Self-adaptive Testing in the Field: Are We There Yet?

Samira Silva
Gran Sasso Science Institute (GSSI)
L'Aquila, Italy
samira.silva@gssi.it

Antonia Bertolino
ISTI-CNR
Pisa, Italy
antonia.bertolino@isti.cnr.it

Patrizio Pelliccione
Gran Sasso Science Institute (GSSI)
L'Aquila, Italy
patrizio.pelliccione@gssi.it

ABSTRACT

Testing in the field is gaining momentum, as a means to detect those failures that escape in-house testing by continuing the testing even while a system is operating in production. Among several approaches that are proposed, this paper focuses on the important notion of self-adaptivity of testing in the field, as such techniques need to adapt in many ways their strategy to the context and the emerging behaviors of the system under test. In this work, we investigate the topic by conducting a scoping review of the literature on self-adaptive testing in the field. We rely on a taxonomy organized in some categories that include the object to adapt, the adaptation trigger, the temporal characteristics, the realization issues, the interaction concerns, the type of field-based approach, and the impact/cost. Our study sheds light on self-adaptive testing in the field by identifying related key concepts and key characteristics and extracting some knowledge gaps to better guide future research.

KEYWORDS

Software testing in the field, Self-adaptive testing, Knowledge gaps

ACM Reference Format:
Samira Silva, Antonia Bertolino, and Patrizio Pelliccione. 2022. Self-adaptive Testing in the Field: Are We There Yet?. In *17th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS '22), May 18–23, 2022, PITTSBURGH, PA, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3524844.3528050

1 INTRODUCTION

Although traditionally software testing was reckoned as an activity for fault removal during development [1], in recent years awareness is growing among both academic researchers and practitioners of the necessity to prolong testing beyond software release, even while the system is operating in the field, e.g., [3, 5, 19, 40]. In fact, due to the high complexity, evolvability, and connectivity of modern software-intensive systems, many of the failures reported in production correspond to faults that would be very hard, if not impossible, to detect by in-house testing [23].

A recent systematic literature review (SLR) of field-based testing techniques [6] selected 80 primary studies published until 2017 and classified them according to several dimensions, including, among others, how, when, and where field tests are generated and activated. For several of the reviewed works, the authors of the SLR noticed that the proposed approaches perform some adaptation strategy, in order to deal with uncertainty or face emergent behaviours of the SUT (System Under Test). Moreover, in summarizing open challenges related to field-test cases generation, they conclude that these "*shall adapt to the production environment*" [6], whereas concerning the oracles for field testing, they observe that these "*need to adapt to the unknown execution conditions that can emerge in the field*" [6].

The concept of adaptive testing is also developed within the recent area of "software cybernetics" [10], which investigates the interplay between software and control processes. In this context, adaptive testing means that a software testing strategy is optimized by leveraging the knowledge collected during the testing that improves our understanding of the tested system [9]. However, this thread does not refer to testing in the production environment, which is our concern here.

In fact, we believe that exactly the same motivations that support moving the testing activity from in-house to the field, such as addressing uncertainty, dynamism and context-dependence, motivate as well the need for a self-adaptive testing strategy. From a more practical point of view, there are two types of systems for which a self-adaptive testing in the field approach would be extremely important: (i) self-adaptive systems (e.g., [20]) and (ii) systems that could be updated over time, due to the raise of new needs, among other reasons (e.g., [36]). In both cases, the precise and automatic adaptation of the testing process would reduce the efforts associated with the system adaptation. That is to say, **self-adaptability represents an extremely important feature when carrying out field testing activities**.

Nowadays self-adaptive systems are actively studied [12, 32, 44]. They are characterized as software-intensive systems that adjust themselves during their operation; such adjustments may be triggered by internal causes or by the context in which they operate, and may depend on specified properties or policies [44]. We think that this depiction well suits also the case of a software framework whose goal is to conduct testing in the field: the test approach itself may need to be adjusted based on what happens in the field, or due to changing requirements or expectations. We refer to such characterization of testing approaches as "self-adaptive testing in the field" (SATF for short). In general, SAFT can be a good choice not to replace traditional testing but to complement it.

Indeed, many of the approaches proposed for testing self-adaptive systems (SAS) are naturally conceived for being applied at runtime, typically when an adaptation occurs, and are themselves conceived as adaptive (we review several such papers in Section 4). However, we do not think that SATF scope is limited to SAS. For sure, it is an advisable approach for the testing of emerging service choreographies, as early prospected in the Choreos European project [7] in

SEAMS '22, May 18–23, 2022, PITTSBURGH, PA, USA

ACM ISBN 978-1-4503-9305-8/22/05…$15.00
https://doi.org/10.1145/3524844.3528050

which policies were defined to adapt the runtime testing strategy. More in general, in a recent opinion paper exploring challenges and promising directions in testing "changing software in a changing world" [8], Bertolino and Inverardi state that the testing approach should provide for the adaptive generation of test plans.

In the above-cited review of field-based testing [6], although self-adaptation aspects emerge in several points throughout the study, they are not studied as a self-standing dimension. This work aims to complement that previous SLR by focusing specifically on self-adaptive approaches to field testing. More precisely, the research questions we aim to answer are:

RQ1 What are the key concepts/definitions in self-adaptive field-based testing?
RQ2 What are the key characteristics of self-adaptive field-based testing?
RQ3 What are the known gaps/challenges in self-adaptive field-based testing?

RQ1 aims at defining SATF in a comprehensive way by looking at existing specific definitions. RQ2 aims at proposing a categorization for SATF. Finally, RQ3 aims at identifying the research gaps and challenges faced when performing SATF. To answer these questions, we conducted a systematic scoping review [39] of literature, and report here our findings. As we further explain in Section 3, scoping review is a better-suited method for the review of emerging topics and for outlining a conceptual framework. Our study reveals that SATF is still in its infancy, and proposes a taxonomy and a list of open challenges to foster future research.

In the next section, we briefly overview related reviews; in Section 3 we describe the search methodology; in Section 4 we report the results of the study, structured into three subsections, one per RQ; finally, in Section 5 we discuss the results, identify limitations and threats to validity, and hint at promising future research directions.

2 RELATED REVIEWS

This work overviews the literature on approaches for self-adaptive testing in the field. While we can report about a few secondary studies that partially overlap with our topic, we are not aware of any previous literature review that specifically addresses the same goal. In 2016, Siqueira et al. [45] conducted a SLR with the aim of identifying and classifying the challenges faced in testing adaptive systems. This paper selected 25 primary studies (spanning from 2003 to 2015) and collected 12 classes of challenges to be addressed. For lack of space, we leave out a summary of such challenges and refer to the cited survey for their enlightening discussion. We find that some of such challenges could be easily adapted when considering adaptive testing in the field. They mention, for example, the high number of configurations to be tested, or context-dependency, among others. Nevertheless, we still think that an *ad hoc* study of SAFT challenges is advisable and worthy of study.

In 2021, two parallel SLRs have been published that focus both on approaches for testing a software system at runtime in its final execution environment. In the already cited SLR on field-based testing [6], Bertolino et al. distinguish between testing approaches that are conducted in production (in-vivo) or use data from production (ex-vivo). On the one side, their search query is broader than ours, in that they do not focus on adaptive approaches as we do, and also include non-adaptive ones. Thus in principle, our set of primary studies would be a subset of theirs. However, on the other side: their search of literature included papers published until 2017, whereas we searched the literature until 2021, and 6 out of the 16 (i.e., 38%) primary studies we selected are indeed newer than 2017. Besides, in [6] the notion of adaptivity is not a primary concern, as for us, and even though existing SAFT approaches are included, the specific aspects of adaption: what is adapted, how and when, are not discussed. The other SLR on runtime testing restricts the study to approaches devoted to dynamically adaptable and distributed systems [33], and in fact they selected a lower number of papers in comparison with Bertolino et al. who did not restrict the domain of the tested application (precisely, 43 from 2006 to 2020, against 80 from 1989 to 2017). The authors of [33] ask eight research questions concerning characteristics of runtime test approaches, and among these the one that would appear as closely related to our study is RQ5 which is formulated as follows: "What kind of dynamic adaptations can these approaches support?". However, looking to how the question is answered, we understand that the adaptation they address refers again to the system under test, whereas the study does not discuss adaptation of the testing approach itself, as we aim to do here.

