Software Architectural Drivers for Cloud Testing

Etiene Lamas, Luiz Alberto Vieira Dias, Adilson Marques da Cunha
Computer Science Division
Aeronautics Institute of Technology, ITA
Sao Jose dos Campos, Brazil
{etiene, vdias, cunha}@ita.br

Abstract—**This paper focuses on the research issues that Cloud Computing imposes on Software Testing. For this purpose, Cloud Testing can be defined as a Software Testing method based on Cloud Computing technology. Software Testing has been an important component within the development process. In order to face the rapid growth of Cloud Computing, Reference Architectures provide a simple and organized environment for applications development. The outage on Cloud Services must be considered an exception not a rule. This research emphasizes the complexity of Cloud Testing, in order to prevent services disruption, as it happened, for example, with the Amazon in April 2011. This research aims to investigate, design, implement, and propose key Software Architectural Drivers for Cloud Testing, focusing on monitoring the quality. Cloud Testing integration may allow monitoring products and services with efficient deliverables. The main advantage that arises from these proposed drivers is the provision of Cloud Testing Reference Architectures to be applied in practice. The main contribution of software architectural drivers is the quantitative monitoring of quality for both end products and services.**

Keywords-cloud testing; software architectural drivers; testing of cloud services; testing of cloud products; reference architectures

I. INTRODUCTION

In general, Cloud Computing changes the way Information Technology (IT) services are delivered. To monitor these changes, Cloud Testing can be defined as a Software Testing method based on Cloud Computing technology [1].

Parveen and Tilley [2] show that not all applications are suitable for testing in the Cloud and nor all types of testing are suitable for the Cloud.

The outage on Cloud Services must be considered an exception not a rule. This research emphasizes the complexity of Cloud Testing, in order to prevent services disruption, as it happened, for example, with the Amazon in April 2011 [3]. This research aims to investigate, design, implement, and propose key Software Architectural Drivers for Cloud Testing (SADCT), focusing on monitoring quality. Thus, an investigation about the generic and the specific theory has been conducted.

The drivers proposed by the authors for Reference Architectures (RAs) are set in order to quantitative monitor Quality of Products (QoP) and Quality of Services (QoS) in the Cloud. The main advantage arising from these proposed drivers is to provide Cloud Testing Reference Architectures to be applied in practice. The main problem is how to monitor and evaluate quantitatively the quality of the Cloud Testing. The main contribution of software architectural drivers is the quantitative monitoring of quality for both end products and services.

This article is organized as follows. Section 2 introduces Reference Architectures. Section 3 describes basic Cloud Computing concepts. Section 4 emphasizes the importance of Cloud Testing and presents the testing of Cloud Services. Section 5 specifies the Cloud Testing Reference Architectures. Section 6 proposes its key architectural drivers. Section 7 includes a Proof of Concept (PoC) study. Finally, Section 8 highlights some conclusions and future works.

II. REFERENCE ARCHITECTURES

In order to face the rapid growth of Cloud Computing, Reference Architectures provide a simple and organized environment for applications development.

A Reference Architecture (RA) is the generalized architecture of several end systems that share one or more common domains. The Reference Architecture defines the common infrastructure to the end systems and also the interfaces of components that will be included in the end systems. The Reference Architecture is then instantiated to create software architecture of a specific system [4].

The principles governing the design and evolution of a system and also the relationships between their components and the environment can be found in a Reference Architecture, which represents its fundamental organization [5].

To facilitate the understanding of the operational intricacies in Cloud Computing, the overview of its Reference Architecture will be presented in the following section.

III. BASIC CLOUD COMPUTING CONCEPTS

Given the rapid growth in its use, it is necessary to define Cloud Computing and Cloud Computing Reference Architectures.

 ISBN: 978-1-61208-233-2

A. Cloud Computing Definition

According to the National Institute of Standards and Technology (NIST) [6], Cloud Computing consists of service models, deployment models, and essential characteristics.

This definition is widely accepted as a valuable contribution toward providing a clear understanding of Cloud Computing technologies and Cloud Services.

It provides a simple and unambiguous taxonomy of three service models available to Cloud Consumers: Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS).

It summarizes the four deployment models describing how the computing infrastructure that delivers these services can be shared: Private Cloud, Community Cloud, Public Cloud, and Hybrid Cloud.

