Adaptive management of applications across multiple clouds: The SeaClouds Approach

Antonio Brogi, Michela Fazzolari, Ahmad Ibrahim,
Jacopo Soldani, PengWei Wang
Dipartimento di Informatica, Università of Pisa, Italy
{*brogi,fazzolar,ahmad,soldani,pengwei*}*@di.unipi.it*

and

Jose Carrasco, Javier Cubo, Francisco Durán, Ernesto Pimentel
Universidad de Malága, Departamento de Lenguajes y Ciencias de la Computación, Spain
{*josec,cubo,duran,ernesto*}*@lcc.uma.es*

and

Elisabetta Di Nitto
Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Italy
elisabetta.dinitto@polimi.it

and

Francesco D'Andria
ATOS, Spain
francesco.dandria@atos.net

Abstract

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 paper we present context, motivations and objectives of the EU research project SeaClouds, which 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. After positioning SeaClouds with respect to related cloud initiatives, we present the SeaClouds architecture and discuss some of its aspect, such as the use of the OASIS standard TOSCA and the compatibility with the OASIS CAMP initiative.

Keywords. Multi-cloud deployment, service orchestration, monitoring, dynamic reconfiguration, cloud interoperability, cloud application management.

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 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 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.

In this paper, we present 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 applications over heterogeneous clouds in an efficient and adaptive way [4]. 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 expected. 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) *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) *Monitoring and run-time reconfiguration operations of services distributed over multiple heterogeneous cloud providers.* Monitoring will be in charge of detecting the possible 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. Reconfiguration may imply updating a service, dynamically replacing malfunctioning services or migrating them to a different cloud provider to leverage its advantages or avoid the shortcomings of another cloud provider.

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

O_4) *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, in particular TOSCA [7] and CAMP [4].

The rest of the paper is organized as follows. Section 2 will introduce a motivating example to illustrate the problems occurring when deploying an application on multiple cloud providers. Section 3 will position SeaClouds with respect to current cloud initiatives and single out the main challenges it wants to overcome. Section 4 will present the SeaClouds approach, while some concluding remarks will be drawn in Section 5.

[1] `http://www.enstratius.com/`

[2] `http://www.bmc.com/`

[3] `http://www.abiquo.com/`

[4] The present paper is an extended version of [9], and it provides an updated progress report on the evolution of the project w.r.t. [9, 10, 11]

2 Motivating Example

In order to motivate our proposal, we introduce an example where a multi-component application is going to be deployed on (potentially) different cloud providers, according to a number of requirements on each of the modules composing the application. After the (multi-cloud) deployment is performed, and components are being executed in different cloud platforms, a monitoring process is in charge of detecting possible requirements violations, which could eventually trigger a reconfiguration. Figure 1 shows the architecture of an online retailing application which is offered to clients as a shopping Webpage. It consists of four modules: A main web page to access the system, two databases (one for users and one for products), and a payment module. The functionalities of the four components of the online system are:

- The Main Webpage gives clients access to the cloud application. Technically speaking, it is a web application providing an HTML-accessible graphical user interface. Its purpose is to show available products by grouping them into categories, and to provide a shopping cart where clients can add products and place orders.
- The User Database records clients data, such as login info and current account data.
- The Stock Database stores information about shopping products and the corresponding stock.
- The Payment module is in charge of handling client payments. It communicates with the User Database to validate the login and the account info.

