	Testing Self-Adaptive Software with Probabilistic Guarantees on Performance Metrics	Undefined
(de Almeida et al., 2020a)	Context-Aware Android Applications Testing	Android
(de Almeida et al., 2020b)	ENVIAR: ENVIronment DAta SimulatoR	Android
(Chen et al., 2021a)	Simulated or Physical? An Empirical Study on Input Validation for Context-Aware Systems in Different Environments	CPS
(Doreste and Travassos, 2023)	CATS: A Testing Technique to Support the Specification of Test Cases for Context-Aware Software Systems	Undefined
(Usman et al., 2020)	TEGDroid: Test case generation approach for android apps considering context and GUI events	Android
(Doreste and Travassos, 2020)	Towards supporting the specification of context-aware software system test cases	Undefined
(Yi et al., 2022)	Improving the Exploration Strategy of an Automated Android GUI Testing Tool based on the Q-Learning Algorithm by Selecting Potential Actions	Android
(Dadeau et al., 2020)	Testing adaptation policies for software components	CPS

Table 1: Selected papers (cont.).

(Dadeau et al., 2021)	Automated Generation of Initial Configurations for Testing Component Systems	Undefined
(Maurio et al., 2021)	Agile services and analysis framework for autonomous and autonomic critical infrastructure	CPS
(Silva, 2020)	Adaptation oriented test data generation for Adaptive Systems	Undefined
(Chen et al., 2022)	Simulation Might Change Your Results: A Comparison of Context-Aware System Input Validation in Simulated and Physical Environments	CPS
(Wang et al., 2023)	Design and implementation of a testing platform for ship control: A case study on the optimal switching controller for ship motion	Embedded

3.1 Overview

A filter of articles found by year of publication was carried out to analyze the number of papers related to testing approaches in DASs in the last four years. The highest number of publications was 2020 (10 articles), followed by 2021 (8 articles), 2022 (5 articles), and the lowest number 2023 (2 articles). The low number of articles in 2023 is expected because the database search was carried out until the beginning of 2023, on March 16, 2023.

The selected papers were also categorized by database, nine publications were identified in Scopus[(Usman et al., 2020), (Doreste and Travassos, 2020),(Dadeau et al., 2020), (Dadeau et al., 2021), (Yi et al., 2022), (Chen et al., 2022), (Dadeau et al., 2022), (Michaels et al., 2022), (Maurio et al., 2021)], 7 in IEEE Xplorer[(Piparia et al., 2021), (DeVries et al., 2021), (Chen et al., 2021b), (Mandrioli and Maggio, 2022), (Shafiei and Rafsanjani, 2020), (Mirza et al., 2021), (Silva, 2020)], 6 in ACM [(Yigitbas, 2020), (Mandrioli and Maggio, 2020), (de Almeida et al., 2020a), (Chen et al., 2021a), (Doreste and Travassos, 2023), (de Almeida et al., 2020b)], and 3 in ScienceDirect[(Fanitabasi et al., 2020), (dos Santos et al., 2021), (Wang et al., 2023)].

3.2 RQ1) What Are the Characteristics of Current Approaches to Testing DAS?

In this Section, the results obtained from the extraction questions RQ1.1, RQ1.2 and RQ1.3 that contribute to answering the research question RQ1 are presented.

3.2.1 Characteristics of the Approaches

The types and levels of testing from Pierre and Richard (Pierre Bourque, 2014) and the definitions of testing activities from Garousi et al. (Garousi et al., 2020) were used to categorize the approaches. Thus, the approaches are listed in Table 2 by type of test (e.g., Performance test), by the level of the test (e.g., Unit), and finally by type of activity (e.g., Test-case design(criteria-based)). It is important to note that the testing activity indicated in the table is related to the final product of the approach, which means that even though the approach helps with other activities, only the final product was considered, considering this is the objective of the approach. In addition, the acronyms "I" and "S" were used to indicate Integration and System levels in the testing level column, respectively. These definitions are presented below:

Type of test (Pierre Bourque, 2014):
Acceptance testing determines whether a system satisfies its acceptance criteria, usually by checking desired system behaviors against the customer's requirements;
Regression testing is the selective retesting of a system or component to verify that modifications have not caused unintended effects and that the system or component still complies with its specified requirements;
Performance testing verifies that the software meets the specified performance requirements and assesses performance characteristics—for instance, capacity and response time.
Security testing is focused on the verification that the software is protected from external attacks. In particular, security testing verifies the confidentiality, integrity, and availability of the systems and its data;
Interface testing aims at verifying whether the components interface correctly to provide the correct exchange of data and control information.

