




---

## SeaClouds Project

### D6.4.2 - SeaClouds periodic evaluation reports

---

Project Acronym	SeaClouds
Project Title	Seamless adaptive multi-cloud management of service-based applications
Call identifier	FP7-ICT-2012-10
Grant agreement no.	610531
Start Date	1 <sup>st</sup> October 2013
Ending Date	31 <sup>st</sup> March 2016
Work Package	WP6 Case Studies implementation and Validation
Deliverable code	D6.4.2
Deliverable Title	SeaClouds periodic evaluation reports
Nature	Report
Dissemination Level	Public
Due Date:	M24
Submission Date:	
Version:	1.0
Status	Draft
Author(s):	Francesco D'Andria (Atos),
Reviewer(s)	Ernesto Pimentel (UMA), Antonio Brogi (UPI)

Dissemination Level

Project co-funded by the European Commission within the Seventh Framework Programme		
	Public	X
	Restricted to other programme participants (including the Commission)	
	Restricted to a group specified by the consortium (including the Commission)	
	Confidential, only for members of the consortium (including the Commission)	

## Table of Contents

1. Executive Summary .....	4
2. Introduction .....	5
3. Functional Evaluation .....	6
4.1 Evaluation Scenario ES01: Local environment .....	7
4.2 Evaluation Scenario ES02: Cloud environment .....	8
4. Non-Functional Evaluation .....	19
4.1 Documentation Testing .....	19
4.2 Installation Testing .....	20
4.2.1 Local Installation test .....	20
4.2.2 Local and Cloud Installation test .....	24
4.3 Interoperability testing .....	25
4.3.1 Int.1.1 .....	26
4.3.2 Int.1.2 .....	26
4.3.3 Int.1.3 .....	27
4.4 Usability Testing .....	28
4. Conclusions and next steps .....	35
5. References .....	36
Annex A. Applications descriptions .....	37
A1. ATOS Case Study .....	37
A2. NURO Case Study .....	39
Annex B. Testing Tools/Software .....	41
B1. HTTP link checkers .....	41
B2. Specific testing tools .....	41
B3. Web-services testing tools .....	42
B5. Tools for Performance/Scalability testing .....	42
Annex C. Test results report format .....	43

## 1. Executive Summary

The task 6.3 of the WP6 is responsible for making an assessment of the quality (effectiveness, efficiency and user satisfaction) of the SeaClouds platform solution, evaluating it from qualitative and quantitative points of view.

This deliverable, the D6.4.2, is the second version of the D6.4.X saga. It will implement the testing activity and reports the results of tests introduced in the D6.4.1 [1], where testing methodology was introduced.

Also, additional information about the different testbeds set-up to validate SeaClouds is added.

## 2. Introduction

The task 6.3 of the WP6 is responsible for the evaluation of the SeaClouds software platform. The Deliverable D6.4.2 includes the results of the first cycle of evaluation measurements and analysis of SeaClouds, evaluating both quantitative and qualitative requirements.

After the selection of tests methods and evaluation scenarios, defined in the deliverable D6.4.1 [1], it is now necessary to define a selection of tools and testbed set-ups to execute those tests and evaluate if the proposed SeaClouds solution fulfills the different requirements defined at the beginning of the project. This deliverable is strongly connected to WP2 to WP5:

- WP2 and WP6 define the technical requirements for SeaClouds. The technical requirements are based on a set of use cases, developed by the partners of the project. The WP2 also highlight the SeaClouds high-level architecture.
- WP3 and WP4 are the responsible of the low-level design and implementation of the SeaClouds design-time and run-time tools. Both components are going to be tested in the context of the WP6.
- The WP5 is responsible for the low-level design and implementation of the SeaClouds GUI.

The conclusions of this document will help to see the actual status of the project and what it is still missing to fulfil the requirements and objectives defined at the beginning of it.

This document is organized as follows:

Section 3 introduces the results of the functional evaluation. A detailed description of all these tests, and the reason way they are done can be found in deliverable D6.4.1 [1]. The objective in this case is to see if SeaClouds fulfils all the functional requirements established at the beginning of the project in deliverable D2.1 [2].

Section 4 presents the results of the non-functional evaluation. A detailed description of all these tests, and the reason way they are done can be found in deliverable D6.4.1 [1]. The objective in this case is to see if SeaClouds fulfils all the non-functional requirements established in D6.4.1.

### 3. Functional Evaluation

The functional evaluation of SeaClouds tries to verify if the software solution as result of this project fulfils the functional requirements established at the beginning of it.

These functional requirements were defined in the deliverable D2.1 – Resubmission [2]. In D2.1 the SeaClouds consortium defined also a list of Use Cases that include a list of steps, which define interactions between actors and the SeaClouds platform as well the internal interactions performed by the SeaClouds platform to provide the overall functionalities.

In this deliverable two different evaluation scenarios (SeaClouds deployment configurations) will be defined to test the SeaClouds functionality: local installation, cloud installation.

**Table 1: Use cases for local testbed**

UC	Local Test Bed
UC1	Design an Application
UC2	Show Cloud offers
UC3	Produce Deployment Plans
UC4	Generate SLA Agreement
UC5	Deploy an Application (on a IaaS and on a PaaS)
UC6	Monitor an Application
UC7	Evaluate Management Policies
UC8	Re-plan Application Deployment
UC9	Migrate Application

**Table 2: Use cases for cloud testbed**

UC	Cloud Test Bed
UC1	Design an Application
UC2	Show Cloud offers
UC3	Produce Deployment Plans
UC4	Generate SLA Agreement
UC5	Deploy an Application (on a IaaS and on a PaaS)

UC6	Monitor an Application
UC7	Evaluate Management Policies
UC8	Re-plan Application Deployment
UC9	Migrate Application

## 4.1 Evaluation Scenario ES01: Local environment

<quite short introduction of the testbed... something like what Andrea showed during the last integration meeting ... two Vms installed locally.. where is locate each SeaClouds component... a diagram may help>

Use Case ID	LUC1
Use Case Name	Design an Application
Purpose	The purpose of this test is design the topology of an application using the SeaClouds GUI <please feel free to add new text>
Initiator	The Software Developer
Primary Actor	The Software Developer
Additional Actors	SeaClouds Operator?
Description	<please here describe the test>
Pre-condition	I suppose the SeaClouds platform is installed correctly; the profile is registered in SeaClouds etc
Post-condition	We finally have designed the topology of the app
Use Case Functionality	
Sequence	Please describe here the sequence....
Alternative	

Use Case ID	LUC2
Use Case Name	Show Cloud offers
Purpose	The purpose of this test is Show Cloud offers <please feel free to

	add new text>
<b>Initiator</b>	The Software Developer
<b>Primary Actor</b>	The Software Developer
<b>Additional Actors</b>	SeaClouds Operator?
<b>Description</b>	<please here describe the test>
<b>Pre-condition</b>	I suppose the SeaClouds platform is installed correctly; the profile is registered in SeaClouds etc we have designed a valid application topology
<b>Post-condition</b>	We finally have cloud offers
<b>Use Case Functionality</b>	
<b>Sequence</b>	Please describe here the sequence....
<b>Alternative</b>	

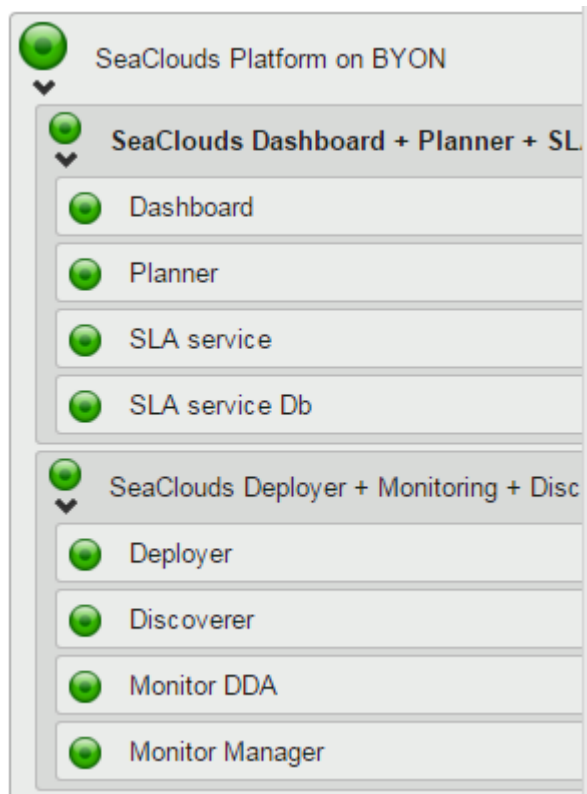
## 4.2 Evaluation Scenario ES02: Cloud environment

The testbed has been prepared in LeaseWeb provider. It consists of two VMs with the following characteristics:

- (cpu info)
- 4GB RAM
- 688GB HDD
- OS: Ubuntu 12.04

The components inside the two VMs have been distributed as shown in the following diagram:



