DOI 10.12694/scpe.v14i1.824
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 14, Number 1, pp. 17–32. http://www.scpe.org © 2013 SCPE

A USER-CENTRIC MULTI-PAAS APPLICATION MANAGEMENT SOLUTION FOR HYBRID MULTI-CLOUD SCENARIOS

DIMITRIS ZEGINIS[§, ¶], FRANCESCO D'ANDRIA[†], STEFANO BOCCONI[‡], JESUS GORRONOGOITIA CRUZ[†], ORIOL COLLELL MARTIN[†], PANAGIOTIS GOUVAS[‖], GIANNIS LEDAKIS[‖], AND KONSTANTINOS A. TARABANIS[§¶]

Abstract. Cloud Platform as a Service (PaaS) is a rapidly growing IT paradigm which enables software developers to deploy applications without the burden of software platform maintenance. Currently, the PaaS market is dominated by a few providers that promote incompatible standards. This introduces adoption barriers that prevent the interoperability between heterogeneous PaaS offerings, so software developers are not able to manage distributed applications spanning multiple public/private clouds. In this paper we present a multi-PaaS application management solution as a result of the Cloud4SOA European project that addresses these challenges. To clarify this approach a distributed deployment and cloud bursting scenarios are used.

Key words: Cloud computing, Platform as a Service, Multi-Cloud Management, interoperability, portability, user-centric, hybrid Cloud

AMS subject classifications. 68M11, 68P10, 68Q55, 68U35

1. Introduction.

Cloud computing is a rapidly growing IT paradigm which transforms the Internet into a global market of on-demand resources. Since 2007, when the term "Cloud computing" started becoming popular, it has managed to evolve into one of the most promising and influential IT trends, occupying the first places in Gartner's list of top strategic technologies for 2013 [12].

Cloud computing first emerged as Infrastructure-as-a-Service (IaaS) boosted by the significance acquired by Amazon Web Services. In parallel, Salesforce was offering a Software-as-a-Service (SaaS) platform (force.com) that included a customization layer and was based on the concepts of Application Service Provision (ASP) and Utility computing dating back to the 70's. Soon, driven by the existence of force.com and the eruption of this market with the entrance of Google's App Engine, the existence of a middle-ware layer in between IaaS and SaaS became clear: Platform-as-a-Service (PaaS).

PaaS [20] enables simplified consumption of Cloud infrastructure and supports Cloud applications. Main consumers of PaaS services are of two user groups: *i)* enterprises with their own internal software development activities and *ii)* Independent Software Vendors (ISV) interested in selling SaaS services on top of a hosted PaaS. The PaaS layer has undertaken a major shift, from mainly accounting on the technologies provided by only two large players as Microsoft (Azure) and Google (App Engine), to the wide variety of capabilities and programming languages found in PaaS offerings today. These new entrants offer a vast range of products, addressing specialized niches as well as broader markets. Often PaaS providers do not even offer their own execution environment, but they rely on IaaS providers for this, becoming effectively a proxy for the consumption of IaaS services. Another trend in the PaaS market is that established IaaS vendors are pushing up the stack to enable programming frameworks on top of their infrastructure, in order to add innovation-driven business models on top of their core IaaS segment. On the other hand, SaaS vendors offer platform tools to tailor their on-demand portfolio with the intention of creating customer loyalty and establishing a wider marketplace around their offering. As a result, there is a very diverse and heterogeneous collection of capabilities offered by PaaS providers.

The heterogeneity among Cloud PaaS providers refers to diversities in supported programming tools (languages, frameworks, runtime environments and databases), in the various types of underlying infrastructure, and even in capabilities available for each PaaS. In such a dynamic and evolving market, application and data portability and semantic interoperability between heterogeneous Cloud PaaS providers become a major issue of concern [23]. The current state of the Cloud PaaS market is therefore far from a global arena open to every player. Giant IT vendors monopolize the market and impose their own standards and formats [10, 11]. Hence, Cloud users, i.e. business and individual developers, are often locked within a specific platform environment

[†]ATOS Spain SA, Av Diagonal, 200, 08018 Barcelona - Spain (francesco.dandria@atos.net, jesus.gorronogoitia@atos.net, oriol.collell.external@atosresearch.eu).
[‡]Cyntelix Corporation BV Baron van Nagellstraat 136, 3771 LL Barneveld - The Netherlands (sbocconi@cyntelix.com).
[§]Centre for Research and Technology Hellas 6th Klm. Charilaou - Thermi Road, 570 01 Thessaloniki - Greece (zeginis@iti.gr, kat@iti.gr).
[¶]Information Systems Lab, University of Macedonia, Thessaloniki, Greece (zeginis@uom.gr, kat@uom.gr).
[‖]Singular Logic, Athens, Greece (pgouvas@gmail.com, g.ledakis@gmail.com).

because it is practically infeasible for them, due to high complexity and switching cost, to move their applications from one platform to another [16]. Reducing the semantic interoperability barriers between different Cloud vendors constitutes the most important step for realizing the global Cloud market vision [18].

