Technical Test Methodology

JANUARY 2015

DELIVERABLE

Project Acronym:	**SDI4Apps**
Grant Agreement number:	**621129**
Project Full Title:	**Uptake of Open Geographic Information Through Innovative Services Based on Linked Data**

D3.5 TECHNICAL TEST METHODOLOGY

Revision no. 05

Authors: Matteo Lorenzini (Hyperborea)
Alessandro Pironi (Hyperborea)

Project co-funded by the European Commission within the ICT Policy Support Programme		
Dissemination Level		
P	Public	X
C	Confidential, only for members of the consortium and the Commission Services	

REVISION HISTORY

Revision	Date	Author	Organisation	Description
01	23/12/2014	Matteo Lorenzini	HYPER	Initial draft
02	07/01/2015	Karel Charvát Martin Kuba	CCSS Masaryk University	Contribution with initial suggestion about specific component will be tested
03	23/01/2015	Martin Kuba	Masaryk University	Contribution with suggestion about functional and non functional testing
04	28/01/2015	Matteo Lorenzini	HYPER	Final Version and formatting
05	30/01/2015	Tomas Mildorf	UWB	Formatting, minor changes

TABLE OF CONTENTS

EXECUTIVE SUMMARY

This document presents the testing methodology adopted in the SDI4Apps framework. The document starts with describing the system architecture that characterises the platform. Then a brief introduction about the cloud testing methods, the most powerful cloud components testing solutions are examined, and finally, our testing methodology adopted and defined for the SDI4Apps project is described.

1 INTRODUCTION

Cloud computing framework has become a new computing paradigm where the cloud can provide both virtualised hardware and software resources that are hosted remotely and provide a use-on-demand service model. Cloud computing offers an ability to access shared resources and common infrastructure, which provide services on demand over the network to perform operations that meet changing business needs. It provides facilities for users to implement, deploy and manage their applications 'on the cloud' which entails virtualization of resources that preserves and accomplishes itself. Cloud testing uses cloud environmental architecture for software testing[1].

Cloud testing is a form of testing in which web applications use cloud computing environment and infrastructure to simulate real world user traffic by using cloud technologies and solutions. . This is where cloud testing has emerged as a fresh approach to testing where cloud computing environments are leveraged to simulate real world with application's performance, consistency, speed, security and functionality.

As defined in the SDI4Apps deliverable D3.1 "Architecture Concept", the main focus of the SDI4Apps project is the development of a cloud based system framework for geographical data integration and enrichment using open API services and protocols in order to connect the world of governments with small and medium enterprises or single user developing applications based on geographical contents.

The principal part of this document describes testing components methodology used for the SDI4Apps framework. The deliverable starts with a short description of the cloud architecture adopted, and, after the definition of testing methodology, the last part concerns the definition of the test workflow adopted in the SDI4Apps project.

1 Organisations pursue general testing that carries some challenges such as limited test budget and meeting deadlines. To serve a good quality product, testing is the last solution to any kind of problem we would face in the future from the customer side.

1 SYSTEM ARCHITECTURE

As described in the deliverable D.3.1 “Architecture Concept”, the SDI4Apps framework is composed of three different Platform As a Service (PAAS) cloud systems and a storage platform that comprise a back end layer system.

Therefore we can identify four different framework layers:

1. Data layer → this level is characterised by the integration of different kind of resources and datasets in the SDI4Apps framework. Thanks to the scalability and interoperability with the other platforms, this particular job is performed by the PostGIS platform.
2. Application layer → this level includes different kinds of components able both to interact with different kinds of data acquired and to process the information in a layer defined mainly as concerning semantical data indexing and geographical representation and visualisation.
3. Deploy → the SDI4APPS infrastructure[2] is characterised by two different cloud service systems:
    1. Software as a Service → used for the service that will not be hosted directly on the infrastructure.
    2. Platform as a Service → used for the service that will be hosted directly on the infrastructure.

   Both SAAS and PAAS will be specific for Spatial Data Infrastructure (SDI) and they will be layered on the top of an IAAS service based on Open Nebula cloud management software.
4. Presentation → this layer characterises the interaction between an end user and the infrastructure. Using an API interface and, according to W3C consortium specifications, HTML5, the infrastructure of the SDI4Apps project will guarantee a complete interoperability with different kinds of devices both on mobile platforms and in local systems.

Furthermore, it’s possible to define two different main groups that will represent the objects of our testing methodology:

Platform as a Service:

- Data collection → different kinds of data will be acquired from different kinds of sources.
- Data transformation → in order to guarantee a common data definition and structure, the data and metadata acquired will be transformed following a common data schema.
- Data provision → data will be available for third parties integrating the SDI4Apps framework ..

