# SeaClouds: Seamless adaptive multi-cloud management of service-based applications*

Antonio Brogi[1], José Carrasco[2], Javier Cubo[2], Francesco D'Andria[3], Ahmad Ibrahim[1], Ernesto Pimentel[2], and Jacopo Soldani[1]

[1] Department of Computer Science, University of Pisa, Italy
[2] Department of Computer Science, University of Malaga, Spain
[3] ATOS, Spain

**Abstract.** How to deploy and manage, in an efficient and adaptive way, complex applications over multiple heterogeneous PaaS platforms is one of the problems that have emerged with the cloud revolution. The recently started EU research project 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 PaaS platforms. In this paper, after presenting context, motivations and objectives of the project, we position SeaClouds with respect to related cloud initiatives and discuss its initial architecture.

**Keywords.** Multi-cloud deployment, service orchestration, monitoring, dynamic reconfiguration.

## 1 Introduction

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 assists 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 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 the cloud, 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 infrastructure, 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

* Work supported by EU-FP7-ICT-610531 SeaClouds project.

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.

There is a need for integrating multiple heterogeneous clouds and to solve the problem of distributing services over several providers [4]. Thus, 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., performance, cost, availability, scalability, etc.). Once the distribution has been decided, the solution 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.

Consider an application consisting of three modules ($A$, $B$, and $C$), each having its own (technological and QoS) requirements ($R_A$, $R_B$, $R_C$). Suppose that four PaaS platforms are available ($\alpha$, $\beta$, $\gamma$, and $\delta$), each providing different offers ($P_\alpha$, $P_\beta$, $P_\gamma$, and $P_\delta$) and based on different (proprietary) technologies. The application developer can deploy the application either on a single PaaS or over multiple PaaS platforms. If she follows the second solution, then she has to figure out the best offer for each module (e.g., $R_A : \{P_\gamma\}$, $R_B : \{P_\beta, P_\gamma\}$, $R_C : \{P_\delta\}$) and deploy them accordingly (e.g., $A \rightarrow \gamma$, $B \rightarrow \beta$, $C \rightarrow \delta$). Afterwards, she has to monitor both the whole application and the single modules. In case some requirements are violated (e.g., $R_B$ is no more satisfied by $P_\beta$), she has to manually reconfigure the deployment (e.g., $A \rightarrow \gamma$, $B \rightarrow \gamma$, $C \rightarrow \delta$). This implies that, to manage multi-cloud applications, the developer must have knowledge of the different employed PaaS platforms and has to continuously monitor the application. This is a costly and cumbersome process and it would be much more efficient to have some framework doing all the work automatically.

In this paper, we discuss the ongoing project SeaClouds (Seamless adaptive multi-cloud management of service-based applications) which focuses on the problem of deploying and managing complex multi-component application over heterogeneous clouds in an efficient and adaptive way. SeaClouds works towards giving organizations the capability of "Agility After Deployment" for cloud-based applications. Its 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 when the monitoring detects that such properties are not respected. So, SeaClouds' main objective is *the development of a novel platform which performs a seamless adaptive multi-cloud management of service-based applications*. More specifically:

$O_1$) *The 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 cloud offerings.

$O_2$) *The 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. As a consequence of monitoring, dynamic reconfiguration will be used to evolve the orchestration by considering all the changes required (without suspending the execution of services not affected by those changes). Reconfiguration may imply updating a service, dynamically replacing erroneous services or migrating them to a different cloud provider to leverage its advantages or avoid the shortcomings of another cloud provider.

$O_3$) *The offer of 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. Such operations will be performed taking into account the synchronization requirements of the application as a whole, providing developers with support beyond the handling of single services.

$O_4$) *The compliance with major standards for cloud interoperability.* SeaClouds will manage applications deployed on technologically diverse cloud platforms, unifying operations such as monitoring and lifecycle management, promoting the adoption of OASIS standards for cloud interoperability (e.g., CAMP [5], TOSCA [6]).

