SeaClouds Open Reference Architecture

White Paper

October 2014

SeaClouds Consortium

www.seaclouds-project.eu

Executive summary

Cloud computing is a model for enabling convenient and on-demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with minimal management effort or service provider interaction **[1].** The cloud helps to reduce time-to-market and provides on-demand scalability at a low cost for the users. Due to its prospective benefits and potential, cloud computing is a hot research area. Many private and public clouds have emerged during the last years, offering a wide range of different services at SaaS, PaaS and IaaS levels aimed at matching different user requirements. To take full benefit of the flexibility provided by different clouds that offer different services, the modules of a complex application should be deployed on multiple clouds depending on their characteristics and strong points.

Current cloud technologies suffer from a lack of standardization, with different providers offering similar resources in a different manner **[2].** This heterogeneity refers to diversities in supported programming tools, in the various types of underlying infrastructures, and even on available capabilities. As a result, cloud developers are often locked in a specific platform environment because it is practically unfeasible for them, due to high complexity and cost, to move their applications from one platform to another **[3]**. Since migrating a single application is a cumbersome and manual process, the deployment, management and reconfiguration of complex applications over multiple clouds is even harder. To overcome the vendor lock-in problem, various standardisation efforts are currently ongoing, such as OASIS Cloud Application Management for Platforms (**CAMP**) **[4],** DMTF Cloud Infrastructure Management Interface (**CIMI**) **[5],** Virtualization Management **(VMAN) [6],** or OASIS Topology and Orchestration Specification for Cloud Applications **(TOSCA) [7],** just to mention some of them. Furthermore, different vendors (e.g., Dell[1], BMC[2], Abiquo[3]) are currently commercialising tools for the provisioning, management and automation of applications in leading public and private clouds. A promising perspective, opened by the availability of different cloud providers, is the possibility of distributing cloud applications over multiple heterogeneous clouds. Indeed, as pointed out by **[8]**, cloud adoption will be hampered if there will be no suitable way of managing data and applications across multiple clouds. In a scenario where a **complex application is distributed on different cloud service providers, a solution is needed in order to manage and orchestrate the distribution of modules in a sound and adaptive way**. Such solution should determine the best cloud provider for each particular module based on client requirements (e.g., availability, cost). Once the distribution has been decided, the solution

[1] http://www.enstratius.com/
[2] http://www.bmc.com/
[3] http://www.abiquo.com/

should support operations such as managing the relationships between the different modules, maintaining all the specified properties and requirements, and monitoring and reconfiguring the distribution in case any problem occurs during operation.

Therefore, ***how to deploy and manage, in an efficient and adaptive way, complex applications across multiple heterogeneous cloud platforms is one of the problems that have emerged with the cloud revolution.***

In this White Paper we present the **SeaClouds Open Reference Architecture**. First, in order to understand the main goal of the European Project SeaClouds, we describe the context, motivations, objectives and positioning with respect to related cloud initiatives and standards. SeaClouds aims at enabling a seamless adaptive multi-cloud management of complex applications by supporting the distribution, monitoring and migration of application modules over multiple heterogeneous cloud platforms. Then, we present the reference architecture and discuss some of its main aspects.

SeaClouds provides the foundation for allowing ***"Agility After Deployment"*** providing necessary tools and a **framework** for **Modelling**, **Planning** and **Controlling Cloud Applications**.

SeaClouds answers questions such as:

- How can a **complex cloud application** be deployed, managed and monitored over **multiple and heterogeneous infrastructures Clouds**?
- How can the underlying cloud providers be **monitored** to check for quality of service compliance?
- How can applications be **reconfigured** if any problem or deviation from normal execution patterns is detected in any component at run time?

Challenges and Positioning of SeaClouds

SeaClouds works towards giving organizations the capability of ***"Agility After Deployment"*** for cloud-based applications. The approach is based on the concept of service orchestration and designed to fulfill functional and non-functional properties over the whole application. Applications will be dynamically reconfigured by changing the orchestration of the services they use when the monitoring will detect that such properties are not respected. So, **SeaClouds' main goal is the development of a novel platform which performs a seamless adaptive multi-cloud management of service-based applications**, with four specific objectives (presented below). For each objective, the SeaClouds consortium plans to tackle a set of challenges, which are described in depth in Deliverable D2.2, related to Initial architecture and design of the SeaClouds platform **[9]**.

Objectives of SeaClouds

Orchestration and adaptation of services distributed over different cloud providers. SeaClouds aims at providing the assisted design, synthesis, and simulation of service orchestrations on cloud providers, distributing modules from a cloud-based application over multiple and heterogeneous clouds.	***Monitoring and run-time reconfiguration operations of services distributed over multiple heterogeneous cloud providers.*** Monitoring will be in charge of detecting the need of redistributing services on several cloud providers, reconfiguring and migrating modules when required.
Offer unified application management of services distributed over different cloud providers. SeaClouds will be able to deploy, manage, scale and monitor services over technologically diverse clouds providers.	***Compliance with major standards for cloud interoperability.*** SeaClouds will manage applications deployed on technologically diverse cloud platforms, promoting the adoption of OASIS TOSCA and CAMP standards for cloud interoperability.

Figure 1 illustrates how SeaClouds intends to relate to existing cloud initiatives and standards **[10]**. The top layer shows the main components generated in SeaClouds, and the other layers depict the relationships with existing efforts.