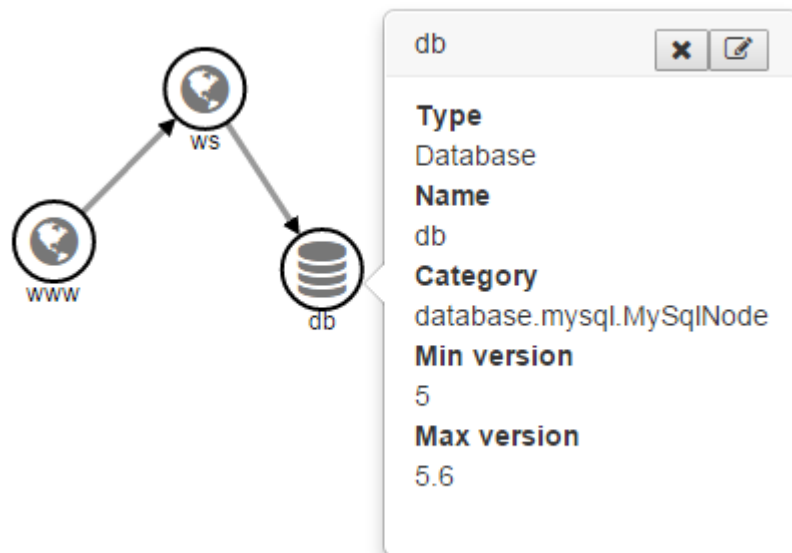


<b>Use Case ID</b>	CEUC1
<b>Use Case Name</b>	Design an Application
<b>Purpose</b>	The purpose of this test is to design the topology and requirements of an application using the SeaClouds GUI.
<b>Initiator</b>	The Software Developer
<b>Primary Actor</b>	The Software Developer
<b>Additional Actors</b>	
<b>Description</b>	<p>The test will cover the design of the topology of the Atos case study, which consists of:</p> <ul style="list-style-type: none"> <li>Frontend module. The technical requirements are: <ul style="list-style-type: none"> <li>Language: Java <math>\geq 7</math></li> <li>To be deployed on PaaS</li> <li>Container: Tomcat</li> <li>Uses the Web Services module 2 times per call in average.</li> </ul> </li> <li>Web services module. The technical requirements are: <ul style="list-style-type: none"> <li>Language: Java <math>\geq 7</math></li> <li>To be deployed on PaaS</li> </ul> </li> </ul>

	<ul style="list-style-type: none"> <li>○ Container: Tomcat</li> <li>○ Uses the database 2.5 times per call in average</li> <li>• Database. The technical requirements are: <ul style="list-style-type: none"> <li>○ MySQL &gt;= 5</li> <li>○ To be deployed on PaaS</li> </ul> </li> </ul> <p>Additionally, the following requirements have been defined:</p> <ul style="list-style-type: none"> <li>• Maximum Response Time: 2000 ms</li> <li>• Availability: 98%</li> <li>• Budget per month: 200 €</li> </ul> <p>The expected workload of the application is 50 requests/second.</p>
<b>Pre-condition</b>	<p>The SeaClouds platform is correctly installed.</p> <p>The browser has the SeaClouds Dashboard loaded.</p>
<b>Post-condition</b>	<p>The topology of the application described above is correctly defined.</p>
<b>Use Case Functionality</b>	
<b>Sequence</b>	<ol style="list-style-type: none"> <li>1. Click the "New application" button</li> <li>2. Fill the application properties <ol style="list-style-type: none"> <li>a. Fill the application name: Atos</li> <li>b. Fill the optimization properties <ol style="list-style-type: none"> <li>i. Response time: 2000</li> <li>ii. Availability: 98</li> <li>iii. Cost: 200</li> <li>iv. Workload: 50</li> </ol> </li> </ol> </li> <li>3. Click next</li> <li>4. Define the topology <ol style="list-style-type: none"> <li>a. Click Web Application button and fill the properties for the frontend module <ol style="list-style-type: none"> <li>i. Name: www</li> <li>ii. Language: Java</li> <li>iii. Min version: 7</li> <li>iv. Max version: 8</li> <li>v. Code container: Tomcat</li> <li>vi. Provider is: PaaS</li> <li>vii. Location: None</li> </ol> </li> <li>b. Click Add. The module is added</li> <li>c. Click Web Application button and fill the properties for the web services module <ol style="list-style-type: none"> <li>i. Name: ws</li> <li>ii. Language: Java</li> <li>iii. Min version: 7</li> <li>iv. Max version: 8</li> </ol> </li> </ol> </li> </ol>

- v. Code container: Tomcat
- vi. Provider is: PaaS
- vii. Location: None
- d. Click Add. The module is added
- e. Click Database button and fill the properties for the database
  - i. Name: db
  - ii. Category: MySQL
  - iii. Min version: 5
  - iv. Max version: 5.6
  - v. Provider is: PaaS
  - vi. Location: None
- f. Click Add. The module is added.
- g. Shift+Click on www and drag to ws; fill the properties for the link
  - i. Average number of calls: 2
- h. Click Edit. The link is added.
- i. Shift+Click on ws and drag to db; fill the properties for the link
  - i. Average number of calls: 2.5
- j. Click Edit. The link is added.

The result is shown in the following figure:



Alternative

Use Case ID CEUC2

Use Case Name Show Cloud offers

<b>Purpose</b>	The purpose of this test is to check that the cloud offerings provided by the planner match the technical requirements expressed in the topology.
<b>Initiator</b>	The Software Developer
<b>Primary Actor</b>	The Software Developer
<b>Additional Actors</b>	
<b>Description</b>	The test will cover the correctness of the plans generated by the planner for the application topology of the Atos case study. The generated plans should contain offerings matching the application requirements.
<b>Pre-condition</b>	<p>The SeaClouds platform is correctly installed.</p> <p>The browser has the SeaClouds Dashboard loaded.</p> <p>We have designed a valid application topology.</p>
<b>Post-condition</b>	<p>An Abstract Application Model (AAM) is generated, is specified in TOSCA and contains the technical requirements expressed in the topology.</p> <p>We finally have a set of cloud offers that matches the application requirements.</p> <ul style="list-style-type: none"> <li>• For www module, a PaaS offering Java <math>\geq 7</math></li> <li>• For ws module, a PaaS offering Java <math>\geq 7</math></li> <li>• For mysql module, a PaaS offering mysql <math>\geq 0</math></li> </ul>
<b>Use Case Functionality</b>	
<b>Sequence</b>	<ol style="list-style-type: none"> <li>1. Create application topology as in CEUC1.</li> <li>2. Click next</li> <li>3. Review generated Abstract Application Model</li> <li>4. Review offerings provided by planner</li> </ol>
<b>Alternative</b>	
<b>Result</b>	<p>The AAM is generated. It contains the technical requirements expressed in the topology.</p> <p>The planner does not return a set of cloud offerings. The needed feature is implemented but not integrated.</p>

<b>Use Case ID</b>	CEUC3
<b>Use Case</b>	Produce Deployment Plans

<b>Name</b>	
<b>Purpose</b>	The purpose of this test is checking that a deployment plan expressed in TOSCA is generated for the plan selected by the user.
<b>Initiator</b>	The Software Developer
<b>Primary Actor</b>	The Software Developer
<b>Additional Actors</b>	
<b>Description</b>	The test will cover the generation of a deployment plan following the TOSCA specification, which should declare that each module is going to be deployed in the selected offering, the SLA agreement and the monitoring rules.
<b>Pre-condition</b>	<p>The SeaClouds platform is correctly installed.</p> <p>The browser has the SeaClouds Dashboard loaded.</p> <p>The user has designed a valid application topology.</p> <p>The user has selected a plan.</p> <p>The user have entered the credentials of the cloud providers</p>
<b>Post-condition</b>	<p>A Deployable Application Model (DAM) is generated, is specified in TOSCA and contains the cloud offerings selected by the user.</p> <p>The credentials for each provider are included in the DAM.</p> <p>An identifier of the generated monitoring rules is included in the DAM.</p> <p>An identifier of the generated SLA agreement is included in the DAM.</p>
<b>Use Case Functionality</b>	
<b>Sequence</b>	<ol style="list-style-type: none"> <li>1. Select plan as in CEUC2.</li> <li>2. Click next</li> <li>3. Enter provider credentials</li> <li>4. Click deploy</li> </ol>
<b>Alternative</b>	
<b>Result</b>	This use case cannot be evaluated because it depends on CEUC2. The needed feature is implemented but not integrated.

<b>Use Case ID</b>	CEUC4
--------------------	-------

<b>Use Case Name</b>	Generate SLA Agreement
<b>Purpose</b>	The purpose of this test is checking that a WS-Agreement agreement is generated.
<b>Initiator</b>	The Software Developer
<b>Primary Actor</b>	The Software Developer
<b>Additional Actors</b>	
<b>Description</b>	This test will cover the correctness of the SLA agreement generated for the ATOS case study.
<b>Pre-condition</b>	<p>The SeaClouds platform is correctly installed.</p> <p>The browser has the SeaClouds Dashboard loaded.</p> <p>The user has designed a valid application topology.</p> <p>The user has selected a plan.</p>
<b>Post-condition</b>	<p>An agreement following WS-Agreement is generated.</p> <p>It contains a guarantee term to assess the desired availability of the application.</p> <p>It contains a guarantee term to assess the desired response time of the application.</p>
<b>Use Case Functionality</b>	
<b>Sequence</b>	<ol style="list-style-type: none"> <li>1. Select plan as in CEUC2.</li> <li>2. Click next</li> <li>3. Click check SLA agreement</li> </ol>
<b>Alternative</b>	
<b>Result</b>	This use case cannot be evaluated because it depends on CEUC2. The needed feature is implemented but not integrated.