The rest of the paper is organized as follows. Section 2 will position SeaClouds with respect to current cloud initiatives and provide the main challenges it wants to overcome. Section 3 will present the SeaClouds approach along with its architecture and expected results. Finally, we will present conclusions and research perspectives in Section 4.

## 2 Positioning and challenges of SeaClouds

In this section we describe how SeaClouds, with the purpose of achieving the main goal, advances the state of the art with respect to the project's objectives $O_1$, $O_2$ and $O_3$, and contributes to obtain $O_4$.

### 2.1 Orchestration and adaptation in the cloud

$O_1$ will be addressed by developing cloud service orchestrators of the cloud-based application modules, and by adapting the specified orchestration . Orchestrators are widely used in the service-oriented computing paradigm [7,8,9,10,11], mainly focusing on behavioural and context-aware adaptation of services, by coordinating the interactions between different services. Several approaches exist that target formal verification and adaptation of orchestrated services, but, to the best of our knowledge, none of these approaches has been extended to the cloud environment. A cloud-compliant orchestration is not a trivial problem. Challenges such as heterogeneity of cloud platforms and migration to different cloud providers have to be addressed, as well as the different standards emerging from distinct vendors. Therefore, existing approaches should be (substantially) extended to operate on heterogeneous cloud providers.

**Challenges in orchestration and adaptation for the cloud.** SeaClouds will address the following challenges in order to extend service-oriented approaches to the cloud:

– Adaptation contracts need to take into account cloud providers characteristics and Service Level Agreement (SLA).
– Violations of Quality of Service (QoS) properties need to be monitored across different cloud platforms.
– Dynamic architecture reconfiguration might involve migrating some components of the application to other cloud providers at runtime.

The latter two challenges (addressed by $O_2$ and $O_3$) are discussed in the following sections.

## 2.2 Monitoring of multi-cloud services

The ongoing EU FP7 Cloud4SOA project (`http://www.cloud4soa.eu`) provides an open source interoperable framework for application developers and PaaS providers. Cloud4SOA facilitates developers in the deployment and lifecycle management and monitoring of their applications on the PaaS offering that best matches their computational needs, and ultimately reduces the risks of a vendor lock-in. The monitoring is based on unified metrics, but Cloud4SOA monitors each application separately and it is not able to aggregate monitoring results of multi-component applications.

Several commercial and open source initiatives target monitoring of cloud applications. Often these initiatives address only particular platforms, for example Appsecute (`http://www.appsecute.com`) monitors only (open-source) CloudFoundry-based platforms. More platform-independent technologies are available for the IaaS level, since the latter has undergone a stronger harmonization effort. Deltacloud (`http://deltacloud.apache.org/`) encapsulates the native API cloud provider to enable management of resources in differents IaaS clouds, such as Amazon EC2. Rightscale (`http://www.rightscale.com`) supports monitoring several public (e.g. Amazon Web Services, Rackspace) and private IaaS clouds (e.g. CloudStack, Eucalyptus, OpenStack). Truly platform-independent monitoring solutions exist, the most known being NewRelic (`http://www.newrelic.com`). NewRelic achieves platform-independency by requiring each provider to implement a monitoring component and integrate it in the offered cloud platform. On the one hand, this approach yields the best results from a monitoring point of view. On the other hand, it forces providers to invest quite some resources in order to implement the monitoring.

**Challenges in monitoring of services on multiple clouds.** In order to address $O_2$, SeaClouds' monitoring will use and enhance existing monitoring functionalities for the PaaS and IaaS levels.

– With respect to the IaaS level, SeaClouds will simply reuse what is available (e.g., Deltacloud).
– With respect to the PaaS level, SeaClouds aims at augmenting the set of metrics currently available from Cloud4SOA (response time and up-time).