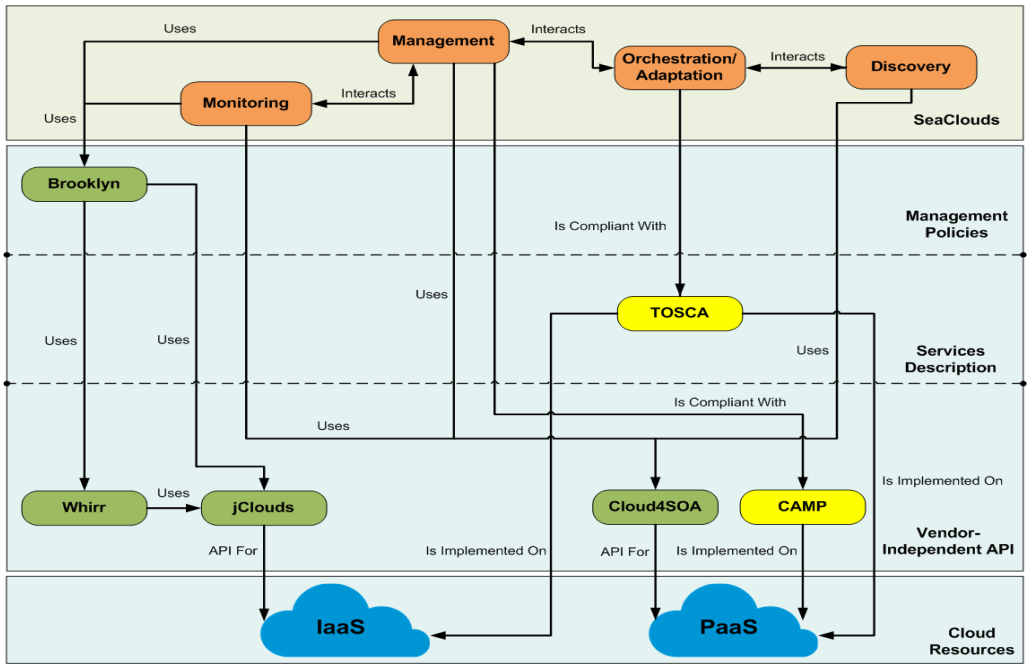

Figure 1. Position of SeaClouds with respect to related initiatives

The SeaClouds Approach

Figure 2 shows the cloud architecture situation before (top) and after SeaClouds (bottom). Without SeaClouds, services can only be deployed, managed and monitored across multiple clouds as standalone applications, and not as part of a composite application. This has the consequence that there is no support for synchronized deployment and unified monitoring, which implies that the QoS of the entire application is difficult to monitor. There is no support for migrating one service and reconfiguring the rest of the application to use the migrated service, in case a provider does not respect its SLA.

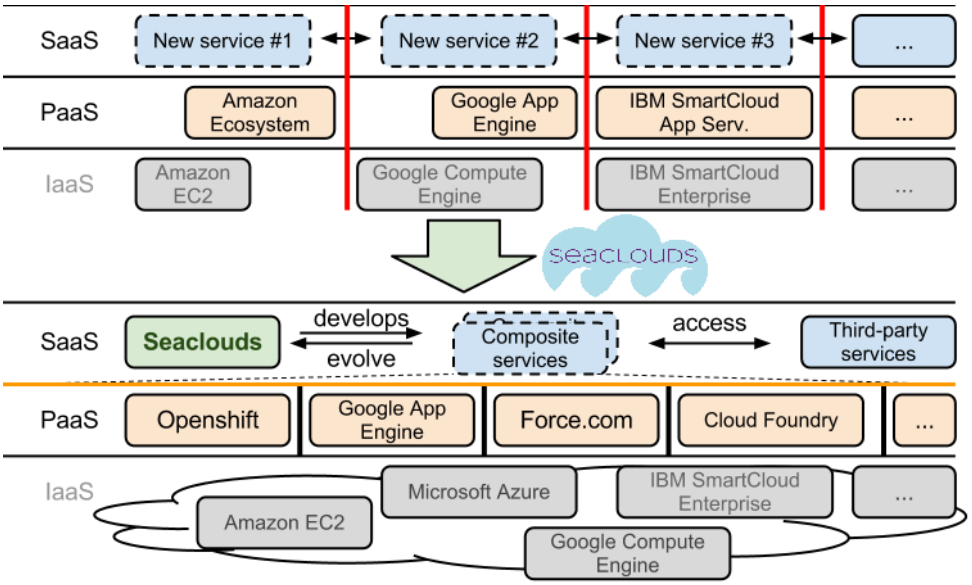

Figure 2. Cloud architecture before and after SeaClouds

SeaClouds aims at homogenizing the management over different providers and at supporting the sound and scalable orchestration of services across them. Moreover, systems developed with SeaClouds will inherently support the evolution of their constituent services, so as to easily cope up with needed changes, even at runtime. The development, monitoring and reconfiguration via SeaClouds include a unified management service, where services can be deployed, replicated, and administered by means of standard harmonized APIs such as CAMP specification and Cloud4SOA project.

In the following, we list some of the current problems and barriers, related to the cloud that will be solved by the main results expected from SeaClouds.

1. ***Support for application deployment and migration to different providers***. SeaClouds will provide support for deploying and migrating applications composed of several services taking care of the synchronization of the services and their reconfiguration, without requiring the user to manually intervene.

2. ***Management and monitoring of underlying providers***. Properties over application and services deployed on multiple clouds can be ensured and managed in a standardized way by using unified metrics and automated auditing.
3. ***Increased availability and higher security***. The usage of formal models to support the management of service-based applications over multi-clouds environments gives more flexibility to reconfigure the distribution as a SLA violation occurs.
4. ***Performance and cost optimization***. The framework gives users freedom to distribute application requirements over different cloud offerings by using needed options in a flexible manner. Organizations can take advantage of useful and powerful services provided by each platform and avoiding its weaknesses. Optimization requirements can also be modelled to consider cost as the main decision parameter.
5. ***Low impact on the code and user-friendly interface***. SeaClouds will tackle different problems for developers and administrators of cloud applications thanks to the proposed orchestration model. First, by simplifying the development process with SeaClouds' range of tools and framework that require minor impact on the code, and second, by simplifying the management of already deployed complex cloud applications thanks to the SeaClouds dashboard.

SeaClouds Open Reference Architecture

This main section presents the reference architecture and design of the SeaClouds platform, and discusses its novel aspects compared to existing initiatives and efforts. Figure 3 depicts the reference architecture of the SeaClouds platform.

Before describing the core components of the architecture of the SeaClouds platform, it is worth observing that the platform features a **Graphical User Interface (GUI)** for two user roles (Designers and Deployment Managers), and that Cloud Provider Systems are considered. The main **stakeholders for the SeaClouds platform** are the following:

- ***Application Designer*** (or ***Developer***) exploits 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, and the maximum acceptable cost for the entire deployment.
- ***Deployment Manager*** *(or* ***Application Manager****)* exploits the GUI through a **Unified Dashboard** that allows them to supervise the deployment and the monitoring of the application.

- ***Cloud Providers*** provide the Cloud Resources (which offer some Cloud Capabilities). They do not necessarily interact directly with the SeaClouds platform, but the services offered are exploited by the platform to run service compositions.

