FROM THE ART OF SOFTWARE TESTING TO TEST-AS-A-SERVICE IN CLOUD COMPUTING

Janete Amaral, Alberto S. Lima, José Neuman de Souza, Lincoln S. Rocha

MDCC, Universidade Federal do Ceará, Fortaleza/CE – Brasil

ABSTRACT

Researchers consider that the first edition of the book "The Art of Software Testing" by Myers (1979) initiated research in Software Testing. Since then, software testing has gone through evolutions that have driven standards and tools. This evolution has accompanied the complexity and variety of software deployment platforms. The migration to the cloud allowed benefits such as scalability, agility, and better return on investment. Cloud computing requires more significant involvement in software testing to ensure that services work as expected. In addition to testing cloud applications, cloud computing has paved the way for testing in the Test-as-a-Service model. This review aims to understand software testing in the context of cloud computing. Based on the knowledge explained here, we sought to linearize the evolution of software testing, characterizing fundamental points and allowing us to compose a synthesis of the body of knowledge in software testing, expanded by the cloud computing paradigm.

KEYWORDS

Cloud Computing; Software Testing; Test-as-a-Service.

1. INTRODUCTION

Cloud Computing is a promising technology that provides computational storage, network resources, and data, employing virtualization technology at the infrastructure layer [01]. Cloud computing is a new frontier of computing that will finally fulfill the dream of providing computing services as a utility [02].

This new model has been changing the way of managing and delivering services in computing, bringing opportunities and new challenges. One of these challenges is Software Testing, known as Test-as-a-Service (TaaS), which has become a topic frequently addressed in different research communities [03]. In TaaS, a service provider performs testing activities for web-based software on a cloud infrastructure and delivers them to customers.

As more applications are moved to the cloud world and more cloud-based systems are developed, there will be an increasing demand for testing services in cloud environments. This implies that many quality assurance professionals need to understand the needs and challenges of testing as a service [04].

Even though several publications discuss architectures, technologies, and management of TaaS [05], this topic has been frequently discussed in the Cloud Computing and Software Engineering communities. The most addressed issues in TaaS are [03] (GAO *et al.*, 2013):

- What is TaaS in the Cloud Computing paradigm?
- What are the scope, motivations, requirements, infrastructure, and unique features of TaaS?
- What processes, environments, and services are provided?
- What are the essential practices, tools, and suppliers?
- What are the differences between conventional software testing services and testing on cloud infrastructure?
- What are the problems, challenges, and special needs?

This article reports research that investigates the integration of software testing requirements into applications provided in the Cloud Computing architecture. The contributions of this article relate to Test as a Service (TaaS). With these goals, this paper presents one TaaS overview, covering the testing of applications migrated to the cloud and those that use the cloud to test traditional applications.

This paper is divided into five sessions to achieve its objective. Session 1 contextualizes the theme of this work. Session 2 presents the background on Software Testing. The idea is to provide, in general terms, the basic concepts that allow the reader to understand the area of Software Testing. For a greater depth, SWEBOK - Software Engineering Body of Knowledge is referenced [06]. Session 3 presents an overview of Automation in Software Testing. Test as a Service is only possible with advances in test automation. Session 4 is the core of thisarticle, in which the specific concepts of testing in a cloud environment are covered as the main remaining issues in TaaS. Finally, the purpose of Session 5 is to conclude the work, presenting considerations about this journey from the "Art of Software Testing" initiated by Glenford Myers [07] to "Test as a Service" of the present day, identifying gaps, as well as research opportunities.

2. SOFTWARE TESTING

Currently, the software handles terabytes of data and requires much time to manually operated. As a result, software testing has come to demand an appropriate environment that is not accessible to all organizations. Therefore, there is a need for a specific service that can facilitate the user in testing his software [08].

To identify the understanding of the topic of Software Testing is necessary to explain its definition. According to SWEBOK [06], "Software Testing is a *dynamic* verification of the behavior of a program in a *finite* set of test cases properly *selected*, from an infinite domain of executions, against *expected* behavior." In this definition, the words in italics correspond to the critical questions in the description of the Software Test knowledge area, namely:

• **Dynamic** - Testing always involves running the program with selected inputs. The input value alone is not always enough to specify a test, as a complex and non-deterministic system can react to the same information with several different behaviors depending on the system. Static techniques are different and complementary to dynamic tests.

• **Finite** - Even in the case of low-complexity programs, so many test cases are theoretically possible that exhaustive tests can take months or years to run. Therefore, a complete set of tests can be considered infinite in practice. Tests are conducted on a subset of all possible tests, determined by risk and prioritization criteria. Testing always implies a trade-off between limited resources on the one hand and inherently unlimited testing requirements on the other.

• **Selected** - Existing testing techniques essentially differ in how the test suite is selected. Identifying the most suitable selection criteria under certain conditions is a complex problem.

Software engineers must know that different selection criteria can produce different degrees of effectiveness.

• **Expected -** It should be possible, although not always easy, to decide whether the observed test results are acceptable. The observed behavior can be verified with the user's needs concerning a specification.

Each test cycle is considered sufficient to find all defects in software. It is desirable to have a systematic approach to minimize the number of cycles and create an application whose behavior is predictable [09]. Otherwise, the testing effort is useless.

One of the significant concerns in software testing is the identification of **Software Testability**. This term has two distinct but related meanings: on the one hand, it refers to the ease with which a given test coverage criterion can be satisfied; on the other hand, it is defined as the probability that a set of test cases exhibit a software failure [06]

According to MYERS *et al.* [07], some of the most important considerations about Software Testing are psychological issues. One can identify a set of vital testing principles or guidelines. Several of these principles may seem obvious, but they are often overlooked. Table 01 summarizes these essential principles.

Table 01. Software Testing Principles [07]

1.	A necessary part of a test case is the definition of the output or its expected result.
2.	A programmer should avoid trying to test his program.
3.	A programming company should not test its programs.
4.	Any testing process must include a thorough inspection of the results of each test.
5.	Test cases must be written for input conditions that are invalid and unexpected, as well as those that are valid and expected.
6.	Examining a program to see if it doesn't do what it's supposed to is, only half the battle; the other half is seeing if the program does what it shouldn't.
7.	Avoid throwaway test cases unless the program is a throwaway program.
8.	Refrain from planning a test effort under the implicit assumption that no errors will be found.
9.	The probability of having more errors in a program section is proportional to the number of errors already found in that section.
10.	Testing is a highly creative and intellectually challenging task.

2.1. Quality Assurance and Software Testing

The difference between Quality Assurance and Software Testing is only sometimes observed. To elucidate this discussion is necessary to distinguish what is meant by software quality and standards. It is understood that the quality of a product or service complies with standards maintained by certifying agencies. In the case of software development, there is a quality measurement for the **Software Development Process** (SDP) and a quality measurement for the Software Product produced in the SDP. o build a good software product, the SDP must also be of good quality. Therefore, each process activity must comply with standards so that the effect produced by this process has good quality. Even with adopting quality processes, software product defects will still exist. Software Testing definitions can detect and correct these defects before they reach your users. Quality for an SDP means preventing defects in the software product, and testing software means detecting and correcting defects. Together, these two processes ensure a good quality software product [06].

