SeaClouds Project

D5.1.1 - Definition of the software developing environment

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	1st October 2013
Ending Date	31st March 2016

Work Package	WP5 Integration, infrastructure delivery and GUI
Deliverable code	D5.1.1
Deliverable Title	Definition of the software developing environment
Nature	Report
Dissemination Level	Public
Due Date:	M8
Submission Date:	15th June 2014
Version:	1.0
Status	Final
Author(s):	Javier Cubo (UMA), Francesco D'Andria (Atos), Elisabetta Di Nitto (Polimi), Michela Fazzolari (UPI), Iván Febles (Atos), Raffaela Mirandola (Polimi), Diego Perez-Palacin (Polimi), Andrea Turli (Cloudsoft), PengWei Wang (UPI)
Reviewer(s)	Roman Sosa (ATOS) Javier Cubo (UMA)

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

List of Figures

1. Executive Summary

This deliverable presents the first version of the SeaClouds integration plan. It describes the integration approach that will be followed during the project as well as the development repository and the supporting tools for the development and the integration. The description of the main components of the SeaClouds architecture together with the preliminary calendar for their integration are also described in the document.

2. Introduction

2.1 Scope and outcome of the Deliverable

The SeaClouds platform is composed of various parts delivered by the workpackages WP3, WP4 and WP5 under the responsibility of various different partners of the SeaClouds consortium. The purpose of this deliverable is to ensure that the development of such platform occurs in a coordinated way. The development approach will be based on two main principles:

- Coordinate development to ensure gradual, seamless and continuous integration. The objective is to avoid the typical 'big bang' integration problem and to ensure that mismatches are discovered and fixed as early as possible. The tools used to ensure such coordination are the plan defined in this document and the adoption of a continuous integration platform.
- Develop the system as an open source project since the very beginning. The purpose is to ensure that the project gradually acquire visibility in the community. The tool to do so is the adoption of github as repository, wiki, and bug tracking system.

Consistently with these principles, this deliverable defines the toolset adopted for the development of the SeaClouds platform and the integration plan together with a preliminary timeline.

2.2 Structure of deliverable

The document is structured as follows:

- Section 3 describes the approach followed in the development of the SeaClouds software platform.
- Section 4 summarises the main components of the SeaClouds architecture described in Deliverable D2.2.
- Section 5 presents the preliminary integration plan and the calendar for the integration of the different components.
- Section 6 reports about the structure of the development repository.
- Section 7 presents an overview of existing tools that can be used as a support in the development and integration process.
- Section 8 concludes the deliverable.

3. Development and integration approach

The software development process follows the Continuous Integration paradigm [1,2]. This paradigm primarily focuses on solving the –usually unpredictably long– task of system integration. To achieve its objective, Continuous Integration proposes to build the system after each small extension or modification of its source code. In such way, many integrations are performed during the project development, while each of these integrations can be easily executed. The rationale for achieving easy integrations is that each of them concerns only a small portion of new code; hence it cannot raise many conflicts or failures.

To be confident about the successfulness of the software development process, the SeaClouds consortium complies with the best practices proposed for the Continuos Integration. Next paragraphs describe them.

- There will only exist one and only one source code repository that keeps and protects the integrity of every source file necessary for the project. This repository will include: source code of the system to build, scripts to build the system from the source code, scripts to test the running system, etc. This source repository will be hosted at https://github.com/SeaCloudsEU and maintained by Atos (the project coordinator).
- The system building from source files will be fully automated. It means that only a simple operation will be needed to create a running system from the source code. All the work that is required to build the system will be automated and launched by such simple operation.
- Tests will be included in the source repository in order to check the goodness of each system build. These tests will be launched just after the building and before updating the mainline of the source code repository. This practice avoids stepping back during the development.
- Every developer who is modifying the code will do so on a separate branch and will commit the code as soon as it will be buildable (one commit per day could be a good frequency). Clearly, before committing the developer will update his/her branch and solve any conflict as needed. Failures or conflicts can still happen during the system building or testing due to integration of commits. However, since each commit represents only a slight variation of the system, it is expected that these problems can be solved quickly (during the same day). Every commit comes with comments containing a detailed description of the performed changes and updates.
- In the beginning, a functional automatic building and testing of the system will be created in order to enable the development of the project code. While the project is developed and more functionalities are included, these test and building processes – even if they are fully automated- are expected to become slow activities that force developers to waste their time. In that moment, the performance of building and test processes will be studied. Regarding the performance of system building, there will be

implemented approaches to speed up the process. Regarding the performance of system testing, the most time-consuming will not be executed before the commit of each code modification but only periodically (e.g., nightly testing).