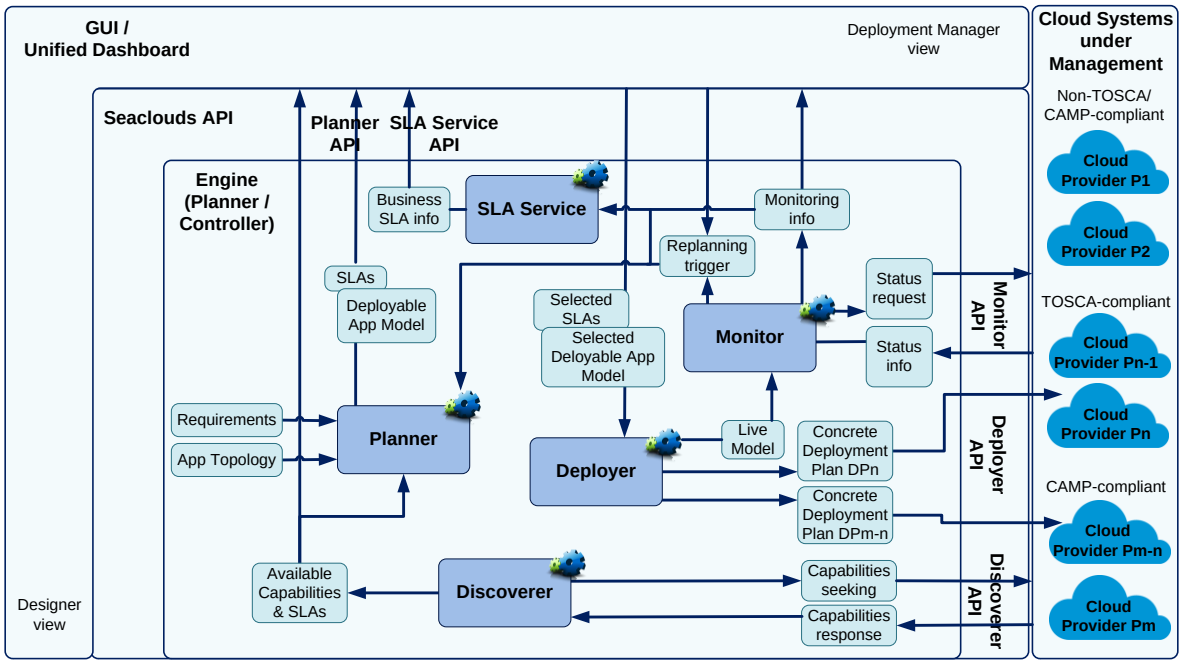

Figure 3. Initial Architecture of the SeaClouds Platform

As regards the **SeaClouds platform functionalities**, five components are identified in the architecture, in addition to the REST **harmonized and unified SeaClouds API** used for the deployment, management and monitoring of simple cloud-based applications through different and heterogeneous cloud providers.

- ***SeaClouds Discoverer***: in charge of discovering (by using the Discoverer API) available capabilities and add-ons offered by available cloud providers.
- ***SeaClouds Planner****:* in charge of implementing planning policy (using the Planner API) to orchestrate the multi-cloud deployment of the application modules.
- ***SeaClouds Deployer***: in charge of taking as input the orchestration specification generated by the Planner, and deploying (by exploiting the Deployer API) the application modules on the specified clouds.
- ***SeaClouds Monitor***: in charge of monitoring (by means of the Monitor API) that the QoS properties of the application modules and the whole application are not violated by the clouds in which they were deployed. Also in charge of generating the reconfiguration suggestions (if needed) to be passed as inputs to the Planner component to trigger the generation of a new adaptive orchestration plan.

- ***SeaClouds SLA Service***: in charge of mapping (by using the SLA Service API) the low level information gathered from the Monitor into business level information about the fulfilment of the SLA defined for a SeaClouds application.

Figure 4 represents the steps necessary to carry out an application deployment from the initial stage where the Application Developer (end-user) provides the Application Model consisting of the Module Profile and the Topology representing the connections among the modules of the cloud application to be deployed (other elements as the SLA restrictions and policies are considered by SeaClouds), as described in detail in Deliverable D3.1 **[11]**, related to the design-time.

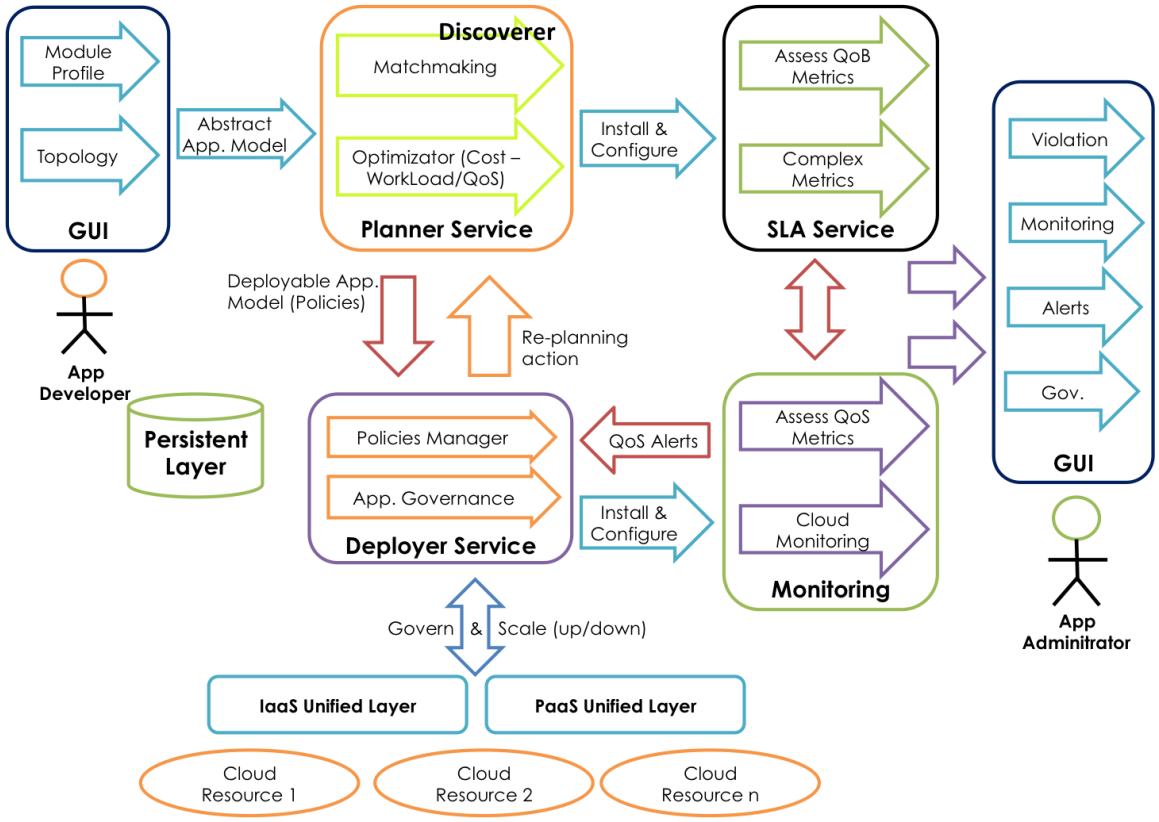

Figure 4. Interaction flow between the SeaClouds components

SeaClouds follows the Application Model Lifecycle depicted in Figure 5. After the Abstract Application Model has been specified, SeaClouds starts the Discoverer and Planner stage. Once the cloud providers have been discovered, the Planner acts with two subprocesses: Matchmaking and Optimizer (described in D3.1). The Planner specifies an Abstract Deployment Plan (ADP) that defines, in an abstract way, where each application module will be deployed.