For both the IaaS as the PaaS level, SeaClouds aims at coordinating, monitoring and aggregating monitoring information at the single service level to serve the purposes of orchestrated services. Thus, SeaClouds aims at:

– Being able to monitor each of the application components, and
– Combining and aggregating the above mentioned data to highlight performance problems and their impact.

## 2.3 Unified management of multi-cloud applications

Brooklyn (`http://brooklyn.io`) is an open source, policy-driven control plane for distributed applications delivered by CloudSoft (a member of the SeaClouds consortium). It enables single-click deployment of applications across machines, locations and clouds. Then it continuously optimizes running applications to ensure ongoing compliance with policies. Brooklyn uses two open source tools to operate on cloud resources, Apache Whirr (`http://whirr.apache.org/`) and Jclouds (`http://www.jclouds.org/`) that supports several IaaS providers. The already mentioned Cloud4SOA project also offers deployment and lifecycle management functionality using a harmonized API layer to encapsulate the providers APIs.

**Challenges in unified application management of services distributed over different cloud providers.** SeaClouds will use Cloud4SOA's management functionality. Specifically:

– SeaClouds' discovery functionality may use and extend existing matchmaking functionalities to match application requirements with PaaS offerings.
– SeaClouds management will use the REST harmonized API for the deployment, management and monitoring of simple cloud-based applications across different and heterogeneous cloud PaaS offerings.

SeaClouds intends to use Brooklyn's policy-driven functionality to integrate support for IaaS providers. Moreover, Brooklyn's approach to policy modeling and enforcing can provide guidance for SeaClouds' orchestration/adaptation and management functionality. On the other hand, 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. Due to a common partner in both initiatives (CloudSoft), Brooklyn can also benefit from integrating SeaClouds' functionality, especially regarding the integration of adaptation techniques in supported policies.

### 2.4 Standards for cloud interoperability

CAMP (Cloud Application Management for Platforms) [5] aims at defining a harmonized API, models, mechanisms and protocols for the self-service management (provisioning, monitoring and control) of applications in a PaaS, independently of the cloud provider. However, CAMP is only a protocol specification, so it needs to be implemented by parties adopting the protocol.

The OASIS TOSCA (Topology and Orchestration Specification for Cloud Applications) [6] Technical Committee aims at enhancing the portability of cloud applications and services. The main aim of TOSCA is to enable the interoperable description of application and infrastructure cloud services, the relationships between parts of the service, and the operational behaviour of these services, independently from the cloud provider. By increasing service and application portability in a vendor-neutral ecosystem, TOSCA aims at enabling portable deployment to any compliant cloud, smoother migration of existing applications to the cloud, as well as dynamic, multi-cloud provider applications.

**Challenges in standards for cloud interoperability.** SeaClouds intends to actively contribute to the standardization effort of CAMP [12] 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 SeaClouds orchestrations, adaptation and monitoring.

SeaClouds will exploit the TOSCA specification to drive the design of the model for specifying cloud service orchestrations in SeaClouds. In doing so, SeaClouds might actively contribute to the standardization effort of TOSCA, by contributing review proposals that will emerge while trying to devise a TOSCA-compliant instances of the SeaClouds service orchestration model. On the other hand, SeaClouds will also focus on developing functionalities that are deliberately out of scope of TOSCA [13] to solve the issues about policies for the dynamic management of service orchestrations.

Although current implementations of TOSCA and CAMP do not support the management of complex application over multiple clouds, SeaClouds will work towards building such management on top of them.

### 2.5 Positioning of SeaClouds

Figure 1 synthesises the relations between SeaClouds and the aforementioned initiatives.