- The concrete programing environment and its setup will be finalized in the future, yet the descriptions of the tool for the automatic building of the system and the service used for the continuous integration are given in Sections 5 and 6 respectively.

4. Main SeaClouds components

In this section, we present the main components of the SeaClouds platform.

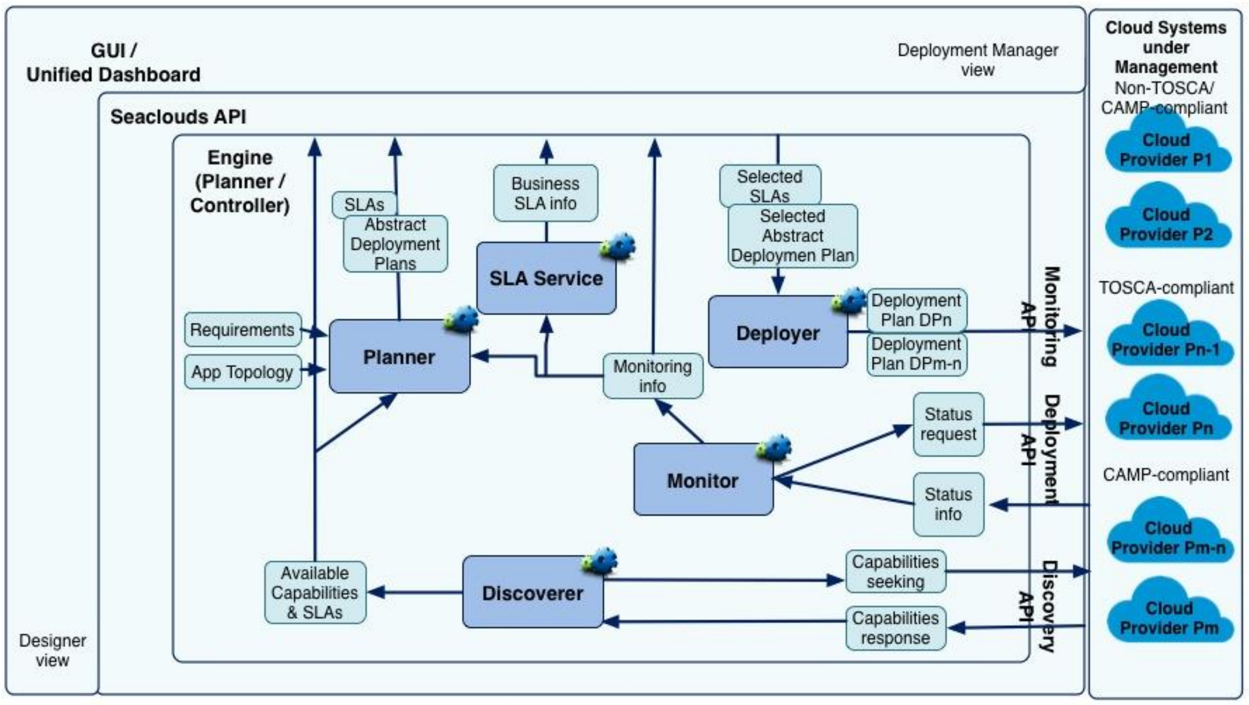

Figure 1. SeaClouds Platform architecture

The SeaClouds architecture is being defined in WP2, D2.2 [3], and ist current state is shown in Figure 1 (it could suffer minimum changes for the firnal version of the deliverable D2.2). The platform is composed of the following elements:

- **Planner.** This component works both at design time and at runtime and aims at generating a plan for the deployment of a SeaClouds application.
- **Discoverer.** This component is in charge of identifying those services that can be used to derive a plan for a SeaClouds application.
- **Deployer.** This component is in charge of deploying and managing the execution of a SeaClouds application.
- **Monitor.** This component is in charge of monitoring the execution of a SeaClouds application on the target clouds that have been selected for its deployment.
- **SLA Service.** This component is in charge of mapping the low level information gathered from the Monitor into business level information about the fulfillment of the SLA defined for a SeaClouds application.

- **GUI.** The SeaClouds platform features the graphical user interface for two user roles of Designers and Deployment Managers.
- **SeaClouds API.** Suitable application programming interfaces are defined to allow the communication among the different SeaClouds components.