Finally, another interesting survey we found is a recent SLR by Siqueira et al. [46] of the types of faults that are encountered when testing adaptive systems and context-aware systems. As the authors explain, such a study can be useful for first understanding the nature of faults specific to those kinds of systems and then conceiving appropriate testing approaches to detect those types of faults. Hence they more specifically target fault-based testing techniques. While relevant, their SLR addresses completely different topics than our current study, that is, they are not providing any characterization on self-adaptive testing in the field, as we aim to do.

3 METHODOLOGY

To answer the research questions described in the introduction, we decided to follow the scoping review research methodology [13, 39, 41]. According to [39], scoping reviews are similar to systematic reviews since they follow a rigorous structured process. However, they have different ambitions and present some different key methodological points. In cases where systematic reviews fail to achieve the necessary aims or criteria of knowledge users, scoping reviews are increasingly considered a viable option.

We believe that scoping review is the most appropriate research methodology since SATF is not yet mature to properly conduct a systematic literature review. This work may also be a helpful precursor to future systematic reviews since it identifies and analyses knowledge gaps in self-adaptive testing in the field. According to the typical purposes for conducting a scoping review [39], this work will help to (i) identify knowledge gaps and key characteristics or factors related to a concept, (ii) scope a body of literature, and (iii) clarify concepts and definitions in the literature. The scoping review research methodology is largely used in health care [13, 39, 41], but is getting attention also in the software engineering community [2]. We make available a replication package of this study at [https:

//samirasilva.github.io/publication/seams_2022/]. We present the review protocol as follows.

3.1 Search Strategy

Considering the research questions described in the introduction, it was established that the search string that would be used for searching the relevant papers should contain variations of the following terms: "software", "online testing", "runtime testing", "field-based testing", "self-adaptive", "self-organizing", "autonomous", "self-managing", and "adaptive". Then, these terms were combined with the Boolean operators AND and OR to be linked. The resulting search string was defined as follows:

```
software AND ("online test*" OR "run-time test*" OR
"runtime test*" OR "field-based test*") AND (self-adaptive
OR "self adaptive" OR self-organizing OR "self organizing"
OR autonomous OR self-managing OR "self managing" OR
"self organising" OR self-organising OR adaptive)
```

To reduce the number of irrelevant search results, we have applied our search over the "full text" of the papers under investigation, that is, only the parts with relevant contents are considered, excluding references, for example.

The databases in which the query above mentioned is applied are presented in Table 1. Our scoping review was conducted using three of the most widely used databases: IEEExplore, ACM Digital Library and Scopus.

Table 1: Search engines for scientific papers.

#No	Database	URL
1	IEEExplore	https://ieeexplore.ieee.org/
2	ACM Digital Library	https://dl.acm.org/
3	Scopus	https://www.scopus.com/

By using the above-mentioned databases and the proposed search string, and considering only papers published from 2012 to 2021, the number of records extracted is equal to 591, 383 and 19, for Scopus, ACM Digital Library and IEEExplore, respectively[1].

3.2 Screening and Duplicate Removal

Since there were several papers that matched the query but were not relevant, we screened the papers by only reading the title and abstract and removed those which were clearly irrelevant. Then, merging the results obtained from screening for different databases into a single set, we removed the duplicates, resulting in 118 papers.

3.3 Selection Criteria

This search is mainly centered on the study of the existing literature on self-adaptive testing in the field. Therefore, the search scope was restricted to this topic through inclusion and exclusion criteria. The inclusion criteria are motivated by the research questions. On the other hand, the exclusion ones are standard quality criteria frequently used. At this stage, we aimed at comprehensiveness, thus we took into consideration papers published in the last ten years that are in English and could be downloaded. It is important to mention that despite the existence of exclusion criteria aiming at removing papers in languages other than English or that could not be downloaded, our search did not report any paper in this situation.

Finally, the application of inclusion and exclusion criteria resulted in 36 papers. Table 2 summarizes the selection criteria.

Table 2: Inclusion and exclusion selection criteria.

Inclusion criteria	Exclusion criteria
I1 - Papers that provide a definition on self-adaptive testing in the field. I2 - Papers that describe characteristics of approaches to perform self-adaptive testing in the field. I3 - Papers that describe gaps/challenges in self-adaptive testing in the field. I4 - Related papers published from 2012 up to 2022.	E1 - Papers that cannot be downloaded. E2 - Studies in languages other than English. E3 - Papers published before 2012. E4 - Unpublished papers. E5 - Secondary studies papers. E6 - Duplicate and out-of-scope papers (i.e. not fulfilling I1, I2 or I3).

3.4 Final List Selection

After applying the selection criteria, the selected papers were merged and assigned to two of the authors to be reviewed. Each reviewer went over the full text of every paper independently deciding whether it should be included or not. All papers that resulted in diverging opinions were discussed in plenary meetings. Then, after a final consensus, we got 16 papers as the final list.

3.5 Data Extraction

In the stage of reading the full text, every author independently also extracted the relevant data for answering the three RQs anticipated in the Introduction (Section 1). For each paper thus we had two independent readings and classifications (in particular, for answering RQ2, we used a spreadsheet available from the replication package), which were then discussed in a series of meetings attended by the authors, for clarifying unclear categories, and solving possible conflicting opinions.

3.6 Review Protocol Summary

This research has focused mainly on papers describing approaches for self-adaptive testing in the field. Figure 1 shows the complete protocol employed in this scoping review. A total number of 134 records were identified from 3 different databases. Then, 118 records were identified to be irrelevant with respect to our topic and were excluded. Thus, at the end of the process, only 16 records were identified to be relevant. The list of primary studies included in this paper is shown in Table 3.

4 RESULTS

In the following, we report the results from our analysis of the 16 selected studies. We structure the discussion into three subsections, correspondingly with the three RQs to be answered.

[1] The searches were conducted on 30/11/2021 (ACM and IEEE) and 06/12/2021 (Scopus).

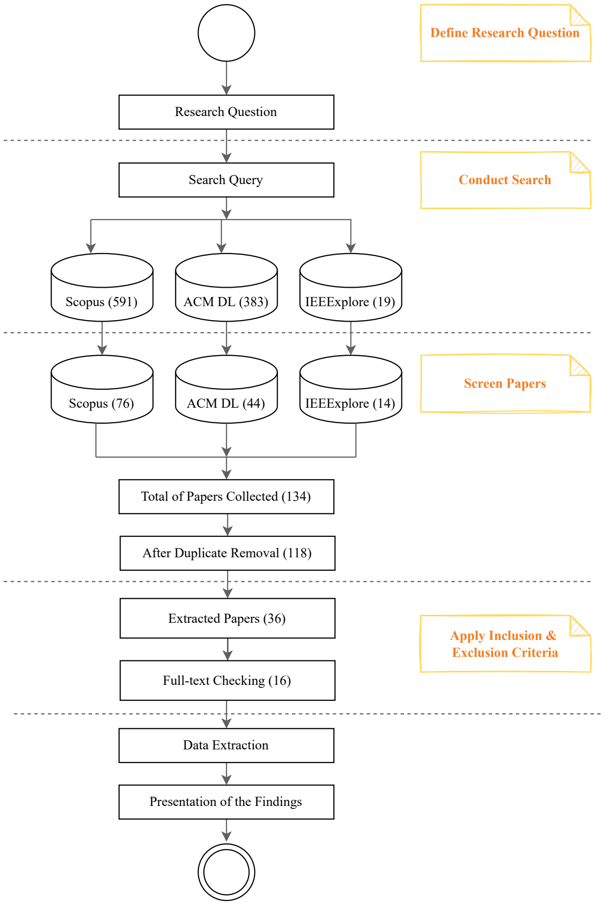

Figure 1: Summary of the selection protocol for this scoping review.