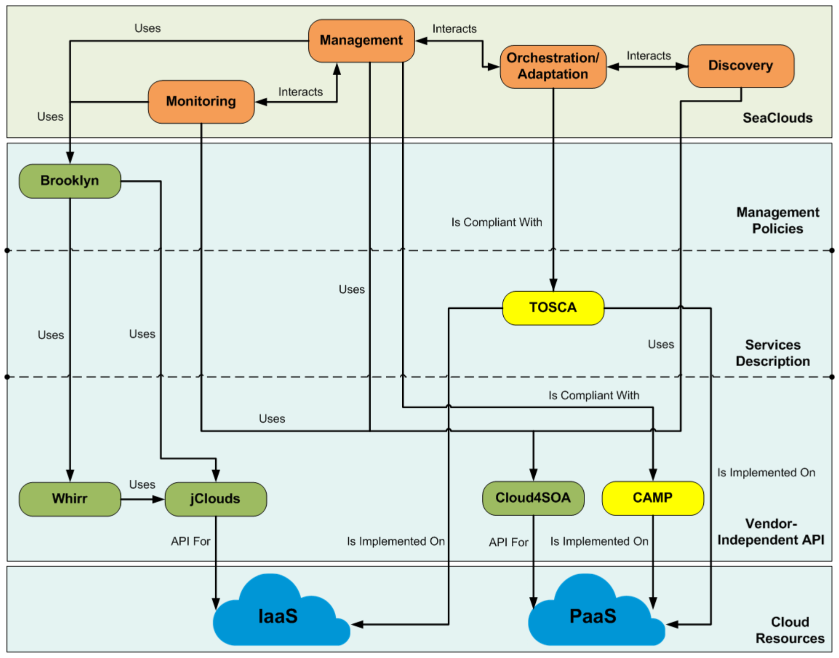

**Fig. 1.** Position of SeaClouds with respect to related initiatives.

## 2.6 Related cloud initiatives

It is important to stress the fact that SeaClouds approach uses adaptation via orchestration and therefore it does not require code modifications to existing services. There are several initiatives and standards that target services deployed on the cloud, and aim at guaranteeing properties such as Quality of Service of those services. These initiatives use different approaches, with the consequence that software developers need to either use special APIs or programming models to code their applications, or to model them using project-specific domain languages. Some of these projects are mentioned in following.

The Broker@Cloud project (`http://www.broker-cloud.eu/`) aims at helping enterprises to transition to the cloud while enforcing quality control on the developed services. Capabilities for cloud service governance and quality control such as lifecycle management, dependency tracking, policy compliance, SLA monitoring, and certification testing are included in the project. Nonetheless, Broker@Cloud targets a brokering architecture, where the above mentioned services are available, and therefore cannot change the orchestration of the deployed services to adapt to changing conditions.

The MODAClouds project (`http://www.modaclouds.eu/`) also aims at providing quality assurance during the application life-cycle, support migration from cloud to cloud when needed, and techniques for data mapping and synchronization among multiple clouds. In order to do so, MODAClouds requires software developers to adopt a Model-Driven Development approach. This approach has therefore, differently from SeaClouds, an impact on the code that needs to be deployed on the cloud.

Also the PaaSage project (`http://www.paasage.eu/`) has Quality of Service has one of its goal. PaaSage also intends to match application requirements against platform characteristics and make deployment recommendations and dynamic mapping of components to the platform(s) selected for the application instantiation. Analogously to MODAClouds, it also requires the developers to adopt a modeling language in order to specify the model of the application.

The mOSAIC project (`http://www.mosaic-cloud.eu/`) aims at providing developers with vendor agnostic APIs, so that the resulting applications can be deployed on different IaaS using a sort of mOSAIC virtual machine. mOSAIC plans also to support SLA negotiation (with monitoring to detect SLA violations) and application life-cycle, but requires developers to adopt the project's API.

## 3 SeaClouds Approach

Figure 2 shows the cloud architecture situation currently before SeaClouds (top), and after SeaClouds (bottom). Without SeaClouds, services can only be deployed, managed and monitored on 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 QoS of the entire application is difficult to monitor. There is also 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.