All above components are part of the SeaClouds Engine. They interact with the SeaClouds GUI and with external systems through the SeaClouds APIs. SeaClouds is able to deploy, manage and monitor applications on clouds that are compatible with TOSCA and CAMP. Moreover, it will be compatible with a selected number of clouds (which will be studied in another deliverable).

In the following of this section, we describe each component of the SeaClouds platform and provide an indication on how it will be developed in terms of:

- programming language

- use of preexisting libraries/components

- dependencies with other components

- software interfaces technology

- license associated to the component

4.1 Discoverer

Short description	The Discoverer component is in charge of identifying the available capabilities offered by cloud providers that will be used by the Planner component to perform the distribution process.
Programming language	Possible languages: Java, PHP, Python, MySQL, PostgreSQL.
Use of preexisting libraries/components	Parts from the matchmaking module Cloud Pier's Lighthouse (Cloud4SOA), Paasify repository.
Dependencies with other components	This component does not depend directly on the interaction with other components of the SeaClouds platform, but it rather depends on the possibilities offered by cloud providers to easily access to their capabilities.
Software interfaces technology	Java, REST accessible API
License associated to the component	Apache License 2.0

4.2 Planner

Short description	The Planner is in charge of determining a distribution of application modules onto multiple available clouds so that the QoS properties and other technology requirements needed for individual application modules are not violated.
Programming language	Possible languages: Java, PHP, Python, MySQL, PostgreSQL.
Use of preexisting libraries/components	-
Dependencies with other components	It will partly depend on the outputs of Discoverer component, i.e., the organization form of available cloud capabilities and SLAs.
Software interfaces technology	Java, REST accessible API
License associated to the component	Apache License 2.0

4.3 Deployer

Short description	The Deployer component is in charge of generating a concrete deployment plan for each target cloud platform. Concrete deployment plans include all the needed steps to be performed to actually deploy a (set of) application module(s) on a specific cloud platform.
Programming language	Possible languages: Java, PHP, Python, MySQL, PostgreSQL.
Use of preexisting libraries/components	Brooklyn (with libraries Apache Whirr, a set of libraries for running cloud services, such as Hadoop, and jClouds , an open source Java library that supports several IaaS providers (no PaaS). At the level of PaaS, Brooklyn could be integrated by reusing the PaaS Unified Library of Cloud4SOA.
Dependencies with other components	Since the plan describes the needed steps to deploy or reconfigure the application, this component will be connected with the Planner component. Then, this plan has to be read and approved by the deployment manager. Also, this component interacts with the monitoring, by executing and initializing the monitoring service and

	the SLA service (that generates the SLA agreements)
Software interfaces technology	Java, REST accessible API
License associated to the component	Apache License 2.0

4.4 Monitor

Short description	The Monitor component is in charge of collecting monitoring information from the targeted cloud platforms, of analyzing such information, and of presenting the results of such analysis (through the SeaClouds GUI dashboard) to the Deployment Manager. The Monitor is also in charge of generating replanning triggers that are passed (possibly filtered by Deployment Manager, depending on the platform configuration) to the Planner in order to start a reconfiguration process.
Programming language	Possible languages: Java, PHP, Python, MySQL, PostgreSQL.
Use of preexisting libraries/components	MODAClouds, Brooklyn
Dependencies with other components	This component interacts with the Deployer by generating the QoS violations, which could require a repair (no changes in the plan, only in the live model topology), or a replanning (changes in the plan and the topology). Therefore, this component also interacts with the Planner component.
Software interfaces technology	Java, REST accessible API
License associated to the component	Apache License 2.0

4.5 SLA Service

Short description	The SLA Service is responsible for establishing, reviewing and cancellation of complex end-to-end- Service Level Agreements (SLAs) between Application Providers and Cloud Suppliers. It provides an operational management with SLA composition and decomposition across functional and organizational Cloud domains. It covers the complete SLA and service lifecycle with consistent interlinking of planning and runtime management aspects by implementing procedures and methods to evaluate and report Business Level Objectives.
Programming language	Java, MySQL
Use of preexisting libraries/components	-
Dependencies with other components	While the Planner component will provide the inputs to create the SLA Agreements, the Deployer component will configure and set up the SLA Service at runtime. Finally, the Monitoring component will generate Business Metrics to evaluate agreements.
Software interfaces technology	Java, REST accessible API
License associated to the component	Apache License 2.0

4.6 GUI

Short description	The SeaClouds platform features the graphical user interface (SeaClouds GUI) for two user roles (Designers and Deployment Managers). Application Designers exploit the GUI to provide a description of the topology of the application to be deployed, together with a set of requirements. These requirements can include QoS properties and technology requirements for the application modules. Deployment Managers instead, exploit the GUI through a unified dashboard that allows them to supervise the deployment and the monitoring of the application.
Programming language	Javascript