Table 3: SATF approaches.

Final list of papers		
Ref.	Year	Title
[38]	2012	Verification and testing at run-time for online quality prediction
[22]	2013	Towards run-time testing of dynamic adaptive systems
[14]	2014	Dynamic test reconfiguration for composite web services
[21]	2014	Towards run-time adaptation of test cases for self-adaptive systems in the face of uncertainty
[20]	2015	Automated generation of adaptive test plans for self-adaptive systems
[36]	2016	Automated workflow regression testing for multi-tenant saas: integrated support in self-service configuration dashboard
[34]	2016	Safe and efficient runtime testing framework applied in dynamic and distributed systems
[30]	2016	Towards autonomous self-tests at runtime
[43]	2017	Self-test framework for self-adaptive software architecture
[28]	2017	Towards collective online and offline testing for dynamic software product line systems
[42]	2018	Run-time reliability estimation of microservice architectures
[26]	2019	A hybrid framework for web services reliability and performance assessment
[35]	2019	The SAMBA approach for Self-Adaptive Model-BAsed online testing of services orchestrations
[11]	2020	A Framework for In-Vivo Testing of Mobile Applications
[27]	2021	On-demand Test as a Web Service Process (OTaaWS Process)
[18]	2021	Runtime testing of context-aware variability in adaptive systems

4.1 Concepts and Definitions

In this section, we aim at answering RQ1, i.e., **what are the key concepts/definitions in self-adaptive field-based testing?**, and at defining SATF by analyzing the current literature on this subject.

We will start from an overview of the different views/proposals about SATF in the selected papers. Generally speaking, each of the works found presents their own vision of SATF, with respect for example to what to monitor, what to adapt, when to adapt, how to adapt, etc. Even though they are not providing a single and standard definition for SATF, their similarities may contribute to the definition of one. The works in [21] and [20] adapt, at runtime, test cases that are aligned with the requirements, according to the system and environmental conditions. Cooray et al. [14] react to service operations, operation arguments, and service composition changes. Similarly, the work in [26] inspects changes occurring in the way the service is provisioned or used to trigger testing sessions. Contrarily, Ceccato et al. [11] monitor the system in order to detect untested configurations and trigger the testing. In the work of Hänsel and Giese [28], the observation of multiple instances of systems derived from a dynamic software product line, along with their applied configurations, is considered with the aim of estimating an up-to-date operational profile. Posteriorly, test cases are incrementally run based on this estimated profile. The work of Heck et al. [30] supports the idea that an "autonomous self-organising system must be capable of self-analysis to detect system components that are faulty". In this sense, it proposes an approach in which the multiple components of a self-organising system are able to test each other. In passive tests, a component evaluates the behavior of the testee during normal system activity. In active tests, it generates test events and observes the testee's reaction.

The online prediction of failures allows the system to anticipate adaptations and prevent further actual occurrence of failures. Metzger et al. [38] make use of SATF for this purpose, by collecting usage rates from constituent services and executing online tests against these services to obtain quality data only if the usage rates are below a predetermined threshold. Then, failure prediction is performed based on the combination of monitoring and testing data. The work by Fredericks et al. [22] proposes a feedback loop to supplement runtime testing strategies named MAPE-T. This is composed of the traditional Monitoring, Analyzing, Planning, and Executing stages of a SAS, here conceived for supporting the Testing activities. With their proposal, they aim at claiming that tests should be treated as "first-class entities that can evolve as requirements change and/or self-reconfigurations are applied" [22]. Besides, they also posit that test evolution is a multidimensional goal, and based on the current context of the system, it must adapt and safely run test cases. Keeping the test cases consistent allows for their reuse at runtime to check not only if they satisfy the specification, but also possible conditions that make the adaptation necessary. A methodology to support the runtime testing of modifications resulting from self-testing components in SAS is also proposed in [43]. Their work aims at making self-testing an implicit characteristic of the systems.

Tests may also be activated on demand. The work in [42] allows for the estimation of microservice reliability at runtime upon reliability assessment requests that can be performed periodically or at every new release of a microservice, for example. Santos et al. [18] evaluate the variability of an adaptive system at runtime by verifying the need for runtime testing after adaptation rules being performed by the system. Whenever it is necessary, their approach generates test cases with abrupt changes of context to encourage adaptation responses and find failures. The SAMBA approach, proposed in [35], addresses functional and regression testing of services orchestrations at runtime since it intends to discover faults originated from evolutionary behaviors, such as the addition of new functionalities. A model is extracted or updated from the orchestration description. If changes are detected in the orchestration, model updates are also triggered. To automatically generate test cases, their approach takes into consideration this model.

Also concerning regression testing at runtime, the approach of Makki et al. [36] does not include a monitor component. Instead, it directly derives test cases from successful executions that are chosen by the tenant administrator. Another approach that also does not include the monitoring phase is proposed by Habibi and Mirian-Hosseinabadi [27] to test a Service- Oriented Architecture (SOA) application. In their approach, test data are automatically generated based on requirements and input data of the consumer's application, and executed on demand. Finally, the work in [34] proposes the evaluation of dynamic and distributed systems through the framework RTF4ADS that performs execution in the field of "test cases covering only software components or compositions affected by the dynamic change" [34].

From our analysis of the selected studies, none of them explicitly provides a definition of SATF, and indeed as we have summarised they also present different approaches. In the already cited SLR about field testing [6], field testing is defined as "any type of testing activities performed in the field", which is a very abstract and comprehensive definition. Based on the above analysis of the current literature in SATF and taking into account the shared key concepts, we adapted the above generic definition as follows:

Self-Adaptive Testing in the Field (SATF) is any type of testing activities performed in the field, which have capability to self-adapt to the different needs and contexts that may arise at runtime.

Such a generic definition can embrace all the previously mentioned studies, which expose the mentioned "capability to self-adapt" in different ways.

4.2 A Taxonomy for SATF

To answer RQ2, i.e., **what are the key characteristics of self-adaptive field-based testing?**, we propose a categorization of self-adaptive approaches to test in the field. The categorization is based on a preliminary taxonomy that we identified by considering 4 highly cited surveys proposed by the self-adaptive systems community [12, 15, 32, 44] and a recent survey in testing in field [6]. This initial categorization is then exercised on the collected papers and refined according to the specific characteristics of self-adaptive approaches to test in the field. In the following, we describe the final version of the categorization. The main categories of self-adaptive field-based testing are (i) the object to adapt, (ii) adaptation trigger, (iii) temporal characteristics, (iv) realization issues, (v) interaction concern, (vi) class of field-based approach, and (vii) impact and cost. It is important to highlight that the work in [22] is not considered for RQ2 since the work is contributing to a feedback loop approach without providing enough information about a concrete approach, as we instead would need to answer RQ2. However, the paper provides very useful information to answer RQ1 and RQ3.

4.2.1 Object to Adapt. In a self-adaptive field-based testing process, its components may self-adapt in order to handle information collected from the field. Considering the components of testing activity, we consider that the components of testing in the field that could need to be adapted can include the *test cases*, the *oracle*, the *monitor* or the *test approach* itself (e.g. the test criterion, the test plan, etc). Table 4 shows the sub-categories of the *Object to adapt* category.