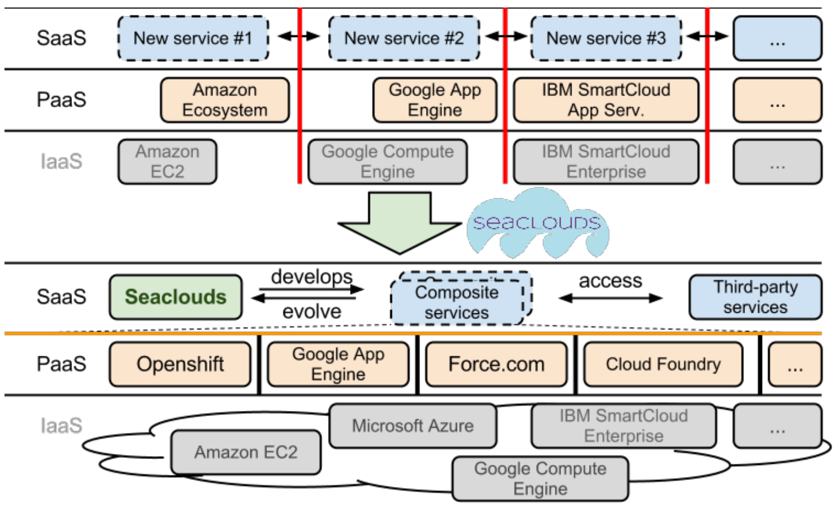

**Fig. 2.** Cloud architecture before and after SeaClouds.

SeaClouds aims at homogenizing the management over different providers and support 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 includes a unified management service, where services can be deployed, replicated, and administered by means of standard harmonized APIs such as CAMP specification [5] and Cloud4SOA (`http://www.cloud4soa.eu/`).

### 3.1 Overall strategy and Architecture overview

We list next 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.* 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.* Using standardized and unified metrics and automated auditing, properties over application and services can be ensured (on multiple clouds in a unified and standardized way).
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, 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 take cost as the main decision parameter.
5. *Low impact on the code and user-friendly interface.* SeaClouds will tackle particular 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 with SeaClouds dashboard.

Figure 3 shows SeaClouds initial architecture, including the various architectural components and modules employed by the SeaClouds platform. The figure also presents the level of relationship with standards and some other already existing solutions. SeaClouds aims to support developers and application managers by providing a set of mechanisms, languages, and tools in the design, deployment and runtime phases. In the figure, the boxes **Architecture and**