Use of preexisting libraries/components	Cisco Curvature, D3, Graphite
Dependencies with other components	The GUI has dependencies with the main components of the architecture (Planner, Deployer, Monitor, Discoverer and SLA Service), as they use inputs from the GUI, or generate outputs to be shown in the GUI.
Software interfaces technology	-
License associated to the component	Apache License 2.0

5. Preliminary integration plan

As discussed in Section 2, the SeaClouds platform will be developed following a continuous integration approach. Nevertheless, according to the Description of Work (DoW), the integration plan aims at setting up specific milestones at which a version of the integrated platform, containing the tools that have been developed till that point, will be released.

Figure 2 overviews the SeaClouds platform integration plan and software components development roadmap, respectively. The figure identifies the main artifacts that will be produced during the lifecycle of the project which are:

- *Tools*, that is, isolated components of the SeaClouds architecture.
- *Integrated platform*, that is, the solution that integrates various tools and is made available on source forge together with proper installation and usage manuals.
- *S/w platform*, that is, an instance of the integrated platform installed and deployed on the Cloud and used as a testbed. This platform is accompanied by videos and other dissemination and training material aiming at showing how SeaClouds has been used by the case studies.
- *Case study prototypes*, that is, the prototypical systems that are developed to validate the SeaClouds results.

The first project integration activity is at M12 and will focus on obtaining the first and simplified version of some of the key components of the SeaClouds architecture. At this point in the project the first s/w platform, that is, the first version of the testbed available for use by the case studies will be made available.

The second phase relevant for integration will be at M19,when we will make available the first integrated platform.

The third phase at M24 will include all the prototypes delivered at M22 as well as the new improvements and developments delivered in the meantime. At M24 both the new integrated platform as well as the s/w platform will be made available and will be exploited by the case studies for their development.

Finally, at M29 the consortium will deliver the last and final integrated Seaclouds platform and s/w platform properly packaged for external use.

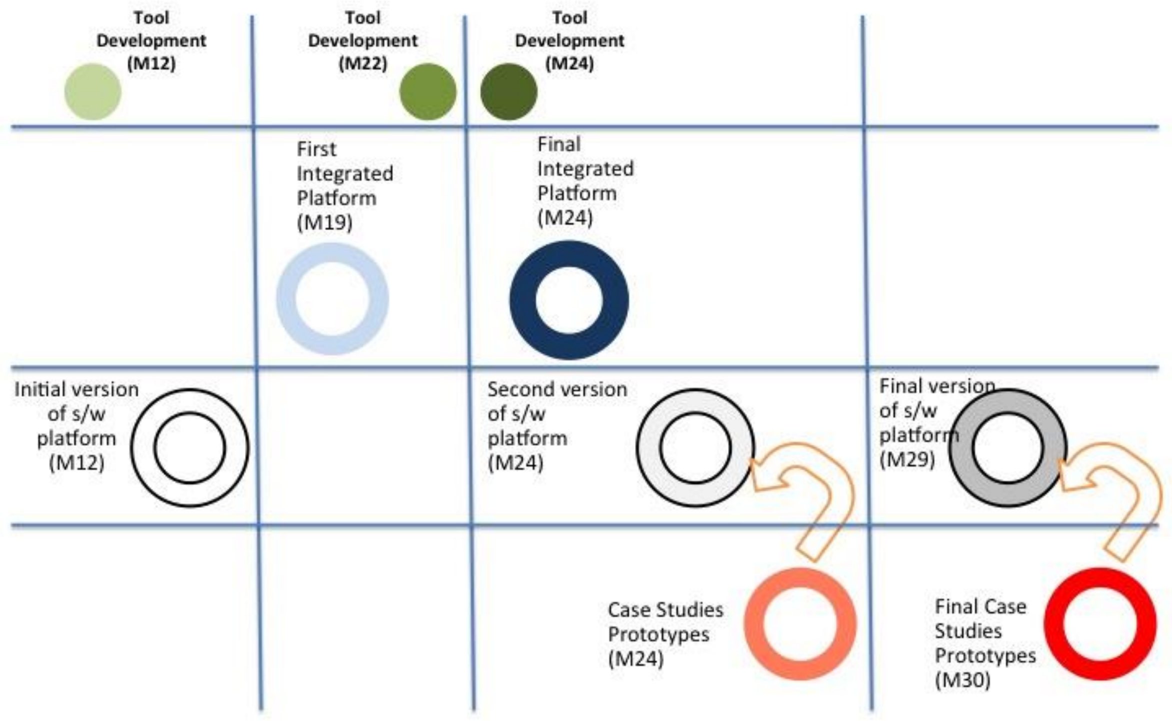
Tool Development (M12)
Tool Development (M22)
Tool Development (M24)
First Integrated Platform (M19)
Final Integrated Platform (M24)
Initial version of s/w platform (M12)
Second version of s/w platform (M24)
Final version of s/w platform (M29)
Case Studies Prototypes (M24)
Final Case Studies Prototypes (M30)