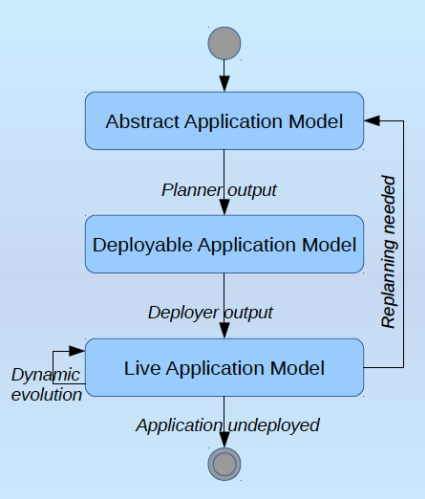

Figure 5. Interaction between the SeaClouds components

Then, as result of the Planner, by means of the ADP, a Deployable Application Model (DAM) is generated, which specifies the concrete cloud services used to distribute the application. The DAM allows the deployment of the application's modules over heterogeneous IaaS and PaaS, using the Deployer component, detailed in Deliverable D4.1 **[12]** related to the run-time environment. Once the application is deployed, the Deployer manages it and notifies the Monitor, which initializes the monitoring of the applications and interacts with the SLA service to manage the violations of the QoS and QoB, and properties. A Live Application Model maintains a track of the dynamic evolution of the deployment and management of the application modules.

A GUI as Dashboard is also used for the interaction with the Application Administrator. Note that a Persistent Layer should be considered to maintain a continue store (e.g., the QoS violations).

A distinguishing aspect of the SeaClouds architecture is that it builds on top of two OASIS standards initiatives: **TOSCA** and **CAMP**. On the one hand, besides employing TOSCA to represent application topologies, TOSCA's plans are also exploited in the deployment phase to generate TOSCA CSARs (Cloud Service ARchives - containing an application specification together with implementation and deployment artifacts) that can be automatically processed by any TOSCA-compliant platform. On the other hand, SeaClouds also employs CAMP, which proposes standardised artifacts and APIs that need to be offered by PaaS clouds to manage the building, running, administration, monitoring and patching of applications in the cloud. It is however worth noting the Deployer does not

require cloud providers to be TOSCA or CAMP compliant, and it actually generates concrete deployment plans for non TOSCA/CAMP compliant providers as needed.

Next, we describe more in detail the components and services, their functionalities, interactions, and the inputs/outputs of the SeaClouds platform.

SeaClouds Discoverer

The main functionality of the Discoverer component is described in Table 1.

Component	***Discoverer***
Description/ Functionality	The ***Discoverer*** component is in charge of discovering available capabilities offered by cloud providers. The description of such capabilities includes technology aspects (e.g., programming languages, development frameworks, runtime environments, add-ons), QoS properties (e.g., availability, reliability), along with the associated SLAs (including the cost associated to each provided service). The Discoverer periodically crawls the cloud providers and stores the discovered capabilities in a repository which is accessible to the Planner component as well as to the Deployment Manager.
Inputs	Cloud provider capabilities (specified as cloud meta-model into TOSCA concepts) and desired SLAs.
Outputs	Cloud provider capabilities available and SLAs to be sent to the planner and the dashboard.
Interactions and Interfaces	This component interacts with the Planner and the Dashboard. The Planner will consume the result of the discoverer to perform a matchmaking process and to decide where to deploy each application module according to its QoS and technology requirements. The GUI/Dashboard will present the result of the discoverer to the end-user. Optionally, this component could also receive automatic updates from cloud providers.

Table 1. Discoverer component description

SeaClouds Planner

The Planner description and architecture is presented in Table 2 and Figure 6, respectively.

Component	***Planner***
Description/ Functionality	The ***Planner*** component receives from the application designer a description of the application topology together with a set of requirements (that include QoS properties and technology requirements) for the application modules, and it determines (by consulting the capabilities repository) how the application modules can be distributed over the available clouds without violating the given set of requirements. The Planner is first triggered by the application Designer input and then by replanning triggers generated by the Monitor component (possibly filtered by the Deployment Manager). The Planner generates an intermediate Abstract Deployment Plan (ADP), describing a feasible distribution of the application modules over the available clouds, with the final result as output of a Deployable Application Model (DAM) specifying the concrete cloud services where distributing the application modules. This is generated in two steps, Matchmaking and Optimization, includes the concrete services associated with each Base module and the policies to manage the scaling mechanism of each module.
Inputs	QoS properties and technology requirements, application topology, available capabilities and SLAs, and replanning trigger.
Outputs	Deployale Application Model (DAM) and related SLAs.
Interactions and Interfaces	The Planner receives the requirements and application topology from the Application Designer, by means of the GUI/Dashboard. It interacts with the Discoverer component to acquire the available capabilities and SLAs. It specifies the ADP used to generate the concrete Deployable Plan, DAM (returned to the Deployment Manager), so it connects also with the Deployer. It receives replanning trigger from the Monitor component, and also from Deployment Manager (connected through the GUI/Dashboard).

Table 2. Planner component description

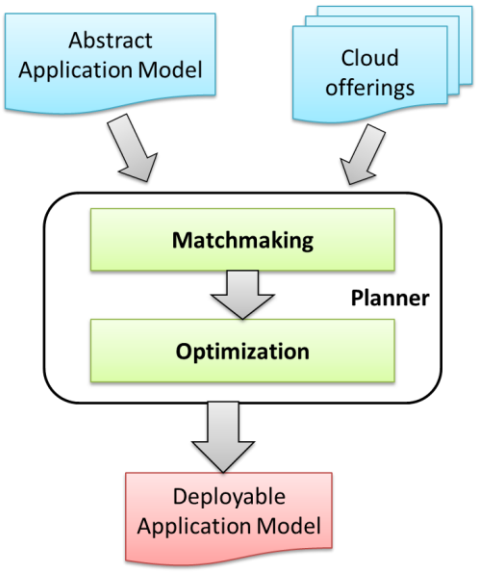

Figure 5. Architecture of the Discoverer and Planner components

As depicted in the architecture, the generation of the final Deployable Application Model using the Abstract Deployment Plan is performed in two steps:

1) Matchmaking: this first step aims to identify the cloud resources that are suitable to allocate each module.
2) Optimization: once a set of suitable cloud services have been identified for each modules, an optimization process can be performed.

SeaClouds Deployer

Table 3 describes the main functionality of the Deployer component, whose architecture is presented in Figure 7.

Component	***Deployer***
Description/ Functionality	The ***Deployer*** component inputs a deployable application model, together with the SLAs of the selected services, and it internally generates 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 (set of) specific cloud platform(s).
Inputs	The Deployable Application Model (DAM), which contains the plan describing the steps to deploy the application. This plan has to be approved by the Deployment Manager. Also, the selected SLAs are

	an input of the Deployer.
Outputs	A Live Model of the managed applications, which contains the services used by an application, the location for each of the application modules and the relationships among modules. Also, the concrete deployment plan, used to deploy the application modules in every cloud provider.
Interactions and Interfaces	The Deployer receives the Deployable plan from the Planner, and by means of the GUI/Dashboard confirms the DAM. Interaction with the target platforms, cloud providers, using the needed services to deploye the application modules. The Monitor is connected to the Deployer by using synchronization events.