Testing level (Pierre Bourque, 2014):
Integration testing is the process of verifying the interactions among software components;

System testing is concerned with testing the behavior of an entire system

Testing activity (Garousi et al., 2020):
Test-case design (criteria-based): Designing test suites (set of test cases) or test requirements to satisfy coverage criteria;
Test execution: Running test cases on the software under test (SUT) and recording the results.

Table 2: Categorization of approaches by type, level and testing activity.

Ref	**Testing type**	**Testing level**	**Testing activity**
(Fanitabasi et al., 2020)	Performance	I	Test execution
(dos Santos et al., 2021)	Acceptance	S	Test execution
(Piparia et al., 2021)	Acceptance	S	Test execution
(DeVries et al., 2021)	Acceptance	S	Test execution
(Chen et al., 2021b)	Regression	S	Test execution
(Mandrioli and Maggio, 2022)	Performance	S	Test execution
(Shafiei and Rafsanjani, 2020)	Acceptance	S	Test-case design (criteria-based)
(Mirza et al., 2021)	Acceptance	S	Test execution
(Yigitbas, 2020)	Interface	S	Test execution
(Mandrioli and Maggio, 2020)	Performance	S	Test execution
(de Almeida et al., 2020a)	Acceptance	S	Test execution
(Chen et al., 2021a)	Acceptance	S	Test execution
(Doreste and Travassos, 2023)	Acceptance	S	Test-case design (criteria-based)
(Usman et al., 2020)	Acceptance	S	Test execution
(Doreste and Travassos, 2020)	Acceptance	S	Test-case design (criteria-based)
(Dadeau et al., 2020)	Acceptance	S	Test execution
(Dadeau et al., 2021)	Acceptance	S	Test execution
(Yi et al., 2022)	Acceptance	S	Test execution
(Chen et al., 2022)	Acceptance	S	Test execution
(Dadeau et al., 2022)	Acceptance	S	Test execution
(Michaels et al., 2022)	Acceptance	S	Test execution
(Wang et al., 2023)	Acceptance	S	Test execution
(Silva, 2020)	Acceptance	S	Test execution
(de Almeida et al., 2020b)	Acceptance	S	Test execution
(Maurio et al., 2021)	Security	S	Test execution

Based on the categorization of Table 2 and following Pierre and Richard (Pierre Bourque, 2014) definition, the percentage of functional and non-functional testing approaches was analyzed. As a result, we obtained 20 functional approaches (dos Santos et al., 2021) (Piparia et al., 2021) (DeVries et al., 2021) (Chen et al., 2021b) (Shafiei and Rafsanjani, 2020) (Mirza et al., 2021) (de Almeida et al., 2020a) (Chen et al., 2021a) (Doreste and Travassos, 2023) (Usman et al., 2020) (Doreste and Travassos, 2020) (Dadeau et al., 2020) (Dadeau et al., 2021) (Yi et al., 2022) (Chen et al., 2022) (Dadeau et al., 2022) (Michaels et al., 2022) (Wang et al., 2023) (Silva, 2020) (de Almeida et al., 2020b), including Acceptance Testing and Regression Testing, and five (Fanitabasi et al., 2020) (Mandrioli and Maggio, 2022) (Yigitbas, 2020) (Mandrioli and Maggio, 2020)

(Maurio et al., 2021) non-functional approaches, divided into Security Testing, Interface Testing, and Performance Testing. The results are shown in Figure 1.

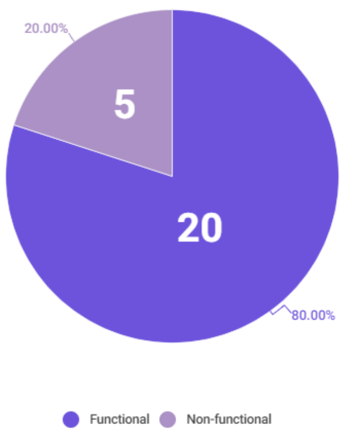

Figure 1: Percentage of functional and non-functional approaches.

3.2.2 Types of SUT Covered by the Approaches

Obtained from RQ1.2, this subsection presents the number of articles for each type of system addressed in the test approaches.