<b>Use Case ID</b>	CEUC5
<b>Use Case Name</b>	Deploy an Application on a PaaS
<b>Purpose</b>	The purpose of this test is checking the correct deployment of the deployment plan.

<b>Initiator</b>	The Software Developer
<b>Primary Actor</b>	The Software Developer
<b>Additional Actors</b>	
<b>Description</b>	This test will cover the deployment of an application in PaaS providers. The topology of the Atos case study defined all the modules to be deployed on PaaS.
<b>Pre-condition</b>	<p>The SeaClouds platform is correctly installed.</p> <p>The browser has the SeaClouds Dashboard loaded.</p> <p>The user has designed a valid application topology.</p> <p>The user has selected a plan where all providers are PaaS providers.</p>
<b>Post-condition</b>	<p>Module www is deployed.</p> <p>Module ws is deployed.</p> <p>A MySQL service for the mysql module is created.</p> <p>The endpoint of ws has been configured for www</p> <p>The MySQL service has been bound to ws.</p> <p>The endpoint, database and credentials of the service have been configured for ws.</p>
<b>Use Case Functionality</b>	
<b>Sequence</b>	<ol style="list-style-type: none"> <li>1. Select plan as in CEUC2.</li> <li>2. Click next</li> <li>3. Enter provider credentials</li> <li>4. Click deploy</li> </ol>
<b>Alternative</b>	
<b>Result</b>	This use case cannot be evaluated because it depends on CEUC2. The needed feature is implemented but not integrated.

<b>Use Case ID</b>	CEUC6
<b>Use Case Name</b>	Monitor an Application
<b>Purpose</b>	The purpose of this test is checking that SeaClouds is able to monitor a

	deployed application.
<b>Initiator</b>	The Software Developer
<b>Primary Actor</b>	The Software Developer
<b>Additional Actors</b>	
<b>Description</b>	This test will cover the monitoring of the application by Tower 4Clouds and the visualization of the monitoring metrics in the SeaClouds dashboard.
<b>Pre-condition</b>	A deployed application
<b>Post-condition</b>	The status view of the dashboard show relevant metrics for the Atos case study.
<b>Use Case Functionality</b>	
<b>Sequence</b>	(hablar con Adrián)
<b>Alternative</b>	
<b>Result</b>	The result is successful.

<b>Use Case ID</b>	CEUC7
<b>Use Case Name</b>	Evaluate Management Policies
<b>Purpose</b>	The purpose of this test is checking the policies management in the SeaClouds platform.
<b>Initiator</b>	The Software Developer
<b>Primary Actor</b>	The Software Developer
<b>Additional Actors</b>	
<b>Description</b>	
<b>Pre-condition</b>	
<b>Post-condition</b>	
<b>Use Case Functionality</b>	
<b>Sequence</b>	
<b>Alternative</b>	



<b>Result</b>	
---------------	--

<b>Use Case ID</b>	CEUC8
<b>Use Case Name</b>	Re-plan Application Deployment
<b>Purpose</b>	The purpose of this test is checking the replanification feature of the SeaClouds platform.
<b>Initiator</b>	The Software Developer
<b>Primary Actor</b>	The Software Developer
<b>Additional Actors</b>	
<b>Description</b>	
<b>Pre-condition</b>	
<b>Post-condition</b>	
<b>Use Case Functionality</b>	
<b>Sequence</b>	
<b>Alternative</b>	
<b>Result</b>	The feature is not implemented.

<b>Use Case ID</b>	CEUC9
<b>Use Case Name</b>	Migrate Application
<b>Purpose</b>	The purpose of this test is checking the migration feature of the SeaClouds platform.
<b>Initiator</b>	The Software Developer
<b>Primary Actor</b>	The Software Developer
<b>Additional Actors</b>	
<b>Description</b>	
<b>Pre-condition</b>	

<b>Post-condition</b>	
<b>Use Case Functionality</b>	
<b>Sequence</b>	
<b>Alternative</b>	
<b>Result</b>	The feature is not implemented.

## 4. Non-Functional Evaluation

While the Functional Requirements specify the set of functions that the SeaClouds system or system component must be able to perform, the Non-Functional Requirements express desired qualities of a problem solution other than those concerning its functionality, e.g. its robustness, its efficiency, its security, its extensibility, its maintainability, its portability, etc.

In the SeaClouds project those Non-Functional Requirements were defined in D6.4.1 [1]. Actually the chapter 5 of D6.4.1 presents a list of testing methods to evaluate the SeaClouds platform. However, at project months 24 (PM24) the SeaClouds system is not yet mature enough to be fully evaluated under the point of view of some of the Non-Functional Requirements listed in D6.4.1. The following table summarizes, on one hand the tests the consortium is going to report in this document, and the test the consortium plans to perform at project months 30, on the other hand.

Non-Functional Requirements tests at M24	Non-Functional Requirements tests at M30
	Performance/Scalability Testing
	Stress Testing
Documentation Testing	Documentation Testing
Local and in Cloud Installation Testing	Local and in Cloud Installation Testing
	Regression Testing
	Long Term Testing
Interoperability testing	Final Interoperability testing
Early Usability Testing	Early Usability Testing

For every method described in D6.4.1, and listed in the left side on the previous table, this chapter specifies the characteristics of the test the environment that host the execution environment and the tools that are necessary to perform the tests (in some cases, the tests use no tools or testbed at all, this will be clarified later).

### 4.1 Documentation Testing

Documentation testing means verifying that the SeaClouds documentation user manuals, including guidelines, tutorials and on-line documentation- are easy to read and understand, grammatically correct and technically accurate.

<b>Test ID</b>	Documentation Testing (DT)	<b>Date</b>	09/10/2015
<b>Tester</b>	Michele Guerriero (Polimi)	<b>Testbed name</b>	Local Testbed and Cloud Testbed.
<b>SeaClouds Platform Version</b>	0.8.0-SNAPSHOT		

SeaClouds documentation Version	README.md from SeaCloudsPlatform 0.8.0-SNAPSHOT <a href="https://github.com/SeaCloudsEU/SeaCloudsPlatform/blob/master/README.md">https://github.com/SeaCloudsEU/SeaCloudsPlatform/blob/master/README.md</a>		
Test Results			
Involved Components	SeaClouds Dashboard, SeaClouds SLA, SeaClouds Monitorr SeaClouds, SeaClouds Discoverer, SeaClouds Planner, SeaClouds Deployer		
Interaction Between Components	Not tested here.		
Passed?	Yes	Bug ID	na
Problems: general observations	The tested documentation just has a missing link at the beginning under the section “Getting Started”.		
Required Changes: specific changes to be made	Remove the missing link in section “Getting Started”		
Cost Estimation	Low		
Comments	There are no issues, everything reported in the current (09/10/2015) documentation available from the github repository of SeaCloudsPlatform worked fine.		

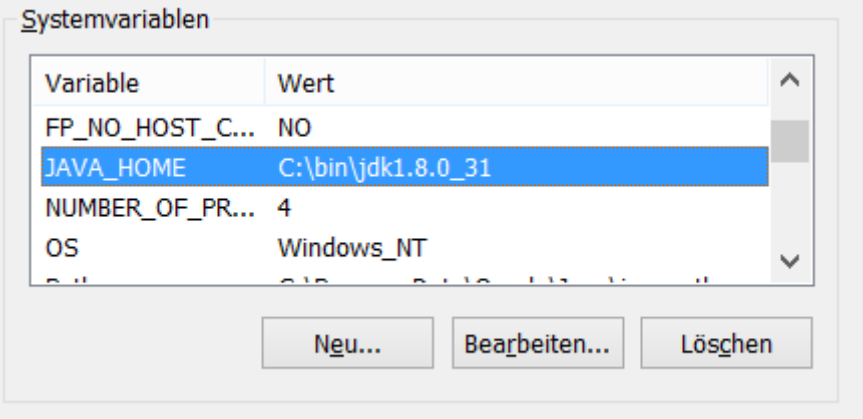
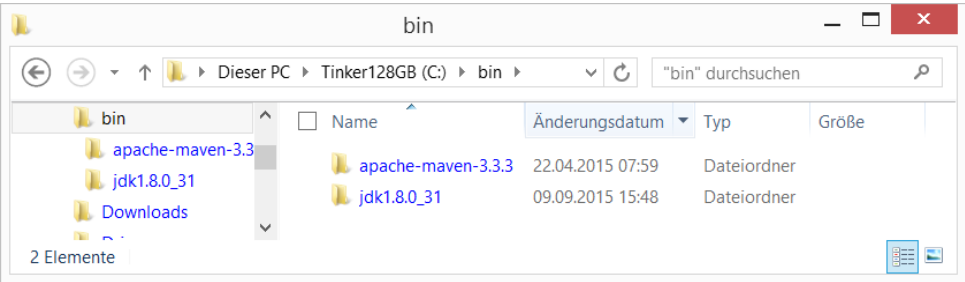
## 4.2 Installation Testing

Installation testing verifies the correct work of the installation procedure of SeaClouds in different configuration environments. The actual report presented here reflects conclusions extracted from two testers in different environments and configurations: Local installation and installation on a Cloud infrastructure.