2.2. Body of Knowledge in Software Testing

To meet the growing demands and match the quality required by the SDP, producing a successful application can become a real challenge. The problem can be even worse by the need to deliver projects on time. Therefore, companies risk releasing an unstable product or even leaving more common errors, such as entering an invoice date or a customer's birth year. Software testing optimizes your company's resource management by preventing problems from appearing later. Software Testing is part of the software development process [06] and can be done by developers or specialized professionals. The procedure aims to anticipate and correct failures that would appear to the end user.

Although it seems simple, its use is essential to avoid the "putting out fires" phenomenon. Different tests are used to prevent surprises and ensure that the system works as planned.

2.2.1. Software Test Levels

Tests are performed at different times in the SDP to identify failures. Your target can range from a single module, a group of modules, or an entire system. These moments are identified at four levels. One of these levels is considered more important than the other [10]. The following are each of the software testing levels:

- **Unit Test** – At this level, the objective is to know how users access the system. t is necessary to test all combinations of input data. For the test to be successful, the result must come out as expected by the user, as specified in the system requirements.

- **Integration Test -** This test analyzes the integration between the different units that make up the system. Aspects of the interface and the dependencies between the components are investigated.

- **Systems Test** – This test is a phase of the process in which the system, already fully integrated, is verified against its requirements in an environment like the production one.

- **User Acceptance Test** – After carrying out the planned set of tests, it will be necessary to perform the user acceptance test. Before launching the product, it is essential to present a beta version to the user to make the last adjustments so that the software is successful when it is commercialized.

2.2.2. Software Testing Techniques

The goal of software testing is to detect as many failures as possible. Many techniques have been developed with this intention. These techniques seek to "break" a program, being as systematic as possible in identifying inputs that will produce expected behaviors, for example, considering input domain subclasses, scenarios, states, and data flow.

ISO/IEC/IEEE 29119-4 [11] classifies testing techniques as specification-based, framework-based, and experience-based. In an approach related to the internal visibility of the item to be tested, there are white box (or glass box) techniques if the tests are based on information about how the software was coded. Or black box if the test cases only depend on the input/output of the software behavior. Then the technique uses multiple approaches, called a gray box [06].

2.2.3. Purpose of Software Testing

The test aims to verify software properties. Its proposition is based on specific objectives, which are stated to varying degrees of precision. Defining test objectives in detail helps to measure and control the testing process. Test cases are designed to verify that the functional specifications are implemented correctly, referred to in the literature as conformance, accuracy, or functional testing. However, several other properties must also be tested - including performance, reliability, usability, and reliability, among others [06]. There is a diversified denomination for the objectives of software testing. This lack of standardization provides varied interpretations, constituting a great difficulty for the apprentice in Software Testing [12].

2.2.4. Software Test Process

The development life cycle of a system plays a vital role in the software industry [08]. The cycle comprises several stages, with each step's output as the next step's input. Testing is a crucial moment in the system lifecycle.

The Software Testing Process [13] is represented in a structuring of phases, activities, artifacts, roles, and responsibilities whose objective is to standardize the work and maximize the organization and monitoring of test projects. Therefore, this process aims to guarantee the quality of the product. In addition, the testing process seeks to ensure that no critical step in the process is forgotten, that is, all activities are carried out correctly.

Successful management of testing activities depends on the software configuration management process used. The macro activities of this process are [06]:

1. **Planning -** Planning includes coordinating personnel, availability of facilities and equipment, creating and maintaining documentation, and planning for possible undesirable outcomes.
2. **Test Case Generation -** It is based on the testing level and the specific testing techniques.
3. **Environment Installation -** The environment must be compatible with the other adopted software engineering tools.
4. **Execution -** It must incorporate basic principles of scientific experimentation: everything must be executed and documented to replicate the results.
5. **Evaluation of Results -** Must determine if the test was successful. In most cases, “successful” means that the software worked as expected. Not all unexpected results are necessarily failures.
6. **Recording -** Activities should be recorded to identify when the test was performed, who performed it, what configuration was used, and other relevant information.
7. **Defect Tracking -** Defects should be tracked and analyzed to determine when they were introduced into the software and why they were created.

The Test Plan must be built to support the software testing process and is focused on ensuring the execution of the process [14]. Through this plan, the technical, functional, structural components, etc., will be verified and validated to guarantee the proper functioning of the program. The Test Plan aims to ensure the reliability and security of software, identifying possible errors and failures during its manufacture or even after. It must be planned with the software proposal, applied at each stage of the project, and not just at the end.

In the Software Development Process (SDP) discussion, it is vital to consider the offshoring model of software projects [15]. Offshoring has brought a new dimension to SDP, proposing changes in many aspects of software development. To broaden their competition and win more business, companies seek to ensure that distance, culture, and time zone restrictions are balanced with delivering productivity, quality, and costs to their customers. Offshoring service providers adopt mature software engineering processes. In Offshoring, robust procedures are followed, ensuring a substantial reduction in software defects.

2.2.5. Software Testing Maturity Model

The Test Maturity Model Integration (TMMi) model was developed as a reference framework for testing process improvement [16]. CMMi is a complementary model to the Capability Maturity Model Integration (CMMi), addressing specific issues for test managers, test engineers, and software quality professionals [17]. Testing, according to TMMi, is applied in its broadest sense to encompass all activities related to the quality of software products.

The TMMi model contains levels an organization passes through as its testing process evolves from an unmanaged (ad hoc) group to a managed, defined, measured, and optimization mode level. Achieving each class ensures that a proper improvement is established as the basis for the next stage.

3. Software Test Automation

Test automation has become the core of the quality control transformation. It is observed that there is more and more talk about automation at all stages of the QA lifecycle and not just execution but how to do it intelligently.

Several tools are on the market for many of the tasks associated with software testing. These tools include management, test case generation, test automation, defect tracking, etc. However, the automation tools show signs that the benefits still need to be fully realized. These tools depend on how the return is measured and communicated to stakeholders. Another observed issue is that tools are getting smarter, but test teams still need to be sufficiently empowered to take full advantage of them [18].

Overall, organizations are working to advance the quality of their products, despite the increasing complexity of the business processes they support, and are achieving greater control and transparency in their Quality Assurance (QA) activities.

RAFI *et al.* [19], in their systematic review of Automated Software Testing (AST), point out that 45% of the professionals surveyed consider that the current AST tools do not meet their needs. The limitations perceived by professionals in AST should outline essential directions for future research.