As shown in Table 1, it was unclear in most papers (9 papers) what type of SUT the approach is aimed. However, the main types identified were Android (8 papers), Cyber-physical systems (CPS) (5 papers), and Web, IoT, and Embedded with one paper each. Figure 2 shows the percentage of target system types per publication.

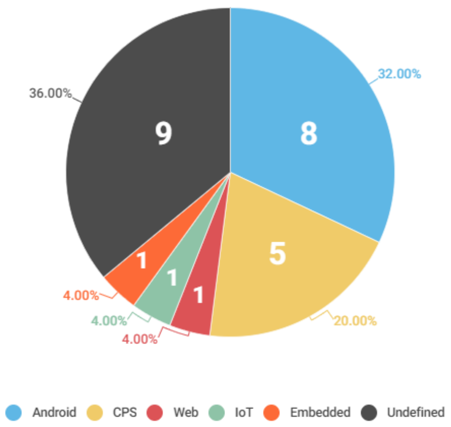

Figure 2: Percentage of target system types per publication.

3.2.3 Characteristics of the Execution Approaches

From RQ1.3, we obtained the characteristics related to when the testing approaches are applied, whether at runtime or design-time.

In most of the publications analyzed (10 articles), it was not possible to identify whether the approach presented by the author was applied at runtime or design-time since the distinction between the two types of execution is the environment in which the approach will be executed. The articles do not explicitly state the focus. Then there are eight runtime approaches, 5 in design-time and two that can be executed at runtime and design-time. Table 3 relates the runtime of the approach presented in the article to the publication reference.

Table 3: Type of execution per article.

Types of execution	**Papers**
Undefined	(Mirza et al., 2021), (Chen et al., 2021a), (Doreste and Travassos, 2023), (Doreste and Travassos, 2020), (Dadeau et al., 2021), (Yi et al., 2022), (Chen et al., 2022), (Dadeau et al., 2022), (Wang et al., 2023), (Silva, 2020)
Runtime	(dos Santos et al., 2021), (Yigitbas, 2020), (de Almeida et al., 2020a), (Usman et al., 2020), (Dadeau et al., 2020), (Michaels et al., 2022), (de Almeida et al., 2020b), (Maurio et al., 2021)
Design-time	(Fanitabasi et al., 2020), (Piparia et al., 2021), (DeVries et al., 2021), (Chen et al., 2021b), (Shafiei and Rafsanjani, 2020)
Both	(Mandrioli and Maggio, 2022), (Mandrioli and Maggio, 2020)

3.2.4 Other Results

By analyzing the articles using RQ1, it was also possible to obtain data related to approaches that used optimization mechanisms in their structure. This result is essential for understanding some of the challenges addressed in Section 3.3.

Most articles do not use optimization in their testing approaches (20 papers). Only five articles use optimization mechanisms to solve challenges related to DAS testing within the definition of Search-Based Software Engineering (SBSE) (Simons, 2013). The following paragraphs describe the mechanisms applied in these five articles (Fanitabasi et al., 2020),(Mandrioli and Maggio, 2022), (Dadeau et al., 2021), (Silva, 2020), (Maurio et al., 2021).

The optimization mechanism used in article (Fanitabasi et al., 2020) was the I-EPOS system (Pournaras et al., 2018), which performs a fully decentralized, self-organizing, and privacy-preserving combinatorial optimization. I-EPOS optimizes an objective for the entire system, measured by a global cost function. (Mandrioli and Maggio, 2022) test problem was finding the limits for an adaptive system's performance parameter. To solve this problem, they defined the following optimization problem: maximize the performance that can always be guaranteed under the constraint that it cannot exceed what is experienced in the tests carried out. In addition, the authors used classic statistical tools such as *Monte Carlo Simulations* (Robert and Casella, 2005) and *Extreme Value Theory* (Haan and Ferreira, 2010). (Dadeau et al., 2021), in turn, present a dedicated combinatorial algorithm that is used to enumerate all possible non-symmetric solutions of the CSP (Constraint Satisfaction Problem) defined by the component model in order to produce initial configurations. This algorithm integrates symmetry elimination patterns that reduce the combinations to be considered. Finally, (Maurio et al., 2021) use a genetic algorithm-based constraint programming approach for their scheduler/allocator to produce an initial configuration containing the start times and locations for all microservices. It is also worth mentioning that (Silva, 2020) claims to use an optimization mechanism in his testing approach, but from the article's analysis, it was impossible to identify what this mechanism might be and how it was used to support the testing activity.