### 4.2.1 Local Installation test

<b>Test ID</b>	Local Installation Testing (LIT)	<b>Date</b>	11/09/2015
<b>Tester</b>	Chrsitian Tismer	<b>Testbed name</b>	Local environment/Windows
<b>SeaClouds Platform Version</b>	Milan GA Sept. 2015 Version		

Test Results	
<b>Involved Components</b>	Any SeaClouds components deployed / launched using Apache Brooklyn. We currently support deployments against Bring Your Own Nodes (BYON) and to all the IaaS provider supported by Apache jclouds.
<b>Environment characteristics</b>	<p>SeaClouds Windows Installation: this installation is based on the SeaClouds installation guide:  <a href="https://github.com/SeaCloudsEU/SeaCloudsPlatform/tree/master/usage/installer">https://github.com/SeaCloudsEU/SeaCloudsPlatform/tree/master/usage/installer</a></p> <p>the installation is done on a laptop with this characteristics:</p> <ul style="list-style-type: none"> <li>• Windows 8.1 (64bit)</li> <li>• 4GB Ram</li> <li>• i3 4030u (1,9 GHz dual core)</li> <li>• 100GB SSD (NTFS compression on)</li> </ul>
<b>Pre-requirements</b>	<p>Software to be installed and configured before to perform the SeaClouds installation:</p> <p><b>git</b>  I have installed git 1.9.5 with global bash support  <a href="https://git-scm.com/download/win">https://git-scm.com/download/win</a></p> <p>As gui I installed TortoiseGit 1.8.14.0  <a href="https://tortoisegit.org/download/">https://tortoisegit.org/download/</a></p> <p><b>bash</b>  to use bash shell scripts with windows you need to install something  I've chosen the git solution (see above), Roman uses Cygwin</p> <p><b>java sdk</b>  Installed "jdk1.8.0_31"  <a href="http://www.oracle.com/technetwork/java/javase/downloads/index.html">http://www.oracle.com/technetwork/java/javase/downloads/index.html</a></p> <p><b>maven</b>  Maven has some issues with Blanks in pathes, thus I copied JDK to C:\bin\  I installed maven also to C:\bin\  I need to set environment JAVA_HOME to "C:\bin\jdk...."  Also PATH needs to be extended by "...;C:\bin\apache-maven-3.3.3\bin\  Download and further information:  <a href="https://maven.apache.org/guides/getting-started/windows-prerequisites.html">https://maven.apache.org/guides/getting-started/windows-prerequisites.html</a></p>

	  <p><b>virtualbox</b> <a href="https://www.virtualbox.org/wiki/Downloads">https://www.virtualbox.org/wiki/Downloads</a></p> <p><b>vagrant</b> <a href="https://docs.vagrantup.com/v2/installation/">https://docs.vagrantup.com/v2/installation/</a></p> <p><b>Changes for the Windows deployment:</b></p> <ol style="list-style-type: none"> <li>1. Prerequisites PATH and JAVA_HOME must be done see above.</li> <li>2. start.windows.sh: this is very minimal but works for me, take care for classpath ':' must be changed to ';'         </li> </ol> <pre> JAVA=\$JAVA_HOME/bin/java  JAVA_OPTS="-Dbrooklyn.location.localhost.address=127.0.0.1 \${JAVA_OPTS}"  \$JAVA \${JAVA_OPTS} -Xms256m -Xmx1024m -XX:MaxPermSize=1024m \ -classpath "conf/;patch/*;*;lib/*" \ eu.seacLOUDS.SeaCloudsMain \ launch "\$@" </pre>
<p><b>Installation steps</b></p>	<ol style="list-style-type: none"> <li>3. git clone "<a href="https://github.com/SeaCloudsEU/SeaCloudsPlatform">https://github.com/SeaCloudsEU/SeaCloudsPlatform</a>" to "C:\ScEvaluation\SeaCloudsPlatform"</li> <li>4. cd C:\ScEvaluation\SeaCloudsPlatform</li> <li>5. mvn clean install (took 7 minutes)</li> <li>6. cd usage\installer\target\seacLOUDS-installer-dist\seacLOUDS-</li> </ol>

- installer\byon
7. vagrant up (took 10 minutes)
8. cd ..
9. (created start.windows.sh based on start.sh see below)
10. (modified VM's to use less RAM see below)
11. bash start.windows.sh
12. <http://127.0.0.1:8081/>
13. used YAML from [raw.githubusercontent.com/SeaClouds/blueprints/seacLOUDS-on-byon.yaml](http://raw.githubusercontent.com/SeaClouds/blueprints/seacLOUDS-on-byon.yaml)
14. <http://192.168.100.11:8000/>

Everything Works!

### Minimize memory usage of the virtual machines

Due to memory problems with my Laptop (4GB, both VM need together 3GB) I changed the memory configuration, I set it to 512M and 1G swap.

#### a) Vagrantfile

changed: box.customize [ "modifyvm", :id, "--memory", "1512" ]  
to: box.customize [ "modifyvm", :id, "--memory", "512" ]

#### b) configure swap

I logged into the vms seacLOUDS-0 and seacLOUDS-1 before I start brooklyn (previous step 9)

I created and configured a swapfile

```

C:\WINDOWS\system32\cmd.exe - vagrant ssh se
vagrant@seacLOUDS-0:~$ sudo sh -
# bash
root@seacLOUDS-0:~# free -m
              total        used        free      shared    buffers     cached
Mem:           491         459           32           0          26         362
-/+ buffers/cache:           70         420
Swap:            0             0             0
root@seacLOUDS-0:~# cat /etc/fstab
LABEL=cloudimg-rootfs / ext4 defaults 0 0
root@seacLOUDS-0:~# fallocaate -l 1G /swapfile
root@seacLOUDS-0:~# echo '/swapfile none swap defaults 0 0' >> /etc/fstab
root@seacLOUDS-0:~# mkswap /swapfile
Setting up swapspace version 1, size = 1048572 KiB
no label, UUID=3b89485f-5308-4a5c-afde-a170680f3a99
root@seacLOUDS-0:~# cat /proc/swaps
Filename                                Type                                Size      Used      Priority
root@seacLOUDS-0:~# swapon -a
root@seacLOUDS-0:~# cat /proc/swaps
Filename                                Type                                Size      Used      Priority
/swapfile                               file                                1048572    0         -1
root@seacLOUDS-0:~# free -m
              total        used        free      shared    buffers     cached
Mem:           491         459           31           0          26         362
-/+ buffers/cache:           70         420
Swap:          1023             0         1023
root@seacLOUDS-0:~#

```

based on <https://jeqo.github.io/blog/devops/vagrant-quickstart/>

Passed?

Yes

<b>Problems</b>	<p>The memory limits of my Test Laptop made it impossible to locally start SeaClouds and the use case in parallel. The local SeaClouds installation worked well to deploy our use case in the cloud.</p> <p>Interoperation of all SeaClouds components was not finished thus some YAML must be injected manually.</p>
<b>Required Changes</b>	Bigger test machine or reduction of SeaClouds resource consumption.
<b>Cost Estimation</b>	
<b>Comments</b>	It is possible to locally install SeaClouds on a Windows machine but you need to install some prerequired Software that is not common for this platform.

#### 4.2.2 Local and Cloud Installation test

Installation testing verifies the correct work of the installation procedure of SeaClouds in different configuration environments. The actual report presented here reflects conclusions extracted from two testers in different environments and configurations: Local installation and installation on a Cloud infrastructure.

Test ID	Installation Testing (IT)		Date	09/10/2015
Tester	Michele Guerriero (Polimi)		Testbed name	Local Testbed and Cloud Testbed.
SeaClouds Platform Version		0.8.0-SNAPSHOT		
Test Results				
Involved Components		SeaClouds Dashboard, SeaClouds SLA, SeaClouds Monitorr SeaClouds, SeaClouds Discoverer, SeaClouds Planner, SeaClouds Deployer		
Environment characteristics		<ul style="list-style-type: none"><li>Local Installation:<ul style="list-style-type: none"><li>Intel Core i7-4500U</li><li>8GB DDR3 L Memory</li><li>Ubunut 14.04</li></ul></li><li>Cloud Testbed: AWS-EC2 m3.medium instances (one for each of the involved components).</li></ul>		
Interaction Between Components		Not tested here.		



<b>Installation steps</b>	The steps followed are those reported in the README.md into the SeaCloudsPlatform github repository in date 09/10/2015.		
<b>Passed?</b>	YES		
<b>Problems</b>	NONE		
<b>Required Changes</b>			
<b>Cost Estimation</b>			
<b>Comments</b>	All the components were successfully installed and were reachable both installing the platform locally and on AWS-EC2.		

### 4.3 Interoperability testing

Interoperability is the “ability to work with other systems”. In the context of SeaClouds this means that component integration with external legacy applications, middleware or COTS components should be guaranteed. In this context the role of standards is primary and then we should consider if:

- We are using standardized (open) protocols
- We are proposing extensions, which conform with the protocol
- We are trying to standardize them

In order to better steer the evaluation phase, we will identify specific aspects of interoperability that are relevant with respect to SeaClouds requirements.