(1) *Test Cases.* The test suit, i.e., the set of test cases, can be adapted by either modifying existing test cases or generating new ones, according to what is observed in the field. As shown in Table 4 in many approaches the object to adapt are the test cases. The strategy of generating new test cases requires more effort with respect to generating them at design time; however, it solves the problem of keeping aligned a pre-generated test suite [26]. In other cases, it becomes inevitable since test cases might become invalid due to system adaptation [21]. Existing test cases might be also adapted, as described, e.g., in [20–22, 28]. Some approaches have also multiple objects to adapt. For instance, the work in [20] adapts test cases for fine-grained adaptation, and adapts the test suites/plans for coarse-grained adaptation.

(2) *Oracle.* Field-based approaches rely on different types of oracle to decide the test outcome [6]. Oracles can be based on specifications, defined by users or some QoS, or they can exploit the detection of crashes or unchecked exceptions. In our study, only one approach has the oracle as the object to adapt [18]. More specifically, oracles are generated at runtime based on the extended Context Feature Model (eCFM) [16], which is exploited to model systems

Table 4: Characteristics of SATF: Object to adapt.

Object to adapt	Test case [11, 14, 18, 20, 21, 26–28, 35, 36, 42, 43]
	Oracle [18]
	Monitor
	Test approach [11, 20, 30, 34, 36, 38, 43]

that adapt their features according to the environment. In the context of this paper, oracles are propositional formulas that declare the correct state of the features.

(3) *Monitor*. The monitor may also be the object to adapt. It is responsible for dynamically collecting and interpreting data about the execution of the SUT [6]. The process of monitoring is essential in field-based testing since this task is based on data that is collected by this component. In the surveyed works we did not find examples of adaptation of monitor. However, we decided to keep it as an option in the object to adapt category since it can make sense to adapt monitors at runtime according to the potential evolution of the SUT.

(4) *Test Approach*. The test approach itself may also be the object to adapt. Possible adaptations are, for instance, the adaptation of the test plan or timing in the case of the tests to be executed are periodically scheduled (periodical trigger of Table 5). Various works adapt the test approach. The work in [34] aims at testing dynamic and distributed systems and performs an adaption of both test selection and test placement, i.e. the assignment of test components to the execution nodes. As another example, the work in [36] adapts both the test case selection and the workflow regression testing for multi-tenant SaaS in order to bypass costly, halting, or time-consuming workflow steps.

4.2.2 Adaptation Trigger. This category describes the adaptation characteristics of self-adaptive approaches to test in the field concerning the trigger of adaptation, i.e., how the adaptation is initiated. Table 5 shows the *Adaptation Trigger* category. A trigger identifies the events that lead to the activation of field test cases. More precisely, "a trigger is any kind of event, scenario, or configuration whose occurrence leads to the execution of some field test cases" [6]. The adaptation can be triggered (i) periodically, (ii) by the internal events of the SUT or changes in its environment or in its technical resources, (iii) by some policy, or (iv) on a request of a testing session from testers, runtime infrastructure/container, etc. Most of the approaches have the SUT as a trigger, which can be a change in the environment or a SUT internal event [11, 21], change in a technical resource [20], upon changes or adaptation in the SUT [18, 34], or when the usage rate of the SUT falls below a certain threshold [42]. Besides of periodical triggers or triggers based on some defined policies, adaptation can be also triggered on demand by, e.g., the testing infrastructure [21] or the tenant administrator [36]. Finally, for the work in [30] the adaptation trigger is not specified.

4.2.3 Temporal Characteristics. Temporal characteristics are related to issues concerning *when* artifacts can/need to be changed [44]. It presents two main sub-facets: the *type of monitoring* and the *foreseeability*. Table 6 shows the subcategories of the *Temporal characteristics* category.

(1) *Type of Monitoring*. This sub-facet defines whether the monitoring process is *Continuous*, that is, it is "continually collecting and processing data" [44] or *Discontinuous* (named Adaptive in [44]), so that a small number of features are monitored and if an anomaly is found, the monitor acts by collecting more data. The decision with respect to this sub-facet influences the monitoring cost and detection time. The majority of approaches have continuous monitors and none of the surveyed works has a discontinuous monitor. There are also some approaches that have no monitor. This is for instance the case of [20], where, since they are testing a SAS, the authors rely on the monitoring component of the SUT.

(2) *Foreseeability*. This sub-facet concerns whether "change can be predicted ahead of time" [12]. The approaches are classified according to the degree of foreseeability: *Foreseen* (taken care of) and *Foreseeable* (planned for) [12]. We did not include *Unforeseen* (not planned for) in the taxonomy since we did not find it suitable for testing approaches. All the surveyed approaches are classified as foreseeable.

Table 5: Characteristics of SATF: Adaptation trigger.

Adaptation trigger	Periodical [27]
	SUT [11, 18, 20, 21, 26, 28, 34, 35, 38, 43]
	Policy [14, 20, 21]
	On-demand [21, 27, 28, 36, 42]

4.2.4 Realization Issues. This facet aims at capturing "*how* the adaptation can/need to be applied" [44]. Table 7 shows the subcategories of the *Realization issues* category.

(1) *Technique*. This sub-facet aims at answering the question "what kind of change is needed?" [32], that is, according to what is observed, what adaptations are necessary. The work in [32] categorizes adaptation techniques into *Parameter*, *Structure*, and *Context*. Parameter techniques perform adaptation through parameter change; examples might be found in [20, 42]. Structure techniques refer to the changes in the structure of the testing system, and they also include the removal/addition of test cases or the update of test reports [14, 28], test placement [34] or test plan and identification on where to test [43]. Finally, context techniques concern changes in the context, such as the testing rate being adapted according to the usage rate [38]. These three adaptation techniques may also be combined. As an example, the work in [20] uses parameters for fine-grained adaptation and structure for coarse-grained.

(2) *Decision Making*. The *static decision-making* consists in hard–coding the decision process, so its modification demands the recompilation and redeployment of the system (or its components). The *dynamic decision-making*, instead, makes policies, rules, or QoS easily manageable during runtime to encompass a new behavior related to functional and non-functional software requirements [44]. All the surveyed approaches present static decision-making.

Table 6: Characteristics of SATF: Temporal characteristics.

Temporal characteristics	Type of monitoring — Continuous [11, 14, 18, 21, 26, 28, 30, 34, 35, 38, 42, 43] — Discontinuous — No monitor [20, 27, 36]
	Foreseeability — Foreseen — Foreseeable [11, 14, 18, 20, 21, 26–28, 30, 34–36, 38, 42, 43]

Table 7: Characteristics of SATF: Realization issues.

Realization issues	Technique Parameter [20, 42] Structure [11, 14, 20, 21, 26–28, 30, 34–36, 43] Context [18, 38]
	Decision making Static [11, 14, 18, 20, 21, 26–28, 30, 34–36, 38, 42, 43] Dynamic
	Type of adaptation Internal [11, 14, 18, 20, 21, 26–28, 30, 34–36, 38, 42, 43] External
	Openness Open [18] Close [11, 14, 20, 21, 26–28, 30, 34–36, 38, 42, 43]
	Degree of decentralisation Decentralised [30] Hybrid [11] Centralised [14, 18, 20, 21, 26–28, 34–36, 38, 42, 43]
	Trigger Proactive [20, 21, 28, 30, 38, 42] Reactive [11, 14, 18, 26, 27, 34–36, 38, 43]

Table 8: Characteristics of SATF: Interaction concern.