3.3 RQ2) What Are the Current Challenges Related to Testing DAS?

In this Section, the results obtained from the extraction question RQ2.1 that contribute to answering the research question RQ2 are presented.

3.3.1 Testing Challenges in DASs

Among the 25 papers analyzed, 18 articles (Fanitabasi et al., 2020), (dos Santos et al., 2021), (Piparia et al., 2021), (DeVries et al., 2021), (Mandrioli and Maggio, 2022), (Mirza et al., 2021), (Yigitbas, 2020), (Mandrioli and Maggio, 2020), (de Almeida et al., 2020a), (Chen et al., 2021a), (Usman et al., 2020), (Doreste and Travassos, 2020), (Dadeau et al., 2020), (Yi et al., 2022), (Silva, 2020), (Dadeau et al., 2022), (de Almeida et al., 2020b), (Maurio et al., 2021) mentioned challenges of testing DASs and only in 7 articles (Chen et al., 2021b), (Shafiei and Rafsanjani, 2020), (Doreste and Travassos, 2023), (Dadeau et al., 2021), (Chen et al., 2022), (Michaels et al., 2022), (Wang et al., 2023) no mention of challenges was identified.

Following the *Grounded Theory* procedures described in Section 2, there was initially open coding where quotes related to the challenges of testing DASs were identified, and codes associated with these quotes were defined. The codes in Table 4 were defined according to the reading of the selected quotes. IDs were defined for each code for better organization and maintenance of the codes.

Table 4: Codes.

ID	Codes
COD01	The adaptation layer that reacts explicitly to uncertainty
COD02	Difficulty in detecting incorrect configurations at runtime
COD03	How to identify an application's context events
COD04	Complexity of the testing activity
COD05	High cost of test maintenance
COD06	Inconsistent and inaccurate context data
COD07	Dependence on dynamic context monitoring at runtime for validation and verification
COD08	Different execution scenarios that can be difficult to reproduce manually
COD09	Fragmented ecosystem
COD10	Explosion of scenario combinations
COD11	Lack of runtime approaches
COD12	Context heterogeneity
COD13	Uncertainties in change that affect validity
COD14	Limited validation and verification techniques
COD15	Limited methodologies that do not consider context
COD16	Limited methodologies that do not consider adaptation
COD17	Continuous change and adaptation
COD18	Need for an adaptation modeling language
COD19	Need for a context modeling language
COD20	Limited testing platforms
COD21	Large number of GUI and context events
COD22	Costly time to test many combinations
COD23	Costly to test many combinations

Table 5 refers to a part of the open coding stage[7] containing the association of the articles, with the extracted citation and the code related to the citation.

In the axial coding stage, the categories were defined based on the previously defined codes: The adaptation layer that explicitly reacts to uncertainty (COD01), Complexity of the testing activity (COD04), and Limitation of validation and verification techniques (COD14)[7].

Table 5: Part of the open coding table.

Ref.	Quote	Code ID
(dos Santos et al., 2021)	"Among the main challenges, the detection of incorrect configurations at runtime in the presence of context changes can be highlighted. "	COD02

Finally, in the selective coding stage, the central category was "Testing challenges in adaptive systems". Figure 3 shows the final result of the GT, with the central category, sub-categories, and their relationships presented by a network view.

4 DISCUSSION

This section discusses the results presented in Section 3, which are organized according to the SLR's objective of presenting challenges and trends. The "Challenges" subsection presents the challenges in testing DASs that are cited in the papers. The "Trends" subsection presents data indicating research opportunities in testing DASs.

4.1 Challenges

In order to identify the current challenges, the data collected in Section 3.3 was analyzed to see how many times the codes in Table 4 were cited in the 25 articles in this research. The most cited challenge among the articles was *Continuous change and adaptation* (cited eight times), followed by *Complexity of the testing activity* and *Explosion of scenario combinations*, both cited seven times. Other challenges involve *Uncertainties in change that affect validity* (cited six times), and finally, *Limited methodologies that do not consider context* (cited four times).