3.1. Benefits of Test Automation

As it is a repetitive process that demands attention, automation helps to perform better-quality work, considering that the tests are executed numerous times. The manual execution of the same procedure exhaustively can cause distraction and negatively influence the result. An overview, the following benefits can be highlighted:

- **Faster feedback -** When the business needs a quick response, automation allows for continuous delivery in an efficient way. This benefit is visibly perceived when there is short-term planning to be fulfilled.
- **Regression testing performance -** In some situations, this type of testing can be frequent, especially when it is necessary to find flaws in the development. The problem is that even

correcting the failure between one change and another, the system may experience a drop in performance. n this case, automation makes the process more practical so that developers can more easily analyze which modification generated the new problem.

- **Time Saving -** Execution requires repeated data input each time it runs. n addition to avoiding incorrect data entry, the developer will not have to worry about placing the same base since the process is done automatically.

Table 02 presents a parallel between the traditional test (manual) and the automated test, emphasizing parameters in which differences are observed.

Table 02. Main differences between Automated Testing and Traditional Testing [26]

Parameter	**Automated Test**	**Traditional Test**
Processing time	The time is significantly less than in the manual approach.	Manual testing is time-consuming and human resource-consuming.
Exploratory Test	Automation does not allow random testing.	Exploratory testing is possible.
Initial investment	The initial investment is higher. However, the ROI is better in the long run.	The initial investment is comparatively less. OI is lower compared to long-term automation testing.
Reliability	By using tools and scripts, it becomes a reliable method. There is no fatigue.	It needs to be more accurate due to the possibility of human errors.
User Interface Change	Even for a trivial UI change, test scripts need to be modified to work as expected.	Small changes in a button's id, class, etc., would not prevent a manual tester from running.
Investment	Investment in tools and human resources is required,	Only necessary investment in human resources.
Test report visibility	All interested parties can log into the automation system and check the test execution results.	They are usually recorded in an Excel or a Word document, and the results are not readily available.
Human observation	It does not involve human consideration. Therefore, it can only guarantee ease of use and a positive customer experience.	It allows for human observation, which can help provide a user-friendly system.
Performance test	Performance tests - load, stress, peak testing, etc. must be tool tested.	Performance testing is not feasible manually
Parallel Execution	It can run on different operating platforms in parallel, reducing test execution time.	They can run in parallel, but they need to increase their human resources, which is expensive.
Batch test	Multiple test scripts can be bundled together for nightly execution.	They cannot be grouped.
Programming knowledge	Programming knowledge is essential.	There is no need for programming.
To set up	Requires a less complex test run setup.	We need to have a more straightforward test run setup
Engagement Model	Tools do engagement. Execution is precise, eliminating the possibility of boring the performer.	Repetitive execution can be tedious and error-prone.
Ideal approach	Useful when frequently executing the same set of test cases	Helpful when the test case only needs to be run a few times.
Build verification test	Useful for Build Verification Testing-BVT.	The Build Verification Test (BVT) is challenging and time-consuming.
Deadlines	It has zero risk of losing a predefined test.	You have a higher risk of missing

		the pre-set trial period.
Structure	They use structures such as Data Drive, Keyword, and Hybrid to accelerate the automation process.	They use structures such as Data Drive, Keyword, and Hybrid to accelerate the automation process.
Documentation	Act as a document that provides training value	Act as a document that includes training value
Test Design	Automated unit tests reinforce/drive test-driven development design.	Automated unit tests reinforce/drive test-driven development design.
DevOps	Help with build verification testing and are an integral part of DevOps	Help with build verification testing and are an essential part of DevOps
When to use it?	It is suitable for regression testing, performance testing, load testing, or highly repeatable functional test cases.	Suitable for regression testing, performance testing, load testing, or highly repeatable functional test cases.

3.2. Types of Tools

It is recorded the existence of a diverse set of software testing tools. It can distinguish into three categories: Vendor Tools, Open Source Tools, and Built-in Tools. Each tool can perform different activities and has different abilities to perform specific functions. They are considered the test tools environment [20].

3.2.1. Supplier Tools

These are tools developed for commercial use by a particular vendor, which sells licenses and provides technical support to its customers. The advantages of these tools lie in the fact that they have technical support when users make a specific complaint. It is updated frequently, and the documentation is always available. Disadvantages include the license cost and the support that is generally only provided for specific operating environments and browsers. Large suppliers usually sell them.

3.2.2. Open-Source Tools

They have source code available on the internet, it can be downloaded and used, but there is no technical support for Open-Source Tools. The advantages of using these tools are that there is no license fee, the source code can be modified (if necessary), and they support various popular operating and browser environments. Disadvantages are that open-source tools need reliable technical support, new features may need to be fixed, and generally, there is a smaller variety of help documents.

3.2.3. Internal Tools

Large companies develop test tools for internal use (private use), such as Microsoft, IBM, Oracle, etc.

3.3. Test Environment and Test Data Management

Test Environment Management (TEM) and Test Data Management (TDM) approaches are still evolving. Some organizations are adopting cloud-based technologies to address their TEM and TDM needs. Service and data virtualization techniques are becoming popular and are being used for TEM and TDM. Some organizations are moving to create a specialized shared services organization for TEM and TDM. It is also possible to observe that governance has emerged as a more significant challenge than technology in this area.

Test environments are critical to successful test automation. Several challenges affect the tasks involved in managing, developing, evolving, configuring, and automating test environments. These challenges are difficult to face, but at the same time, they are interesting because they are involved with other activities and functions in different application areas and research topics. Thus, promising solutions will be available from research in software engineering and quality management [20].

3.4. Automated Testing Without Code

Software development has undergone numerous changes with the advancement and growth of technology and the emergence of practices such as DevOps and Agile. o support these changes, Software Testing even extends to automated testing without code.

The World Quality Report 2020–2021, presented by the companies Capgemini, Sogeti, and Micro Focus [18], reports that not having the know-how and experience in advanced programming is considered one of the challenges when applying automated tests. The development of codeless testing tools is a promising solution to this challenge. Automated testing without code means running the automation test without using a script. Writing automated scripts can be challenging for quality control, as the code required to be written in each programming language is arduous and time-consuming. Running automated tests without code is not entirely code-free; however, there are many reasons why you should consider implementing codeless test automation in your software testing process. Several no-code testing tools on the market cover a set of built-in features without any complex coding.

4. CLOUD SOFTWARE TESTING

Software Testing has been identified as a priority area to migrate to the cloud environment. Virtualization [1] was initially used to create virtual computing resources with different operating systems (OS), to allow testing software on various platforms. Testing new software often requires servers, storage, and network devices to be used for a limited time. These computing resources are often underutilized after testing, incurring extra costs in the organization's budget.