2 SDI4Apps deliverable D.3.1.2 “Architecture Concept”

SDI4Apps cloud storage: the storage system is characterised by Postgres-XL and PostGIS. This solution guarantees the management of big datasets [3] composed of alphanumerical, geometrical and geographical features. At this level the activity related to data transformation will be performed.

3 Thanks to the scalability offered by Postgres-XL

2 TEST METHODOLOGY OF CLOUD FRAMEWORK COMPONENTS

2.1 What is cloud testing?

According to Wikipedia definition, “*cloud testing is a form of software testing in which Web applications that leverage Cloud computing environments ("cloud") seek to simulate real-world user traffic as a means of load testing and stress testing web sites. The ability and costs to simulate Web traffic for software testing purposes has been an inhibitor to overall Web reliability*[4]." More practically, cloud-based software testing refers to testing and measurement activities on a cloud-based environment and infrastructure by leveraging cloud technologies and solutions. It has four major objectives.

1. To assure the quality of cloud-based applications deployed in a cloud, including their functional services, business processes, and system performance as well as scalability based on a set of application-based system requirements in a cloud.
2. To validate software as a service (SaaS) in a cloud environment, including software performance, scalability, security and measurement based on certain economic scales and pre-defined SLAs.
3. To check the provided automatic cloud-based functional services, for example auto-provisioned functions.
4. To test cloud compatibility and inter-operation capability between SaaS and applications in a cloud infrastructure, for example, checking the APIs of SaaS and their cloud connectivity to others.

2.2 Why is cloud testing important?

Compared with the other kinds of testing methodologies, cloud-based testing has several unique advantages listed below.

- Reduce costs by leveraging computing resources in clouds – This refers to effectively using virtualised resources and shared cloud infrastructure to eliminate required computer resources and licensed software costs in a test laboratory.
- Take the advantage of on-demand test services (by a third-party) to conduct large-scale and effective real-time online validation for internet-based software in clouds.
- Easily leverage scalable cloud system infrastructure to test and evaluate system (SaaS/Cloud/Application) performance and scalability.

2.3 Different types of testing

Following the bibliographical survey carried on for the definition of our testing methodology we can identify some general common testing approaches:

4 Based on a recent literature survey, there is a few of published papers addressing cloud-testing concepts, issues, and challenges.

- Availability Testing: Cloud offerings must be available at all times. It is the responsibility of the Cloud maintainer to ensure that there are no abrupt downtimes. In addition the business of the users must not be adversely affected in the case of any planned downtime
- Security Testing: There must be no unauthorised access to data. Shared data integrity should be maintained and secured at all times. At present several organisations and communities are formalising industry standards to define the acceptability criteria for Cloud offerings in terms of security.
- Stress testing: it is used for performance testing which focuses on determining an application's robustness, convenience, and consistency under extreme conditions. The aim of stress testing is to recognise application issues that become apparent under risky conditions. These conditions can include dense loads, high concurrency, or limited computational resources. Right stress testing is useful in outcome synchronization and effective bugs interconnect problems, priority problems, and resource loss bugs. The plan is to stress the system to the breaking point in order to find bugs. The system is not expected to process the overload without adequate resources, but to behave (e.g. failure) in an acceptable manner (e.g. not corrupting or losing data or loss).Stress tests typically involve simulating one or more key production scenarios under a variety of hectic conditions. For example, you might deploy your application on a server that is already running a processor-intensive application; in this manner, your application is immediately "starved" of processor resources and must compete with the other application for processor phases. You can also stress-test a single item such as a stored procedure or class or a single Web page.
- Load testing: The process of analysing software applications and supporting infrastructure to determine acceptable performance, capacity and transaction handling capabilities of real world data with usage conditions and executing them against the application and supporting infrastructure under test. The basic approaches to performing load testing on a Web application are:
  - Identify the performance-serious states.
  - Identify the workload status for distributing the entire load among the key scenarios.
  - Identify the metrics to verify them against your performance objectives.
  - Design tests to simulate the load.
  - Use available tools to implement the load according to the designed tests, and capture the metrics for proper load analysis.
  - Identify and analyse the metric data captured during the tests; make a record for proper load spreading. By such an iterative testing process, we achieve our performance objectives. There are several reasons for load-testing to be accomplished over Web applications. The basic need of load testing is used to govern the Web application's behaviour under both usual and foreseen peak load conditions. Performance testing eliminates the constraints of traditional testing solutions like hardware availability, software licensing and installation, version control, test creation, system monitoring, and the cost of hiring and training staff.