We pointed out to two different critical points:

- Internal interoperability. Related to the communication between internal modules belonging to the SeaClouds system: the Deployer Component and a light version of the MODAClouds Monitoring Platform (<http://www.modaclouds.eu/software/monitoring/>).
- External interoperability. Related to the communication with other systems that useful to exploit SeaClouds capabilities: the Discover Component and Paasify (<http://www.paasify.it/vendors>) and CloudHarmony (<https://cloudharmony.com/>) services.

Table 3: interoperability scenarios

Scenario Id	Scenario Description
Int.1.1	Tests will be performed to evaluate the communication between the Deployer Component and a light version of the MODAClouds Monitoring Platform
Int.1.2	Tests will be performed to evaluate the communication between the Discover Component and Paasify.
Int.1.3	Tests will be performed to evaluate the communication between the Discover Component and CloudHarmony .

### 4.3.1 Int.1.1

Table 4: Results of the Interoperability Test Int.1.3

Test ID	Int.1.	Date	09/10/2015
Tester	Michele Guerriero	Testbed ID	Local Testbed
SeaClouds Version	0.8.0 -SNAPSHOT		
Test Results			
Involved Components	SeaClouds Dashboard, SeaClouds SLA, SeaClouds Monitor, SeaClouds Planner.		
Interaction Between Components	The SeaClouds Dashboard automatically install the required monitoring rules for a given application into Tower 4Clouds at deploy time. In the meanwhile the SeaClouds Dashboard also triggers the deploy of the application coupled with all the required data collectors. When the monitoring rules are installed the Dashboard notifies the SeaClouds SLA which at this point starts the SLAs enforcement process by observing the violations occurring on the conditions specified over some QoS metrics. In the meanwhile a the reconfiguration-data-collector monitor the status of each managed application and if one goes down an Tower 4Clouds automatically triggers the replanning process.		
Passed?	Partially	Bug ID	
Problems	The SeaClouds Dashboard currently does not generate automatically the required monitoring rules.  The replanning process is still not implemented and currently it can be just triggered.		
Required Changes	Having the SeaClouds Planner integrated with the SeaClouds DAM Generator and with the Dashboard.  Implement the replanning process after the replanning is triggered		
Cost Estimation	1 month and an half of implementation.		
Comments	-		

### 4.3.2 Int.1.2

Table 5: Results of the Interoperability Test Int.1.2

Test ID	Int.1.2	Date	30/09/2015
---------	---------	------	------------

Tester	Paolo Cifariello	Testbed ID	Local Installation (LI)
SeaClouds Version	0.8.0 (According to github pom file)		
Test Results			
Involved Components	PaaSify spider, PaaSify, SeaClouds Discoverer (crawler manager and CORE).		
Interaction Between Components	The crawler manager of the Discoverer triggers the PaaSify spider to retrieve the list of cloud offerings and their metrics from PaaSify. The PaaSify spider interacts with PaaSify by cloning the github repository where the cloud offerings are located. The cloud offerings in PaaSify are represented using a JSON format. The spider converts the offerings into TOSCA YAML, and sends it to the CORE discoverer, which stores it in the repository of the SeaClouds discoverer.		
Passed?	YES	Bug ID	-
Problems	None.		
Required Changes	Synchronize the taxonomy (e.g. metrics naming, cloud offering naming,etc.) with the rest of components of SeaClouds.		
Cost Estimation	-		
Comments	-		

### 4.3.3 Int.1.3

Table 6: Results of the Interoperability Test Int.1.3

Test ID	Int.1.3	Date	30/09/2015
Tester	Simone Zenzaro	Testbed ID	Local Installation (LI)
SeaClouds Version	0.8.0 (According to github pom file)		
Test Results			
Involved Components	CloudHarmony spider, CloudHarmony, SeaClouds Discoverer (crawler manager and CORE).		
Interaction Between Components	The crawler manager of the Discoverer triggers the CloudHarmony spider to retrieve the list of cloud offerings and their metrics from CloudHarmony. The CloudHarmony		

	spider uses the RESTful API provided by CloudHarmony to interact with it, through HTTP/REST protocol. The spider converts the offerings into TOSCA YAML, and sends it to the CORE discoverer, which stores it in the repository of the SeaClouds discoverer.	
Passed?	Partially	Bug ID
Problems	Some of the offerings provided by CloudHarmony are currently not retrieved	
Required Changes	Retrieve all the offerings of CloudHarmony and improve the list of metrics it gets.	
Cost Estimation	1 month of refactoring.	
Comments	-	

## 4.4 Usability Testing

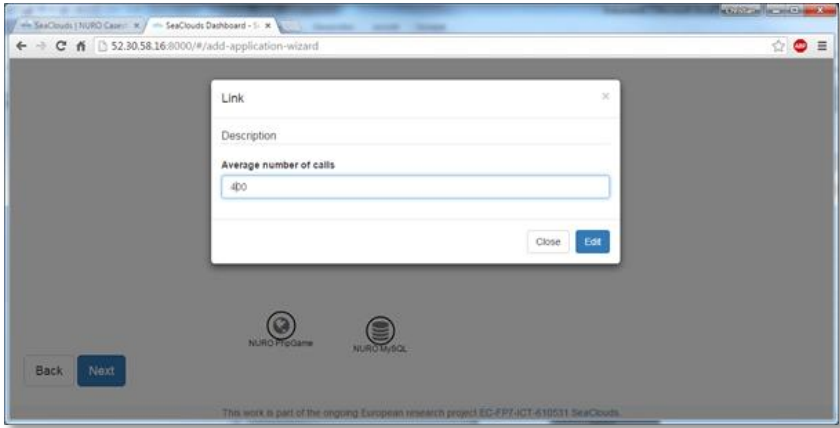
Through time many definitions for usability have been proposed. Two of the most established definitions can be found in international standard for the evaluation of software ISO 9241-11 [1] and ISO 9126 [4].

The Guidance on usability in ISO 9241-11 outlines the usability as “the level to which a (software) product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use”.

On the other hand, in the standard ISO 9126, usability is defined as “the capability of a software product to be understood, learned, used and attractive for the user, when it is used under specified conditions”. In depth, usability studies relate to evaluating a product by testing it on representative users while they focus not only on how well users can learn and use a product to achieve their goals but also on how satisfied users are with that process. This can be seen as an irreplaceable usability practice since it gives direct input on how real users use the system. Usability studies examine three principles: effectiveness, efficiency and overall satisfaction of the user. [6]

In the context of the SeaClouds project usability testing is a perceptual test depending of the tester. No tasks are done in an automatic way. It is important to note that the testers could use one of the testbeds to perform the usability tests or to install the SeaClouds platform in a virtual machine (similar to the installation test). Due to the subjectivity of this test, it is going to be performance by at least two different partners of the project.

<b>Test ID</b>	Usability Testing 1 (UT1)	<b>Date</b>	Week 46 2015
<b>Tester</b>	Christian Tismer (Nuro)	<b>Testbed name</b>	Cloud Testbed

<b>SeaClouds Platform Version</b>	Presentations Cloud Deployment of Oktober 2015
<b>Test Results</b>	
<b>Involved Components</b>	SeaClouds Designer, SeaClouds Dashboard, SeaClouds Monitor
<b>Interaction Between Components</b>	<p>The integration between the Components was not finalized at testing time thus the focus is “Designer” look and feel and “Monitoring” interaction with the NURO case study. Optimizer, Deployment and Replanning is out of focus of this test.</p> <p><b>SeaClouds Dashboard</b></p> <ul style="list-style-type: none"> <li>- Design Wizard: intuitively and fun to use, modules and parameters are suitable for this research level implementation. A real world implementation needs more modules, e.g. load balancer and refined parameters</li> </ul>  <p><b>Figure 1 SeaClouds Application Wizzard</b></p> <ul style="list-style-type: none"> <li>- “Application Model” generation: a great advantage to the previous version. Intuitive and easy to use.</li> </ul>