Category 1 (The adaptation layer that reacts explicitly to uncertainty) and 3 (Limited validation and verification techniques) have the most subcategories, eight respectively. It can be seen that the significant challenges in testing DASs are the adaptation layer related to:

- High cost of test maintenance
- Different execution scenarios that can be difficult to reproduce manually
- Context heterogeneity
- Uncertainties in change that affect validity
- Continuous change and adaptation
- Large number of GUI and context events
- Costly time to test many combinations
- Costly to test many combinations

Furthermore, the limitation of techniques to help validate and verify these systems, which are associated with:

- Difficulty in detecting incorrect configurations at runtime
- Dependence on dynamic context monitoring at runtime for validation and verification
- Lack of runtime approaches
- Limited methodologies that do not consider context
- Limited methodologies that do not consider adaptation
- Need for an adaptation modeling language
- Need for a context modeling language
- Limited testing platforms

In addition, Category 2 (Complexity of the testing activity) is also significant, with four subcategories, as it can also be seen that several factors make testing these systems complex (How to identify an application's context events, Inconsistent and inaccurate context data, and Explosion of scenario combinations).

4.2 Trends

Based on the subsection 3.2.1 results, there are few approaches to non-functional testing; only five articles deal with this type of testing. In addition, only the non-functional Performance, Interface, and Security tests are covered. Furthermore, most of the approaches focus more on the activity of executing tests on DASs and system-level testing. In this way, approaches to non-functional testing that focus on other activities and levels of testing are an opportunity for study.

[7]More information is available at https://github.com/isabelycosta/Testing-DAS-Challenges-and-Trends

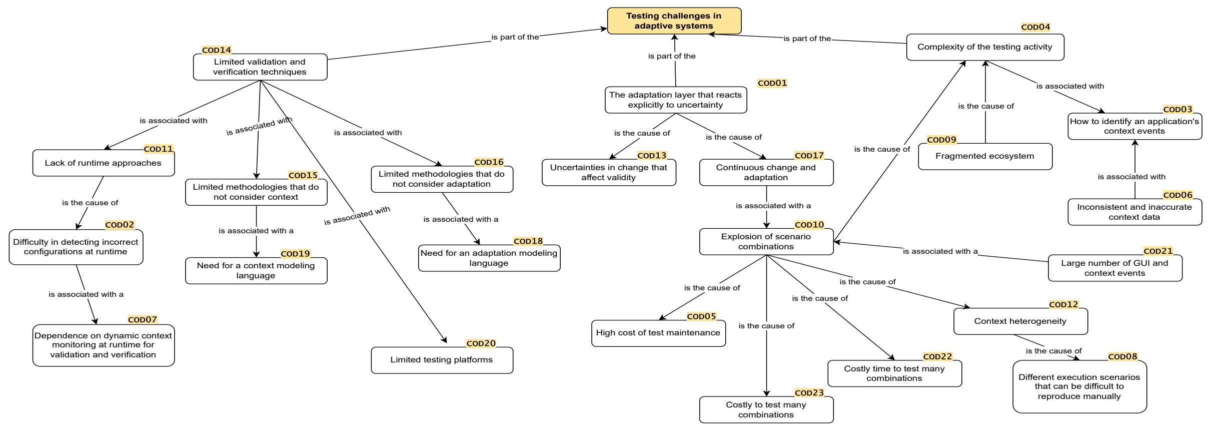
Testing challenges in adaptive systems
is part of the
is part of the
is part of the
COD14
Limited validation and verification techniques
COD01
The adaptation layer that reacts explicitly to uncertainty
COD04
Complexity of the testing activity
is associated with
is associated with
is associated with
is associated with
COD11
Lack of runtime approaches
COD15
Limited methodologies that do not consider context
COD16
Limited methodologies that do not consider adaptation
is the cause of
is the cause of
COD13
Uncertainties in change that affect validity
COD17
Continuous change and adaptation
is the cause of
is the cause of
COD09
Fragmented ecosystem
is associated with
COD03
How to identify an application's context events
is associated with
COD06
Inconsistent and inaccurate context data
is the cause of
COD02
Difficulty in detecting incorrect configurations at runtime
is associated with a
COD19
Need for a context modeling language
is associated with
is associated with a
COD18
Need for an adaptation modeling language
is associated with a
COD10
Explosion of scenario combinations
is associated with a
COD21
Large number of GUI and context events
is associated with a
COD07
Dependence on dynamic context monitoring at runtime for validation and verification
COD20
Limited testing platforms
is the cause of
COD05
High cost of test maintenance
is the cause of
is the cause of
is the cause of
COD12
Context heterogeneity
is the cause of
COD22
Costly time to test many combinations
COD08
Different execution scenarios that can be difficult to reproduce manually
COD23
Costly to test many combinations

Figure 3: Network view of categories and sub-categories.