Finally, the NIST definition also provides a unifying view of five essential characteristics that all Cloud Services exhibit: on-demand self-service, broad network access, resource pooling, rapid elasticity, and measured service [6].

The three service models identified by the NIST, i.e., SaaS, PaaS, and IaaS, offer to the consumers different types of service management operations and expose different entry points into Cloud Systems.

B. Cloud Computing Reference Architecture

The overview of the NIST Cloud Computing Reference Architecture [7] is a logical extension to the NIST Definition of Cloud Computing.

According to Liu et al. [7], it is a generic high-level conceptual model that is an effective tool for discussing the requirements, structures, and operations of Cloud Computing. Also, according to [7], it defines a set of actors, activities, and functions that can be used in the process of developing cloud computing architectures. It describes five major actors with their roles and responsibilities, using the newly developed Cloud Computing Taxonomy. The five major participating actors are: (i) Cloud Consumer - a person or organization that maintains a business relationship with, and uses service from, Cloud Providers; (ii) Cloud Provider - a person, organization, or entity responsible for making a service available to interested parties; (iii) Cloud Broker - an entity that manages Cloud Services; (iv) Cloud Auditor - a party that can conduct independent assessment of Cloud Services; and (v) Cloud Carrier - an intermediary that provides connectivity and transport of Cloud Services. Each actor is an entity (a person or an organization) that participates in a transaction or process and/or performs tasks in Cloud Computing [7].

The NIST Cloud Computing Reference Architecture [7] focuses on the requirements of "what" Cloud Services provide, not on "how to" design solutions and implementations.

In order to improve the quality of Cloud Services, the interactions between the actors in Cloud Testing scenarios will be discussed in the following section.

IV. TESTING OF CLOUD SERVICES

Software Testing has been an important component within the development process. This paper focuses on the research issues that Cloud Computing imposes on Software Testing.

The architecture described by Blokland and Mengerink [8] consists of detailed risks that may occur when one starts using Cloud Computing, grouped into themes. Next to these risks, in their book, there are sets of test measures. Some measures do exist, like load testing, but polished to fit the new needs for applying performance testing in the Cloud.

The important asset of [8] is the link made from each individual risk to the different measures needed to cover the risk.

According to Blokland and Mengerink [8], there are also new measures that stretch the definition of test, like test in production. These measures are new because they are Cloud specific and present the complexity of testing in the Cloud.

It should be emphasized that some aspects of quality can be tested "on live" and it is very wise to continuously test them, because of the ever-changing situation in the Cloud Environment.

Testing activities must continue even after the system has gone live. But, there are other aspects that should be 'more traditionally' tested, before a new version of the system is put into the live Cloud Environment. Testing is not done only under the main implementation phases (Unit Testing; Integration Testing; System Testing; and Acceptance Testing) as it used to be, but it will be done also during selection (when the Cloud Services are selected). The criteria for selection are chosen for mitigating risks.

Because once in Cloud production everything might change, it is needed to continuously test the software under production. Some measures are specific for the Cloud, like how to deal with rules and regulations in different countries [7], like Migration Testing.

When software is running "in house", most of the failures are under control; but when "in the Cloud" everything is different, because failures are not under control any more. Due to the mutability of the Cloud Environment, it is necessary to verify if the services are still working after the deployment. The testing under production will validate the functionalities in this environment.

V. CLOUD TESTING REFERENCE ARCHITECTURE

Architectural Drivers are defined as the major quality attribute goals that shape the Cloud Reference Architectures [9]. This research aims to investigate, design, implement,

 ISBN: 978-1-61208-233-2

and propose key Software Architectural Drivers for Cloud Testing, focusing on monitoring quality.

Aiming to understand Reference Architecture roles for Cloud Testing, Figure 1, adapted from [10], presents the interaction of the software tester role (Cloud Tester) with the Cloud Environment specified for this research. Figure 1 also highlights those that provide and consume services.

Notice that IaaS supports PaaS that, in turn, supports SaaS.

The main characteristics that distinguish Cloud Testing from regular Software Testing are related to risks across all three layers of the Cloud stack (IaaS, PaaS, and SaaS), as seen in Figure 1. It is important to keep this basic stack in mind as the building blocks of the Cloud Computing system.