Interaction concern	Human involvement No human involvement [20, 21, 26, 27, 34, 35, 38, 42, 43] Human involvement [11, 14, 18, 28, 36]
	Trust [30, 34]

(3) *Type of Adaptation.* The type of adaptation is related to the separation of the adaptation mechanism and application logic [44]. In this sense, approaches can be categorized into *Internal* or *External*. Internal approaches merge the application and adaptation logic. In this category, sensors, effectors, and adaptation processes are mixed with the application code. It can be useful for handling local adaptations, although it may result in problems with scalability and maintainability. External approaches make use of an external adaptation engine comprehending the adaptation processes. A significant advantage of external approaches is the reusability of the adaptation engine. It can be modified to handle different configurations of applications. Each of the approaches we surveyed is internal. In the future, we might imagine having external approaches to benefit from the reusability of the adaptation engine and to facilitate the transformation of non-adaptive existing field--based testing towards self-adaptiveness.

(4) *Openness.* It refers to the openness of the set of adaptive actions [44]. A *close-adaptive* system presents an established number of adaptive actions, that is, no new action can be introduced during runtime. Differently, an *open-adaptive* system can be extended, and therefore new adaptive actions can be added, resulting also in the possibility of including new entities in the adaptation mechanism. All the surveyed approaches are close with the only exception of the work in [18], which is open. In their work, the adaptation is based on adaptation rules that might be changed throughout time.

(5) *Degree of Decentralisation.* This sub-facet corresponds to the degree of decentralisation held by the adaptation logic [32]. A *centralised adaptation* logic is recommended for small systems, containing few resources to be managed. On the other hand, large systems, comprising a large number of components to manage, require a *decentralised adaptation* approach, in order to split responsibilities and improve the system performance for adaptation. Finally, *hybrid adaptation* approaches introduce central components to decentralised approaches or split the adaptation logic functionality into sub-systems. Since the testing system is often composed of a relatively small number of components, most of the approaches are centralised. We mention a few exceptions to decentralised testing. The work in [30] aims at testing a self-adaptive system composed of a number of independent agents. Each agent can test other agents. When an agent identifies another agent as malfunctioning it can also happen that the testing agent is faulty. For this reason, before reporting a malfunctioning, there should be more than k agents reporting a malfunction of the same node, according to the k-resilience notion [29]. Another example of decentralisation is described in [11], where the testing in the field is applied to the mobile domain. In vivo tests are decentralised but there is a central server orchestrating the overall testing activities. For this reason, we categorize this approach as hybrid. The other works are categorized as centralised.

(6) *Trigger.* This sub-facet is related to the question “When should we adapt?”. Therefore, approaches are categorized according to the time they adapt. According to [32], the temporal aspects of the adaptation can be divided into two dimensions: *Reactive* and *Proactive*. By adapting to testing the general description in [44], in the case of reactive approaches, the testing system responds when a change has already happened, while in the case of proactive approaches, the testing system predicts when the change is going to occur and can anticipate its self-adaptation. The majority of approaches surveyed are reactive, but some of them are proactive. A clear example of a proactive approach is presented in [38] in which external and local failures of the system are predicted and the testing process self-adapt to handle these predictions. On the other hand, the work in [28] performs adaptations based on an estimated operational profile.

4.2.5 Interaction Concern. This facet concerns issues involving the interaction with humans and/or other self-adaptive systems/elements through the use of interfaces [44]. In this case, the approaches are categorized according to *Human Involvement* and the “Trust". Table 8 shows the subcategories of the *Interaction concern* category.

(1) *Human Involvement.* This sub-facet consists in who is the agent of change. For some approaches, there is *No Human Involvement*. On the other hand, we also identified works that require human involvement. The work in [36] focuses on testing Multi–Tenant SaaS. The tenant administrator (human) triggers the testing

Table 9: Characteristics of SATF: Class of Field Test (FT).

Class of FT	Ex-vivo testing	Functional
		Non-functional [26]
	Offline testing	Functional [43]
		Non-functional
	Online testing	Functional [11, 14, 18, 20, 21, 27, 28, 30, 34–36, 38, 42, 43]
		Non-functional

and instructs the test cases generation by selecting from a previous execution of the workflow. The work in [11] requires the involvement of developers when unknown configurations are found. The work in [14] requires human involvement for the analysis of test reports by the service provider. Finally, the work in [28] requires humans to decide whether to trigger online tests or analyze reports.

(2) *Trust.* With this sub-facet, approaches are evaluated with respect to the trust they present. According to [44], "trust is a relationship of reliance, based on past experience or transparency". It may involve aspects concerning security [17], assurance, dependability [24] and predictability [31, 37]. On one side, testing can be considered an instrument that can help in improving trust towards the SUT. This is especially true for field-based testing since it is devoted to testing systems in the field or via the use of data coming from the field. However, trust is also extremely important in the approach that is used for testing the system. Besides trusting that testing will reliably find bugs, in field-based testing, and especially in online testing, trust takes also the dimension of building confidence that the testing will not compromise the correct behavior of the SUT and will "not interfere or delay required adaptations" [22]. As identified in [34] a key prerequisite for enabling safe runtime testing is the runtime testability, which includes test sensitivity, i.e. whether the component under test can be tested without unwanted side-effects, and test isolation to prevent test processes from interfering with the SUT behavior. Trust is a very important sub-facet but not much considered so far in self-adaptive field-based testing approaches. The approach in [30] investigates the problem of trusting the results of testing. The work aims at testing various independent agents and, more specifically, an agent is tested by other agents. In order to trust a malfunction result of a test, the authors propose the collect at least k malfunction reports before recording the presence of a malfunction.

4.2.6 *Class of SATF.*

According to [6], field-based testing approaches can be categorized into *Ex-vivo*, *Offline* and *Online*, depending on when testing activities are performed and whether the actual system is used by these activities. Finally, they are also classified according to the type of faults they are approaching. An approach may aim to find *Functional* or *Non-Functional* faults. Table 9 shows the subcategories of the *Class of field-based approach* category.

(1) *Ex-vivo Testing*. This sub-facet comprises testing approaches "performed in the development environment using information extracted from the field" [6]. We only found one approach that is performing ex-vivo testing, i.e. [26]. More precisely the work is performing online non-functional testing for estimating the reliability and/or performance of a web service.

Table 10: Characteristics of SATF: Impact and cost.

Impact and cost	Measured [11, 18, 20, 26, 34]
	Not measured [14, 21, 27, 28, 30, 35, 36, 38, 42, 43]

(2) *Offline Testing*. This sub-facet comprises testing approaches "performed in the production environment on a SUT separated from the actual system" [6]. We only found one work that is performing offline and online testing [43].

(3) *Online Testing*. This sub-facet comprises testing approaches "performed in the production environment on the actual system" [6]. The majority of the surveyed works perform online functional testing.

4.2.7 *Impact and Cost.*

According to [44], the impact "describes the scope of after effects", and concerns the "execution time, required resources, and complexity of adaptation actions". This facet refers to what and how the adaptation action will be applied to. Table 10 shows the subcategories of the *Impact and cost* category. In this work, we make a distinction between works that measure the impact, cost, or overhead of using the approach and those that instead are not measuring them. The work in [44] takes into account both impact and cost and classifies adaptation actions into the *Weak* and *Strong* categories. We found that with the information available in the surveyed papers we were not able to provide a judgment on whether the impact and cost are weak or strong. We might conclude that impact and cost are important aspects to be considered when performing self-adaptive testing in the field. However, we did not find many approaches that seriously discuss them.

4.3 Challenges

In this section, we aim at answering RQ3, i.e., **what are the known gaps/challenges in self-adaptive field-based testing?**, and at revealing the main research gaps and challenges in SATF. We list some of them as follows.

4.3.1 *Uncertainty.*

One of the toughest problems in SATF is uncertainty. This difficulty may arise from the fact that the system under test might face different operating situations that are hard to predict [21], and such uncertainty is of course reflected in the testing system itself. When testing mobile applications, for example, the number of possible configurations can be exponential [11]. The work in [20] also emphasizes that adapting testing to a self-adaptive system as it reconfigures is challenging. Besides, at runtime, interactions that are not predicted could be found [34] and the testing approach should be able to handle them. A concrete example on the impact of uncertainty is provided in [22]. They show that changes in the requirements or in the environment may compromise the effectiveness of test cases and/or oracles. In their work, they also raise the problem of how to know when to test and what characteristics of the system should be monitored by the testing approach.