Based on the results presented in Subsection 3.2.2, there has been a lack of approaches focused on Embedded, Web, and IoT systems in recent years. Therefore, presenting new approaches for DAS aimed at these systems could be an interesting topic for further research.

In Subsection 3.2.3, it can be seen that most of the studies that indicate the type of execution of the testing approach are aimed at runtime testing, and only two studies present approaches that can be both runtime and design-time. Proposing testing approaches for flexible DASs that can be executed at runtime and design-time could also be an opportunity for further research.

Work-related to optimization approaches is scarce. As presented in Section 3.2.4, only five articles in 4 years address optimization mechanisms within testing approaches in DASs. It is noteworthy that the use of optimization mechanisms in testing approaches for DAS may be promising due to the benefits already observed from using Search-Based Software Testing in other domains (McMinn, 2011).

4.3 Threats to Validity

The main threat related to the process of this systematic review is selection bias. To mitigate this, a strict research protocol was followed, with string testing and discussions about the results between the researchers at all stages of the review. In addition, relevant Software Engineering databases were used, which are the most frequently used in work in this area. A second threat identified concerns the data extraction process due to inaccuracy in extraction and bias in data synthesis. The research protocol was strictly followed to mitigate this threat, and the researchers frequently reviewed the data collected.

5 RELATED WORKS

There are already papers in the literature that address testing challenges and approaches in dynamically adaptive systems. However, we have identified a gap concerning an up-to-date overview of the challenges and characterization of testing approaches in DAS. The following is a summary of the main related works found in the literature so that it is possible to see the differences to our work, whose main contribution is to provide an up-to-date overview (2020-2023) of testing in DASs, seeking to identify its characteristics, associated challenges and trends in the area of testing in DASs.

(de Sousa Santos et al., 2017) present a *quasi*-systematic review of the literature on test case *design* techniques for CASS (Context Aware Software Systems) about quality characteristics evaluated, coverage criteria used, type of test and testing techniques. The quick *review* by (Matalonga et al., 2022) presents how researchers deal with context variation when testing CASS (Context-Aware Software Systems) developed in non-academic environments. (Priya and Rajalakshmi, 2022) provides an overview of the various CAA (Context Aware Applications) testing techniques available in the literature, research challenges in testing such applications, and what can help researchers in this field. Finally, (Lahami and Krichen, 2021) present a review of the literature focused on studies related to runtime testing in DAS, as well as approaches, frameworks, and test isolation techniques for these systems.

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS

6 CONCLUSION

Through the systematic review presented in this article, we can get an overview of the context of testing DASs over the last three years and envision challenges and opportunities. We were also able to identify a need for approaches focused on Embedded, Web, and IoT systems and that only two approaches are flexible in terms of the type of execution, being possible to execute them at runtime and design-time. In addition, there is a lack of approaches aimed at non-functional testing that support various testing levels and activities. Furthermore, of the 25 articles selected, only five applied optimization mechanisms in their testing approaches.

Finally, we realized several challenges related to testing DASs, mainly linked to the adaptation layer, the limited number of testing techniques, and the complexity of testing these systems. Our results can help future research on testing dynamically adaptive systems and encourage scientific production that seeks to mitigate the challenges identified.

ACKNOWLEDGMENTS

The authors would like to thank CNPQ, which provided the Industrial Technological Development Scholarship of Isabely do Nascimento Costa DTI-B (*N^o* 384087 / 2023-0) and the Productivity Scholarship in Technological Development and Innovative Extension of Rossana M. C. Andrade 1D (*N^o* 306362 / 2021-0). Furthermore, to Funcap/Iplanfor for their financial support in executing the ”Big Data Fortaleza” project.

REFERENCES

Chen, J., Qin, Y., Wang, H., and Xu, C. (2021a). Simulated or physical? an empirical study on input validation for context-aware systems in different environments. Internetware '20, page 146–155, New York, NY, USA. Association for Computing Machinery.

Chen, J.-C., Qin, Y., Wang, H.-Y., and Xu, C. (2022). Simulation might change your results: a comparison of context-aware system input validation in simulated and physical environments. *Journal of Computer Science and Technology*, 37(1):83–105.