Figure 2. SeaClouds Platform integration plan

5.1 Dependency graph

Figure 3 describes the main components of the SeaClouds architecture together with their dependencies and dates of delivery of the prototypes implementation. Starting from the bottom, the figure shows that the Monitor, Discoverer and Deployer components depend on existing cloud infrastructures. The Planner component performs the activity of generating a distribution of the different application modules onto available clouds. To this end, it depends on the results obtained by the Discoverer -on available cloud capabilities and their SLAs- and on triggers received by the Monitor component in case of an event that requires replanning arises. The SLA service establishes, reviews and eventually cancels complex end-to-end SLAs between application providers and cloud infrastructure suppliers. SLA service requires the Planner since it provides the required inputs to establish the SLAs; the Deployer since it passes information to configure and set up the SLA service behavior at runtime; and the Monitor since it will generate the metrics of interest for the application to evaluate the satisfaction of agreements. The Deployer component is in charge of deploying the application modules as suggested by the devised plan and to this end it depends on the Planner results. The GUI allows the use of the SeaClouds platform both to application designers and deployers and therefore it depends on all the main components, Deployer, Planner, SLA service, Monitor and Discoverer.

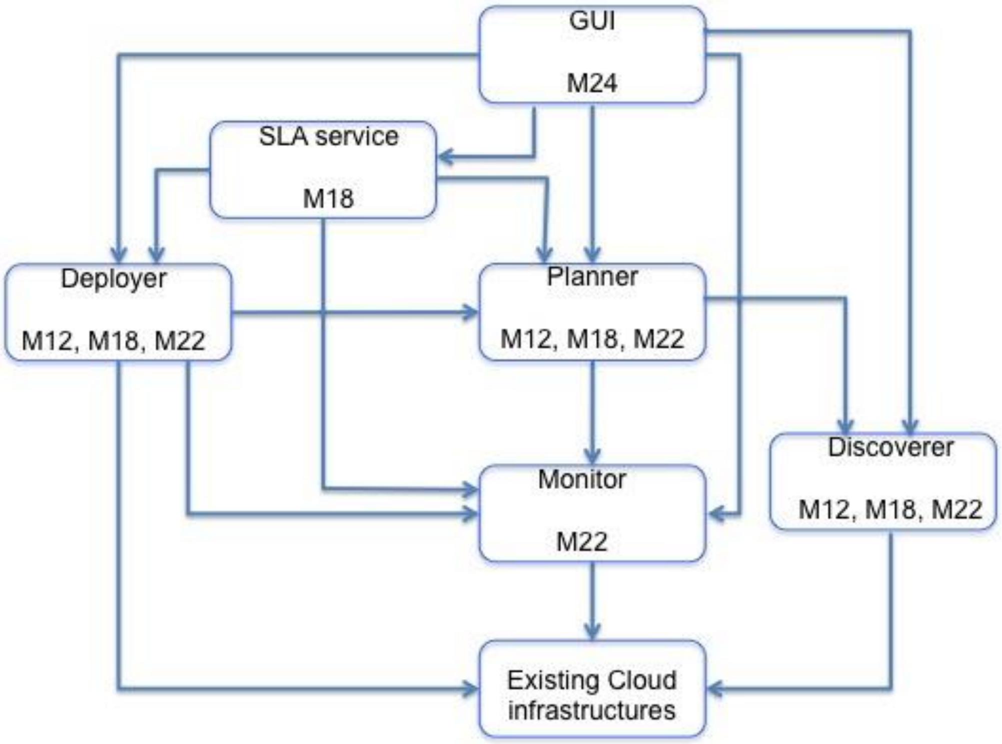

Figure 3. Software component's dependencies and development roadmap

5.2 Preliminary calendar for integration

The first release of the software platform is at **M12**. For this date, a first prototype version of Deployer, Planner and Monitor components will be released, as illustrated in Figure 4. The Planner at this step will provide a first version of the plan using a simple static matchmaking approach. The Deployer will use the data sent by the Planner with an asynchronous data passing through a file. The Monitor interacts with the Planner through a REST API in case of requirements violations. In this first release, components Deployer, Planner and Monitor will be deployed on the same Virtual Machine. A possible storyboard for this first release is described in Section 4.3.