Another set of important issues related to the acceptance and spread of Cloud computing is security and privacy. Many organizations with highly sensitive data (e.g. customers' data or health records), because of security or regulatory concerns, cannot move directly to public Clouds. Currently there exist three different deployment models for Cloud computing [20]:

(i) *Public Cloud.* The Cloud infrastructure is provisioned to the general public for open use. It may be owned, managed, and operated by a business, academic, or government organization. It exists on the premises of the Cloud provider.

(ii) *Private Cloud.* The Cloud infrastructure is provisioned locally for exclusive use by a single organization. It is owned, managed, and operated by that same organization and exist within the premises of the organization.

(iii) *Hybrid Cloud.* The Cloud infrastructure is a composition of the two Cloud infrastructures (private, public) that remain unique entities, but are bound together by standardized or proprietary technology enabling data and application portability.

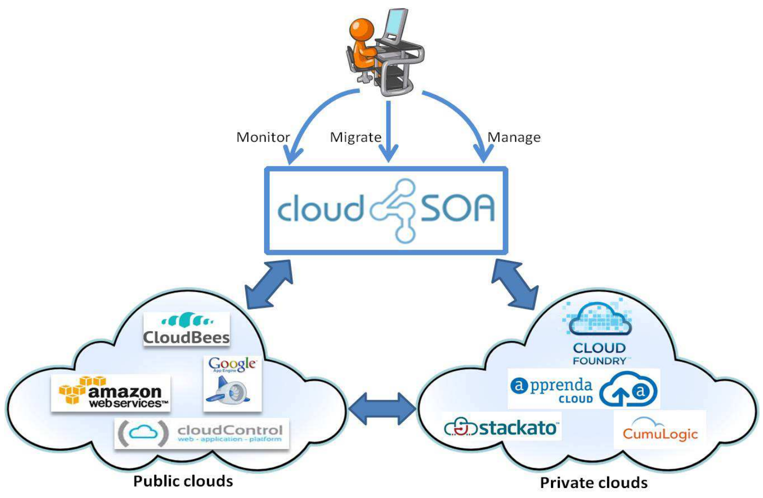


FIGURE 1.1. *The Cloud4SOA: multi-PaaS application management*

The main benefits of a private Cloud with respect to a public one is the possibility to have a more secure environment, since the Cloud is dedicated entirely to the use of the organization avoiding applications from other sources to interfere with them; moreover it offers a localized and completely customizable environment that can be adapted to the organization's needs. On the other hand, a private Cloud has some key drawbacks with respect to a public Cloud, including the ongoing investment of local infrastructure (perhaps a blocking point for SMEs) and scaling limitations, in addition to the lack of the pay-per-use model that helped foster the rise of the Cloud. Organizations that need the benefits of both private and public Clouds can exploit the hybrid model. The hybrid Cloud seeks to alleviate the inherent limitations of purely public and private approaches by combining them into a multi-Cloud that leverages their strengths [25].

In this paper we use a proof-of-concept hybrid multi-Cloud scenario to demonstrate the multi-PaaS application management solution developed by the Cloud4SOA project[1] (Fig. 1.1). Cloud4SOA introduces a broker-based architecture [17, 19] whose main goal is to address and tackle semantic interoperability challenges at the PaaS layer. The architecture is equipped with management and monitoring services providing the appropriate flexibility to handle public, private and hybrid deployment models.

This is an extended version of the work originally presented in [8]. The remainder of this paper is organized as follows. Section 2 presents the related work. Section 3 presents the Cloud4SOA architecture in a nutshell introducing the main layers and the core functionality. Section 4 further describes the layers of the architecture

[1]`http://www.Cloud4soa.eu/`

that enable the multi-PaaS application management and monitoring. Section 5 presents the hybrid usage scenario to clarify the Cloud4SOA functionality. Finally, Sect. 6 concludes this paper and presents future directions.