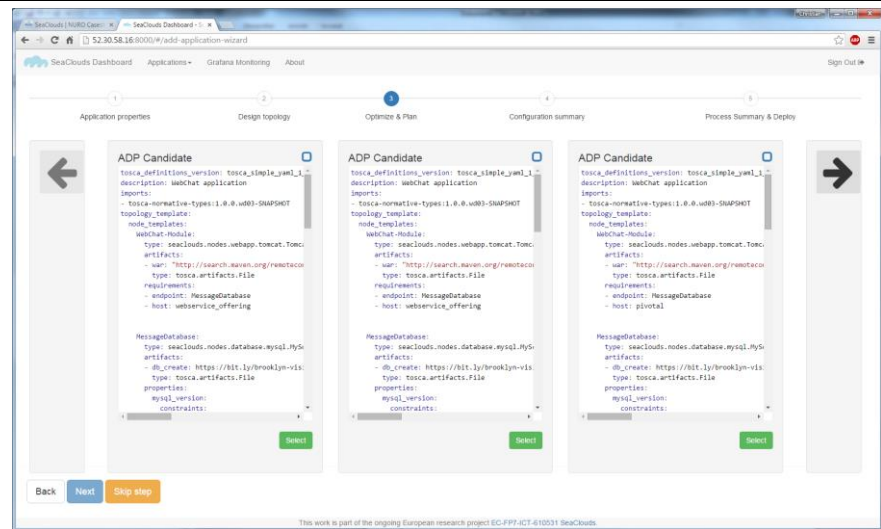


Figure 2 SeaClouds proposed Deployment Candidate Models

- APP Status Overview: intuitively to use

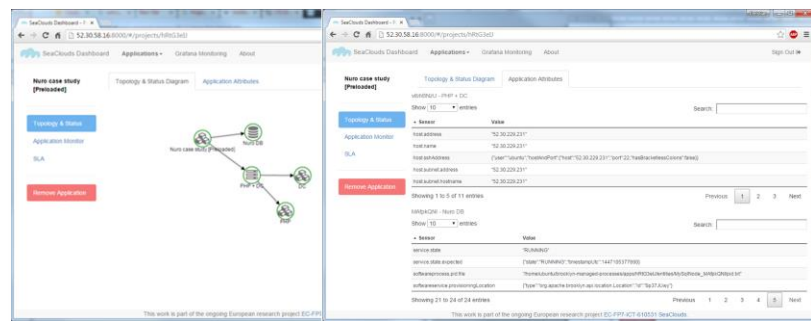
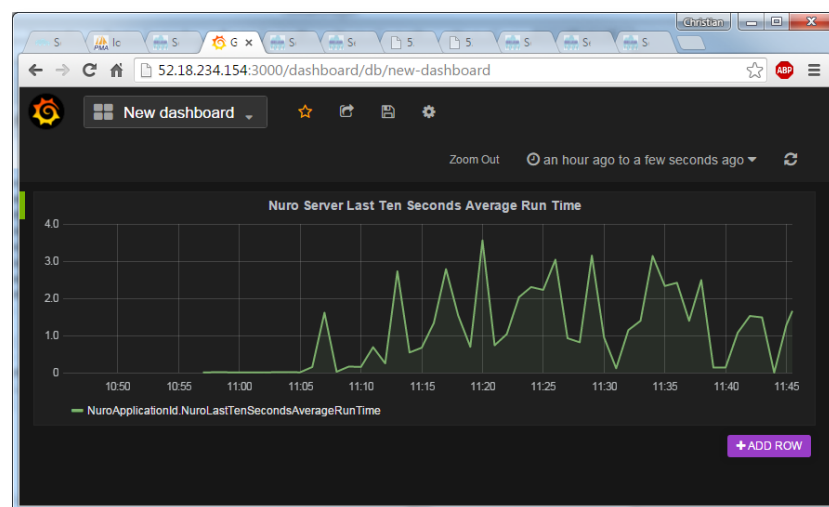


Figure 3 SeaCloud APP Status Overview

- Grafana Monitoring: Worked basically, feels not integrated to the dashboard



## SeaClouds Deployment

Deployment was tested by the partners, it is reported, NURO case study was deployed successfully to all desired test beds.

- private deployment: succeeded
- IaaS deployment: succeeded
- PaaS deployment: succeeded

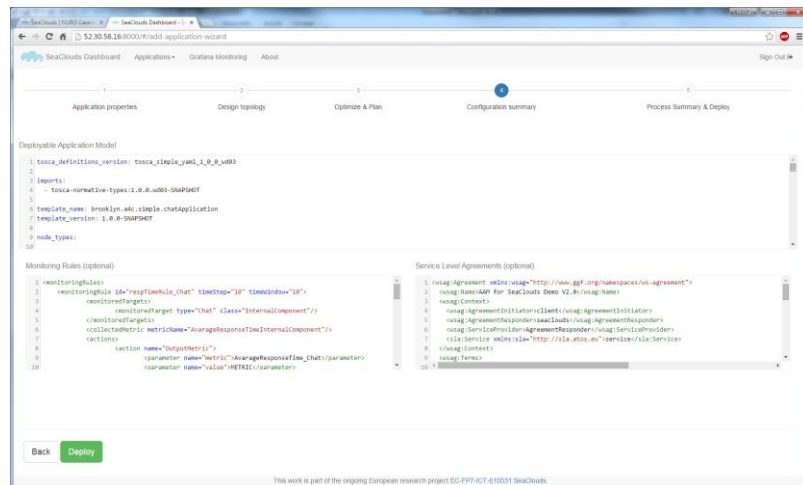


Figure 4 SeaClouds pre deployment summary

## SeaClouds Monitor

Configuration of the monitoring was supported by POLIMI

- accessing NURO sensor: succeeded
- accessing NURO effector: succeeded
- trigger violations: succeeded

NURO simulator and SeaClouds monitoring call the same effector to log events.



Figure 5 Extract from NURO's analytics: Documented simulation with violation

This figure is an extract of NURO's runtime analytics. It represents the metrics of a time group. In this case the analytics of a minute interval. The messages were sent to the effector by the NURO simulator and the SeaClouds monitoring.

<b>Passed?</b>	Yes / Partialy <b>Bug ID</b>
<b>Problems</b>	Due to the maturity of the system, interoperation between the components was not final at the testing time. Human interaction was needed where in the final version the processing should be automated.
<b>Required Changes</b>	None, SeaClouds development team works on the integration.
<b>Cost Estimation</b>	
<b>Comments</b>	Reconfiguration and replanning was not tested with this test iteration.

Test ID	Usability Testing 2 (UT2)	Date	13/11/2015
Tester	Roi Sucas (ATOS)	Testbed name	Local Testbed and Cloud Testbed
SeaClouds Platform Version		0.8.0-SNAPSHOT	
Test Results			
Involved Components	SeaClouds <i>dashboard</i> , SeaClouds <i>deployer</i> , SeaClouds <i>monitor</i>		
Interaction Between Components	SeaClouds Dashboard		Impression
	Wizard navigation		Intuitive and easy to use and understand
	Application deployment model generation		Also intuitive and easy to use. It offers a lot of options in the definition of each application component.  We had to do the deployment model manually.
	Grafana monitoring		-not tested-
	SeaClouds Deployer		Impression
	Deployment of the application in different		This component deployed successfully all the Software components in the selected



	PaaS providers	PaaS providers: <i>Pivotal, Cloud Foundry</i> and <i>IBM Bluemix</i>	
	<b>SeaClouds Monitor</b>	<b>Impression</b>	
	Monitoring of the deployed components	<p>After the deployment of the Software components, those that were going to be monitored could connect successfully with the monitoring component.</p> <p>We could also generate some violations and check them later using different observers.</p>	
<b>Passed?</b>	Yes / Partially	<b>Bug ID</b>	
<b>Problems</b>	<p>All the SeaClouds components we used are still under development, and most of the problems we have encountered are related with this.</p> <p><b>SeaClouds Dashboard:</b> Some minor bugs (overlapping issues with some components) in the user interface of the dashboard (with Chrome):</p> <div data-bbox="456 1064 1286 1265"> <p>Available Modules</p> <div> <div>Back</div> <div>Next</div> </div> <div>Database</div> <div>Web application</div> <p>This work is part of the ongoing European research project</p> </div> <div data-bbox="456 1332 1286 1505"> <p>Back Next Skip step</p> <p>tosca_definitions_version: tosca_simple_yaml_1_0_0_wd03</p> <p>description: WebChat application</p> </div> <p>We had to generate the deployment model manually.</p> <p><b>SeaClouds Deployer:</b> We had to use this component separately in order to use the last updates / changes needed for a PaaS deployment.</p> <p><b>SeaClouds Monitor:</b> Monitoring platform was also deployed manually.</p>		
<b>Required Changes</b>	-		
<b>Cost Estimation</b>	-		
<b>Comments</b>	As the components are still under development we had to use the SeaClouds tools separately.		



## 4. Conclusions and next steps

The deliverable D6.4.2 is the second document of the D.6.4.x saga. It has highlighted the results of the first cycle of evaluation measurements and analysis of the SeaClouds platform, evaluating both quantitative and qualitative requirements. The information has been separated into two main sections; the section 3 introduced the different configurations set-ups to perform a functional evaluation analysis while the section 4 has been devoted to detail some non-functional evaluation analysis. Due to the fact the SeaClouds software was not totally mature; the consortium postponed some non-functional tests to M30.

Moreover, in this document (in the Annex B) a collection of tools to be used during the testing and validation phase of the SeaClouds project has been presented.

In summary “The initial version of the SeaClouds software solution probes a great part of the functionality described in the deliverable D2.1 [2] although it is still missing some key features. Once these main features will be added to the system, developers need to fix stability problems to achieve all non-functional requirements.

From the point of view of the non-functional requirements, the situation has to be improved in the next months. The current release of SeaClouds presents several stability issues. These issues have not allowed performing Performance/Scalability tests as well as Long-Term tests.