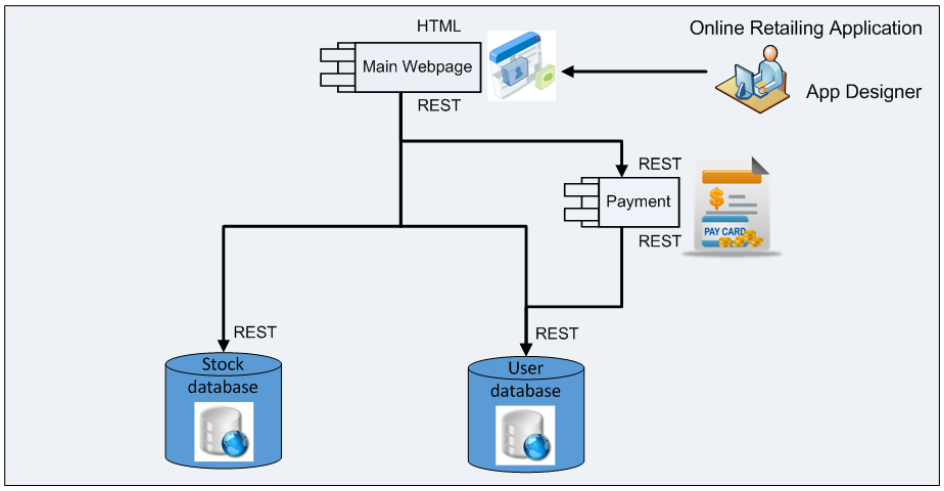

Figure 1: A component diagram for the online retailing system.

Together with the application topology provided by the component diagram above described, the SeaClouds platform will also input a collection of requirements both for the application as a whole, and for each module. These requirements may be divided into technology requirements and Service Level Agreement (SLA) / Quality of Service (QoS) requirements. Among the technology requirements, for this example, we consider the following ones:

- The modules of the application communicate through REST interfaces.
- The User and Stock databases are encoded in MySQL.
- The application is totally written in Java, with no specific version/API.

On the other hand, requirements concerning the SLAs could also be specified by the application designer. For instance:

- Availability of databases has to be greater than 99%.
- Error Rate for every separate component should be less than 5%.
- Average Response Time for the Payment component has to be less than 1s.
- Response Time of the Main Webpage has to be less than 500ms.

The SeaClouds platform gives an application designer the possibility of describing her application in terms of its modules and of the relationships between them. For each module, the platform gives the possibility of specifying the technology requirements and QoS parameters, providing minimum and maximum thresholds for automatic assignment and release of resources.

In order to perform a matchmaking process between the user requirements and the providers' offers, an automatic mechanism will discover capabilities provided by cloud platforms and will check dynamic changes in these offers. These capabilities will be considered when determining a deployment plan for application modules, together with the terms of the SLAs offered, with the aim of selecting the most suitable cloud provider for each application module.

Going back to our example, the programming language requirement allows the application to be deployed on almost any cloud platform (such as Google App Engine, AppFog, Windows Azure, AWS Elastic Beenstalk, Heroku, Cloud Foundry, Force, Cloudify, or CloudBees), since it uses a widely supported programming language. However, the requirement on the technology used for the databases prevent from deploying them on some cloud platforms. Furthermore, the SLAs published by the different cloud platforms provide information that permits matching the availability and error rate requirements. All requirements will be monitored at runtime, to detect possible violations, in which case the SeaClouds platform will trigger a reconfiguration.

Taking into account both the application topology and requirements, the SeaClouds platform will propose different deployment alternatives (Figure 2), among which the application deployment manager should choose one.