Chen, Y., Chaudhari, N., and Chen, M.-H. (2021b). Context-aware regression test selection. In *2021 28th Asia-Pacific Software Engineering Conference (APSEC)*, pages 431–440. IEEE.

Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educational and psychological measurement*, 20(1):37–46.

Dadeau, F., Gros, J.-P., and Kouchnarenko, O. (2020). Testing adaptation policies for software components. *Software Quality Journal*, 28:1347–1378.

Dadeau, F., Gros, J.-P., and Kouchnarenko, O. (2021). Automated generation of initial configurations for testing component systems. In *Formal Aspects of Component Software: 17th International Conference, FACS 2021, Virtual Event, October 28–29, 2021, Proceedings 17*, pages 134–152. Springer.

Dadeau, F., Gros, J.-P., and Kouchnarenko, O. (2022). Online testing of dynamic reconfigurations wrt adaptation policies. *Automatic Control and Computer Sciences*, 56(7):606–622.

de Almeida, D. R., Machado, P. D., and Andrade, W. L. (2020a). Context-aware android applications testing. In *Proceedings of the XXXIV Brazilian Symposium on Software Engineering*, pages 283–292.

de Almeida, D. R., Machado, P. D. L., and Andrade, W. L. (2020b). Enviar: Environment data simulator. SBES '20, New York, NY, USA. Association for Computing Machinery.

de Sousa Santos, I., de Castro Andrade, R. M., Rocha, L. S., Matalonga, S., de Oliveira, K. M., and Travassos, G. H. (2017). Test case design for context-aware applications: Are we there yet? *Information and Software Technology*, 88:1–16.

DeVries, B., Fredericks, E. M., and Cheng, B. H. (2021). Analysis and monitoring of cyber-physical systems via environmental domain knowledge & modeling. In *2021 International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, pages 11–17. IEEE.

Doreste, A. C. d. S. and Travassos, G. H. (2020). Towards supporting the specification of context-aware software system test cases. In *CIbSE*, pages 356–363.

Doreste, A. C. D. S. and Travassos, G. H. (2023). Cats: A testing technique to support the specification of test cases for context-aware software systems. SBQS '22, New York, NY, USA. Association for Computing Machinery.

dos Santos, E. B., Andrade, R. M., and de Sousa Santos, I. (2021). Runtime testing of context-aware variability in adaptive systems. *Information and Software Technology*, 131:106482.

Fanitabasi, F., Gaere, E., and Pournaras, E. (2020). A self-integration testbed for decentralized socio-technical systems. *Future Generation Computer Systems*, 113:541–555.

Garousi, V., Felderer, M., Kuhrmann, M., Herkiloğlu, K., and Eldh, S. (2020). Exploring the industry's challenges in software testing: An empirical study. *Journal of Software: Evolution and Process*, 32(8):e2251.

Haan, L. and Ferreira, A. (2010). *Extreme value theory: an introduction*. Springer.

Jamovi (2022). The jamovi project.

Kitchenham, B., Sjøberg, D. I., Brereton, O. P., Budgen, D., Dybå, T., Höst, M., Pfahl, D., and Runeson, P. (2010). Can we evaluate the quality of software engineering experiments? In *Proceedings of the 2010 ACM-IEEE*

International Symposium on Empirical Software Engineering and Measurement, pages 1–8.

Kitchenham, B. A., Budgen, D., and Brereton, P. (2016). *Evidence-based software engineering and systematic reviews*. CRC press.

Krupitzer, C., Roth, F. M., VanSyckel, S., Schiele, G., and Becker, C. (2015). A survey on engineering approaches for self-adaptive systems. *Pervasive and Mobile Computing*, 17:184–206.

Lahami, M. and Krichen, M. (2021). A survey on runtime testing of dynamically adaptable and distributed systems. *Software Quality Journal*, 29(2):555–593.

Mandrioli, C. and Maggio, M. (2020). Testing self-adaptive software with probabilistic guarantees on performance metrics. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 1002–1014.

Mandrioli, C. and Maggio, M. (2022). Testing self-adaptive software with probabilistic guarantees on performance metrics: Extended and comparative results. *IEEE Transactions on Software Engineering*, 48(9):3554–3572.

Matalonga, S., Amalfitano, D., Doreste, A., Fasolino, A. R., and Travassos, G. H. (2022). Alternatives for testing of context-aware software systems in non-academic settings: results from a rapid review. *Information and Software Technology*, 149:106937.