**At the same time, this release is nothing more than a demo version, it proves that a set of specific functionality can be done, but it is far from a product that can be used effectively and in an user-friendly and productive environment.**

---

### Non-Functional Requirements tests at M30

Performance/Scalability Testing

Stress Testing

Documentation Testing

Local and in Cloud Installation Testing

Regression Testing

Long Term Testing

Final Interoperability testing

Early Usability Testing

---

## 5. References

- [1]. SeaClouds D6.4.1 - SeaClouds periodic evaluation reports [http://www.seaclouds-project.eu/deliverables/SEACLOUDS-D6.4.1\\_SeaClouds\\_periodic\\_evaluation\\_reports.pdf](http://www.seaclouds-project.eu/deliverables/SEACLOUDS-D6.4.1_SeaClouds_periodic_evaluation_reports.pdf)
- [2]. SeaClouds D2.1. Requirements for the SeaClouds Platform: [http://www.seaclouds-project.eu/deliverables/SeaClouds-D2.1-Requirements\\_for\\_the\\_SeaClouds\\_Platform.pdf](http://www.seaclouds-project.eu/deliverables/SeaClouds-D2.1-Requirements_for_the_SeaClouds_Platform.pdf)
- [3]. SeaClouds D6.1. Case study extended description [http://www.seaclouds-project.eu/deliverables/SeaClouds-D6.1-Case\\_study\\_extended\\_description.pdf](http://www.seaclouds-project.eu/deliverables/SeaClouds-D6.1-Case_study_extended_description.pdf)
- [4]. ISO 9241-11:1998, Ergonomic requirements for office work with visual display terminals (VDTs) – Part 11: Guidance on usability, Retrieved from [http://www.iso.org/iso/home/store/catalogue\\_tc/catalogue\\_detail.htm?csnumber=16883](http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=16883).
- [5]. ISO/IEC 9126:1991. Information Technology - Software Product Evaluation - Quality Characteristics and Guidelines for the User.
- [6]. [https://en.wikipedia.org/wiki/Usability\\_testing](https://en.wikipedia.org/wiki/Usability_testing)
- [7]. Initial architecture and design of the SeaClouds Platform [http://www.seaclouds-project.eu/deliverables/SeaClouds-D2\\_2-Initial\\_architecture\\_and\\_design\\_of\\_the\\_SeaClouds\\_platform.pdf](http://www.seaclouds-project.eu/deliverables/SeaClouds-D2_2-Initial_architecture_and_design_of_the_SeaClouds_platform.pdf)

## Annex A. Applications descriptions

To test the effectiveness and functionality of the SeaClouds platform, the tests are going to be performed by real applications in typical usage scenarios. These applications are going to be provided by the two Case Study partners of the project: ATOS and NURO Game.

### A1. ATOS Case Study

The ATOS case study is about an e-health and social networking application system composed by several applications and modules that aim to ease the lives of elderly people, and also the work of the social workers and doctors that take care of them. The applications that compose this solution are the following:

- **Desktop application:**

This *.NET* desktop application will be used by each one of the elderly users. It is ready to be deployed in PCs or small devices, and it is responsible for collecting the medical of these elderly. This application is also responsible for offering them all the multimedia and social content of the solution.

- **Web services application:**

This java Web application is responsible of the main logic of the application components. It is also responsible for the connections with the main database.

- SoftCare Web GUI applications:

- **Users application:**

This web application will offer most of the services offered by the desktop application, like the medical data collection.

- **Administration application:**

This java Web application will be used by social workers and doctors in order to do the follow-up of the elderly people, and also to assign them social and multimedia content.

- **SoftCare Database:**

This database stores the data of all users, including the medical data of the elderly. This implies that the database has to be stored in a private environment that ensures a correct management of the privacy and confidentiality of the stored data.

- **Forum Web application & database:**

This java Web application is responsible for maintaining a forum service for elderly people, their families and friends.

- **Multimedia repository application:**

Finally, this application is responsible for the management of the multimedia content that is offered to the elderly people.

The architecture of this solution is depicted in the next image:

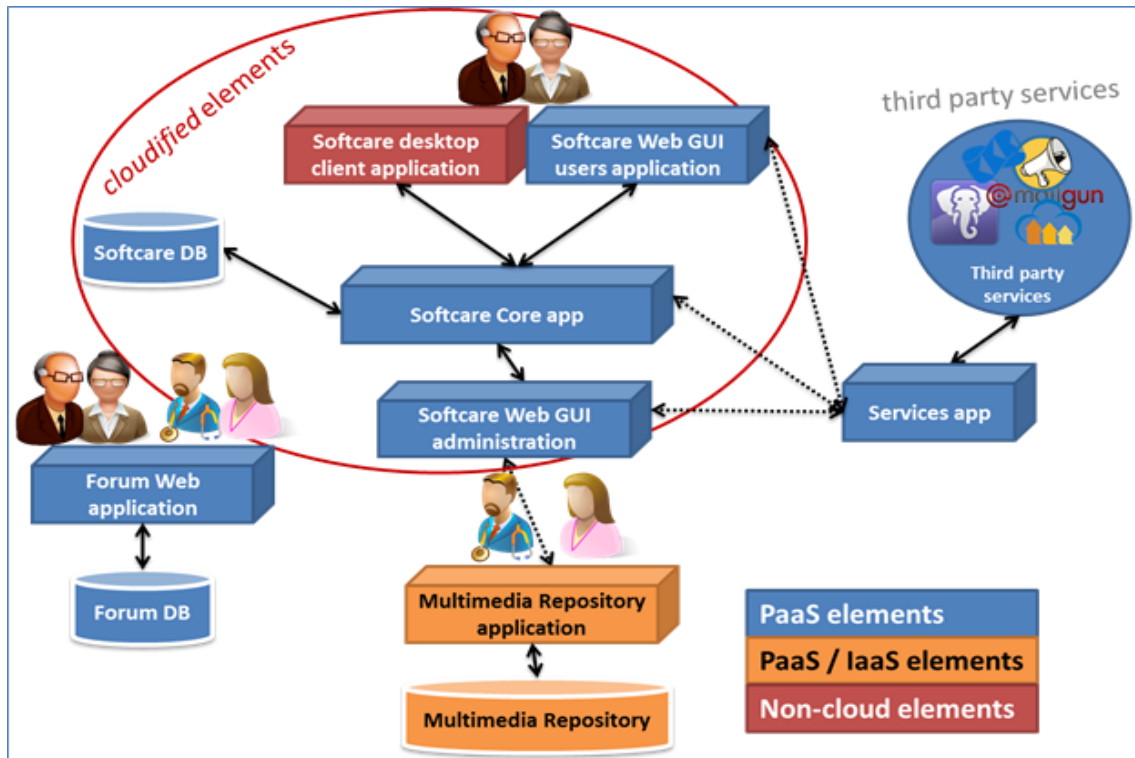


Figure 6: ATOS Case Study architecture – SoftCare solution

The SeaClouds platform will be used to design, deploy and manage all the previous described Softcare applications / components, except the desktop application for elderly people, which is out of the SeaClouds scope.

## A2. NURO Case Study

Nurogames GmbH (NURO) is a software development company focused on high quality games, gamification solutions and research. Both, customers' products and their own productions are on the market and in deployment state.

The NURO case study is based on their game servers engine, a typical so called LAMP solution (Linux Apache MySQL Php) a popular open source based technique for webserver based applications.

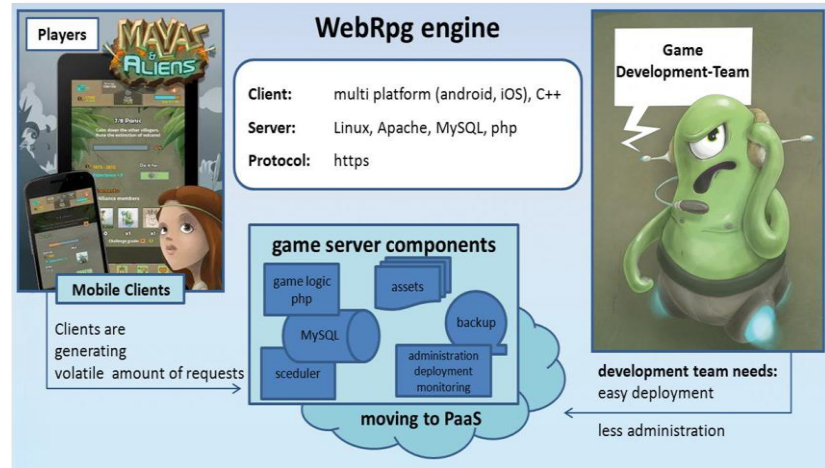


Figure 7: NURO case study - techniques

Game clients interact via HTTP(S) with the server.

The server application processes the client requests and stores the persistent data.

Cost efficiency and performance are the decisive factors for the choice of deployment setup.

Games have a very volatile usage with regional, cultural, daytime and event based influences.

The NURO cases study is focused to find by the SeaClouds System a simple to use tool to find the best deployment solution for the game and adjust it to the games' needs. A mix of private and multi cloud resources should be possible.

To evaluate this NURO implemented a simplified server based on their engines.

In the simple Setup it consists of a "Database" and a "PHP" module.

For this project NURO developed some components to simulate load scenarios and to provide an API to interact with the seaclouds system see D6.3.2.