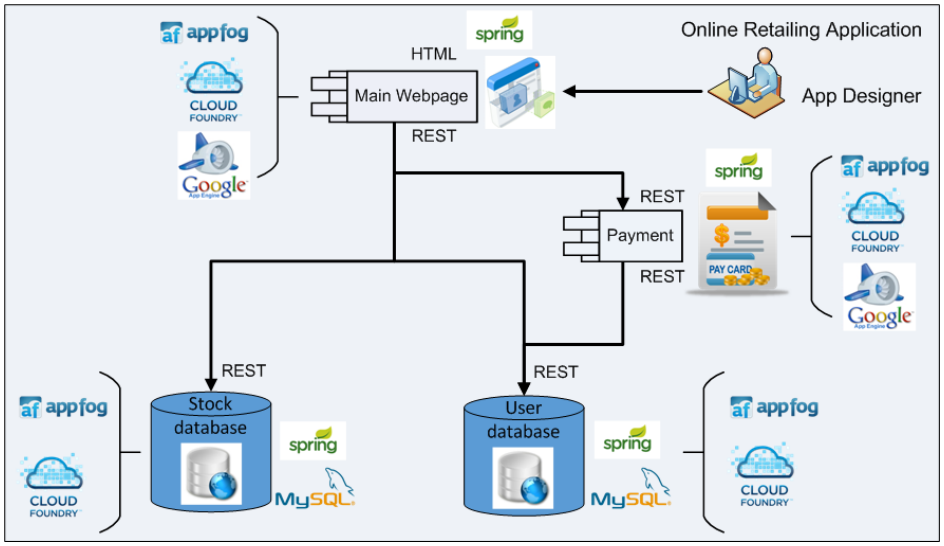

Figure 2: Different deployment alternatives for the online retailing system.

Once each application component has been deployed on a specific platform, different situations may occur at runtime, such as:

- Some of the cloud platforms possibly fail. The original cloud provider claims high availability rates but their services are not entirely exempt of failures.
- The elastic resources cause unexpected costs. Resources transparently scale as need, but this may provoke cost spikes.
- The business model of the original cloud provider changes (volatile market). The application hosted on that cloud provider could require rearranging it to exploit services, platforms and infrastructures which present the most beneficial cost/value ratio.

Considering the changing situations listed above, the application manager would need to have some modules of the application being moved from the platforms in which they were originally deployed to other platforms. To this end, the SeaClouds platform will provide monitoring facilities allowing the user to observe a set of standardised and unified metrics of different types, based on underlying cloud providers monitoring systems. This functionality will permit the runtime monitoring of the deployed modules so that the required QoS level can be assured.

If an issue related to the violation of the QoS or SLA is identified, some procedures may need to be executed for enforcing the SLA or for managing the violation through some reconfiguration of application modules. These procedures may require to stop, migrate and restart the service(s), and possibly migrate data, tackling challenges such as:

1. to perform this migration process as automatically as possible,
2. to minimise the downtime (viz., the time period in which the application will not be available),
3. to take care of synchronising or coordinating the rest of the application with the newly deployed service(s), and
4. to manage dynamically the new deployment of the application.

3 SeaClouds in Context

In this section we describe how SeaClouds, with the purpose of achieving its main goal, advances the state of the art with respect to the project's objectives O_1, O_2 and O_3 (presented in Section 1), and contributes to obtain O_4.

3.1 Orchestration and adaptation in the cloud

Objective 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 service-oriented computing [12, 13, 14, 15, 16], 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. 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 further in the following sections.

3.2 Monitoring of multi-cloud services

The 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 the 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 probably NewRelic (`http://www.newrelic.com`). NewRelic achieves platform-independency by requiring each provider to implement a monitoring component and to 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 IaaS and PaaS 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 and 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 application component, and at
- Combining and aggregating the above mentioned data to highlight performance problems and their impact.

3.3 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 support 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. Thanks 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.

3.4 Standards for cloud interoperability

CAMP (Cloud Application Management for Platforms) [4] 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) [7] 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 [17]. 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 [18] 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.

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.

3.5 Positioning of SeaClouds

Figure 3 synthesises the relations between SeaClouds and the aforementioned initiatives.