4.3.2 Overhead. The overhead in terms of memory, network, and execution time is the concern of several testing approaches [4]. However, in SATF, addressing this problem is even more imperative since the testing process may introduce some overhead to the SUT. It comes from the fact that self-adaptive testing may consume resources to perform adaptations and provoke an overhead that might harm the system's execution. To make such type of testing acceptable to the end-user, the overhead incurred should be minimized [11]. The approach in [11] analyzes the overhead imposed by the monitoring stage and concludes that in their case, it is unnoticeable to negligible. The work in [34] proposes an approach to test dynamic and distributed systems and assesses that the overhead, in terms of execution time and memory consumption, is relatively low and tolerable, mainly if dynamic adaptations are not commonly requested. The size of test sequences may be an important factor to evaluate the overhead. Small test sequences may result in a low overhead, while large test sequences may impact considerably the system execution [18]. The approach proposed in [22] states the importance of the balance between maximizing test coverage and minimizing test overhead. In their approach, the schedule for the test execution is organized in a way that testing does not affect negatively the system performance. This balance between the gain obtained and the cost incurred is left as future work for the approach proposed in [26]. Considering the overhead in terms of memory consumption and execution time, the work in [20] concludes that for their SATF approach, a significant impact on the system execution time was observed, while memory is insignificantly affected. Finally, the work in [27] reports an increase in the response time to the user due to an overhead that they are not able to handle.

4.3.3 Human Intervention. SATF approaches may need, at some point, manual intervention. In an ideal scenario, the testing approach should be fully self-adaptive. Generally speaking, it reduces costs related to human intervention, not only financially, but also in terms of computational resources. However, this is not the scenario of several approaches [11, 14, 28, 36]. In this sense, many self-adaptive testing approaches still require a certain degree of human intervention. In the testing of critical systems, for example, this intervention may even be crucial. However, for some approaches, it may be important to reduce human intervention, as it can involve costs that can be avoided.

4.3.4 Test Isolation. A well-known issue for any field testing approach is test isolation as stated in [6]. This term means that the testing process should not be intrusive, that is, "should not interfere with the processes running in production and their data" [6]. This challenge may become even more important in SATF, since the adaptations performed in the testing process may result in unexpected access to parts of the system that are critical. Approaches to testing in the field must provide a strategy to minimize the system's sensitivity to its execution at runtime and this leads to a reduction of potential impacts on the system and environment [18]. The system's sensitivity describes which operations, that are part of the testing, when performed interfere with the running system or the environment in an undesirable way [25]. The work in [22] isolates particular test cases to prevent failures from affecting the real system's operation. Also, new bugs may emerge as a result of runtime adaptations of component-based systems, which may result in malfunctions and drive its execution to an unsafe state [34]. Although the approach proposed in [26] does not face this challenge, it mentions that online testing sessions can have an impact on how the services work or in some circumstances jeopardize the correct operation of the service. The work in [36] affirms that when in the production environment, a test workflow should be conducted in such a way that the operational database is not impacted. Some examples of this undesirable impact are the modification of mission-critical data or sharing them with unauthorized people. Finally, the work in [11] makes use of managed profiles[2] to isolate the runtime testing session from the normal user session.

4.3.5 Other Challenges. The previous subsections summarize considerably the key challenges when performing SATF. However, it is important to mention that other more specific and minor challenges related to the SATF were also found but they are not detailed here. This list includes constraints arising from the specific techniques employed [14, 28] (e.g. with respect to the complexity of data type handled), strategies to make test events indistinguishable from normal events [30], proper reaction to identified malfunctions [30], provisions for reliable traceability between test cases and requirements [22], specification of adaptation constraints [22], etc.

5 DISCUSSION AND FINAL REMARKS

We presented the first review of literature on SATF, i.e, testing approaches that *i)* are conducted in the field, and *ii)* undergo different kinds of adaption to face uncertainty, changing requirements, or evolution. Based on a final set of 16 primary studies, we provided a tentative definition of SATF concepts, a taxonomy of its characteristics mapped onto the selected papers, and a discussion of the main challenges.

5.1 Retrospective Discussion

Our study evidenced - as we expected - that the topic of SATF is still immature and not yet recognized by software testing researchers as a self-standing emerging discipline. One immediate sign of this is of course the low number of papers we could eventually select, which contrasts against the wealth of challenges to solve and the clear need for adaptive techniques to test an application in its operative environment. Other signs of immaturity also emerge from our analysis of the selected papers: there exists neither a common terminology nor a shared structuring of the papers. One would expect for example that if a test approach is proposed for execution in the field, its overhead should be at least considered, if not measured, but very few of the selected papers did so, with the large majority not even mentioning the argument.

Another observation that is worth attention is the ambiguity of the term "online", which was also included in our search query. After having selected 36 papers based on title and abstract, we eventually reduced the set to 16, because many of the 20 excluded papers used online testing with the meaning of deriving the next test cases based on the results of the previously executed ones, but were not addressing field testing. Such ambiguity was already

[2] https://source.android.com/devices/tech/admin/managed-profiles

recognized in [6] (see their exclusion criterion 4), but could even become more complicated by considering that in some approaches also the adaptation could be interleaved with test execution.

Of course, being SATF approaches a subclass of the broader topic of field testing previously surveyed by Bertolino et al. in [6], they inherit all the characteristics, concerns, and challenges already widely discussed in that SLR. In this paper, we wanted rather focus on the additional aspects, issues, and opportunities as well, stemming from self-adaptation capability, and for this reason, our taxonomy for RQ2 was mostly inspired by the ones adopted in the SAS community. However, while proceeding with the analysis, we realized that a self-adaptive testing system may expose -luckily- some simplifications with respect to a generic concept of a SAS. For example, almost all of the studies we analyzed are close-adaptive and centralized, and this can make their development easier.

A clear demarcation descends from the nature of the application under test. Eleven[18, 20–22, 27, 28, 30, 34, 35, 38, 43] out of the 16 selected studies targeted self-adaptive systems, and this is easily explainable when testing such types of systems. The need for testing approaches that are themselves adaptive (at the very least for adjusting the test cases to the new system version) is immediately evident. Nevertheless, as we anticipated in the introduction, SATF is not exclusively conceived for SAS, and few works demonstrate that it can be useful also for field testing of not adaptive systems, for example for mobile [11] or service-oriented [14] applications. We think indeed that the need for self-adaptation should be a first-class citizen for any field-testing approach, and we expect that the growing attitude toward continuing testing after deployment will automatically also increase the attention towards SATF concerns.

5.2 Limitations and Threats to Validity

This study only aimed at providing a first characterization of SATF, which is a topic not yet well-established in the literature, to understand its relevance and comprehension among the authors that propose such kinds of approaches. As such, in this scoping review, we relaxed some of the usually adopted inclusions and exclusion criteria in similar studies, for the sake of collecting visions and definitions on the topic, even if not supported by rigorous validation. For this reason, in our set of primary studies, we also include a couple of opinion papers, which could provide information not supported by scientific experimentation. Also, having only two authors' opinions on a paper could be seen as a threat, but as mentioned this is mitigated by the fact that we discussed all papers that we were not crystal-clear about the inclusion during plenary meetings.

Another possible threat to validity is lack of completeness: our review is based on a rigorous search over three popular repositories, but we have not (yet) completed the collection process with a snowballing cycle. The reason is that the process of selecting the final set of studies has been very effort-prone, requiring careful reading and several discussions among the authors for reaching a consensus, also due to the above-mentioned lack of shared terminology which hindered comprehension. In other terms, in this study we prioritized depth vs. breadth: we might have missed some additional approaches, but for the purpose of a scoping review we consider a more thoughtful analysis of concepts more useful.