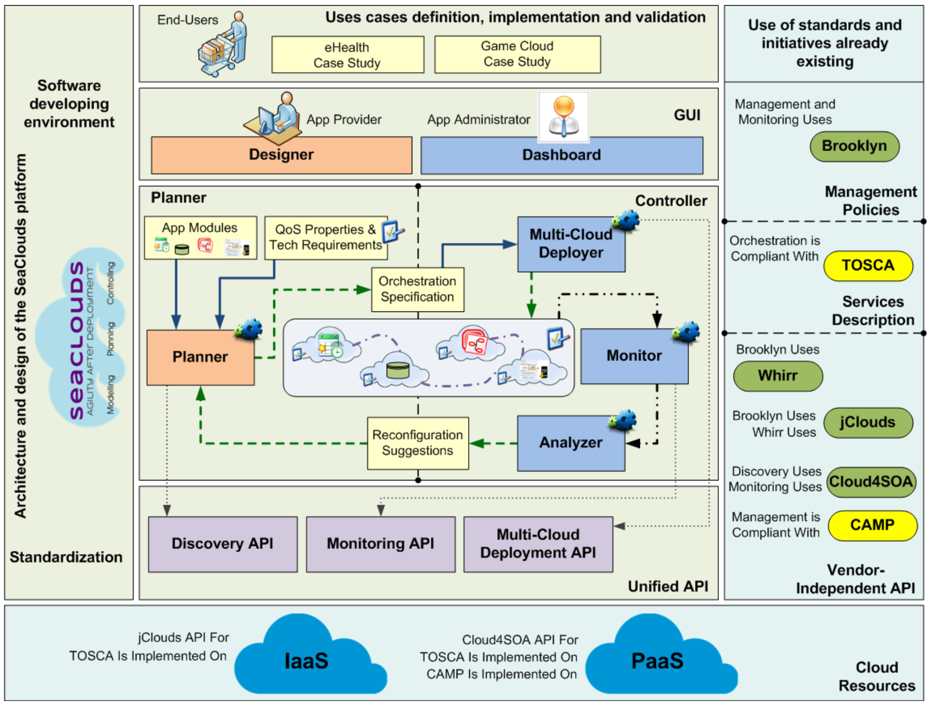

**Fig. 3.** SeaClouds Architecture.

**design of the SeaClouds platform**, **Uses cases definition, implementation and validation**, **GUI**, **Planner/Controller**, and **Unified API** represent the contributions of SeaClouds (i.e., the expected outcomes of the SeaClouds project), where the set of mechanisms and tools generated will be organized in a framework. This framework will be available either as software to install on premises or as SaaS or a combination of both. The boxes **Use of standards and initiave already existing**, **Management Policies**, **Services Description**, **Vendor-Independent API**, and **Cloud Resources** correspond to the different standards, solutions, libraries, or projects used, promoted or aligned by SeaClouds, and to the situation in the cloud computing technologies (SaaS, PaaS, IaaS).

The **Architecture and design** of the SeaClouds platform is in charge of generating the final framework containing all the mechanisms and tools specified and developed during the project life. It also receives feedback for the definition and implementation of the software developing environment, GUI, and the standardization activities.

The **Uses cases definition, implementation and validation** represents the definition, development and validation of the SeaClouds case studies prototypes that will operate in the frame of preselected realistic use-cases and scenarios. The performance of the evaluation of the case studies and the generation of lessons learnt and methodological adoption guidelines for cloud computing will be also considered here.

In order to obtain the main goals of SeaClouds as regards the development of the platform and the definition, implementation and validation of the uses cases, SeaClouds provides a foundation to allow **"Agility After Deployment"** providing necessary tools and a framework for *Modelling*, *Planning* and *Controlling* cloud applications (Planner/Controller, GUI, and Unified API). Here, two main areas (separated by a black dotted line) are identified in the figure that corresponds with the Planner and Controller.

The arrow's coding in Figure 3 is the following: normal and dashed lines are the inputs and outputs to/from the components, respectively; dashed-dotted lines refer to internal executions between components; and dotted lines are the connection between the components and the corresponding APIs.

The **Planner** module will implement SeaClouds planning policy to orchestrate the multi-cloud deployment of the application modules. The Planner will take the input data provided by the SeaClouds users, which will specify which are the application modules to be deployed on multiple clouds, as well as the desired QoS properties for the SLA of the whole application and/or the desired QoS properties and technology requirements needed for individual application modules. The Planner will exploit the **Discovery API** to discover the capabilities and add-ons featured by available clouds, and it will generate as output an *orchestration specification* of the application modules over the chosen clouds.

The **Controller** module will implement the multi-cloud deployment of the application modules and SeaClouds monitoring policy. In particular, the **Multi-Cloud Deployer** component will input the orchestration specification generated by the Planner, and it will deploy (by exploiting the **Multi-Cloud Deployment API**) the application modules on the specified clouds. The **Monitor** component of the Controller will be in charge of monitoring (by exploiting the **Monitoring API**) that the QoS properties of the application modules are not violated by the clouds in which they were deployed, and that the whole application satisfies the QoS properties specified for the whole application. If the monitoring component will detect a (non-transient) property violation, it will trigger the **Analyzer** component which is in charge of generating the *reconfiguration suggestions* (if needed) to be passed as inputs to the Planner module to trigger the generation of a new adaptive orchestration plan.