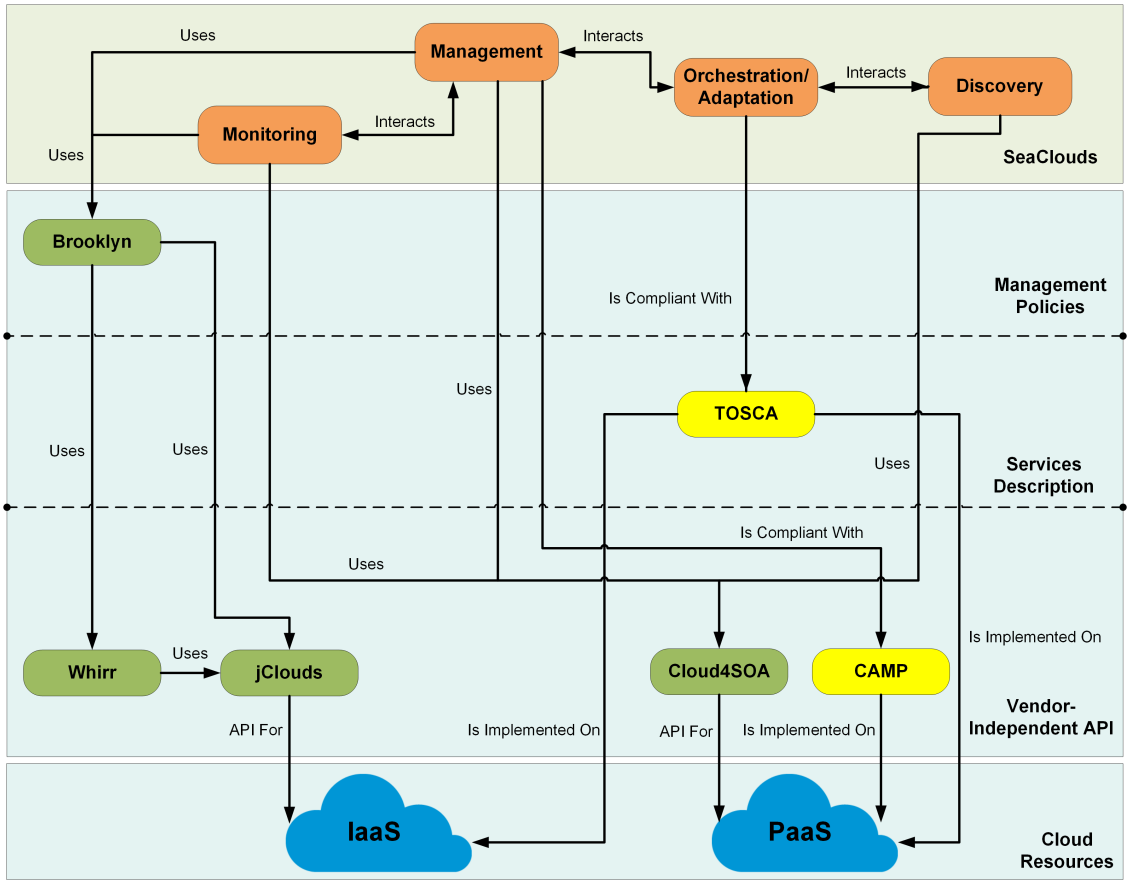

Figure 3: Positioning of SeaClouds with respect to related initiatives.

3.6 Other 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 move to the cloud while enforcing quality control on 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 it 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, supporting migration from cloud to cloud when needed, and techniques for data mapping and synchronization among multiple clouds. To do so, MODAClouds requires software developers to adopt a Model-Driven Development approach. The monitoring platform developed in MODAClouds overcomes the limitation of the one offered by Cloud4SOA by gathering data of various kinds from components, containers and cloud resources distributed and replicated on multiple clouds. Although this approach has differently from SeaClouds, an impact on the code that needs to be deployed on the cloud, we are considering incorporating some of the MODAClouds functionalities related to the usage of data collectors which we are currently analyzing.

Also the PaaSage project (`http://www.paasage.eu/`) has Quality of Service assurance as one of its goal. PaaSage 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/`) also shares some of the SeaClouds' goals. More precisely, mOSAIC aims at developing an open-source platform that enables applications to negotiate Cloud services as requested by their users. Using a Cloud ontology, applications will be able to specify their service requirements and communicate them to the platform via a vendor agnostic API, so that the resulting applications can be deployed on different IaaS using a sort of mOSAIC virtual machine. The platform will implement a multi-agent brokering mechanism that will search for services matching the applications' requests, and possibly compose the requested service if no direct hint is found. Like in other projects, the final result is a platform supporting the user when developing her code. Our proposal avoids the creation of a middleware platform by allowing the user to select at runtime the platform(s) which better fix his application's preferences and requirements in order to deploy or migrate the corresponding module(s). mOSAIC also plans to support SLA negotiation (with monitoring to detect SLA violations) and application life-cycle, but requires developers to adopt the project's API.