Table 3. Deployer component description

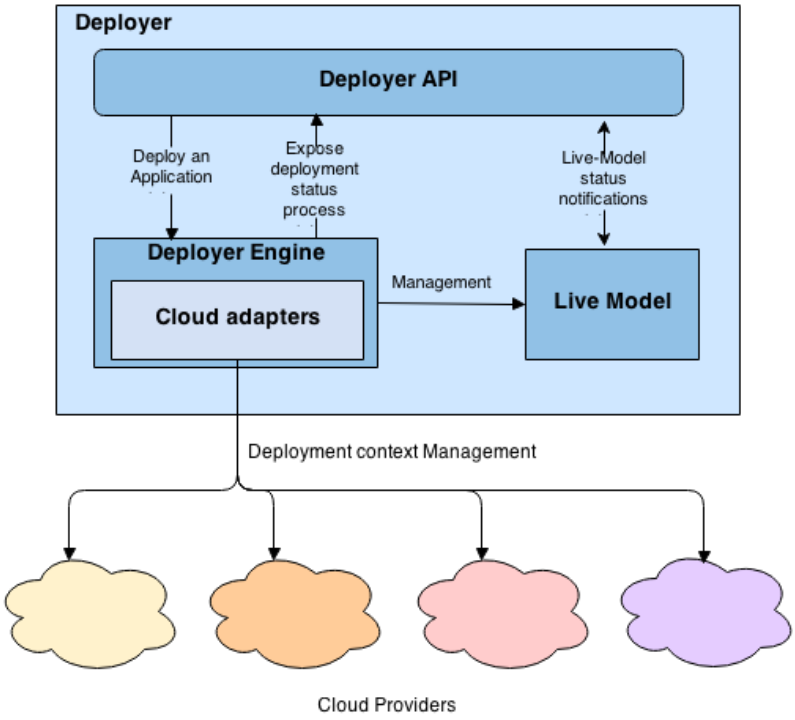

Figure 6. Architecture of the Deployer component

The deployer is composed of several elements. The main element of our SeaClouds Deployer is the Deployer Engine. The Deployer Engine receives a Deployable Application Model (DAM) through its Deployer API and executes the DAM. As the Deployer Engine is cloud-agnostic, it is able to deploy applications on different cloud providers using multiple Cloud Adapters (PaaS and IaaS levels). Once the application has been deployed, the

Deployer Engine uploads/upgrades the Live Model, which contains the data structure (components and relationship between these) in order to maintain topology of the application. Currently, SeaClouds is using Brooklyn **[13]** as Deployer Engine to accomplish the multideployment of the application components and the Live Model generation and management. The application components could be deployed over different cloud providers simultaneously (using jClouds **[14]** as Cloud Adapter at the IaaS level). Thus, we define the DAM based on the YAML Blueprint specification of Brooklyn **[15]**.

Figure 8 shows the steps performed during the deployment of a cloud application using the SeaClodus Deployer, considering the interaction with the rest of components (the full description can be found in **[12]**).

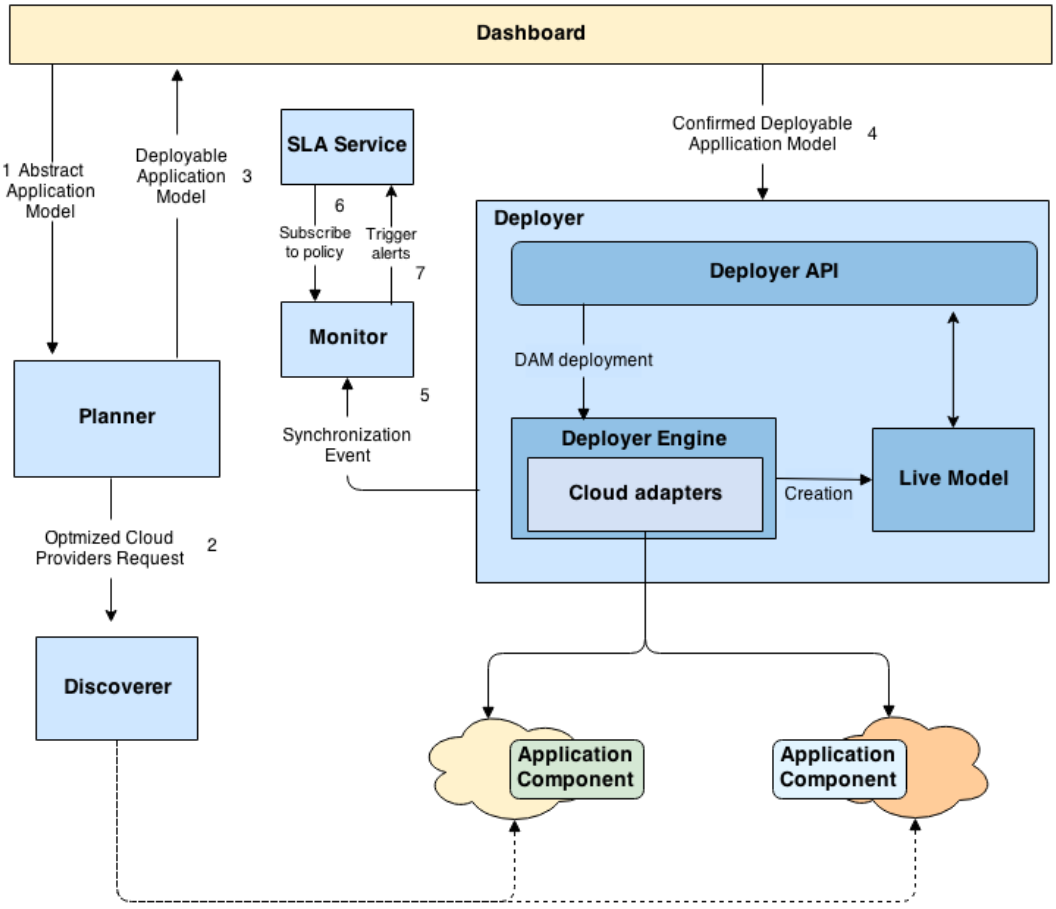

Figure 7. Deployment strategy: interaction between components

SeaClodus Monitor

In Table 4 and Figure 9 are presented, respectively, the functionality and architecture of the Monitor component.