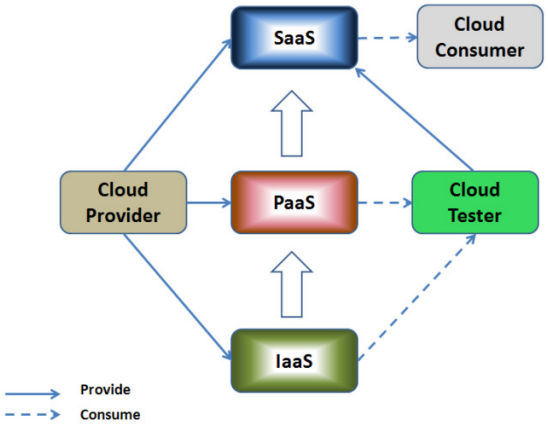

Figure 1. The Reference Architecture roles for Cloud Testing.

The Cloud Provider is responsible for providing, managing, and monitoring the entire structure to the solution of Cloud Computing, freeing the Cloud Tester and the Cloud Consumer from these types of liability. To do this, the Cloud Provider provides services for Cloud Consumers.

According to Veras [10], this organization in roles helps to define the actor and their different interests on Cloud Computing. Actors can assume different roles at the same time, according to their interests, and only the Cloud Provider supports all three functions of Cloud Services (IaaS, PaaS, and SaaS). From the viewpoint of interaction, among the three functions of service, IaaS provides computing resources (hardware or software) to PaaS. In its turn, PaaS provides resources, technologies, and tools for the development and the delivery of services to be implemented, becoming available as SaaS.

It is important to mention that an organization that provides Cloud Services needs does not necessarily provide all three-service functions. That is, a Cloud Provider can provide the option IaaS without necessarily also providing a PaaS [10].

This actor/role-based model is intended to serve the expectations of the stakeholders by allowing them to understand the overall view of roles and responsibilities, in order to assess and assign risks [7].

VI. THE KEY ARCHITECTURAL DRIVERS

According to Kazman et al. [9], the project manager describes what business goals are motivating the development effort and hence what will be the primary Architectural Drivers (e.g., high availability, time to market, or high security).

Aiming to understand the key architectural drivers for Cloud Testing Reference Architecture, the purpose of Figure 2, suggested by the authors, is to provide the guidance for the Cloud Testers to acquire knowledge on all needed testing categories.

Some proposed drivers for Cloud Testing Reference Architectures were based on traditional concepts that allow products with recognized quality (QoP), and another proposed drivers, specific for the Cloud, that allow better quality for Cloud Services (QoS).

These drivers were defined based on concepts from the traditional testing management environment, as seen in the bottom side of Figure 2, and also on concepts and elements, for the Cloud Testing management environment, as seen in the upper side of Figure 2.

A. *Traditional testing management environment*

The most important concepts for traditional test management environment can be clustered in two groups. The first group of concepts involves a set of definitions to support the Cloud Testing with Noncloud standards. These definitions relate to the guidelines for Software product Quality Requirements and Evaluation (SQuaRE) [11] and the appropriate breadth and depth of test documentation. The second group of concepts involves a set of methods supporting Cloud Testing with Noncloud methodologies. These definitions relate to effective methods for Software Testing [12] and these specific methods are listed below.

The authors suggest the use of an Agile Software Development Methodology, in order to deliver as much quality software as possible, within a series of short time boxes called Sprints, which last about a month. This methodology is characterized by short, intensive, and daily meetings involving the whole developers' team [13]. Agile is iterative and incremental. This means that the testers must test each increment of coding as soon as it is finished [14].