The REMICS project (`http://www.remics.eu/`) focuses its work on developing advanced model-driven methodology and tools for the reuse and migration of legacy applications to interoperable Cloud services. Although the REMICS methodology focuses on legacy applications, our proposal shifts the migration paradigm to cloud services.

In contrast to our proposal with a specific goal of solving Cloud lock-in at the PaaS layer, 4CaaSt (`http://4caast.morfeo-project.org/`) is a project with very generic goals. The 4CaaSt project indeed defines an advanced PaaS platform which will support the optimised and elastic hosting of internet-scale applications. This platform will facilitate programming of rich applications and enable the creation of a business ecosystem where applications coming from different providers can be tailored to different users, mashed up and traded together.

The Cloud-TM project (`http://www.cloudtm.eu/`) defines a programming paradigm to facilitate the development and administration of Cloud applications. It focuses on self-optimising middleware platform that aims at simplifying the development and administration of applications deployed on large scale Cloud Computing infrastructures, while the intention of SeaClouds' approach is to administer complex applications but tackling the distribution and migration issues.

In the same scope of SeaClouds' approach, Cloud Federations have gained momentum in the last years. A Cloud Federation, in the sense of inter-cloud operation, is a platform where the user can select the infrastructure in which to deploy her software across a set of third-party solutions. As an example, Paraiso et al. [19] propose to define a PaaS level allowing selection on the level of IaaS. The Open Source Cloudware (`http://www.ow2.org/view/Cloud/`) community (OW2) has motivated many projects in the line of Cloud interoperability. One example is The Open Cloudware project (`http://www.opencloudware.org/`), which aims at building an open software engineering platform for the collaborative development of distributed applications to be deployed across multiple Cloud Infrastructures. In the last years, the term *Open Cloud* is becoming more popular, with projects like Cloudify, CloudStack, OpenStack, OpenShift, etc. These platforms separate the PaaS and IaaS layers, thus giving to the users more decision capability on how to deploy their applications, allowing the IaaS selection and defining a set of common services for the communication and data storage.

In the scope of commercial solutions, we can find some new platforms that are working on providing flexibility to users allowing the IaaS selection, and in some cases the migration over some specific PaaSes. On the one hand, one example is AppFog, a kind of Cloud Federation, whose system enables the user to select the infrastructure on which her software is going to be deployed among several commercial solutions. Although

when the software is deployed, it can be migrated from one IaaS to another with a minimal user interaction. However, it is centred on IaaS portability and does not address the lock-in at level of PaaS. On the other hand, at the level of PaaS, Cloud Foundry Core project is growing fast. The Cloud Foundry Core defines a baseline of common capabilities to promote Cloud portability across different instances of Cloud Foundry. This introduces a new level of lock-in, where one can migrate over platforms but it is required to implement a set of capabilities defined by the Cloud Foundry Core. The SeaClouds project goes one step further, and expects to allow the migration over all platforms, so as to strongly mitigate the vendor lock-in problem.

4 The SeaClouds Approach

This section presents the overall strategy of SeaClouds, describes the reference architecture, and presents its novel aspects compared to previous initiatives and efforts.

4.1 Overall strategy

Figure 4 shows the cloud architecture situation currently before SeaClouds (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 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.