Especially for some applications, Software Testing is resource intensive. s an example, to test the performance and scalability of a banking application, it must be exercised with millions of user requests in a short time. This scenario needs to be tested because many users use their bank accounts regularly at the end of the month. Reproducing such a scenario requires the provider to configure test equipment to emulate the actions of millions of users. Likewise, mobile app providers often have to maintain the quality of their services across many platform combinations. Today's computing platforms span multiple browsers, supported by different backends, running on multiple operating systems. Therefore, providers must test their services on multiple platforms to ensure reliable service.

One of the drivers of cloud computing adoption is the economy of scale. Cloud computing provides a pay-per-use type of service, thus eliminating, in many cases, the initial investment. Testing tools and services are no exception. Development teams can benefit from this paradigm for using test tools when and in the amount needed, saving license fees.

AKERELE *et al.* [21] present a classification of Cloud Testing in three categories: Cloud Testing Infrastructure-as-a-Service (CTIaaS), Cloud Testing Platform-as-a-Service (CTPaaS), and Cloud Testing Software-as-a-Service (CTSaaS). In this classification, the author uses the resources used by the customer at the provider (the entire infrastructure, the platform, or just the software) as a basis for his category. Table 03 shows each type's proposed cloud test types and service provider examples. A short description of these types follows.

Table 03. Classification of Cloud Testing [21].

CLOUD TESTING		
CTIaaS	**CTPaaS**	**CTSaaS**
Amazon Google API VMware	Microsoft Azure Force.com Soasta	JMeter LoadImpact

- **Cloud Testing Infrastructure-as-a-Service (CTIaaS) -** Provides secure internet access to storage, hardware, network components, and servers for testing and development purposes. The service provider owns the infrastructure and is generally hosted, managed, and maintained by the infrastructure. Customers pay for the infrastructure needed for testing, which the service provider supports solely for the customer's use. All major testing activities are done on the customer's website. Organizations have a high level of control over their instances, and this category is considered the safest, although it is the most capital-intensive. The ease and low cost of building and dismantling the server make this an attractive option for organizations concerned about their data security. Customer pricing is prepaid, with pricing varying linearly with the number of server instances and the software environment installed on them. Network traffic and the volume of data hosted on the server also influence pricing.

- **Cloud Testing Platform-as-a-Service (CTPaaS)** – Provides a platform for development teams to perform functional testing. This allows you to leverage Integrated Development Environments (IDEs) in the cloud to perform functional testing and edit test scripts for automation. CTPaaS providers provide a platform that spans application development, testing, and deployment. CTPaaS can be considered a complete platform for developing cloud computing systems. CTPaaS eliminates the need for substantial capital that would otherwise be required to set up a test/development environment by helping to deliver platform-specific configurations via the web browser interface. Therefore, the service provider's platform can be leveraged for software testing without investment. Cost implication and installation setup time can be eliminated. t also eliminates staffing concerns and the personnel expertise required to acquire and maintain the necessary infrastructure. CTPaaS allows users to select test requirements through the service provider's browser interface. n some cases, these tests must be written in the service provider's Domain-Specific Language (DSL). The main disadvantage is the power the service provider has to hold customers in providing the service due to the high cost of migrating to another provider/platform.

- **Cloud Testing Software-as-a-Service (CTSaaS)** – This is the most popular category of cloud testing and is often understood as the only option for cloud testing. In this category, tests run on browsers offered by service providers. Users can choose the operating system, browser type, version, and the number of concurrent users, as well as the geographic locations of simulated traffic. This category allows the testing of cloud applications by running the tests using the generated data traffic as input.

In the systematic literature review performed by BERTOLINO *et al.* [02], three different categories were identified in research carried out around the Cloud Test, namely:

● **Testing in the cloud** - is about leveraging scalable cloud technologies to validate non-cloud applications. This category includes testing solutions for different domains and platforms (mobile or web environments), which are validated by exploiting large-scale simulations and elastic capabilities offered by the cloud. The main benefits of these solutions consist of: (i) reducing costs through the exploitation of cloud computing resources, (ii) avoiding developing and maintaining test infrastructure, and (iii) performing on-demand testing services provided by third parties for the online validation of large-scale systems.

● **Testing of the cloud** – refers to validating the quality of applications and infrastructure deployed in the cloud. The focus is on the specific testing problems posed by systems residing in the cloud. Therefore, research reports in this category study the automatically provided cloud-based services and validate their performance, scalability, elasticity, and security. In addition, applications can be deployed on different clouds (e.g., private, public, or hybrid). Testing also focuses on compatibility and interoperability between heterogeneous cloud resources.

● **Testing of the cloud in the cloud** - refers to applications and infrastructure deployed in the cloud and tested by leveraging cloud platforms. Surveys in this category fill the intersection area between Testing of the cloud and Testing in the cloud.

In the rest of this session, the main characteristics of Test as Services (TaaS) will be presented, as well as the advantages and disadvantages of TaaS, extracted from research carried out by [05] in academic publications from 2008 to 2018.

4.1. Test-as-a-Service

GAO *et al.* [03] point out that the term "Testing-as-a-Service" (TaaS) was initially introduced in Denmark in 2009 by the company Tieto (http://www.tieto.com), and IBM nominated its respective TaaS solution for the Software Innovation Award 2009.

Recently, TaaS has received much attention in the academic and commercial communities for providing a scalable test environment, cost savings, utility-based service models, and on-demand test services. Many researchers, such as the following [05], [02], [22], [23], [24], have produced systematic literature reviews on terms inherent to TaaS and/or Cloud Testing in different periods. With this projection in the research, several definitions of TaaS emerged, some emphasizing different perspectives. Some of these definitions will be highlighted later in this presentation.

O TaaS is an outsourcing model in which software testing is performed by a third-party service provider rather than by employees of the organization. n TaaS, testing is done by a specialized service provider in a test environment, like the real world, to find defects in the software product.
TaaS is sometimes used when the company needs more skills or internal resources to perform specific tests and wants to keep its costs the same. One can also consider the TaaS use case when the company wants in-house developers not to influence the testing process results (which they could do if the tests were done in-house). With TaaS, the company can increase test execution speed while reducing the total time of software development.

CANDEA *et al.* [25] identify three service delivery models for TaaS:

- **TaaSC Certification Services** - Public certification that independently assesses software reliability, security, and protection.
- **TaaSH for End Users** - Demand test - home editions - for customers to review the software they want to install on their PCs or mobile devices.

- **TaaSD for Developers -** A programmer's sidekick service that fully allows developers to test code with a minimal initial investment.

Still, in their research, CANDEA *et al.* [25] point to different levels of use of TaaS. Each of them with different focus and objectives:

- **TaaS for cloud web-based software -** In this format, web-based software is deployed to a cloud and validated using testing services provided by TaaS providers. Its main objective is to take advantage of large-scale test simulations and elastic cloud computing capabilities.
- **Cloud TaaS -** In this model, cloud-based applications and SaaS systems are deployed, run, and validated on a cloud infrastructure or platform. Unlike the previous form, TaaS in the cloud must address and validate the scalability and multi-tenancy of SaaS systems. In addition, a SaaS system is usually built on specific cloud technologies, and its services, SaaS connectivity protocols, and APIs must also be validated.
- **TaaS over clouds** - SaaS applications that cross hybrid clouds are deployed and validated against different clouds (such as private and public clouds). n a hybrid cloud infrastructure, multiple test-on-demand services are provided and provided by a TaaS vendor.