Finally, the second group of concepts involves also a set of techniques supporting Cloud Testing with Noncloud techniques. These are related to functional and structural techniques for Software Testing. These groups are:

a) Noncloud Standards: In this group of drivers, as seen in the bottom left side of Figure 2, the standards ISO/IEC 25000 named Software product Quality

 ISBN: 978-1-61208-233-2

Requirements and Evaluation (SQuaRE) should be applied [11], and the IEEE Std 829-2008, named IEEE Standard for Software and System Test Documentation, should be also applied [15]; and

b) Noncloud Testing Methodologies and Techniques: In this group of drivers, in the bottom right side of Figure 2, traditional phased software agile methodologies and effective methods for Software Testing should be applied. Traditionally, most of the test effort occurs after the requirements have been defined and the coding process has been completed. But, in the Agile approaches [14], most of the test effort is on-going. Newer development models, such as Agile, often employ Test Driven Development (TDD) and place an increased portion of the testing in the hands of the developer, before it reaches a formal team of testers. In a more traditional model, most of the test execution occurs after the requirements have been defined and the coding process has been initiated [14]. Also, in this group of drivers, in the bottom right side of Figure 2, the techniques can be divided into functional and structural. The main functional system testing techniques are: (i) Requirements - system performs as specified; (ii) Regression - verifies that anything unchanged still performs correctly; (iii) Defects Handling - defects can be prevented or detected, and then corrected; (iv) Manual support - the people-computer interaction works; (v) Control - controls reduce system risk to an acceptable level; and (vi) Parallel - old system and new system run and their results are compared to detect unplanned differences [12]. The main structural testing techniques are: (i) Stress - system performs with expected volumes; (ii) Execution - system achieves desired level of proficiency; (iii) Recovery - system can be returned to an operational status after a failure; (iv) Operations - system can be executed in a normal operational status; (v) Compliance - system is developed in accordance with standards and procedures; and (vi) Security - system is protected in accordance with the importance to organization [16].

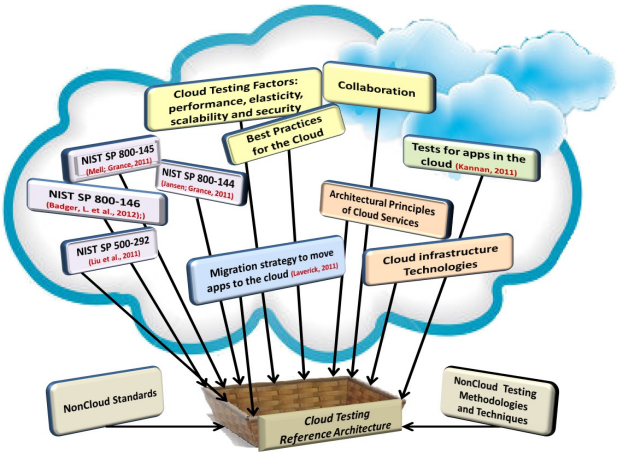

Figure 2. Software Architectural Drivers for Cloud Testing (SADCT).

B. *Cloud Testing management environment*

The most important concepts and elements for the Cloud Testing management environment can be clustered in five groups. The first group is a set of definitions to support the Cloud Testing with standards. These are related to adoption, development, and provision of testing and security for Cloud Computing. The second group is the set of best practices supporting Cloud Testing with collaboration and relevant factors. These are related to the Cloud Environment and essential characteristics for Cloud Services. The third group is a set of techniques supporting Cloud Testing with challenges. These are related to the testing techniques. The fourth group is a set of concepts supporting Cloud Testing with architectural principles. These are related to the technologies comprised in the Cloud Infrastructure. In the Cloud, not all applications are equally created. Finally, the fifth group is a set of steps supporting Cloud Testing with strategy for porting applications to the Cloud. These steps are related to Cloud Migration strategies. These groups are:

a) Cloud Standards: In the grey group of drivers, in Figure 2, the NIST Definition of Cloud Computing should be applied [6]; the NIST Guidelines on Security and Privacy in Public Cloud Computing should be applied [16]; the NIST Cloud Computing Synopsis and Recommendations should be applied [17]; and the NIST Cloud Computing Reference Architecture should be also applied [7]. The logical step to take after the formation of the NIST Cloud Computing definition is to create an intermediate reference point from where one can frame the rest of the discussion about Cloud Computing and begin to identify sections in the Reference Architecture in which standards are either required, useful, or optional [7];

b) Testing Factors, Collaboration, and Best Practices for the Cloud: In the yellow group of drivers, in Figure 2, it is important to mention that testing for Cloud-based applications presents its own specific challenges. Understanding how these applications are structured goes a long way in designing and executing appropriate test plans for them. These tests are done in addition to the usual Unit; Integration; System; Acceptance, and Performance. For example, the Performance should be achieved in the Cloud by testing for bandwidth, connectivity, scalability, and quality of the end-user experience. When testing Cloud applications, it is needed to validate and verify specific Cloud functionalities such as redundancy, failover, and performance scalability. Also, in the yellow group of drivers, suggested by the authors, it is important to mention that the Cloud provides an environment that supports global collaboration and knowledge sharing, as well as, group decision-making. Shared sites can be easily set up, replicated, and torn down as needed to meet the collaboration requirements of a given project. For collaboration, the best practices are to: (i) continuously monitor from users' perspective and end-user response time; (ii) implement end-to-end diagnostics; (iii) design for fault-tolerance; and (iv) load test to determine the breaking point.