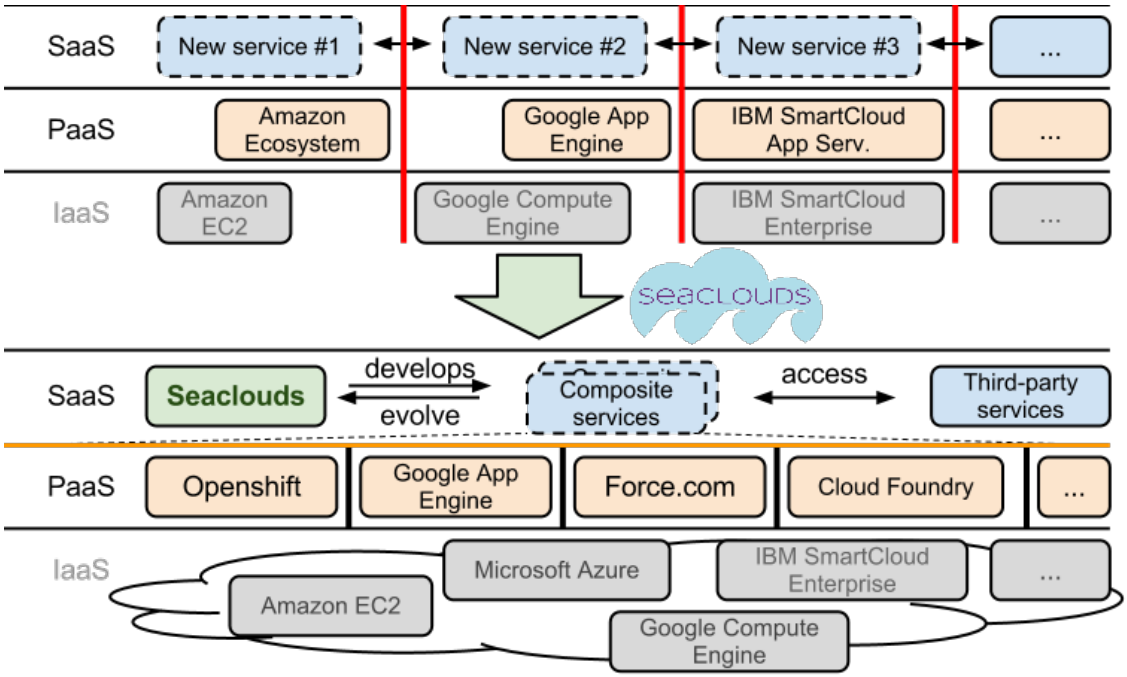

Figure 4: Cloud architecture before and after SeaClouds.

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 includes a unified management service, where services can be deployed, replicated, and administered by means of standard harmonized APIs such as CAMP specification and Cloud4SOA.

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 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 take 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 with SeaClouds dashboard.

4.2 SeaClouds architecture

We present the SeaClouds reference architecture, and discuss its novel aspects compared to previous initiatives and efforts.

4.2.1 Description

Before describing the core components of the architecture of the SeaClouds platform (Figure 5), it is worth observing that the platform features a graphical user interface (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, and the maximum acceptable cost for the entire deployment. Deployment Managers instead exploit the GUI through a unified dashboard that allows them to supervise the deployment and the monitoring of the application.