CANDEA *et al.* [25] also record the possibility of using three types of cloud-based test environments:

- A private/public cloud test environment where vendors deploy SaaS applications on a private (or public) cloud to validate their quality.
- A cloud-based enterprise test environment where application vendors deploy web-based applications to a cloud to validate their quality on a cloud infrastructure. This cloud-based enterprise test environment is helpful for testing cloud-based enterprise service programs.
- In a hybrid cloud test environment, vendors deploy cloud-based applications on a hybrid cloud infrastructure to verify their quality.

TaaS takes advantage of cloud technology and solutions. TaaS on a cloud infrastructure makes it possible to deliver various products to the customer, like on-demand automated test execution, control services, and test project management services. All of them are defined based on well-defined service level agreements (SLAs) and predefined Quality of Service (QoS) standards.

4.1.1. Main Activities in TaaS

IT organizations may choose to outsource to TaaS service providers various types of Testing activities. The TaaS services offered are generally categorized into 03 classes, namely [26]:

- **Functional Testing as a Service**: TaaS Functional Testing can include UI/GUI Testing, Regression, Integration, and User Acceptance Testing, but it is unnecessary to do the functional part.

- **Performance Testing as a Service -** Performance testing services include analysis of user scenarios and transactions in a simulated production environment. TaaS mimics a real-world user environment by creating virtual users and performing load and stress testing. TaaS providers can run simulations with a predetermined number of transactions, users, and views to assess the performance of their application, under different server loads and conditions, with multiple users accessing the application simultaneously.

- **Security Testing as a Service**: The purpose of security testing is to identify potential vulnerabilities or attack vectors that could be exposed in a cyber-attack. TaaS scans applications and websites for any vulnerabilities. TaaS service provider uses its security analysis tools to assess an application's exposure and make recommendations to reduce remediation and the perception of security flaws.

Among the activities most frequently cited in the test as a service, there are the following:

- Provide a self-service portal for running functional and load tests for applications.
- Maintain a test library with security controls, saving all test assets to end users.
- Maximize hardware utilization, and facilitate sharing of cloud hardware resource pool while complying with security policies.
- Make complete testing available on-demand to labs, including the ability to deploy complex, multi-tiered applications, test scripts, and test tools.
- Ensure, from the monitoring, the detection of bottlenecks and the resolution of problems of the application under test
- Allow, from the measurement resources, to track and charge for services used by customers.

4.1.2. Using TaaS - Step by Step

After user scenarios are created, and testing is designed, service providers provide servers to generate virtual traffic. According to [29], the cloud testing lifecycle presents the following activities:

A. **Develop User Scenarios -** It is the actual requirement of organizations and users to ensure that different stakeholders verify the application.

B. **Design Test Cases** - A test case contains a set of actions performed on the application to verify its functionality. It is necessary to consider the inputs, the effort to be performed, and the respective output.

C. **Select a Cloud Service Provider** - Each organization has specific requirements and evaluation criteria. Based on these criteria, an organization selects the cloud provider. There are some common characteristics for evaluating service provider quality. These characteristics can help choose a service provider that will benefit an organization to deliver the best quality service for which the cloud is expected.

D. **Configure the Infrastructure** - When installing the cloud infrastructure, the architect must consider the critical requirement that needs to be addressed.

E. **Start Testing** - The test team or test administrator can proceed with the test.

4.2. Advantages and Disadvantages of using TaaS

TaaS is useful when testing an application requires extensive automation and a short test execution cycle. It is also considered appropriate to perform TaaS when the in-depth design or system knowledge is not needed or for resource-intensive tests. Among the possible TaaS benefits, the most cited are the following:

- Flexible test execution and test assets.
- Achieving a quick return on investment, eliminating the investment made after purchasing, managing, and maintaining hardware, licensing software, etc.
- Faster product delivery through rapid procurement, project setup, and execution.

- Assurance of data integrity and accessibility anytime and anywhere.
- Reduction of operational costs, maintenance costs, and investments.
- Pay as you go.

4.2.1. Main Advantages of using TaaS

GIRARDON *et al.* [05] identify an extensive list of possible advantages to be obtained in the use of TaaS, namely:

- **Accuracy -** TaaS will be able to simulate production servers more efficiently because the resources are in the cloud. The load on the servers will be able to simulate a real-world scenario, both from a scale perspective and a geographic distribution perspective. Simulating different browsers and platforms using cloud infrastructure is also very effective.
- **Cross-Platform Testing -** Analyzes application behavior in different environments. Cross-platform testing helps identify issues that may vary depending on platforms or settings, such as consistency, user interface, usability, and performance.
- **Cost-effective -** The main advantage of using the TaaS model is cost reduction. TaaS requires no investment in server configuration, tools, or operating systems, significantly reducing capital and depreciation costs. Substantial savings are also made in maintenance costs in locating and fixing defects in released software. Another cost reduction concerns the user only paying for the actual service and the time spent; this helps control costs, resulting in a better return on investment (ROI).
- **Less testing knowledge required -** TaaS allows teams to perform automation with minimal technical expertise and effort. Some platforms assist in writing and executing test code, either by automation or human intervention.
- **Reduced execution time -** With the high availability of resources, tests tend to run faster. This affects the speed at which you get test results and the total cost of testing.
- **Availability of tools and options -** Unlike the strategy of performing on-premises testing, which causes project delays, testing in the cloud provides unlimited, uninterrupted access to servers, tools, and programs anywhere, without waiting. These speeds up the entire testing cycle, resulting in better time-to-market.
- **On-demand Services -** On-demand services promise to transform how development groups purchase test services and manage their budgets using a pay-as-you-go model. This model allows for a shift from capital expenditures to operating expenses, dramatically reducing your quality assurance budgets and IT maintenance overhead.
- **Efficiency -** The TaaS model allows testers to focus on essential processes while synchronizing tools, people, and operations. With a standardized infrastructure, fewer configuration errors occur, which increases the efficiency of the entire testing process.
- **Global Availability -** Probability that a system is working when needed and under normal operating conditions. Then the system is up and running, and the organization can continue to produce results and run tests.
- **Scalability -** The TaaS environment allows testers to scale applications easily and quickly under test and expand capacity limits for multiple users simultaneously to meet the demands of Agile and DevOps strategies. This model allows organizations to add or remove hardware or software as needed dynamically. o exemplify such a situation, a high-end server may only be necessary once the load, stress, or performance tests are started. The server can be taken down as soon as these tests are completed. Likewise, in TaaS, a license for an application for a specific type of test can be counted only while said test is running.