 ISBN: 978-1-61208-233-2

For performance, the best practices are to: (i) understand where all bottlenecks are; (ii) mitigate bottlenecks; (iii) test performance for understanding normal and peak load to baseline "normal"; and (iv) continuously monitor performance from users' perspective. For scalability, the best practices are to: (i) architect for elasticity; (ii) use an elastic platform to scale services and data; (iii) isolate functions to scale them separately; (iv) implement a Cloud bursting strategy for load balancing between Clouds [17]; (v) automate scaling to quickly scale-up and down; and (vi) execute the load test in your application;

c) Testing techniques: In the green group of drivers, in Figure 2, Kannan [18] exemplifies challenges for: (i) browsers testing; (ii) service provisioning/de-provisioning testing; (iii) distributed Cloud Testing; (iv) multi-tenancy testing; (v) Cloud Portability Testing; among others. Cloud-based software applications have some additional characteristics compared to Noncloud-based ones. These pose additional challenges but with a systematic and comprehensive approach to test planning, to be appropriately handled;

d) Cloud infrastructure and architectural principles: In the pink group of drivers, in Figure 2, also suggested by the authors, it is important to mention that Cloud infrastructure should never go down for a day. Clouds are characterized by various technologies including: (i) virtualization (hypervisor); (ii) automation; (iii) monitoring; and (iv) service portal/service catalog. Currently, there are Cloud architectural principles for high availability: (i) monitoring; (ii) fault tolerance; and (iii) fixable.

e) Migration strategies: In the blue group of drivers, in Figure 2, it is important to mention that in the Cloud, not all applications are created equal, and some are completely wrong for the infrastructure model. To make the right decision about which applications to move, it is needed a solid migration strategy. It is also needed to consider the application portfolio and the business requirements to prevent problems such as poor application performance and latency, data leakage, or issues with compliance or other regulations [19]. Here is how to develop a foolproof strategy for moving the right applications to the cloud, which starts by outlining clear objectives, then focuses on your application portfolio's characteristics and business requirements to determine the best fit. These steps ensure that moving to the Cloud will be possible.

VII. PROOF OF CONCEPT (POC)

A Proof of Concept (PoC) is an exercise to test a design idea or assumption. Software developers tend to utilize PoCs instinctively when they experiment with technology.

The presented drivers could be used in a PoC to quantify the quality monitoring throughout the key Software Architectural Drivers for Cloud Testing (SADCT). This will be elaborated using the Multi-Attribute Global Inference of Quality (MAGIQ) technique for Software Testing [20]. The MAGIQ technique uses Rank Order Centroids (ROC) [21] to convert system comparison drivers into normalized numeric weights, and then computes an overall measure of quality as a weighted (by comparison drivers) sum of system ratings.

The PoC was applied to an academic project named "Fraud Detection and Unauthorized Access (FDUA)", developed at the Brazilian Aeronautics Institute of Technology (*Instituto Tecnologico de Aeronautica* - ITA) aiming to evaluate the feasibility of the Software Architectural Drivers for Cloud Testing propositions.

Given the FDUA Test Scenario and the Software Architectural Drivers for Cloud Testing Hierarchical Diagram, as seen in Figure 3, suggested by the authors, the students (Cloud Testers), all seasoned testers, were asked to rank the Software Architectural Drivers for Cloud Testing items.

At this point, the Software Architectural Drivers for Cloud Testing Hierarchical Diagram was performed as a hierarchical decomposition of the proposed Software Architectural Drivers for Cloud Testing by using MAGIQ technique for Software Testing [20].