2. Related work. This section reviews the literature related to multi-PaaS management of applications. Specifically, we reviewed the existing Cloud computing architectures that have been proposed by EU-funded and national R&D projects. We mainly focus on architectures that deal with Cloud computing (semantic) interoperability, even if this is not their primary objective. Moreover, we reviewed brokers and multi-cloud managers which provide an abstract interface that masks the differences among heterogeneous Cloud providers enabling them to interoperate and collaborate.

2.1. R&D projects. The 4CaaSt project [21] envisions an advanced PaaS platform which supports the optimized and elastic hosting of internet-scale applications. This platform facilitates 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. 4CaaSt introduces a broker-based architecture which enables developers to control their applications' lifecycle.

CumuloNimbo [14] aims to provide a scalable PaaS Service which enables secure and un-partitioned data transactions resulting in consistent applications and at the same time ensuring the independent and optimized use of resources at a minimum cost. The CumuloNimbo subsystems is self-healing, automatically repairing themselves in the case of technical failures in order to avoid problems during the service provisioning.

The CONTRAIL project [13] enables enterprises to be both a Cloud provider when its infrastructure is not used at its maximum and a Cloud customer in periods of peak activity. This way, cooperation and resource sharing over Cloud federations is supported by the means of standardized interfaces. To this end, CONTRAIL provides a system in which resources that belong to different operators are integrated into a single homogeneous federated Cloud that users can access seamlessly.

The mOSAIC project [9] aims at the development of a platform that serves as an environment for competition between Cloud providers. mOSAIC focuses both on IaaS and PaaS layers by allowing applications to specify their service requirements (requested by their users) in terms of a Cloud ontology, while the platform, using a brokering mechanism, will perform a search of services in order to find the best matches with application's requirements. The main outcome of the mOSAIC project is a common API for communicating multi-Cloud resources. Unlike Cloud4SOA, the mOSAIC approach is based on the personalization of the IaaS to meet user requirements at the PaaS layer.

The CloudTM project [6] defines a programming paradigm to facilitate the development and administration of Cloud applications. It develops a self-optimized distributed transactional memory middleware that supports interoperability with heterogeneous persistent stores utilizing abstraction techniques. It is composed by two main parts: the Data Platform and the Automatic Manager.

The RESERVOIR project [22] aims at the development of an innovative service-oriented infrastructure that allows the dynamic interoperability of Cloud providers for the reliable delivery of services. To this end, it separates the roles of service provider and infrastructure provider. Service providers interact with the end-users, understand and address their needs. They do not own the computational resources; instead, they lease them from infrastructure providers which interoperate with each other creating a seamlessly infinitive pool of IT resources.

The SLA@SOI [24] developed a service-oriented infrastructure that allows the exchange of IT services with flexibility and well defined conditions and costs. This infrastructure leads towards the evolution of a service-oriented economy with formally specified SLAs and automated transactions between services.

The Cloud@Home is a national project that aims to resolve the problem of incompatibilities among resources composing an interoperable Cloud environment [7]. The Cloud@Home server-side is divided into two main subsystems: the management and the resource. On the one hand, the management subsystem is responsible for enrolling and managing the distributed resources and services providing a unique point of access. On the other hand, the resource subsystem implements the lower level functionalities (e.g. execution, storage).

Table 2.1 summarizes the R&D projects that deal with Cloud computing interoperability. For each project we investigated the targeted Cloud layer (i.e. IaaS, PaaS, SaaS), the usage of semantics as a main feature of the proposed architectures, the resolution of interoperability between diverse Cloud offerings, the portability of applications between Cloud offerings and finally the user-centricity (i.e. the focus on the needs of the end-users).

Based on the results of table 2.1, we see that only mOSAIC and Cloud4SOA follow a user-centric approach

Table 2.1
Overview of the R&D projects

Project	Cloud Layer	Semantics	Interoperability	Portability	User-centric
4CaaSt	PaaS	✗	✗	✓	✓
CumuloNimbo	PaaS	✗	✗	✓	✓
CONTRAIL	IaaS/PaaS	✗	✓	✓	✗
mOSAIC	IaaS/PaaS	✓	✓	✓	✓
Cloud-TM	IaaS	✗	✗	✓	✓
RESERVOIR	IaaS	✗	✓	✓	✓
SLA@SOI	IaaS	✓	✗	✓	✓
Cloud@Home	IaaS	✗	✓	✓	✓
Cloud4SOA	PaaS	✓	✓	✓	✓

to achieve semantic interoperability and portability of applications across diverse heterogeneous PaaS offerings. Cloud4SOA is focused only on the PaaS layer, while mOSAIC focuses both at IaaS and PaaS.