- **Maximizing Resource Utilization** - Effective use of resources. n general, TaaS will require less administration effort compared to traditional testing. The organization can thus use resources more effectively. There is an increasing need for industries to be environmentally responsible by going green, and the IT industry is no exception. Cloud testing organizations enhance Green Testing. y sharing test resources in the cloud, companies utilize IT resources exclusively on demand, eliminating waste by eradicating infrastructure idleness.
- **Compatibility** - Ensures the software works correctly on all required platforms, ensuring an optimal user experience. Compatibility can refer to interoperability between hardware and software, products of the same type or different types, or different versions of the same product.
- **Geographic distribution** - Geographically distributed clouds allow the simulation of multiple users from different locations.
- **Multi-tenancy** - In cloud computing, the meaning of multi-tenancy architecture has been expanded because of new service models that take advantage of virtualization and remote access. TaaS providers can run an instance of their application on an instance of a database and provide web access to multiple customers. n this scenario, data for each tenant is isolated and remains invisible to other tenants. This also applies to common SaaS platforms.
- **Flexibility** - TaaS imparts tremendous flexibility by providing capabilities to grow and tear down the test environment without connecting to new tools and infrastructure. More importantly, on-demand, unit-based testing allows companies to pay for a specific unit of the test rather than spending on the entire stack, which somewhat reduces budget lock-in.
- **Automated testing** - Automated testing perform specific teststhrough automation (e.g., a set of regression tests). Test automation refers to the automation of the process of tracking and managing different tests.
- **Third-party quality certification** - When an independent organization evaluates the software development process and declares that the product conforms to specific safety, quality, or performance standards.
- **Accessibility** – The ability to dynamically provision the server, and the willingness to deploy or remove software, allows organizations to reduce the initial investment cost. If the organization decides to stop the operation for financial or strategic reasons, it will be easier to make this decision because it does not have many resources. The availability of different licensing models, such as pay-per-use, can further help reduce costs.
- **Agility -** Ability to create and respond to change. t is a way of coping and succeeding in an uncertain and turbulent environment. TaaS makes it possible to reduce testing time and costs without compromising quality, allowing organizations to be more agile in delivering critical applications to their users.
- **Easy to update** - The testing service provider can update their tools transparently to the user whenever necessary. In traditional testing, the user must report defects or undesirable behavior to the tool vendor, who reproduces, corrects, and send updates to the user, installing them.

4.2.2. Main Disadvantages of using TaaS

GIRARDON *et al.* [05] identify disadvantages in Testing as a Service, the special needs required to carry out these tests, namely:

- **Lack of standards -** Studies that report processes or methodologies that offer standards for the entire TaaS process are not easily found in the literature. Therefore, the lack of standards can be considered a disadvantage.

- **Special Techniques Required** - Cloud testing may require special technical skills to properly utilize the TaaS platform to generate test cases and scripts.
- **Data security and integrity** - Data security protects data from unauthorized access and corruption throughout its lifecycle. Data security includes encryption, tokenization, and essential management practices that protect data across all applications and platforms. Many providers allow users to submit binaries or executables for testing. This is also a significant concern, especially when testing activities are outsourced to a cloud testing service provider.
- **Possibility of non-availability** - TaaS platforms need to proactively identify misconfigurations and single points of failure and help prevent disruptions before they impact the business. The provider's challenge is maintaining constant global availability, testing services, and technical and support skills that meet or exceed customer needs.
- **Reliability** - Probability of fault-free software operation for a specified period in a given environment. Software reliability is also an essential factor that affects system availability.
- **Lack of automation** - Some platforms require the test user to enter parameters at each execution step. These tests are not fully automated and may require user knowledge.
- **Vendor lockout** - Known as owner lockout or customer lockout, makes the customer dependent on a testing platform or service, unable to use another vendor without substantial switching costs. Lock-in costs that create barriers to market entry can result in antitrust actions against a monopoly.

4.3. Tools and Providers in TaaS

Several tools were proposed to help developers and testers in their tasks to increase the productivity of the Software Testing process. However, writing test cases using these tools still requires much effort.

Test automation is often addressed when testing software in the cloud. Many test automation tools meet different requirements in a test lifecycle (e.g., automated test data generation, test case generation, test execution, and test evaluation). It is observed that over time, the tool market and its respective suppliers evolve very frequently. The following tools are considered pioneers in cloud test automation:

- **Testingbot** (https://testingbot.com) - Provides a cloud with more than 2,300 browsers and devices. Runs automated and manual tests. Run automated Selenium and Appium tests with high concurrency.
- **Saucelabs** (https://saucelabs.com/platform) - Conducts live, automated, and continuous testing for applications on the web and mobile platforms on large cloud-based testing platforms. Tests applications across thousands of browser and operating system combinations for desktop and mobile in the cloud.
- **Browserstack** (https://www.browserstack.com) - Provides users with a close-to-real experience, testing on more than 2,000 devices and browsers. It is a cloud-based web and mobile testing platform that allows developers to test their websites and mobile apps in browsers on demand.
- **Bitbar** (https://www.bitbar.com) – Offers a flexible, cloud-based mobile testing solution. With unlimited users and concurrency, it adapts to existing CI/CD (continuous integration /continuous delivery) processes and tools. Use any framework to run manual or automated tests on thousands of real devices.

KAUR *et al.* [24] also add a list of the leading providers of testing software as a service:

- **IBM** - The Smart Business Development & Test service, aimed at developers using IBM's Rational toolset.

- **SOASTA** - A forerunner in the cloud testing market, it has a unique cloud testing product suite (SOASTA CloudTest) to support standard web application testing in performance and load testing, function testing, and UI testing.
- **SKYTAP** - Offers a hybrid cloud model where customers use their test services with their on-premises tools and processes.
- **VMLogix** - Virtual Machine Management Specialist, VMLogix LabManager, Cloud Edition, runs on Amazon's EC2 (Elastic Compute Cloud).
- **ZEPHYR** - Manages the test lifecycle. The Zephyr platform provides test desktops and dashboards from a hosted management platform, which incorporates test applications and APIs.

4.4. Targeting Research in TaaS

Research carried out by [27], [28], [29] addresses TaaS in the following dimensions: Application, Management, and Legal and Financial. For each of these axes, questions in the scope of their research are detailed:

A. Application Problems

1) What types of applications can be tested in the cloud - Vendors and customers interested in cloud testing would like to know about the applications that can be readily tested in the cloud. This would help in migrating tests to the cloud. Then would it be safe to migrate?
2) Online Performance Testing Suite ready for any customer - Many applications and the cloud itself need to be tested on various performance attributes (response time, speed, and throughput).
3) Quality checks for applications tested in the cloud - Current technological advances have influenced a high demand for quality service from users. More is needed to carry out tests in the cloud, and it is necessary to guarantee the reliability of the test process and the tested application.
4) Harmonization of Testing Processes Across Multiplayers - With the increasing number of cloud providers, customers have various choices for cloud services. If the customer purchases services from more than one cloud testing provider, would it be efficient to have methods, tools, and facilities to monitor and manage the different processes and providers?
5) Test solutions for e-business systems - E-business systems were already used before cloud computing. One point for research is how cloud computing improves e-business models and what risks are involved.