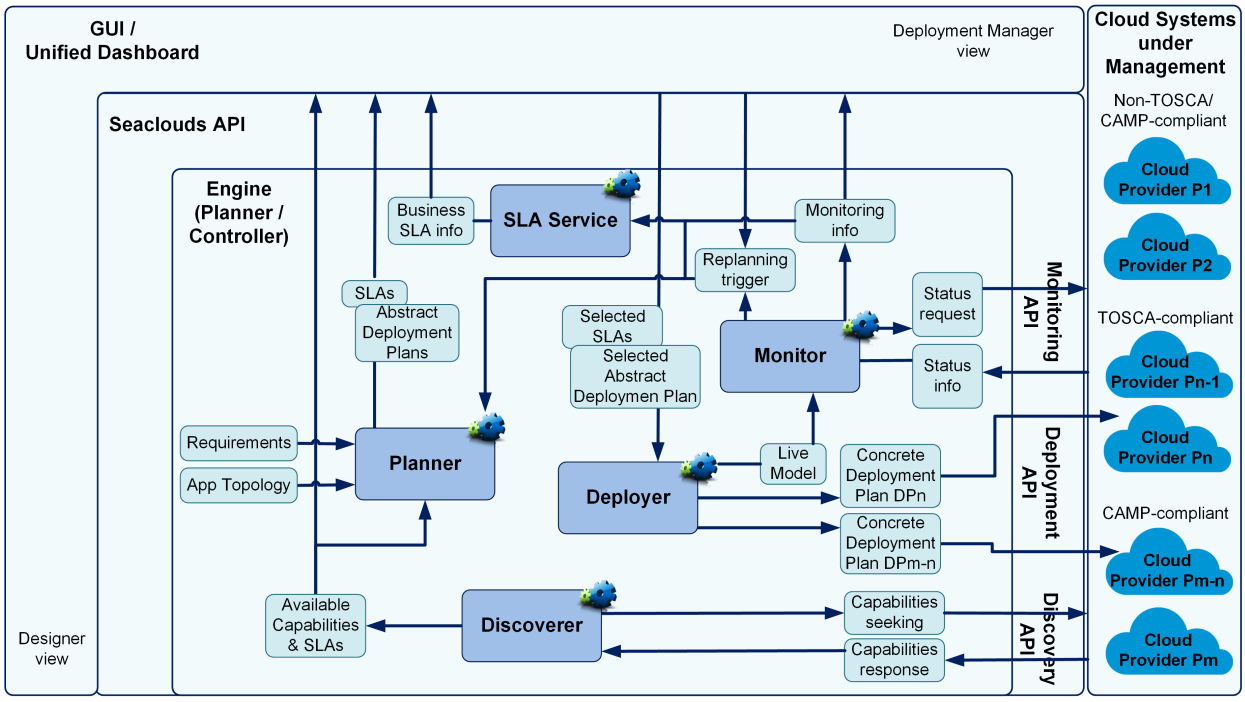

Figure 5: Architecture of the SeaClouds Platform.

Discoverer. 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.

Planner. The Planner component receives from application designers 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 a set of abstract deployment plans, each describing a feasible distribution of the application modules over the available clouds. One of such abstract plans is selected either by the Deployment Manager (through the SeaClouds GUI) or automatically, depending on the platform configuration. The selected abstract plan is hence passed to the Deployer, to the Monitor, and back to the Planner itself.

Deployer. The Deployer component inputs an abstract deployment plan, together with the SLAs of the selected services, and it 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). 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 that 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.

Monitor. 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.

SLA Service. 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.

Before concluding this presentation, let us briefly mention some other important aspects of the architecture: the Discoverer component also makes use of OASIS CAMP, to access the capabilities featured by (CAMP compliant) cloud providers. The Discoverer, the Deployer and the Monitor rely on CAMP-compliant adapters to interact with non CAMP-compliant providers. When new cloud capabilities are found after deployment, the Discoverer informs the Monitor as this may suggest the generation of replanning triggers. Deployment plans generated because of replanning triggers are actually reconfiguration plans that include both undeployment and deployment operations.

Coming back to the example of Section 2, Figure 6 shows a possible multi-cloud deployment of the different modules of the online retailing system as performed by SeaClouds. It applies the main concepts (modelling, planning and controlling) of the SeaClouds platform to achieve a seamless adaptive multi-cloud management.

4.2.2 Discussion

There are some novel aspects for the proposed SeaClouds platform 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