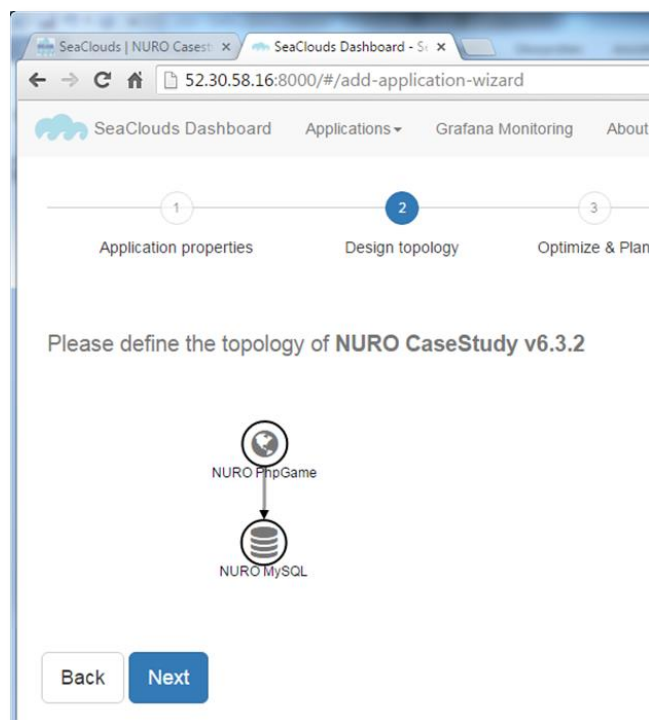


Figure 8: NURO case study - modules

Based on a flexible implementation all components can be also accessed by any web browser. The response is HTML or JSON, we recommend the JSONview plugin to display JSON responses in a human friendly way.



Figure 9: NURO case study - components

These D6.3.2 Components are:

- benchmark.php - Frontend to Apache benchmarking tool
- simulator.php - NURO Scenario Simulator (Under development)
- sensor.php - NURO Sensor, returns server metrics
- effector.php - NURO Effector, accepts event requests
- analytics.php - NURO Analytics, returns runtime analytics

Also a quiz game server and client have been developed, included this components and others of NUROs engines. The quiz game has not been tested with this evaluation.

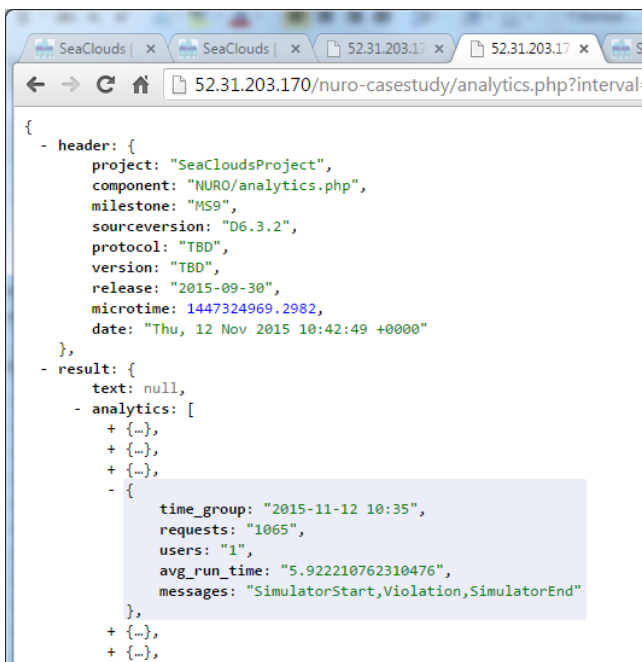


Figure 10: NURO cases study - analytics.php response

Figure 9 is an analytics result after a simulation with a SLA violation.

Both the “NURO simulator” and the “SeaClouds Monitoring” use the same effector.php API to report events.

- SimulatorStart
- Violation
- SimulatorEnd

These events are reported by the analytics.php at node: result.analytics[3].messages



## Annex B. Testing Tools/Software

In this deliverable and in the deliverable D6.1 several tests are presented that need to be performed in different scenarios and SeaClouds installations. The objective is to try to automate those tests as much as possible. The idea is to create different scripts to make the tests automatic, to write those scripts, open source or free software tools will be used.

In the following sections possible options to perform different tasks are presented. It is the tasks of the each person assigned to perform a test (see deliverable D6.1) to select the best one to write the testing scripts.

### B1. HTTP link checkers

There are two possible options:

- W3C Link Checker (<http://validator.w3.org/checklink>), only valid for public online webpages.
- Xenu (<http://home.snaful.de/tilman/xenulink.html>), a Microsoft Windows application that reports broken links for online webpages and local webpages.

### B2. Specific testing tools

The following tools can help the different testers to create the necessary scripts to validate the functionality of the SeaClouds platform.

The responsible to write a specific script should look and see what is the best option for her/him (this option includes to use no tool at all or, just a typical scripting language such as bash, perl, python, etc.).

The different options

- JSystem (<http://www.jssystemtest.org>) - It is an open source framework made in Java to create and run different testing projects. It is a modular project that covers all the possibility of testing, from unit tests to acceptance tests. In the specific case of the SeaClouds project, there are modules that may be used to run tests scripts using a CLI interfaces, to monitor computers or to test web-applications (it uses Selenium - <http://seleniumhq.org/>).
- QMTest (<http://www.codesourcery.com/qmtest>) - Another testing management tool. In this case it is made in python. It can test any kind of application based in its input and output values.
- Texttest (<http://texttest.carmen.se>) - It is a more simple tool than the two previous ones. It compares the log output of an application with a previous log output of what was expected as right behaviour of the application.
- Staff (<http://staf.sourceforge.net/>) - It is a framework to develop testing suites. It could be useful for the project, although it looks like a complex solution in comparison with the previous ones.

### B3. Web-services testing tools

As it was commented at the beginning of this document, for several of the test maybe it is necessary to write some web-services tests to verify the functionality of those tasks that can not be performed using the CLI interface.

The different tools are

- SoapUI (<http://www.eviware.com>) - It is a open source java desktop application that, among other features, it can perform functional, load and, compliance web-services tests. It provides plugins for the most common Java IDEs (Eclipse, Netbeans and, Idea). There is a commercial version with extended features, but the open source one is more than enough for our testing objectives.
- PushToTest TestMaker (<http://www.pushtotest.com>) - Open source tool that allows the creation of functional tests, load tests and monitoring. It also allows the integration of unit tests inside the framework, but it fall outside of the scope of the WP6.
- WebInject (<http://www.webinject.org/>) - Open source tool written in perl that can perform functional and regression test over web-services and web applications. The test are written in XML and can be only performed over applications that use http or https protocols.

### B5. Tools for Performance/Scalability testing

Useful tool that can be used during the performance/scalability testing and stress testing are:

- Apache JMeter (<http://jakarta.apache.org/jmeter>) - JMeter is a java application designed to test client/server software, including web applications. JMeter can be used to simulate heavy load in a server and to see how the system changes its behaviour under different load conditions.
- VisualVM (<https://visualvm.dev.java.net>) - VisualVM is a tool to monitor and troubleshoot Java applications. It runs on Sun JDK 6, but is able to monitor applications running on JDK 1.4 and higher. It utilizes various available technologies like jvmstat, JMX, the Serviceability Agent (SA), and the Attach API to get the data and automatically uses the fastest and most lightweight technology to impose minimal overhead on monitored applications.

## Annex C. Test results report format

This section introduces the template that the tester has to fill for each one of the tests mentioned in the deliverable D6.1 and in this deliverable.

The table 7 includes all this categories

- **Scenario ID/Quality test ID** – Provides the unique identifier that refers to the different quality tests and scenarios defined in this deliverable.
- **Date** – Date in which the test was completed.
- **Pass/Fail** – Indicate if the tests was successful passed by SeaClouds or it failed.
- **Tester Name** – Name of the tester that performed the different tests that are included in the corresponding table report.
- **Testbed/Machine used** – Name of the testbed or machine where some requirement of SeaClouds was tested.
- **Comments about the Testbed/Machine** – Any possible comment about changes or clarification to the information about the testbed or the machine commented in this deliverable or in the deliverable D6.1 (e.g. a new Java Virtual Machine was intalled, new version of the operating system, etc.).
- **SeaClouds version** – Version of SeaClouds tested.
- **Third party software used** – Additional software used in the tests (e.g ATOS Use Case, Nuro Use Case, the dummy application, etc.). It should be specified the exact version of those applications.
- **Third party testing software used** – In the case the tester uses any of the tools stated in the 0, it should be mentioned here.
- **Involved Components** – A list of all SeaClouds architecture components involved in order to carry out the related test or scenario.
- **Description of interactions among components** – It provides a brief description about how the different components interact to achieve the scenario/test.
- **Possible problems and necessary changes** – During the tests and possible changes needed to make to the system to pass the tests in new versions of SeaClouds.
- **Comments** – Any helpful commentary that the tester considers necessary.

Table 7 Template to fill the results of the tests.

Scenario ID/Quality test ID	
Date	
Pass/Fail	
Tester Name	
Testbed/Machine used	

<b>Comments about the Testbed/Machine</b>	
<b>SeaClouds version</b>	
<b>Third party software used</b>	
<b>Third party testing software used</b>	
<b>Involved Components</b>	
<b>Description of interactions among components</b>	
<b>Possible problems and necessary changes</b>	
<b>Comments</b>	

To fill all the results of the different tests, a web application is going to be created. The tester will fill some forms and each test result is going to be automatically stored into a database.