- Interoperability Testing: Any developed or migrated Cloud application must work in multiple environments and platforms. The application should also have the capability to be executed across various Cloud platforms. It should be easier to move the Cloud applications and platforms from one Infrastructure (as a service) to another Infrastructure. As with Security Testing, standards are being formalized for interoperability between diverse Cloud offerings too.
- Disaster Recovery Testing: It is preferred that a Cloud offering is available all the time, though it is not 100% achievable even for on-premise applications. In case of a failure, the disaster recovery time must be low. Verification must be done to ensure the service is back online with minimum adverse effects on the client's business.
- Multi-tenancy Testing: Multi-tenancy refers to multiple clients and organizations using an on-demand offering. Considering the requirements to be verified for multi-tenancy, the offering should be customizable for each client and should provide data and configuration level security to avoid any access related issues. A Cloud offering should be thoroughly validated for each client whenever multiple clients are active at a given time
- Unit Testing: this kind of testing is demanded to the specific components of the framework. This ensures that all the framework components will provide their specific functional requirements.

Different testing methods mentioned before were successfully applied in different kinds of domains and, the SDI4Apps framework project represents a well-defined starting point for the definition of our custom solution.

3 TESTING METHODOLOGY PIPE-LINE ADOPTED IN SDI4APPS PROJECT

Conventional testing methodologies are ideal for on-premise applications and environments. Any product or service offered on-premise or on Cloud should meet its functional requirements. Specifically for Cloud offerings, equal (or may be more) emphasis is required on non-functional requirements.

Following both the architecture explained in paragraph 2 and the main goals of the SDI4Apps project, the technical test methodology proposed in this deliverable refers to the verification and validation of applications, environments and infrastructure that are available on demand. This aspect is one of the most important parts of the framework, from the moment in which the SDI4Apps platform acquired different kinds of data from different kinds of resources and make it available dynamically following a common and interoperable structure. As defined in deliverable D.3.1 "Architecture Concept" during testing activity attention will be paid to generic and specific enablers thanks to which the complete interaction between different components will be guaranteed that characterises the SDI4Apps framework and its integration with third parties. In particular to Open Nebula cloud solution as generic enabler and PostgreSQL as specific enabler.

3.1 Functional testing

Functional cloud computing testing is performed for both remote and local applications. Functional cloud testing is the testing of all the features and functions of a system which includes hardware and software. It is conducted on a complete, integrated software platform to check its compliance with the requirements. In functional cloud testing, the process of verification against system specifications or requirements is carried out in cloud instead of on-premise software testing. Functional testing is not comprehensive enough to identify all the combinations of circumstances a site will be subjected to and its performance under stress conditions.

3.1.1 Component functional testing

Component functional unit testing techniques are used to prove its functional behaviour within its own boundaries. It is critical to prove that the system as whole functions as it has been designed when the single components work together, and the inputs and outputs are as expected.

3.1.2 Integration testing

Integration cloud testing allows the business to verify that the cloud solution will work within the current infrastructure and environments which ultimately prove that the implementation of a cloud solution does not detrimentally impact any existing systems. Finally, the business requirements must be verified and validated to prove that the end result of the Cloud solution will meet the documented needs of the business. In the SDI4Apps framework will follow a top down approach starting from CERIT-SC cloud infrastructure. Testing will start checking the interaction between Open Nebula and components adopted for data management: PostgreSQL and Sesame Triplestore. Each component will be tested:

1. PostgreSQL
   - Accessibility
   - Security protocols
   - Scalability
   - Performance
   - PostGIS templates integration and data management
2. Sesame triple store
   - Interoperability with PostgreSQL
     - Metadata mapping (D2RQ library)
   - Metadata management
   - Accessibility
     - SPARQL query performance

Furthermore will be tested the integration and interoperability between Web Map Service (WMS) and PostgreSQL-PostGIS via mapserver. In order to check the proper functionality of the system, all the functional operations will be individually tested.

3.1.3 User acceptance testing

User Acceptance Testing will be done to prove that delivered cloud solution meets business requirements so that the user accepts the developed cloud solution. User acceptance testing is done on both on-premise and Software as a Service[5] (SAAS). However, the onsite testing allows immediate control and monitoring of test progress.

3.2 Non-functional testing

Non-functional testing is done to ensure that an application meets its all requirements not covered in the list of functional requirements. In a cloud framework the most important non-functional testing involve system testing, business requirements testing, performance and scalability testing.