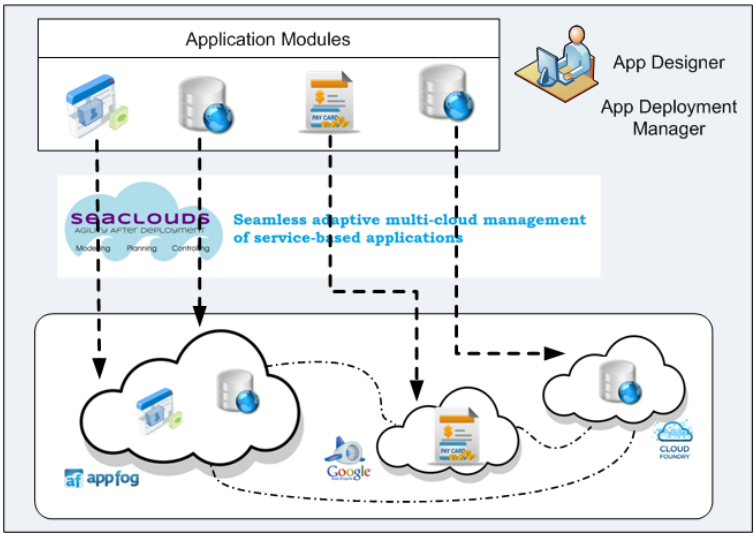

Figure 6: Example of SeaClouds multi-cloud deployment for the online retailing system.

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 component planner is only in charge of determining a distribution of application modules onto multiple available clouds in the form of abstract plans, while the component deployer is in charge of instantiating a concrete plan so as to actually deploy the application modules. 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 [17], 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 providers 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 standardisation 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.

5 Conclusions

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 cloud platforms. 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. The proposed architecture can well support this process, and also the exploitation of the best available offering for each application component at any time. 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).

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.

Acknowledgments

This work was partly supported by the EU-FP7-ICT-610531 SeaClouds project.

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, April 2010.

[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] 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 v1.0.0," http://dmtf.org/sites/default/files/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 and A. Chaddha, "Cloud Interoperability and Standardization," *SETLabs Briefings - Infosys*, vol. 7, no. 7, pp. 19–26, 2012.

[9] A. Brogi, J. Carrasco, J. Cubo, F. D'Andria, A. Ibrahim, E. Pimentel, and J. Soldani, "SeaClouds: Seamless adaptive multi-cloud management of service-based applications," in *XVII Ibero-American Conference on Software Engineering (CibSE '14)*, April 2014.

[10] A. Brogi, A. Ibrahim, J. Soldani, J. Carrasco, J. Cubo, E. Pimentel, and F. D'Andria, "SeaClouds: a European project on seamless management of multi-cloud applications," *ACM SIGSOFT Software Engineering Notes*, vol. 39, no. 1, pp. 1–4, January 2014.

[11] A. Brogi, J. Carrasco, J. Cubo, F. D'Andria, A. Ibrahim, E. Pimentel, and J. Soldani, "EU Project SeaClouds: Adaptive Management of Service-Based Applications Across Multiple Clouds," in *4th International Conference on Cloud Computing and Services Science (CLOSER '14)*, April 2014, pp. 758–763.

[12] 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, pp. 81–95.

[13] C. Canal, P. Poizat, and G. Salaun, "Model-Based Adaptation of Behavioural Mismatching Components," *IEEE Transactions on Software Engineering*, vol. 34, no. 4, pp. 546–563, 2008.

[14] 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, pp. 627–630.

[15] 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, pp. 731–740.

[16] 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, pp. 388–404.

[17] A. Brogi, J. Soldani, and P. Wang, "TOSCA in a Nutshell: Promises and perspectives," in *Service-Oriented and Cloud Computing*, ser. Lecture Notes in Computer Science, M. Villari, W. Zimmermann, and K.-K. Lau, Eds. Springer Berlin Heidelberg, 2014, vol. 8745, pp. 171–186. [Online]. Available: http://dx.doi.org/10.1007/978-3-662-44879-3_13

[18] OASIS, "Cloud Application Management for Platforms (CAMP) Technical Committee Charter," https://www.oasis-open.org/committees/camp/charter.php, 2013.

[19] F. Paraiso, N. Haderer, P. Merle, R. Rouvoy, and L. Seinturier, "A federated multi-cloud PaaS infrastructure," in *IEEE 5th International Conference on Cloud Computing*, 2012, pp. 392–399.