In the MAGIQ analysis technique, after the attributes of the systems under evaluation have been determined, rank order centroids are used to assign relative weights to each comparison attribute [20].

The Cloud Testers examine the comparison attribute set at each level of the hierarchical decomposition of the attributes, and ranks the Software Architectural Drivers for Cloud Testing in the set from most important to least important, and then assigns relative weights to each Software Architectural Drivers for Cloud Testing using rank order centroids [21].

For each item, the Cloud Testers should assign a weight (in the range 0 to 1), which will be composed with the MAGIQ coefficients, in order to evaluate quantitatively QoP and QoS.

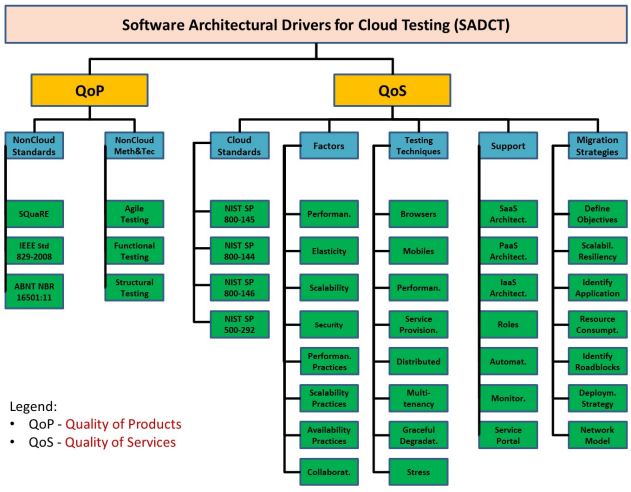

Figure 3. The SADCT Hierarchical Diagram.

Numerically, the quality of Cloud Testing is obtained by Software Architectural Drivers for Cloud Testing.

Equation (1), proposed by authors, quantitatively monitors the quality of Cloud Testing:

$$SADCT = QoP + QoS. \quad (1)$$

Software Architectural Drivers for Cloud Testing (SADCT) = 0.910																				
Quality of Products (QoP) (Rank = 0.750) (Result = 0.719)						Quality of Services (QoS) (Rank = 0.250) (Result = 0.191)														
Ranks	Weight	Part.Res.	Ranks	Weight	Part.Res.	Ranks	Weight	Part.Res.	Ranks	Weight	Part.Res.	Ranks	Weight	Part.Res.	Ranks	Weight	Part.Res.	Ranks	Weight	Part.Res.
0.250	100%	0.25	0.750	83%	0.708	0.124	88%	0.118	0.457	75%	0.435	0.257	47%	0.097	0.124	57%	0.107	0.040	20%	0.008
0.111	1	0.028	0.611	1	0.458	0.104	0.5	0.006	0.184	1	0.084	0.016	0	0.000	0.192	1	0.024	0.299	0.2	0.002
0.611	1	0.153	0.111	0.5	0.042	0.521	1	0.065	0.025	0	0.000	0.079	0.25	0.002	0.192	1	0.024	0.156	0.2	0.001
0.278	1	0.070	0.278	1	0.209	0.104	1	0.013	0.184	1	0.084	0.236	1	0.029	0.109	1	0.014	0.299	0.2	0.002
						0.271	1	0.034	0.340	1	0.155	0.236	1	0.029	0.370	1	0.046	0.059	0.2	0.000
									0.081	1	0.037	0.111	0.5	0.007	0.032	0	0.000	0.109	0.2	0.001
									0.081	1	0.037	0.054	0	0.000	0.032	0	0.000	0.020	0.2	0.000
									0.081	1	0.037	0.033	0	0.000	0.073	0	0.000	0.059	0.2	0.000
									0.025	0	0.000	0.236	1	0.029						
Noncloud Standards			Noncloud Meth&Tech			Cloud Standards			Factors			Cloud Testing Techniques			Support			Migration Strategies		

Figure 4. The Q1 Results.

The drivers for traditional concepts will focus on Quality of Products (QoP), and the additional drivers, specific for the Cloud, will focus on the Quality of Services (QoS).

VIII. PoC Results

The Proof of Concept (PoC) was applied in four different Agile Testing Quadrants or phases (Q1, Q2, Q3, and Q4) of the Cloud Testing [14].