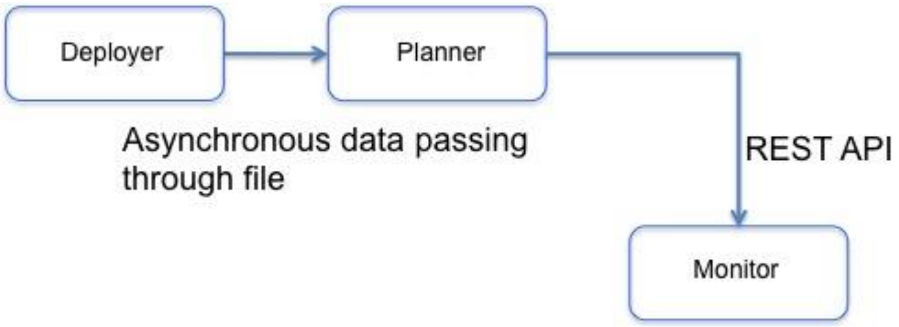

Figure 4. M12 Software components

The second release at **M19** will consist of an extended version of M12 software platform, including the prototypes released at M18. In particular, the new prototypes will be well tested components properly packaged for external use. This release will deploy software components on two different Virtual Machines: the first one will execute GUI, Planner, Deployer, Monitor and SLA services; while the second one will execute the Discoverer component together with the persistency task for models and artifacts.

At **M24** the software platform will integrate the M22 prototypes and will include all the components of the SeaClouds architecture. Figure 5 illustrates the software components with their dependencies and the specification of the API.