5.3 Future Research Directions

We can hint at various relevant research directions for progressing the SATF topic. Our review extracted several open challenges from the studies analyzed, as we discuss in Section 4.3, and consequently of course future research in SATF should investigate approaches that can help address those challenges.

In addition to these challenges, throughout the development of this research, we noticed that some important characteristics of SATF approaches that are present in our taxonomy are not completely addressed by the works found. Therefore, these characteristics can still be seen as open challenges and they are described in the following of this section. First, we emphasize the need to establish trust in the testing results. Considering a test technique that can autonomously change its approach and/or test cases, how can reliance be posed on the test outcome? Probably together with a self-adaptation strategy, the approach should also account for a self-evaluation following own changes. With the pervasiveness of critical software systems, we see this as a crucial research challenge.

Another aspect that we did not encounter was oracle adaptation, although this seems a necessity. The oracle problem remains a tough challenge for any testing approach, but when the test cases have to change based on uncertainty or context-dependence, their oracle should adapt as well. Surprisingly, with the only exception of [18], this evident need was not explicitly considered in any of the surveyed studies. The question of how to ensure that the test oracle remains up-to-date in the face of evolution remains open.

On a different plane, we can think of challenges for monitoring the context and the application under test in lightweight and unobtrusive mode, for understanding when the test strategy needs to be updated. So far, in fact, we have only found SATF approaches whose adaptation is statically decided. How could we conceive dynamic adaptation of field testing, so to autonomously decide when the test strategy is no longer effective or valid? Such challenge is somehow related to uncertainty, already mentioned in Section 4.3.1.

The technical aspects beyond the conceptual design of a SATF approach also will require considerable attention. We already mentioned the need to limit the overhead of tools implementing SATF. Indeed, SATF is itself applying a MAPE loop, and as such requires a complex architecture for implementing its components.

Finally, in view of the newness of the topic, and of the many open challenges along with different directions, we close with an invitation: the most rapid way to attract awareness and establish a common vocabulary could be the organization of a dedicated workshop or seminar, in the “Dagstuhl style”. This could perhaps stem even from the testing researchers within the SEAMS community.

ACKNOWLEDGMENTS

This paper has been partially supported by the Italian MIUR PRIN 2017 Project: SISMA (Contract 201752ENYB).

REFERENCES

[1] Algirdas Avizienis, J-C Laprie, Brian Randell, and Carl Landwehr. 2004. Basic concepts and taxonomy of dependable and secure computing. *IEEE transactions on dependable and secure computing* 1, 1 (2004), 11–33.

[2] Morena Barboni, Antonia Bertolino, and Guglielmo De Angelis. 2021. What We Talk About When We Talk About Software Test Flakiness. In *Quality of Information and Communications Technology*, Ana C. R. Paiva, Ana Rosa Cavalli,

Paula Ventura Martins, and Ricardo Pérez-Castillo (Eds.). Springer International Publishing, Cham, 29–39.

[3] Luciano Baresi and Carlo Ghezzi. 2010. The disappearing boundary between development-time and run-time. In *Proceedings of the Workshop on Future of Software Engineering Research (FoSER)*.

[4] Antonia Bertolino. 2007. Software testing research: Achievements, challenges, dreams. In *Future of Software Engineering (FOSE'07)*. IEEE, 85–103.

[5] Antonia Bertolino, Guglielmo De Angelis, Sampo Kellomaki, and Andrea Polini. 2012. Enhancing Service Federation Trustworthiness through Online Testing. *Computer* 45, 1 (2012), 66–72. https://doi.org/10.1109/MC.2011.227

[6] Antonia Bertolino, Pietro Braione, Guglielmo De Angelis, Luca Gazzola, Fitsum Kifetew, Leonardo Mariani, Matteo Orrù, Mauro Pezzè, Roberto Pietrantuono, Stefano Russo, et al. 2021. A Survey of Field-based Testing Techniques. *ACM Computing Surveys (CSUR)* 54, 5 (2021), 1–39.

[7] Antonia Bertolino, Guglielmo De Angelis, and Andrea Polini. 2012. Governance policies for verification and validation of service choreographies. In *International Conference on Web Information Systems and Technologies*. Springer, 86–102.

[8] Antonia Bertolino and Paola Inverardi. 2019. *Changing Software in a Changing World: How to Test in Presence of Variability, Adaptation and Evolution?* Springer International Publishing, Cham, 56–66. https://doi.org/10.1007/978-3-030-30985-5_5

[9] Kai-Yuan Cai. 2002. Optimal software testing and adaptive software testing in the context of software cybernetics. *Information and Software Technology* 44, 14 (2002), 841–855. https://doi.org/10.1016/S0950-5849(02)00108-8

[10] Kai-Yuan Cai, João W Cangussu, Raymond A DeCarlo, and Aditya P Mathur. 2003. An overview of software cybernetics. In *Eleventh Annual International Workshop on Software Technology and Engineering Practice*. IEEE, 77–86.

[11] Mariano Ceccato, Davide Corradini, Luca Gazzola, Fitsum Meshesha Kifetew, Leonardo Mariani, Matteo Orrù, and Paolo Tonella. 2020. A Framework for In-Vivo Testing of Mobile Applications. In *2020 IEEE 13th International Conference on Software Testing, Validation and Verification (ICST)*. IEEE, 286–296.

[12] Betty H Cheng, Rogério Lemos, Holger Giese, Paola Inverardi, Jeff Magee, Jesper Andersson, Basil Becker, Nelly Bencomo, Yuriy Brun, Bojan Cukic, et al. 2009. Software Engineering for Self-Adaptive Systems: A Research Roadmap. In *Software Engineering for Self-Adaptive Systems*. 1–26.

[13] Heather L. Colquhoun, Danielle Levac, Kelly K. O'Brien, Sharon Straus, Andrea C. Tricco, Laure Perrier, Monika Kastner, and David Moher. 2014. Scoping reviews: time for clarity in definition, methods, and reporting. *Journal of Clinical Epidemiology* 67, 12 (2014), 1291–1294. https://doi.org/10.1016/j.jclinepi.2014.03.013

[14] Mark B Cooray, James H Hamlyn-Harris, and Robert G Merkel. 2014. Dynamic test reconfiguration for composite web services. *IEEE Transactions on Services Computing* 8, 4 (2014), 576–585.

[15] Rogério de Lemos, Holger Giese, Hausi A. Müller, Mary Shaw, Jesper Andersson, Marin Litoiu, Bradley Schmerl, Gabriel Tamura, Norha M. Villegas, Thomas Vogel, Danny Weyns, Luciano Baresi, Basil Becker, Nelly Bencomo, Yuriy Brun, Bojan Cukic, Ron Desmarais, Schahram Dustdar, Gregor Engels, Kurt Geihs, Karl M. Göschka, Alessandra Gorla, Vincenzo Grassi, Paola Inverardi, Gabor Karsai, Jeff Kramer, Antónia Lopes, Jeff Magee, Sam Malek, Serge Mankovskii, Raffaela Mirandola, John Mylopoulos, Oscar Nierstrasz, Mauro Pezzè, Christian Prehofer, Wilhelm Schäfer, Rick Schlichting, Dennis B. Smith, João Pedro Sousa, Ladan Tahvildari, Kenny Wong, and Jochen Wuttke. 2013. *Software Engineering for Self-Adaptive Systems: A Second Research Roadmap*. Springer Berlin Heidelberg, Berlin, Heidelberg, 1–32. https://doi.org/10.1007/978-3-642-35813-5_1

[16] Ismayle de Sousa Santos, Magno Luã de Jesus Souza, Michelle Larissa Luciano Carvalho, Thalisson Alves Oliveira, Eduardo Santana de Almeida, and Rossana Maria de Castro Andrade. 2017. Dynamically Adaptable Software Is All about Modeling Contextual Variability and Avoiding Failures. *IEEE Software* 34, 6 (2017), 72–77. https://doi.org/10.1109/MS.2017.4121205

[17] Simon Dobson, Spyros Denazis, Antonio Fernández, Dominique Gaïti, Erol Gelenbe, Fabio Massacci, Paddy Nixon, Fabrice Saffre, Nikita Schmidt, and Franco Zambonelli. 2006. A Survey of Autonomic Communications. ACM Transactions on Autonomous and Adaptive Systems. (2006).