[5] Off-premise software

3.2.1 System testing

In the SDI4Apps infrastructure system testing will be made directly on the CERIT-SC cloud infrastructure and it will take in exam the functional design of the system. Will be consider:

1. CPU-bound performance
    1. Will be performed intensive read and write operations on map image.
    2. Will be performed different geo-processing operations.
2. RAM-bound performance
    1. Management of graph in triple store.
    2. Timing of generation of different map images in memory and their restitution on client server.
3. Disk-bound performance
    1. Reading of data files.
    2. Intensive read and write operations on the PostgreSQL platform.

3.2.2 Business requirements testing

Before migrating their business to a cloud computing solution, the organisations and cooperating parties must carefully analyse and document their business requirements clearly, precisely and unambiguously. Business requirements are foundations for building a cloud computing solution. These business requirements can be achieved through reviews, periodical customer meetings and workshops. Later, this in turn provides that a perfect system is constructed which is capable of delivering the business requirements.

3.2.3 Cloud scalability and performance testing

Cloud Scalability is another major area of concern where adequate amount of testing is needed. Cloud Computing solutions always claim to be scalable on demand. Load or Stress testing can be used to prove that the developed cloud solution can be scale as required with the help of software tools. Hence Cloud solution can be accurately measured and capacity verified. Cloud Performance testing techniques allow us to measure the cloud system performance accurately. Performance testing with the load testing techniques allows getting an accurate image of the solution’s ability on the cloud. Performance is generally tied to an application’s capabilities within the cloud infrastructure. Finding out thresholds, bottlenecks and limitations is a part of performance testing. For this, testing performance under a particular workload and vary the nature of traffic on-demand is necessary.

- Cloud Load and stress testing → Application stability is a major factor as the user count is expected to increase. Load testing of an application involves creation of heavy user traffic and measuring its response. There is also a need to tune the performance of any application to meet certain standards. Measure response times and isolate issues related to specific actions while system is subjected to increasing load from different locations and multi user operations. Stress Test is used to determine ability of application to maintain a certain level of effectiveness beyond breaking point or maximum expected capacity or beyond the expected usage. It is essential for any application to work even under excessive stress and maintain stability. Stress testing assures this by creating peak loads using simulators.

- Latency Testing →Cloud testing is utilized to measure the latency between the action and the corresponding response for any application after deploying it on cloud.

3.3 Ability testing techniques

Ability testing is done to ensure that the cloud environment is able to gives its service on-demand to users. In this category, the compatibility, interoperability and multi-tenancy ability of cloud computing environment is tested

3.3.1 Compatibility and interoperability testing

In cloud environment, different software's and operating systems is used and created on demand which makes the compatibility testing must. A cloud application must capable to work and executed across multiple environments and various cloud platforms. Hence, it is very easy to migrate cloud applications and platforms from one infrastructure to another infrastructure

3.3.2 Disaster recovery

The cloud service provider always prefers that its cloud services should be available all the time to end-users but actually it is not achievable. There may be some chance of failure so the disaster recovery time must be low. Cloud verification must be done to ensure the service is back online with minimum adverse effect on business.

3.3.3 Multi-tenancy testing

Multi-tenancy testing ensures that the multiple clients and organisations using on-demand services activated at a given time. Cloud service should be customisable for each client and provide data and security level to avoid any access related issues.

3.4 Data quality testing

In parallel with functional and non-functional testing also the quality of metadata managed by triple store will be checked. The quality of the information derived from different kind of resources and the level of semantical definition of the information inferred during the normalization process will be examined.

Data quality testing will interest mainly data schema and style sheet defined in order to guarantee metadata mapping.

4. CONCLUSION

The analysis and definition of new cloud testing methodology is becoming one of the most important topics among new research activities as witnessed by many projects carried out during the last years. However, despite the huge quantities of cloud based projects and frameworks, it's very difficult to find a well-defined common testing methodology[6].

Despite this gap, thanks to the subdivision in functional and non-functional testing proposed in our project we are able to define a common testing strategy able to work both on a general architecture and single components.

This approach matches completely with a scalable and interoperable architecture as we can find in the SDI4Apps framework.

6 On the one hand, we have a lot of powerful well-defined testing methods that can be adapted on a specific architecture. On the other hand, it's very difficult to find a common methods that can be applied to different architectures.

REFERENCES

Crispin L., Gregory J., “Agile Testing: A practical Guide for tester and Agile Teams”, 2008

Jerry G., Xiaoying B., Wei-Tek T., “Cloud Testing-Issues, Challenges, Needs and Practice”, SEIJ, September2011

Pundhir Y.S, “Cloud computing applications and their testing methodology”, Bookman International Journal of Software Engineering, Vol.2 No.1, March 2013

Vanitha Katherine A., Alagarsamy K.,”Software Testing in Cloud Platform: A Survey”, IJCA, May2012