Component	*Monitor*
Description/ Functionality	The ***Monitor*** component gets the set of SLAs of the services in the selected deployment plan, and is in charge of collecting monitoring information from the targeted cloud platforms, of analysing such information, and of presenting the results of such analysis (through the SeaClouds dashboard) to the Deployment Manager. The Monitor is also in charge of generating replanning triggers that are passed (possibly filtered by the Deployment Manager, depending on the platform configuration) to the Planner in order to start a reconfiguration process.
Inputs	Available capabilities and a set of the services in the selected deployment plan SLAs. It has the knowledge about the live model storing information about how the distribution of the application modules is done and deployed in cloud providers.
Outputs	Monitoring information related to the metrics and replanning triggers, alerts.
Interactions and Interfaces	Connection with the Dashboard to receive the information about the deployment plan, the SLAs. The Monitor provides the collected monitoring information (or the results from its analysis) to the GUI/Dashboard. The Monitor triggers a replanning, connected to the Planner and the SLA Service, if problematic situations that require it are detected.

Table 4. Monitor component description

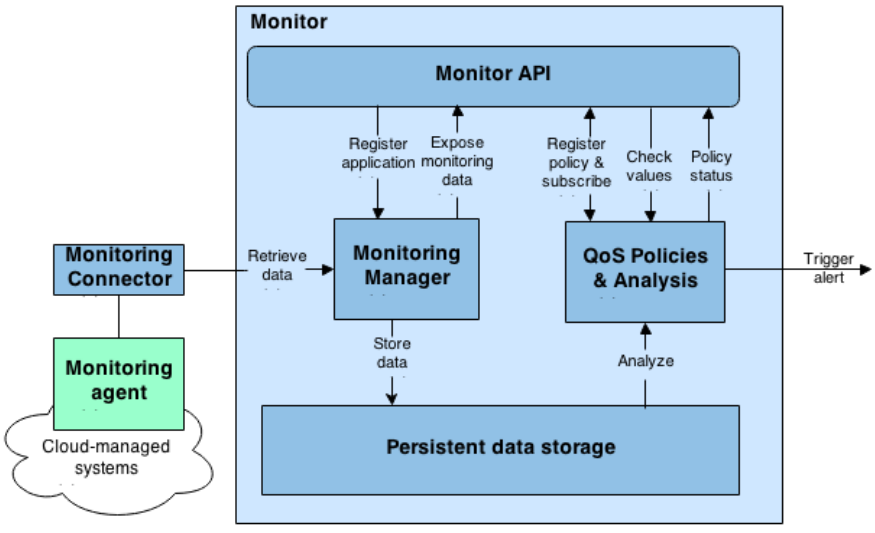

Figure 8. Architecture of the Monitor component

The Monitor functionality is centralized on the Monitoring Manager, which acts as a registry for new applications deployed and it is also the mechanism to manage the data generated from the Monitoring Agent by retrieving it, exposing it through the Monitor API and storing it for later usage and analysis. As an initial approach, we make use of Brooklyn as the main Monitoring Agent, taking advantage of the management capabilities and mechanisms incorporated in this tool, such as sensors, data feeds, enrichers and policies (we are also studying the incorporation of MODAClouds **[16]** in the SeaClouds Monitor). In this way we can define new data retrieving mechanisms for any application managed by SeaClouds that will be gathered as QoS properties on the Monitor platform.

Figure 10 shows an overall diagram of the interaction between the Monitor and the rest of components in SeaClouds, checking and analyzing the violations of QoS policies, and triggering the corresponding alerts. The procedure of registering a new application into SeaClouds is one of the most useful processes to explain how monitoring data is managed. The monitoring process starts once the target application enters the deploying phase. At this point, the Deployer module will notify the Monitoring Manager, through Monitor API, registering the new application in the Monitoring Manager. Registering process will require information about the Monitoring Agent in use, so that metrics can be collected and exposed to the rest of the SeaClouds modules using Monitor API.

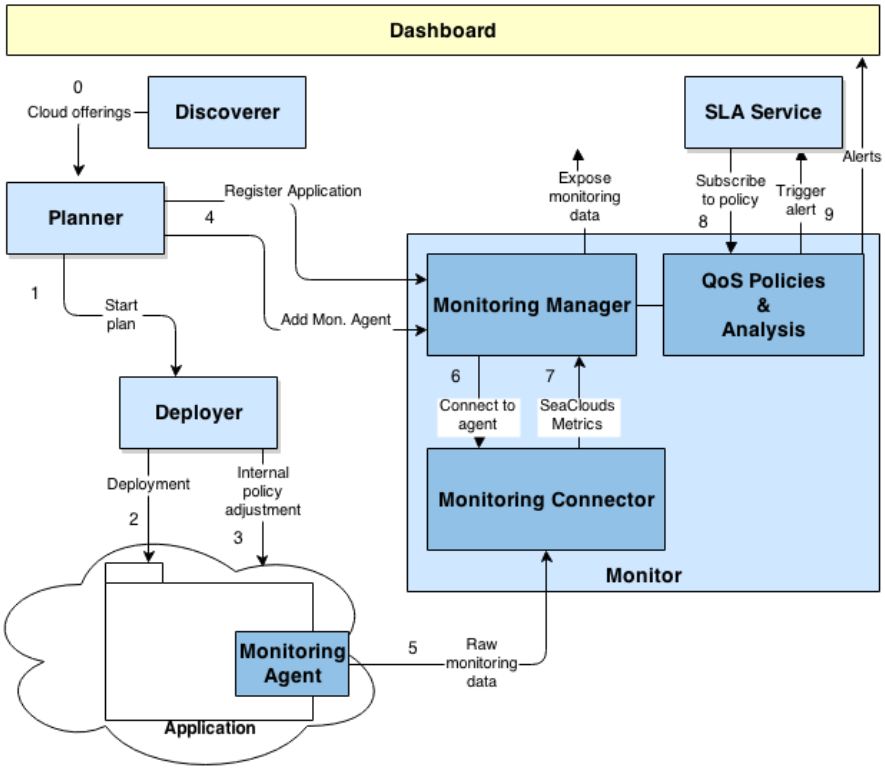

Figure 9. Monitoring strategy: interaction between components

SeaClouds SLA Service

As regards the SLA Service, its functionality and architecture is described in Table 5 and Figure 11, respectively.

Component	***SLA Service***
Description/ Functionality	The ***SLA Service*** is in charge of mapping the low level information gathered from the Monitor into business level information about the fulfilment of the SLA defined for a SeaClouds application. It is responsible for establishing, reviewing and cancelling of complex end-to-end- Service Level Agreements (SLAs) between Application Providers and Cloud Suppliers. 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.
Inputs	Monitoring info, and trigger alerts.
Outputs	Business SLA info.
Interactions and Interfaces	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, by using the GUI/Dashboard. Finally, the Monitor component will generate business metrics to evaluate agreements, by using the monitoring info and according to the trigger alerts.