Maurio, J., Wood, P., Zanlongo, S., Silbermann, J., Sookoor, T., Lorenzo, A., Sleight, R., Rogers, J., Muller, D., Armiger, N., et al. (2021). Agile services and analysis framework for autonomous and autonomic critical infrastructure. *Innovations in Systems and Software Engineering*, pages 1–12.

McMinn, P. (2011). Search-based software testing: Past, present and future. In *2011 IEEE Fourth International Conference on Software Testing, Verification and Validation Workshops*, pages 153–163. IEEE.

Mendes, E., Wohlin, C., Felizardo, K., and Kalinowski, M. (2020). When to update systematic literature reviews in software engineering. *Journal of Systems and Software*, 167:110607.

Michaels, R., Piparia, S., Adamo, D., and Bryce, R. (2022). Data driven testing for context aware apps. page 206 – 211.

Mirza, A. M., Khan, M. N. A., Wagan, R. A., Laghari, M. B., Ashraf, M., Akram, M., and Bilal, M. (2021). Contextdrive: Towards a functional scenario-based testing framework for context-aware applications. *IEEE Access*, 9:80478–80490.

Pierre Bourque, R. E. F. (2014). *Guide to the Software Engineering Body of Knowledge (SWEBOK(r)): Version 3.0*. IEEE Computer Society Press, 3rd edition.

Piparia, S., Adamo, D., Bryce, R., Do, H., and Bryant, B. (2021). Combinatorial testing of context aware android applications. In *2021 16th Conference on Computer Science and Intelligence Systems (FedCSIS)*, pages 17–26. IEEE.

Pournaras, E., Pilgerstorfer, P., and Asikis, T. (2018). Decentralized collective learning for self-managed sharing economies. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 13(2):1–33.

Priya, S. S. and Rajalakshmi, B. (2022). Testing context aware application and its research challenges. In *2022 International Conference on Smart Technologies and Systems for Next Generation Computing (ICSTSN)*, pages 1–7. IEEE.

Robert, C. P. and Casella, G. (2005). *Monte Carlo statistical methods*. Springer.

Shafiei, Z. and Rafsanjani, A. J. (2020). A test case design method for context aware android applications. In *2020 25th International Computer Conference, Computer Society of Iran (CSICC)*, pages 1–8. IEEE.

Silva, D. N. A. d. (2020). Adaptation oriented test data generation for adaptive systems. In *2020 15th Iberian Conference on Information Systems and Technologies (CISTI)*, pages 1–7.

Simons, C. L. (2013). Whither (away) software engineers in sbse? In *2013 1st International Workshop on Combining Modelling and Search-Based Software Engineering (CMSBSE)*, pages 49–50. IEEE.

Siqueira, B. R., Ferrari, F. C., Serikawa, M. A., Menotti, R., and de Camargo, V. V. (2016). Characterisation of challenges for testing of adaptive systems. In *Proceedings of the 1st Brazilian Symposium on Systematic and Automated Software Testing*, pages 1–10.

Siqueira, B. R., Ferrari, F. C., Souza, K. E., Camargo, V. V., and de Lemos, R. (2021). Testing of adaptive and context-aware systems: approaches and challenges. *Software Testing, Verification and Reliability*, 31(7):e1772.

Strauss, A. and Corbin, J. (1990). *Basics of qualitative research*. Sage publications.

Usman, A., Ibrahim, N., and Salihu, I. A. (2020). Tegdroid: Test case generation approach for android apps considering context and gui events. *International Journal on Advanced Science, Engineering and Information Technology*, 10(1):16.

Wang, L., Li, S., Liu, J., Hu, Y., and Wu, Q. (2023). Design and implementation of a testing platform for ship control: A case study on the optimal switching controller for ship motion. *Advances in Engineering Software*, 178:103427.

Yi, G. K., Baharom, S. B., and Din, J. (2022). Improving the exploration strategy of an automated android gui testing tool based on the q-learning algorithm by selecting potential actions. *Journal of Computer Science*, 18(2):90 – 102.

Yigitbas, E. (2020). Model-driven engineering and usability evaluation of self-adaptive user interfaces. *ACM SIGWEB Newsletter*, (Autumn):1–4.

SCIENCE AND TECHNOLOGY PUBLICATIONS