2.2. Brokers and multi-Cloud managers. LibCloud[2] supported by the Apache Software Foundation, is a library that can abstract and handle IaaS resources from diverse Cloud providers. Specifically, it provides a unified interface to the computing resources and can manage Cloud servers and storage from different Cloud providers. It also offers security and pricing functionalities.

DeltaCloud[3] abstracts the differences between diverse Clouds offering a single entry point for all the Cloud offering. Specifically, DeltaCloud enables the management of IaaS resources (i.e. compute and storage). The compute resources include the management of the instances, images, firewalls, IP addresses etc. The storage resources include the storage volumes that can be attached to a running instance and blob storage.

RightScale[4] has released the Cloud Management Platform, a Web-based platform which facilitates the deployment and the management of applications spanning multiple Cloud infrastructures - private, public, or hybrid. It delivers complete automation, support for complex deployments, while ensures flexibility, control, and portability.

Enomaly[5] has introduced the Enomaly Elastic Computing Platform (ECP) which empowers service providers (hosting providers, and managed service providers) with a complete Cloud-in-a-box platform that enables them to offer revenue-generating Cloud hosting (infrastructure-on-demand or IaaS) services to their customers.

OpenStack[6] is a free, open-source platform that service providers can use to offer infrastructure services similar to Amazon Web Services' EC2 and S3. It has two main parts: (i) Nova, originally developed by NASA for its computer processing services, and (ii) Swift, the storage service component developed by Rackspace. The OpenStack API can handle compute and storage resources.

OpenNebula[7] is an open-source Cloud computing toolkit for managing heterogeneous distributed data center infrastructures. OpenNebula orchestrates storage, network, virtualization, monitoring, and security technologies to deploy multitier services as virtual machines on distributed infrastructures, combining both datacenter resources and remote Cloud resources, according to allocation policies.

Table 2.2 summarizes the Cloud Brokers which provide an abstract interface that masks the differences among heterogeneous Cloud offerings. For each broker we investigated whether they deal with IaaS resources management (i.e. compute, network, storage), whether they deal with PaaS management (i.e. management of applications) and whether they deal with Cloud monitoring (i.e. monitor resources, applications etc.).

In order to implement a multi-PaaS application management solution the two main features required are the PaaS management (to manage applications at different PaaS) and the Cloud monitoring (to monitor the applications across PaaS). Based on the results of table 2.2 only Cloud4SOA satisfies these two requirements. Moreover Cloud4SOA offers a matchmaking service that allows searching among the existing PaaS offerings those that best match the developer's needs.

[2]http://libCloud.apache.org/
[3]http://deltaCloud.apache.org/
[4]https://www.rightscale.com
[5]http://www.enomaly.com/
[6]http://www.openstack.org/
[7]http://opennebula.org/

TABLE 2.2
Overview of the brokers and multi-cloud managers

Cloud Broker	IaaS management	PaaS management	Cloud monitoring
LibCloud	✓	✗	✗
DeltaCloud	✓	✗	✗
RightScale	✓	✗	✓
Enomaly	✓	✗	✓
OpenStack	✓	✗	✗
OpenNebula	✓	✗	✓
Cloud4SOA	✗	✓	✓

3. Cloud4SOA architecture in a nutshell. The Cloud4SOA reference architecture facilitates Cloud-based application developers in searching for, deploying and governing their business applications on the PaaS offerings that best match their needs. Additionally, switching between PaaS providers is supported in an easy and guided way, without putting the application or the underlying data at risk. This functionality is enabled by semantically interconnecting heterogeneous PaaS offerings. As described in the introduction, heterogeneity refers to the effect of numerous PaaS providers entering the Cloud market with dissimilar architectures, services, models and programming paradigms. Therefore, in the scope of Cloud4SOA, interoperability is defined as the ability of applications and their data to seamlessly be deployed on and/or migrated between Cloud PaaS offerings that are using the same technological background but different data (information) models and Application Programming Interfaces (APIs).

The Cloud4SOA reference architecture has been designed to enable interoperability and cross-platform application management. The Cloud4SOA reference architecture consists of five layers (see Fig. 3.1):

(i) The **Front-end layer** supports the user-centric focus of Cloud4SOA and the easy access of both Cloud-based application developers and Cloud PaaS providers to the provided functionalities.

(ii) The **Semantic layer** is the backbone of the architecture and spans the entire architecture, resolving interoperability conflicts by providing a common basis for publishing and searching different PaaS offerings.