B. Management Problems

1) **Group of testers** - Cloud service providers often promise 24x7 availability. Crowdsourcing is an exciting dynamic for cloud testing. There is potential to investigate how crowdsourcing supports cloud testing and the different crowdsourcing models that can be practiced. One model is to leverage a global pool of testers for customers' on-demand testing needs. Another model is testing specific software by a user community or interest group. The feedback obtained from the community would then be used to improve the tested application. Incorporating crowdsourcing, or human as a service (HaaS), into cloud testing would raise social issues such as trust and communication. The research problem is to investigate these impacts on collaborative testing activities in the cloud.
2) **Effects on the customer's business** - Some issues arise when an organization wants to take a cloud approach to acquire trial services. What would be the technical or commercial factors to consider? Then can a customer safely migrate testing to the cloud?

C. Legal and Financial Issues

1) **Test Data -** Test data management is a delicate task. Some tasks depend on the customer or production data for a test to take place effectively. n some cases, due to rules and regulations, customers are prohibited from providing confidential or production data to third parties.
2) **Pricing Models and Test Service Descriptions -** In general, pricing models for cloud computing services need to be well thought out so that customers are up to date and can estimate costs.

5. CONCLUSION AND FURTHER RESEARCH WORK

Cloud computing has impacted all business sectors in general. With the growing popularity of platform-as-a-service (PaaS) infrastructure, many applications are designed to run on multiple devices worldwide. Consequently, there is a need for various test configurations, including platforms, browsers, devices, networks, and the ability to mirror user access in real time. This is becoming a challenge for the entire testing process. Automated tests, management suites, and on-demand test infrastructure need to be rigorously analyzed when choosing vendors for test applications.

Through studies on Software Testing, it was observed that researchers consider that this area will only be partially free of challenges. However, applying new trends in TaaS can mitigate many of these obstacles. n this direction, TaaS promises benefits for many of today's problems in software testing. It can effectively revolutionize the future of testing, simplifying the creation and execution time of test suites with little cost and effort for your users.
The concepts defined in the Art of Software Testing, when looking for the principles of Software Testing, are still fundamental. However, research shows changes in the infrastructure of application availability, bringing Software Testing to join this evolution in ensuring software quality.

Software Testing has made and will continue to significantly contribute to the growth of cloud computing and vice versa. Cloud computing testing is expected to reach maturity in the next few years. As cloud testing standards evolve, testing strategies will be modified and adapted to test the quality of service parameters based on these future standards. Standardization will also allow concerns like security and interoperability to be addressed with more objective data.

Studies and research on TaaS can already be considered voluminous. For this work, we analyzed various research publications and literature reviews on TaaS. From the results of these analyses, it was observed that TaaS still has several needs that must be satisfied:

- Pricing models must be transparent and detailed to allow customers to estimate their spending better.
- The description of the services must be detailed so that it has the service definitions and reportable metrics.
- Quality standards specific to TaaS must also be created.
- Consumers and providers must have well-defined service standards that spell out details about software testing and quality to help develop fair Service Level Agreements.
- Solutions to cross-cutting vulnerabilities in cloud platforms and applications must be identified.
- TaaS can only succeed with innovative methods and solutions to increase demand and maintain its relevance in the future.

Regarding future research, it is observed that in TaaS, there is still room for much research on topics such as application problem-solving (which applications are best suited for testing in the cloud), management issues (how to organize human resources for tests based on and how to determine the effects of TaaS on a customer's business), legal and financial matters (how to manage test data in different global jurisdictions and how to handle test data, especially when it relates to sensitive customer information), and economic issues (how to design appropriate pricing models).

Finally, in the research carried out, researchers unanimously agree that TaaS will be the future of Software Testing and that there is an urgent need to develop new standards, techniques, platforms, and solutions for Testing as a Service.

REFERENCES

[1] D. A. Menace, Virtualization: Concepts, applications, and performance modeling, The Computer Measurement Groups'2005 International Conference, 2005.

[2] Antonia Bertolino, Guglielmo de Angelis, Micael Gallego, Boni García, Francisco Gortázar, Francesca Lonetti, Eda Marchetti. 2019. A Systematic Review on Cloud Testing. *ACM Computing Surveys*, Article 1 (January 2019), 41 pages.

[3] J. Gao, X. Bai, W. Tsai, and T. Uehara, Testing as a Service (TaaS) on Clouds, 2013 IEEE Seventh International Symposium on Service-Oriented System Engineering, pp. 212-223.

[4] MARKETSANDMARKETS. Press release, cloud testing market worth 10.24 billion USD by 2022. 2019) Available: http://www.marketsandmarkets.com/PressReleases/ cloud-testing.asp Accessed: 10/03/2021.

[5] Gustavo Girardon, Victor Costa, Rodrigo Machado, Maicon Bernardino, Guilherme Legramante, Fábio Paulo Basso, Elder de Macedo Rodrigues,AníbalNeto. 2020. Testing as a Service (TaaS): A Systematic Literature Map. In: The 35th ACM/SIGAPP Symposium on Applied Computing (SAC '20), March 30-April 3, 2020, Brno, Czech Republic. ACM, New York, NY, USA, 8 pages.

[6] BOURQUE, Pierre; FAIRLEY, Richard E. (Orgs.). SWEBOK: Guide to the Software **Engineering Body of Knowledge**. 3ª. ed. Los Alamitos, CA: IEEE Computer Society, 2014. Available: <http://www.swebok.org/>. Accessed: 25 fev. 2021.

[7] MYERS, Glenford J.; SANDLER, Corey; BADGETT, Tom. The Art of Software Testing. Ed. NJ: John Wiley & Sons, 2012.

[8] Saeed, Soobia & Khan, F & Khan, S & Islam, Noman. (2018). Conceptions of software testing as a service. 0. 1369-1384. 10.4314/jfas.v10i5s.114.

[9] Nitin Dangwal, Neha Mehrotra Dewan, Sonal Sachdeva. Testing the Cloud and Testing as a Service. n: Encyclopedia of Cloud Computing, Edited by San Murugesan and Irena Bojanova. USA: John Wiley & Sons, Ltd. 2016.

[10] Aderson Bastos, Emerson Rios, Ricardo Cristalli, Trayahú Moreira. Base de Conhecimento em Teste de Software - 3ª Ed. SP: Martins Editora, 2012.