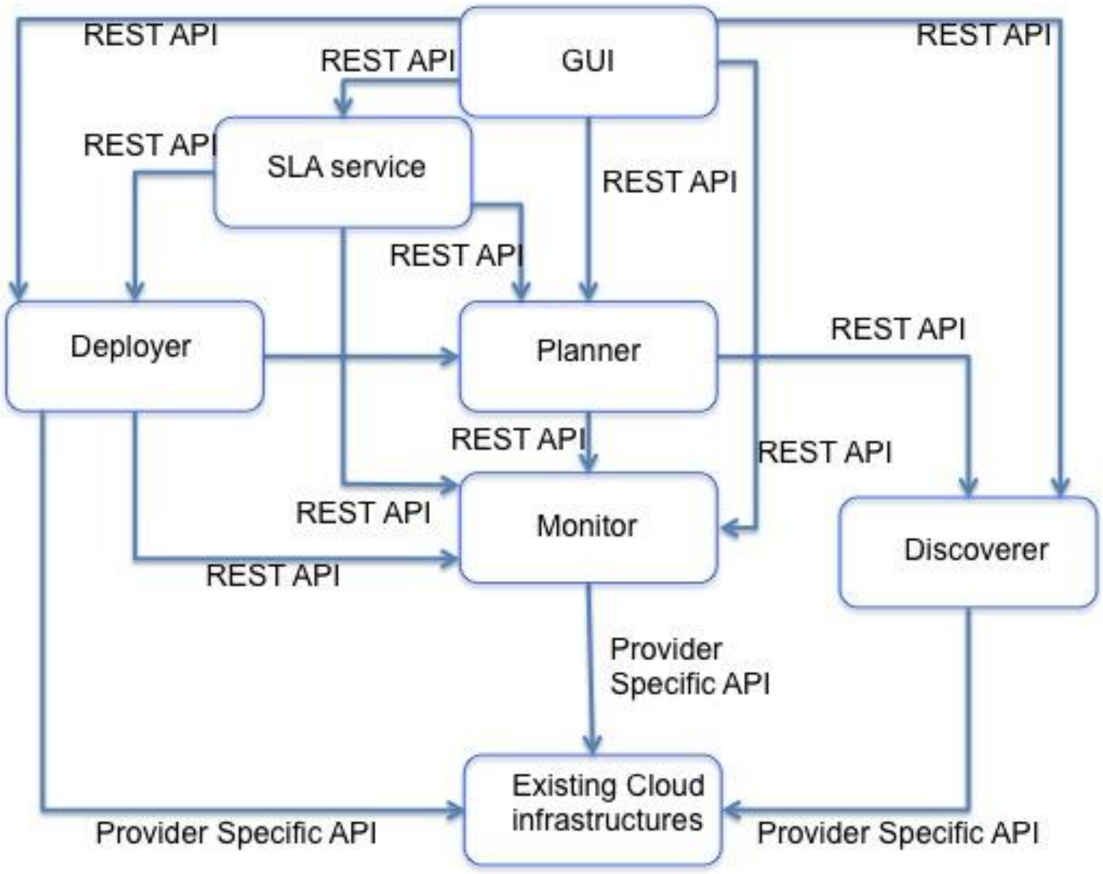

Figure 5. M24 Software components

The final integrated platform will be released at **M29** together with the GUI allowing its exploitation to both SeaClouds and external users.

5.3 Demo Story Board (M12)

For M12, a first version of the NURO application will be demonstrated using the M12 SeaClouds platform. This application is described in Deliverable D6.1 [4]. In particular, only two modules of the application, the PHP-Worker and the database, will be taken into account and a multi-cloud solution will be provided. There will be four allowed connections, inbound ping, inbound HTTP(S) for Client

Requests to the PHP-Worker, one connection from PHP-Worker to the database and, if needed, an inbound ssh connection to PHP-Worker and database. At M12 we will focus on IaaS aspects while we will tackle PaaS at M24. The realization of the NURO application will follow the development, deployment and execution approach defined in SeaClouds. More specifically, the team involved in the development and operation of the application will go through the following the steps:

Step 1 Definition of the Module Profile

Step 1.a. Provide the functional requirements (hard requirements) of the two modules belonging to the app

- Example: Technology (Java 1.6 or PHP >=5.2 or SQL BD); resources (DB=1G space, or RAM memory; Type of service; Location

Step1.b. Provide the non-functional requirements (hard and soft requirements) of the two modules.

- Example: Scalability: growing; Fixed, bronze; QoS: RT; Availability;

Step 2: Definition of the Abstract Plan Model

This step will occur outside the SeaClouds platform and will be based on some pre-existing approach that will be selected in the next months. The abstract plan will be the input to the planner service.

Step 3: Execution of the Planner Service:

At M12 services will be statically matched. The outcome of this phase will be the generation of a Concrete Plan.

Step 4: Execution of the Deployer Service:

Step 4.a. Read the Concrete Plan

Step 4.b. Retrieve the artifacts

Step 4.c. Prepare the cloud environments

Step 4.d. Deploy the artefact on top of them

Step 4.e. Configure the application

Step 4.f. Generate and Store the Topology (Live Model)

Step 4.g. Initialize the Monitoring Service (Monitoring Rules)

Step 4.h. Initialize the SLA Service (generate SLA Agreements)

Step 5: Execution of the Monitoring Service

Step 5.a. Monitor Application

Step 5.b. Enforce the Monitoring Rules

Step 5.c. Generate a soft QoS Violation , that is, a violation that change the Topology only, not the plan.

Step 6: Execution of the SLA Service

Step 6.a. Ensure fulfillment of SLA Agreements

Step 6.c. Generate a QoB (Quality of Business) Violation

Step 7: Execution of the Deployer Service:

Step 7.a. Change the Topology

Step 7.b. Enforce the changes

Step 7.c. Store the new topology

6. Development repository

In this section, we will overview the structure of the development repository that will follow the structure of the architecture.

By “structure of the architecture”, the SeaClouds consortium means the decisions the consortium makes concerning how the project best meets its objective. We need to consider how to best leverage the language programming features to create a clean and effective software. In practical terms, “structure” means making clean code whose logic and dependencies are clear as well as how the files and folders are organized in the folders.

Here the consortium will first overview the approach followed to automatically package and integrate the SeaClouds software and second how such software can be organized on a common repository.

Automatic Software Packaging and Integration

Software packaging is a process that automatically integrates a diverse collection of computer programs based on the types of components involved and the capabilities of available translators and adapters in an environment.

SeaClouds consortium, leveraging an SOA approach, aims at releasing its code in separate (but interdependent) modules that belog a multi-module project. In this respect, branches are important to be created, managed and merged. This may happen whenever a new feature is going to be introduced, a bug is fixed or an enhancement is applied.

Definitely, for a project of this scale and modularity, it is also required to use build automation techniques.