Figure 4, suggested by the authors, is a data sheet in order to calculate values for Quality of Products (QoP) and Quality of Services (QoS), applied to the Software Architectural Drivers for Cloud Testing, as in (1):

a) Q1 – In the Unit Testing, the results are obtained through the calculations from the data sheet presented in Figure 4. Summarizing, the value for the obtained Software Architectural Drivers for Cloud Testing is 0.910, because QoP is 0.719 and QoS is 0.191;

b) Q2 – In the Integration Testing, the results are obtained through similar calculations. Summarizing, the value for the obtained Software Architectural Drivers for Cloud Testing is 0.715, because QoP is 0.563 and QoS is 0.151;

c) Q3 – In the System Testing, the results are obtained through similar calculations. Summarizing, the value for the obtained Software Architectural Drivers for Cloud Testing is 0.830, because QoP is 0.682 and QoS is 0.149; and

d) Q4 – In the Acceptance Testing, the results are obtained through similar calculations. Summarizing the value for Software Architectural Drivers for Cloud Testing obtained is 0,901 because QoP is 0,226 and QoS is 0,675.

Figures 5 and 6, suggested by the authors, show numerically the Quality of Products (QoP) and Quality of Services (QoS) for each Cloud Testing phases.

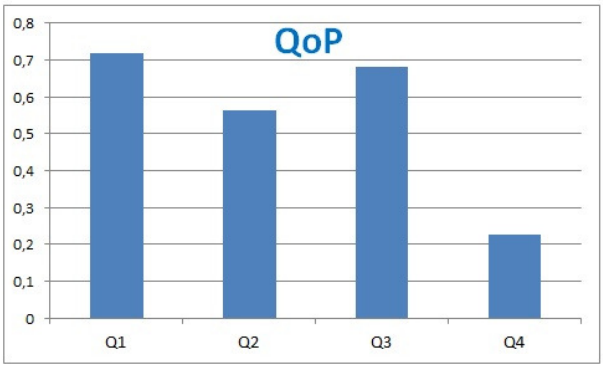

Figure 5. The QoP Results for each Cloud Testing phases.

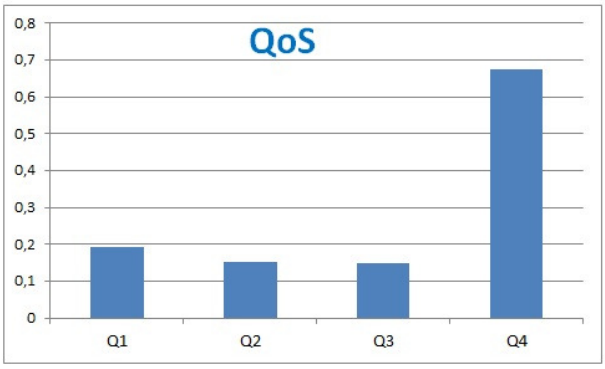

Figure 6. The QoS Results for each Cloud Testing phases.

IX. Conclusion and Future Work

This research has provided the investigation, design, and implementation of some key Software Architectural Drivers for Cloud Testing, focusing on monitoring the quality for both end products and services.

 ISBN: 978-1-61208-233-2

The Software Architectural Drivers for Cloud Testing, proposed by the authors, was evaluated through priorities and weights assigned to them, by seasoned testers. Other drivers could have been included and theirs effects measured for Cloud Testing. However, in this research, the proposed drivers have been proved appropriated, based on the above criteria.

The drivers, prioritized and weighted by experts, have allowed the quantitative monitoring of quality on Cloud Testing.

As a result of this research, it was obtained a way to numerically calculate the quality of Cloud Testing.

From this research, it is possible to evaluate the influence of each Software Architectural Drivers for Cloud Testing, by prioritizing and weighting each driver. It is also possible to measure the influence of each individual driver on the overall quality of Cloud Testing, by assigning it a numerical value.

Within the Cloud, the test must start earlier (in a very early stage); the test scope is widened because of its nonfunctional requirements; and the test must never stop (due to the fact that there are a lot of continuous services to be performed and also due to constant environment changes). This assures that the software is tested thoroughly.

The authors recommend the continuation of this research in the Cloud Production. A question that arises from this work is: "Software Architectural Drivers for Cloud Testing in production can be applied?"