Table 5. SLA Service description

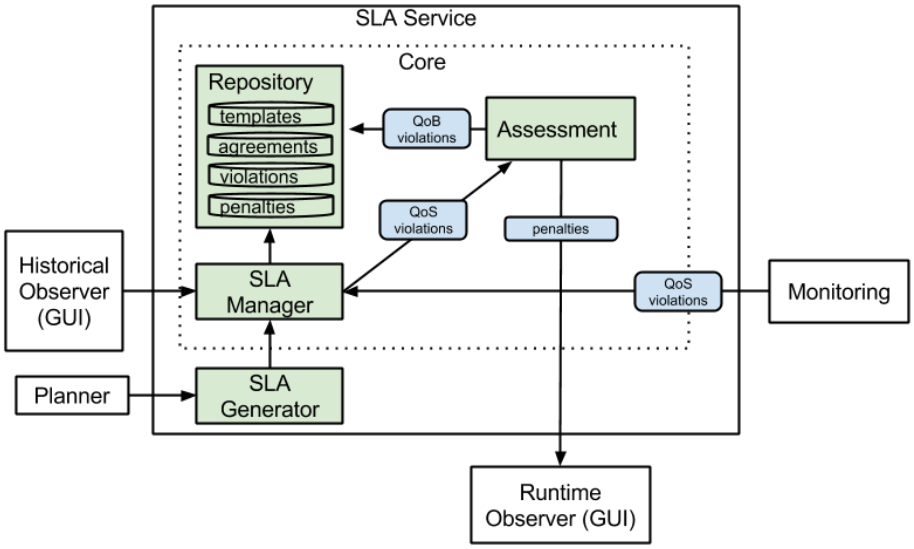

Figure 10. Architecture of the SLA Service

The SLA Service enables the Service Level Agreements (SLA) management of business oriented policies. The SLA Service is an implementation of the WS-Agreement specification **[17]**. The main responsibilities of the SLA service are: generating and storing WS-Agreement templates and agreements, and assessing that all the agreements (SLA guarantees) are respected by evaluating the business penalties. Currently, Brooklyn is used as Monitoring Service to accomplish the constraints evaluation.

Novel Aspects in the SeaClouds Architecture

There are some novel aspects for the proposed SeaClouds platform, and its proposed architecture, compared to the previous initiatives and efforts, and we would like to discuss some of them in the following.

Multi-cloud orientation. As described previously, cloud computing has proven a major commercial success in the last years, with the appearance of many different vendors. What followed is a need for integrating multiple heterogeneous clouds and to solve the problem of distributing the services over several providers. In particular, the need of orchestration is more evident when complex applications move to cloud environments. With the current cloud technologies, services can only be deployed, managed and monitored on multiple clouds as stand-alone applications, and not as part of a composite application. Thus, in a scenario where a complex application is distributed on different cloud service providers, a solution is needed to manage and orchestrate the distribution of modules in a sound and adaptive way. The SeaClouds platform is proposed to solve these problems and advance the field by supporting the orchestration and deployment to

multiple clouds and management thereon, including resilience and migration of modules that compose cloud-based applications over multiple and technologically diverse clouds offerings. Based on the concept of cloud-based services orchestration, SeaClouds can realise the automated arrangement, coordination, deployment and management of multiple services as a single aggregated complex application in an efficient and adaptive way, without the need of modifying the code of the services. This allows organisations to embrace cloud solutions and avoid risks of unreliability and vendor lock-in. By solving the problems caused by the multiple-vendor scenario, the SeaClouds architecture would benefit not only application developers and cloud providers, but also the whole market, by reducing the adoption barrier for new players.

Separation between (abstract) planning and (concrete) deployment. In the SeaClouds platform, the planner component is in charge of determining a distribution of application modules onto multiple available clouds in the form of abstract plans (as deployable application model), while the deployer component is finally in charge of instantiating a concrete plan so as to actually deploy the application modules (using the information defined in the deployable application model, and deploying in the concrete cloud providers). This is based on a strategy of separating high-level planning and low-level deployment. With this separation, the SeaClouds platform becomes more scalable and flexible. Since the plans generated by the component planner are high-level abstract ones, and they are not bound to a specific language for deployment and management, the platform is capable of generating the concrete deployment plan either TOSCA-compliant or CAMP-compliant, or both. Moreover, if some new standard or language for cloud application deployment and management, which is widely accepted by industry, appears in the future, the SeaClouds platform can be scaled more easily to adapt to the new situation, by just modifying the deployer component or adding an interpreter into it for the new emerged language. This scalability and flexibility can also help SeaClouds to avoid the risks resulting from the possible delay or even failure of CAMP and TOSCA standardisation activities, or the case that they are not widely accepted by the industry. In addition, by this separation, the planner and deployer components can be more easily reused, respectively, by other developers or initiatives that need them in multi-cloud scenarios.

Use of TOSCA to represent the application topology. In the SeaClouds platform, the newly emerged OASIS standard TOSCA is employed to specify the topology of complex applications. TOSCA enables the interoperable description of application and infrastructure cloud services, the relationships between parts of the services, and the operational behavior of these services, independently from the supplier creating the services, and any particular cloud provider or hosting technology. In line with the main

goals of TOSCA, the use of this standard will ease automated deployment and management, and will enhance the portability and reusability of multi-cloud applications and their components. In addition, it will also allow the SeaClouds platform to generate TOSCA-compliant orchestration specifications, which will ease the matching and interoperation with TOSCA-compliant PaaS offerings.

Compatibility with CAMP. The SeaClouds platform is compatible with the novel OASIS standard CAMP, which is one of the major standards for cloud interoperability. Its objective is to define standardised artefacts and APIs that a PaaS should offer to allow the management, building, administration, monitoring and patching of cloud-based applications. Obviously, the availability of CAMP results can simplify the development of the Discovery API, Monitoring API, and part of the Multi-Cloud Deployment API of SeaClouds. By extending and incorporating CAMP, we can cover all future CAMP-compliant providrs or tools, allowing application developers to manage applications hosted on multiple clouds environments. Furthermore, by leveraging CAMP, SeaClouds will attract a significant user base (as this standard has a lot of interest but no reference implementations, so far) and advance the standard, ensuring the long-term viability of the benefits implied in SeaClouds, i.e., management and monitoring of underlying providers, performance optimisation, low impact on the code, formal methods support, flexibility to include new services and react to problems at runtime. On the other hand, SeaClouds can provide valuable feedback and contribute to the standardization effort of CAMP, both by implementing a CAMP-compliant interface towards PaaS providers for management, and by contributing review proposals that will possibly emerge while specifying properties of the included orchestrations, adaptation and monitoring. This can be particularly valuable since, as mentioned, there are no CAMP implementations at the moment, and therefore the protocol has not been tested.