Maven [Maven http://maven.apache.org/] is a build automation tool typically used for Java projects. Maven serves a similar purpose to the Apache Ant tool, but it is based on different concepts and works in a profoundly different manner. It can also be used to build and manage projects written in C#, Ruby, Scala, and other languages. Maven is hosted by the Apache Software Foundation, where it was formerly part of the Jakarta Project. Although it is simple to use the command line to pass a single source module to a compiler and then to a linker to create the final deployable object, when attempting to compile and link many source code modules, in a particular order, using the command line process is not a reasonable solution, and Maven as a build automation tool was used.

The typical "maven" way to organize multi-module projects is to store them hierarchically, where your modules exist within your multi-module project. Your modules may even have more modules within them. However, the "Eclipse" way to organize projects is in a flat manner, where each module is a project located at the root level of your workspace.

Version Control System

The SeaClouds consortium is composed by six European partners. Their developers are not centrally placed but dispersed throughout different countries, so the code decoupling to a central repository is vital and therefore no client-server solution could be followed. The consortium adopted a “Git approach” as the primary VCS system and the SeaClouds code will be hosted in a public GitHub repository. https://github.com/SeaCloudsEU

Actually, GitHub is a web-based hosting service for software development projects that use the Git revision control system.

7. Supporting tools

The SeaClouds consortium agreed on having a productive toolset to facilitate the communications and the knowledge sharing.

As described on the previous section, code produced by SeaClouds project will be publicly available at https://github.com/SeaCloudsEU. Github is a powerful collaboration, code review, and code management for open source and private projects. This is greatly used in a variety of successful OpenSource projects and it is particularly appreciated also because it provides developers with a set of generic useful tools. In fact, SeaClouds consortium will also take advantage of the Github Wiki pages, already available at https://github.com/SeaCloudsEU/SeaCloudsPlatform/wiki, which is extremely developer-friendly as it follows the same github workflow process to edit documentation, and Github issues, already available at https://github.com/SeaCloudsEU/SeaCloudsPlatform/issues to keep track of the project's backlog.

To track the active tasks, SeaClouds consortium will evaluate trello.com. Trello is a collaboration tool that organizes your projects into boards. In one glance, Trello tells you what's being worked on, who's working on what, and where something is in a process.

Github integrates easily on many Continuous Integration tool. SeaClouds consortium has identified CloudBees[1] and its BuildHive cloud based continuous integration service for Github projects. It combines cloud-powered Jenkins with your GitHub development projects. It shows your build status on GitHub and easily verify the quality of incoming pull requests. BuildHive and Jenkins automatically detect your build platforms so your builds and tests run seamlessly. If Jenkins says your builds are blue, they are good-to-go.

As a collaborative editing tool, needed for example to prepare official documentation, SeaClouds consortium will use Google Drive, a free tool that keep your files backed up and easy to reach from any phone, tablet, or computer. It offers also live docs, also it has recency views and tagging support for documents.

From the Google office-replacement tools, SeaClouds consortium will use private Google calendar, where it is possible to co-ordinate meetings that can happen on Google Hangouts and that can be integrated with Doodle[2], when SeaClouds consortium need to schedule a meeting.

Seaclouds consortium is also considering to replace the current mailing list available at seaclouds@lists.atosresearch.eu with a Private Google Group as it is easier to add members and have access to the email history.

For real-time communication, SeaClouds identified gitter.im as a IRC-like chat based on github accounts, so to minimizy also the need for multiple accounts. SeaClouds already setup different chat rooms at:

[1] cloudbees.com
[2] doodle.com

- https://gitter.im/SeaCloudsEU/SeaCloudsPlatform, for public conversations
- https://gitter.im/SeaCloudsEU, for private conversations

It is also possible to create rooms per-repository (e.g. https://gitter.im/SeaCloudsEU/WP3) for sub-project specific chat.

8. Conclusions

This deliverable has provided an overview of the development approach the SeaClouds consortium plan to follow and of the plan we have for integration. The document might be updated in the future based on the findings we will identify during the development of the project.

9. References

[1] A. Fuggetta and E. Di Nitto. Software Process. In Proceedings of the on Future of Software Engineering. FOSE'14. New York, NY, USA, 2014, ACM.

[2] M. Fowler. Continuous Integration. May 2006. http://martinfowler.com/articles/continuousIntegration.html

[3] SeaClouds deliverable D2.2. Initial architecture and design of the SeaClouds platform. June 2014. *In progress*

[4] SeaClouds deliverable D6.1 . Case study extended description. December 2013. http://www.seaclouds-project.eu/deliverables/SeaClouds-D6.1-Case_study_extended_description.pdf