Users will exploit the **GUI** module to interact with SeaClouds. The Graphical User Interface (GUI) module will contain a **Designer** component to allow users to provide SeaClouds input data (viz., application modules, or QoS properties and technology requirements). The GUI will also contain a **Dashboard** component to provide users with a friendly management and visualization of the state of the multi-cloud deployed applications, by providing a unified access to the Discovery, Deployment and Monitoring APIs (**Unified Management API**).

As regards the alignment and promotion of SeaClouds with standards, solutions, libraries, or projects, the figure shows (on the right side) the interaction of this project with the **standards and initiatives already existing**. Thus, Sea-

Clouds will implement the protocols defined in the OASIS standards (viz., CAMP, TOSCA), and will use and expand existing tools and APIs (e.g., Brooklyn).

To sum up, SeaClouds platform will provide a set of services, components, and /modules which ease the design and implementation of service orchestrators to perform a seamless adaptive multi-cloud management of service-based applications, with added-value, which can be deployed, monitored, dynamically migrated, elastically scaled and distributed among several different PaaSs.

But also the idea behind SeaClouds is to look for the non-dependence with the SeaClouds platform, so the applications deployment with SeaClouds will not require SeaClouds services throughout their entire life cycle, and thus will be inherently independent both from their underlying infrastructure and our provided services, and therefore, they are not locked in by any PaaS, IaaS or middleware. This means that once the SeaClouds components have helped the user to the decision of distributing and deploying the modules of his/her application, that application can run without using SeaClouds services. Then, apart from the regular way of working of SeaClouds, previously described, it may also offer the user the possibility of (i) simply getting a plan from SeaClouds using the SeaClouds Planner module (without letting SeaClouds performing the deployment), or of (ii) going ahead without SeaClouds after the first deployment, in which case the Multi-Cloud Deployer simply performs the deployment, (without triggering the Monitoring module). In both cases, modifications in the modules already developed by SeaClouds will not be necessary.

## 3.2 Expected results

The SeaClouds project will result, through both reports and prototypes, in:

- *Formal orchestration of services on the cloud.* High-level orchestration specifications will be used to describe the mapping between the services (dealt as modules that compose the cloud-based applications) to compose and the goal expected from their orchestration. This will ease the development of composite services since the developer does not need to deal with concurrency issues and will allow the process to be agnostic to the actual implementation language or the underlying cloud provider.
- *Monitoring process.* SeaClouds performs a monitoring process of the QoS properties of both the application modules and the whole application in order to determine whether it is required to perform an evolution, or migration of some cloud services distributed on several PaaS.
- *Traceability of QoS violations.* Monitoring will have to trace possible property violations back to the causes of such violations, so as to suggest to the reconfiguration mechanisms the provider or the service to be replaced in order to re-establish the soundness of the deployed adaption.
- *Reconfiguration mechanism.* Our proposal allows a real flexible reconfiguration, with lighter services without extra migration logic at the cost of slightly higher time-to-service. During monitoring SeaClouds considers the verified properties

determined in the verified adaptation objective, with the idea of preserving the soundness of SeaClouds orchestrating adaptors.