[11] ISO/IEC/IEEE 29119-4. Software and systems engineering - Software testing - Part 4: Test techniques. Geneva, Switzerland: International Organization for Standardization, 2015.

[12] Filipe B. Barbosa, Isabelle V. Torres. O Teste de Software no Mercado de Trabalho, 2011. Available: <http://revista.faculdadeprojecao.edu.br/revista/index.php/projecao2/article/viewFile/82/70>. Accessed: 25 fev. 2021.

[13] Josenilson dos Santos Silva; Grazielle Bandeira Viana; Giselle Barbosa Gomes Machado; Rogério Oliveira da Silva. O Processo de Teste de Software. Tecnologias em Projeção, Distrito Federal, v.7, n. 2, p. 99. 2016.

[14] Júlio Viegas. Teste de software: introdução, conceitos básicos e tipos de testes. Sofist - Intelligent Software Testing. Available: https://blog.onedaytesting.com.br/teste-de-softwareAccessed: 02/02/2021.

[15] Ashfaq Ahmed. Software Testing as a Service. L: CRC Press, 2010. 212p.

[16] TMMi Foundation. Test Maturity Model integration (TMMi) - Diretrizes para Melhoria do Processo de Teste-Framework R1.2. 018. Available: <https://tmmi.org/tm6/wp-content/uploads/ 2018/11/TMMi-Framework-R1-2.pdf> Accessed: Feb. 2021.

[17] ISACA – CMI. Performance Solutions. Available: <https://cmmiinstitute.com/> Accessed: 20 Mar. 2021.

[18] CAPGEMINI; SOGETI; MICRO FOCUS. World Quality Report, 2020-21. Available: < https://www.capgemini.com/research/world-quality-report-wqr-20-21> Accessed: 25 Feb. 2021.

[19] Dudekula Mohammad Rafi, Katam Reddy Kiran Moses, K. Petersen, and M. V. Mäntylä, "Benefits and limitations of automated software testing: Systematic literature review and practitioner survey," *2012 7th International Workshop on Automation of Software Test (AST)*, 2012, pp. 36-42.

[20] R. Ramler and J. Gmeiner, "Practical Challenges in Test Environment Management," 2014 IEEE Seventh International Conference on Software Testing, Verification and Validation Workshops, 2014, pp. 358-359.

[21] Akerele O., Ramachandran M., Dixon M. (2013) Testing in the Cloud: Strategies, Risks, and Benefits. n: Mahmood Z., Saeed S. (eds) Software Engineering Frameworks for the Cloud Computing Paradigm. omputer Communications and Networks. pringer, London.

[22] Jayashree, J.& Vijayashree, J (2015), Software Testing in Cloud, *International Journal of Engineering Research and General Science, 3*(1).

[23] K. Inçki, I. Ari and H. Sözer, A Survey of Software Testing in the Cloud, *2012 IEEE Sixth International Conference on Software Security and Reliability Companion*, 2012, pp. 18-23.

[24] Amandeep Kaur, Navjeet Singh, Gurdev Singh. An overview of cloud testing as a service. International Journal of Computers & Technology V.2, N.2, Apr. 2012. SSN: 2277–3061 (online).

[25] George Candea, Stefan Bucur, Cristian Zamfi. Automated software testing as a service. *n:SoCC '10 proceedings of the 1st ACM symposium on cloud computing*, pp. 155–160. 2010.

[26] GURU99. Software Testing Tutorial. Available: <https://www.guru99.com/software-testing.html>Accessed: 26 Feb. 2021.

[27] L. M. Riungu, O. Taipale, and K. Smolander, Research Issues for Software Testing in the Cloud, *2010 IEEE Second International Conference on Cloud Computing Technology and Science*, 2010, pp. 557-564.

[28] L. M. Riungu, O. Taipale, and K. Smolander, Software Testing as an Online Service: Observations from Practice, *2010 Third International Conference on Software Testing, Verification, and Validation Workshops*, 2010, pp. 418-423.

[29] Leah Riungu-Kalliosaari, Ossi Taipale, Kari Smolander, Ita Richardson. Adoption and use of cloud-based testing in practice. n: Software Quality Journal. 24, 337–364. 2016.

AUTHORS

Janete Amaral is a Ph.D. Student in Computer Science at the Universidade Federal do Ceará and a Master's in Computer Science. Training in Process Consulting, Teaching Didactics, Facilitation of Group Processes, Coaching, etc. Professor and coordinator of bachelor's degree in Computer Science. he has experience in Software Engineering, working in Software Development Environments, Software Processes, Virtual Education, Usability, Testing, and Distributed Artificial Intelligence.

Alberto S. Lima is a Ph.D. in Eng. Teleinformatics from the Universidade Federal do Ceará. Master's in Computer Science from the Universidade de Fortaleza. Post-doctoral fellow at Universidade Federal de Campina Grande.Professor in Computer Science and master's degree in Public Policy and Management of Higher Education. He has experience in Corporate Governance, Strategic Planning, Computer Systems, working in Information and Communication Technology Management, Corporate and IT Governance, Computer Networks, Software Engineering, Applied Computational Intelligence, Multimedia/Web Projects, New Technologies in Education, Management, and Evaluation of Education. Reviewer of journals JNCA, COMNET, IEEE TNSM, IEEE TCC, IEEE Latin America Transactions, IJCC, Revista Ingenieria e Investigacion and Revista Electrónica Iberoamericana Sobre Calidad, Eficacia y Cambio en Educación.

José Neuman de Souza is a Senior Member of the IEEE. Professor at the Universidade Federal do Ceará. Doctor in Informatique - Universite de Paris VI. He developed Senior Post-Doctoral activities at LNCC-Rio de Janeiro. Senior Editor on the Editorial Board of the Journal of Network and Computer Applications (Elsevier). Member of the editorial board of journals (Elsevier) Computer Communications, Computer Networks, International Journal (Wiley Interscience) of Network Management. Works in

Computer Networks, Network Management, Information Systems, Grids, and Computational Clouds..Guest Professor at the University of Versailles Saint Quentin en Yvelines, University of Paris Nord (Paris XIII), University of Ottawa, University of Evry Val D'essonne. Guest Researcher CNRS at the University of Bordeaux I. Representative at IFIP/TC6 (Communication Systems).

Lincoln S. Rocha is a Ph.D. in Computer Science from Universidade Federal do Ceará. He specializes in exception handling, software product line, software adaptability, software architecture, and software defect prediction. His research seeks to solve software development problems in other areas of knowledge, such as Mobile and Context-Aware Computing, Ubiquitous Computing, Mobile Cloud Computing, and the Internet of Things. He acts as a reviewer of articles in workshops (VEM and SBCUP), conferences (FSE, SBES, SBLP, SugarLoafPlop, and WebMedia), and specialized journals (IST, JSERD, and CEE). Member of the SBC (Sociedade Brasileira de Computação).