The answer to this question can be obtained through further experiments.

As future works, it is suggested the application of these drivers into other experiments and a statistical in-depth evaluation about its effects on Cloud Testing.

This would foster better QoS, as end products, by fulfilling some existing gaps of knowledge within the Cloud Computing environment.

REFERENCES

[1] W. Jun and F. Meng, "Software Testing Based on Cloud Computing," International Conference on Internet Computing and Information Services, 2011.

[2] T. Parveen, and S. Tilley, "When to Migrate Software Testing to the Cloud?," In proc. 2nd International Workshop on Software Testing in the cloud (STITC), 3rd IEEE International Conference on Software Testing, Verification and Validation (ICST), April 2010, pp. 424-427.

[3] A.W.S. Team, "Summary of the Amazon EC2 and Amazon RDS Service Disruption," Amazon Web Services [Online]. Available: <http://aws.amazon.com/pt/message/65648/> 10.18.2012.

[4] B. Gallagher, "Using the Architecture Tradeoff Analysis Method to Evaluate a Reference Architecture: A Case Study," Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, Technical Note CMU/SEI-2000-TN-007, 2000.

[5] B.Batke and P. Didier, "The importance of Reference Architecture in Manufacturing Networks," CIP Networks Conference, 2007. Available:<http://www.odva.org/Portals/0/Library/CIPConf_AGM/ODVA_12_AGM_The_Importance_of_Reference_Architectures_Didier_Batke.pdf> 10.18.2012.

[6] P. Mell and T. Grance, "The NIST Definition of Cloud Computing," SP 800-145, National Institute of Standards and Technology's, U.S., 2011.

[7] F. Liu, J. Tong, J. Mao, R. Bohn, J. Messina, L. Badger, and D. Leaf, "NIST Cloud Computing Reference Architecture," SP 500-292, National Institute of Standards and Technology's, U.S., 2011.

[8] K. Blokland and J. Mengerink, "Cloutest®: Testen van cloudservices," Uitgeverij Tutein Nolthenius, 2012.

[9] R. Kazman, M. Klein, and P. Clements, "ATAM: Method for Architecture Evaluation," Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, Technical Report CMU/SEI-2000-TR-004, 2000.

[10] M. Veras, "Virtualização: Componente Central do Datacenter," Brasport, Sérgio Martins Oliveira, 2011.

[11] ABNT, "NBR ISO/IEC 25000. software quality requirements and evaluation," Associação Brasileira de Normas Técnicas, 2008.

[12] W. E. Perry, "Effective Methods for Software Testing," N.Y.: Wiley, 2006.

[13] M. Cohn, "Succeeding with Agile: Software Development Using Scrum," Addison-Wesley Professional, 2009.

[14] L. Crispin and J. Gregory, "Agile Testing: A Practical Guide for Testers and Agile Teams," Addison-Wesley Professional, 2009.

[15] IEEE Std 829-2008, "IEEE Standard for Software and System Test Documentation," Institute of Electrical and Eletronics Engineerd, 2008.

[16] W. Jansen and T. Grance, "Guidelines on Security and Privacy in Public Cloud Computing, " SP 800-144, National Institute of Standards and Technology's, U.S., 2011.

[17] L. Badger, T. Grance, R. Patt-Corner and J. Voas, "Cloud Computing Synopsis and Recommendations - SP800-146," National Institute of Standards and Technology's (NIST), 2012.

[18] N. Kannan, "Ten tests for software applications in the cloud," SearchCloudComputing, TechTarget, Inc. 275 Grove St. Newton, MA 02466, 2011.

[19] M. Laverick, "Private Cloud e-zine," vol. 1, SearchCloudComputing, TechTarget, Inc. 275 Grove St. Newton, MA 02466, 2011.

[20] J. D. McCaffrey, "Using the Multi-Attribute Global Inference of Quality (MAGIQ) Technique for Software Testing," Information Technology: New Generations, 2009. ITNG '09. Sixth International Conference on, 2009, pp. 738-742.

[21] J. Jia, G. W. Fischer and J. S. Dyer, "Attribute weighting methods and decision quality in the presence of response error: a simulation study," Journal of Behavioral Decision Making, 1998, vol. 11, no. 2, pp. 85-105.

 ISBN: 978-1-61208-233-2