- *Unified management API for clouds.* SeaClouds facilitates the access and the administration of both public and private cloud providers providing multi-cloud management tools as well as offering cloud providers and consumers a REST-based approach to cloud-based application management. Using this API, SeaClouds can deploy, stop, start, and update applications, increasing consumers' ability to port their applications between cloud provider offerings.
- *Dashboard for administration of services distributed between PaaS.* This will constitute a graphical Web interface so to have a single-point where to visualize the current configuration of the deployed services and perform changes in their scalability parameters, resource consumption, migration and general management.
- *Web-based development environment for the design of service orchestrations.* This will provide an alternative, visual access to the RESTful API provided by orchestration services. This will support the orchestration specifications and it will show the result of the verification via model-based simulation.
- *Contribute to the development of CAMP and TOSCA.* Provide feedback to CAMP by proposing new functionalities/resources to be added to working version of the specification, and implement a CAMP-compliant interface to talk to PaaS providers. Contribute to core concepts and usage patterns of Service Templates of TOSCA, as well as to the standardization effort of TOSCA, by providing feedback that will emerge while trying to devise a TOSCA-compliant instance of the SeaClouds service orchestration model. In TOSCA, issues like portability and service composition might benefit from some of the adaptation techniques in SeaClouds, driving the design of the model for specifying cloud service orchestrations. Also the composition of service templates could be improved with a dynamic reconfiguration mechanism, providing more flexible conditions to substitute service templates.

## 4 Conclusions and Perspectives

In this paper we have presented our ongoing research in 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. 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 PaaS platforms. Such orchestration is then deployed and monitored according to standard metrics. If requirements are violated, then the SeaClouds analyzer will generate reconfiguration information which leverages the creation of a different orchestration of the application. Please note that, thanks to the seamless distribution over several different PaaS platforms, applications developed in SeaClouds will also take advantage of higher availability (via inter-PaaS redundancy), higher security (via inter-PaaS data partition) and higher throughput (via inter-PaaS load balancing).

Our goal is to provide both the open source framework and the standardization of the PaaS monitoring metrics. We aim at obtaining such results by maintaining compliance with OASIS emerging standards for cloud applications.

## 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 (2010) 50–58
3. N. Loutas et al: D1.1 Requirements Analysis Report. Cloud4SOA Project Deliverable, `http://www.cloud4soa.eu/sites/default/files/Cloud4SOA%20D1.1%20Requirements%20Analysis.pdf` (2011)
4. Parameswaran, A., Chaddha, A.: Cloud Interoperability and Standardization. SETLabs Briefings - Infosys **7** (2012) 19–26
5. OASIS: CAMP 1.0 (Cloud Application Management for Platforms), Version 1.0. `http://docs.oasis-open.org/camp/camp-spec/v1.1/camp-spec-v1.1.html/` (2012)
6. 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` (2012)
7. A. Brogi, J. Camara, C. Canal, J, Cubo and E. Pimentel: Dynamic contextual adaptation. In: In Proc. of Fifth International Workshop on the Foundations of Coordination Languages and Software Architectures 2006 (FOCLASA'06), vol. 175, no. 2, Elviser (2007) 81–95
8. C. Canal, P.P., Salaun, G.: Model-Based Adaptation of Behavioural Mismatching Components. IEEE Transactions on Software Engineering **34** (2008) 546–563
9. J. Cámara, J.A. Martín, G. Salaün, J. Cubo, M. Ouederni, C. Canal and E. Pimentel: Itaca: An integrated toolbox for the automatic composition and adaptation of web services. In: Proc. of International Conference on Software Engineering 2009 (ICSE'09), IEEE Computer Society Press (2009) 627–630
10. H. Nezhad, G.Y. Xu and B. Benatallah: Protocol-aware matching of web service interfaces for adapter development. In: Proc. of 19th International Conference on World Wide Web 2010 (WWW'10), ACM (2010) 731–740
11. J. Cubo and E. Pimentel: Damasco: A framework for the automatic composition of component-based and service-oriented architectures. In: In I. Crnkovic, V. Gruhn, M. Book (editors), European Conference on Software Architecture 2011 (ECSA'11), Lecture Notes in Computer Science 6903, Springer-Verlag (2011) 388–404
12. OASIS: Cloud Application Management for Platforms (CAMP) Technical Committee Charter. `https://www.oasis-open.org/committees/camp/charter.php` (2013)
13. OASIS: OASIS Topology and Orchestration Specification for Cloud Applications (TOSCA) Technical Committee Charter. `https://www.oasis-open.org/committees/tosca/charter.php` (2013)