[18] Erick Barros dos Santos, Rossana MC Andrade, and Ismayle de Sousa Santos. 2021. Runtime testing of context-aware variability in adaptive systems. *Information and Software Technology* 131 (2021), 106482.

[19] Brian Fitzgerald and Klaas-Jan Stol. 2017. Continuous software engineering: A roadmap and agenda. *Journal of Systems and Software* 123 (2017), 176–189.

[20] Erik M Fredericks and Betty HC Cheng. 2015. Automated generation of adaptive test plans for self-adaptive systems. In *2015 IEEE/ACM 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*. IEEE, 157–167.

[21] Erik M Fredericks, Byron DeVries, and Betty HC Cheng. 2014. Towards runtime adaptation of test cases for self-adaptive systems in the face of uncertainty. In *Proceedings of the 9th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*. 17–26.

[22] Erik M Fredericks, Andres J Ramirez, and Betty HC Cheng. 2013. Towards runtime testing of dynamic adaptive systems. In *2013 8th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*. IEEE, 169–174.

[23] Luca Gazzola, Leonardo Mariani, Fabrizio Pastore, and Mauro Pezze. 2017. An exploratory study of field failures. In *2017 IEEE 28th International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 67–77.

[24] John C Georgas, André van der Hoek, and Richard N Taylor. 2005. Architectural runtime configuration management in support of dependable self-adaptive software. *ACM SIGSOFT Software Engineering Notes* 30, 4 (2005), 1–6.

[25] Alberto González, Eric Piel, and Hans-Gerhard Gross. 2009. A model for the measurement of the runtime testability of component-based systems. In *2009 International Conference on Software Testing, Verification, and Validation Workshops*. IEEE, 19–28.

[26] Antonio Guerriero, Raffaela Mirandola, Roberto Pietrantuono, and Stefano Russo. 2019. A hybrid framework for web services reliability and performance assessment. In *2019 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*. IEEE, 185–192.

[27] Elaheh Habibi and Seyed-Hassan Mirian-Hosseinabadi. 2021. On-demand Test as a Web Service Process (OTaaWS Process). In *2021 7th International Conference on Web Research (ICWR)*. IEEE, 16–23.

[28] Joachim Hänsel and Holger Giese. 2017. Towards collective online and offline testing for dynamic software product line systems. In *2017 IEEE/ACM 2nd International Workshop on Variability and Complexity in Software Design (VACE)*. IEEE, 9–12.

[29] Henner Heck, Christian Gruhl, Stefan Rudolph, Arno Wacker, Bernhard Sick, and Joerg Haehner. 2016. Multi-k-Resilience in Distributed Adaptive Cyber-Physical Systems. In *ARCS 2016; 29th International Conference on Architecture of Computing Systems*. 1–8.

[30] Henner Heck, Stefan Rudolph, Christian Gruhl, Arno Wacker, Jörg Hähner, Bernhard Sick, and Sven Tomforde. 2016. Towards autonomous self-tests at runtime. In *2016 IEEE 1st International Workshops on Foundations and Applications of Self* Systems (FAS* W)*. IEEE, 98–99.

[31] Markus C Huebscher and Julie A McCann. 2008. A survey of autonomic computing—degrees, models, and applications. *ACM Computing Surveys (CSUR)* 40, 3 (2008), 1–28.

[32] Christian Krupitzer, Felix Maximilian Roth, Sebastian VanSyckel, Gregor Schiele, and Christian Becker. 2015. A survey on engineering approaches for self-adaptive systems. *Pervasive and Mobile Computing* 17 (2015), 184–206.

[33] Mariam Lahami and Moez Krichen. 2021. A survey on runtime testing of dynamically adaptable and distributed systems. *Software Quality Journal* (2021), 1–39.

[34] Mariam Lahami, Moez Krichen, and Mohamed Jmaiel. 2016. Safe and efficient runtime testing framework applied in dynamic and distributed systems. *Science of Computer Programming* 122 (2016), 1–28.

[35] Lucas Leal, Andrea Ceccarelli, and Eliane Martins. 2019. The SAMBA approach for Self-Adaptive Model-BAsed online testing of services orchestrations. In *2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC)*, Vol. 1. IEEE, 495–500.

[36] Majid Makki, Dimitri Van Landuyt, and Wouter Joosen. 2016. Automated workflow regression testing for multi-tenant saas: integrated support in self-service configuration dashboard. In *Proceedings of the 7th International Workshop on Automating Test Case Design, Selection, and Evaluation*. 70–73.

[37] Julie A McCann, Rogeŕio De Lemos, Markus Huebscher, Omer F Rana, and Andreas Wombacher. 2006. Can self-managed systems be trusted? some views and trends. *The Knowledge Engineering Review* 21, 3 (2006), 239–248.

[38] Andreas Metzger, Eric Schmieders, Osama Sammodi, and Klaus Pohl. 2012. Verification and testing at run-time for online quality prediction. In *2012 First International Workshop on European Software Services and Systems Research-Results and Challenges (S-Cube)*. IEEE, 49–50.

[39] Zachary Munn, Micah DJ Peters, Cindy Stern, Catalin Tufanaru, Alexa McArthur, and Edoardo Aromataris. 2018. Systematic review or scoping review? Guidance for authors when choosing between a systematic or scoping review approach. *BMC medical research methodology* 18, 1 (2018), 1–7.

[40] Christian Murphy, Gail Kaiser, Ian Vo, and Matt Chu. 2009. Quality assurance of software applications using the in vivo testing approach. In *2009 International Conference on Software Testing Verification and Validation*. IEEE, 111–120.

[41] Micah Peters, Christina Godfrey, Hanan Khalil, Patricia McInerney, Deborah Parker, and Cassia Baldini Soares. 2015. Guidance for conducting systematic scoping reviews. *International Journal of Evidence-Based Healthcare* 13, 3 (2015), 141 – 146. https://doi.org/10.1097/XEB.0000000000000050

[42] Roberto Pietrantuono, Stefano Russo, and Antonio Guerriero. 2018. Run-time reliability estimation of microservice architectures. In *2018 IEEE 29th International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 25–35.

[43] Y Mohana Roopa and M Ramesh Babu. 2017. Self-test framework for self-adaptive software architecture. In *2017 International conference of Electronics, Communication and Aerospace Technology (ICECA)*, Vol. 2. IEEE, 669–674.

[44] Mazeiar Salehie and Ladan Tahvildari. 2009. Self-adaptive software: Landscape and research challenges. *ACM transactions on autonomous and adaptive systems (TAAS)* 4, 2 (2009), 1–42.

[45] Bento Rafael Siqueira, Fabiano Cutigi Ferrari, Marcel Akira Serikawa, Ricardo Menotti, and Valter Vieira de Camargo. 2016. Characterisation of challenges for testing of adaptive systems. In *Proceedings of the 1st Brazilian Symposium on Systematic and Automated Software Testing*. 1–10.

[46] Bento R Siqueira, Fabiano C Ferrari, Kathiani E Souza, Daniel SM Santibáñez, and Valter V Camargo. 2020. Fault Types of Adaptive and Context-Aware Systems and Their Relationship with Fault-based Testing Approaches. In *2020 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*. IEEE, 284–293.