Compatibility with different target platforms. In addition to the above-mentioned cloud standards as TOSCA and CAMP, SeaClouds also uses several existing platforms and initiatives, such as Brooklyn, Whirr, jClouds, Cloud4SOA, and MODAClouds. The Cloud4SOA project provides an open source interoperable framework for application developers and PaaS providers. It facilitates developers in the deployment and lifecycle management of their applications on the PaaS offering that best matches their computational needs, and ultimately reduces the risks of a vendor lock-in. SeaClouds will leverage and extend Cloud4SOA outcomes, such as multiplatform matchmaking, management, cloud monitoring and migration, to ease and accelerate the implementation. SeaClouds will use Brooklyn's policy-driven functionality to integrate support for IaaS providers. In addition, Brooklyn's approach to policy modelling and enforcing will provide guidance for the orchestration, adaptation, and management

functionality. From these different platforms and initiatives, SeaClouds will take advantage of the compatibility and reuse of relevant tools and APIs provided by them, and also obtain a good user base. On the other hand, SeaClouds will definitely be able to contribute back to them. For example, Brooklyn only targets the IaaS level and has no support for orchestration. Beyond what Brooklyn provides, SeaClouds will therefore extend policy-driven functionality to the PaaS level and also add support for adaptation and orchestration. Thus, Brooklyn can benefit from integrating the proposed functionalities, especially regarding the integration of adaptation techniques in supported policies, thereby increasing the adoption rates and the market size of the Brooklyn platform.

Conclusions

In this document, we have presented the results of our research as regards the ***Open Reference Architecture of the SeaClouds project***. The project aims at providing an open source framework to address the problem of deploying, managing and reconfiguring complex applications over multiple and heterogeneous clouds.

The SeaClouds approach works towards achieving *"Agility After Deployment"* by tackling the problem from the service orchestration perspective. Following the architecture proposed, first, the exploitation of the best available offering for each application component at any time is performed. Then, a complex application, which consists of modules and (technological and QoS) requirements, is provided as input to the SeaClouds planner. The latter generates the orchestration by assigning (groups of) modules to different cloud providers. Such orchestration is then deployed and monitored according to standard metrics. If requirements are violated, then the SeaClouds monitor will generate reconfiguration information which leverages the creation of a different orchestration of the application***.***

Therefore, ***the proposed architecture can well support this process of deploying and managing a cloud application over multiple clouds***.

Note that, thanks to the seamless distribution over several different platform and infrastructure clouds, applications developed in SeaClouds will also take advantage of higher availability (via inter-cloud redundancy), higher security (via inter-cloud data partition) and higher throughput (via inter-cloud load balancing).

A key ingredient in our proposal is the use of two OASIS standards initiatives for cloud interoperability, namely CAMP and TOSCA, which allow us to describe the topology of user applications independently of cloud providers, provide abstract plans, and discover, deploy/reconfigure, and monitor our applications independently of the particularities of the cloud providers.

Acknowledgements

The SeaClouds project is partially supported by the European Commission grant no. FP/-ICT-2013-10-610531 (SeaClouds).

Authors

This architecture oriented paper has been created by SeaClouds Scientific and Technical Team according to the work made by all members of SeaClouds' Technical and Management Team:

- **ATOS**: Francesco D'Andria, Román Sosa
- **UMA**: Ernesto Pimentel, Javier Cubo, Francisco Durán, Jose Carrasco, Miguel Barrientos, Adrián Nieto
- **UPI**: Antonio Brogi, PengWei Wang, Michela Fazzolari, Jacopo Soldani, Ahmad Ibrahim
- **POLIMI**: Elisabetta Di Nitto, Raffaela Mirandola, Diego Pérez
- **CloudSoft**: Alex Heneveld, Andrea Turli
- **NURO**: Christian Tismer

References

1. P. Mell, T. Grance. The NIST definition of cloud computing. NIST Special Publication, http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf, 2011.

2. M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, M. Zaharia. A view of cloud computing. Commun. ACM, no. 4, p. 50-58, 2010.

3. N. Loutas et al. Deliverable D1.1 Requirements Analysis Report. Cloud4SOA Project Deliverable, 2011.

4. OASIS. CAMP 1.1 (Cloud Application Management for Platforms), Version 1.1, http://docs.oasis-open.org/camp/camp-spec/v1.1/camp-spec-v1.1.pdf, 2014.

5. DMTF. Cloud Infrastructure Management Interface (CIMI) Model and REST Interface over HTTP, An Interface for Managing Cloud Infrastructure, Version 1.0.0, http://dmtf.org/sites/default/ les/standards/documents/DSP0263 1.0.0.pdf, 2012.

6. DMTF. Virtualization Management (VMAN), http://dmtf.org/standards/vman, 2014.

7. OASIS. TOSCA 1.0 (Topology and Orchestration Specification for Cloud Applications), Version 1.0, http://docs.oasis-open.org/tosca/TOSCA/v1.0/TOSCA-v1.0.pdf, 2013.

8. A. Parameswaran, A. Chaddha. Cloud Interoperability and Standardization. SETLabs Briefings - Infosys, vol. 7, no. 7, pp. 19-26, 2012.

9. SeaClouds Project. Deliverable D2.2 Initial architecture and design of the SeaClouds platform (SeaClouds Consortium), http://seaclouds-project.eu/deliverables/SeaClouds-D2_2-Initial_architecture_and_design_of_the_SeaClouds_platform.pdf, 2014.

10. A. Brogi, J. Carrasco, J. Cubo, F. D'Andria, A. Ibrahim, E. Pimentel, J. Soldani. EU Project SeaClouds: Adaptive Management of Service-Based Applications Across Multiple Clouds. Proceedings of the 4th International Conference on Cloud Computing and Services Science (CLOSER 2014), SCITEPRESS, 2014.

11. SeaClouds Project. Deliverable D3.1 Discovery, Design and Orchestration Functionalities: First Specification (SeaClouds Consortium), http://seaclouds-project.eu/deliverables/SEACLOUDS-D3.1-Discovery_Design_and_Orchestration_Functionalities%20_First_Specification.pdf, 2014.

12. SeaClouds Project. Deliverable D4.1 Definition of the multi-deployment and monitoring strategies (SeaClouds Consortium), 2014.

13. Apache Brooklyn. https://brooklyn.incubator.apache.org/, 2014.

14. Apache jClouds. The Hava Multi-Cloud Toolkit, https://jclouds.apache.org/, 2014.

15. Brooklyn YAML Blueprint Reference, http://brooklyncentral.github.io/v/0.7.0-SNAPSHOT/use/guide/defining-applications/yaml-reference.html, CloudSoft, 2014.

16. MODAClouds European Project, http://www.modaclouds.eu/, 2014.

17. Guide to WS-Agreement Language, Open Grid Forum https://packcs-e0.scai.fraunhofer.de/wsag4j/wsag/wsag-language.html, 2014.