(iii) The **SOA layer** comprises of a toolbox that is accessible through the Service Front-end layer. It implements the core functionalities offered by the Cloud4SOA system (see below).

(iv) The **Governance layer** implements the business-centric focus of Cloud4SOA where PaaS providers and application developers can establish business relationships through SLA agreements and can manage and monitor Cloud-based applications.

(v) The **Repository layer and harmonized API** acts as an intermediary between the platform and the various PaaS offerings by providing a harmonized API which enables the management of applications across different PaaS offerings.

Cloud4SOA's reference architecture provides four main functionalities briefly described in the following:

1. **Semantic Matchmaking** allows to searching among the existing PaaS offerings those that best match the developer's needs. Matchmaking heavily capitalizes on semantic technologies to: *i)* align the user requirements and the PaaS offerings even if they are expressed in different terms and *ii)* resolve the semantic conflicts between diverse PaaS offerings and allow matching of concepts between different PaaS providers that may use different naming or even different measurement units.

2. **Multi-platform application Management** supports the efficient deployment and governance of applications in a PaaS-independent way. The developers can manage the life-cycle of their applications is a homogenized way independently of the specific PaaS offering the application is deployed.

3. **Application Migration** offers the functionality to migrate already deployed applications from one PaaS offering to another. Moving an application between PaaS offerings consists of two main steps: i) moving the application data and ii) moving and redeploying the application itself to the new PaaS offering.

4. **Monitoring** offers a unified platform-independent mechanism, to monitor the health and performance of business-critical applications hosted on multiple Clouds environments in order to ensure that their performance consistently meets expectations defined by the SLA. In order to consider the heterogeneity of different PaaS offering Cloud4SOA provides a monitoring functionality based on unified platform independent metrics.

4. Enabling user-centric multi-PaaS application management and monitoring. This section justifies and presents the Cloud4SOA layers that enable the multi-PaaS application management and monitoring.

A common pitfall in building interfaces is to focus more on the technology rather than on how the user will

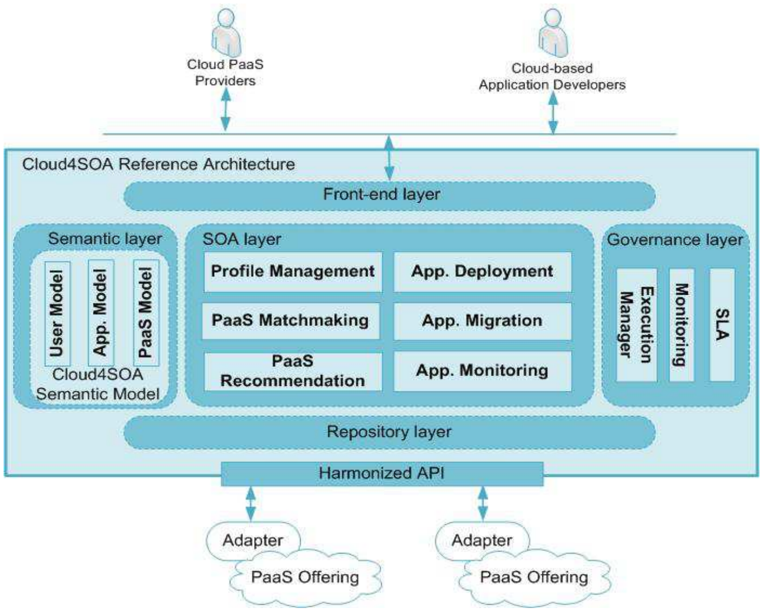


Figure 3.1. *The Cloud4SOA Reference Architecture*

make use of this technology. In such cases interfaces mirror what the technology can offer, without providing the user with a usable abstraction above this technology. Complexity and too many details are a common result of this mindset. The design of the Cloud4SOA interface (see Sect. 4.1) considers the user's point of view.

As discussed in the introduction, even when developers use technologies that are standardized (e.g. J2EE), they face a severe lock-in problem while trying to deploy/migrate their applications to a different PaaS provider since the APIs used by each provider differ. Thus, in order to enable multi-PaaS application management and monitoring there is a need for a harmonized API (see Sect. 4.2) which covers functionalities offered by the majority of PaaS providers.

Finally, in a multi-Cloud scenario it is vital to constantly monitor business-critical applications hosted on various Cloud environments, to ensure that their performance consistently meets expectations and that Cloud resources are being effectively used, independent of the technology used to implement such applications. Section 4.3 presents the approach proposed by Cloud4SOA to enable the multi-PaaS application management and monitoring.

4.1. Front-end layer - a user-centric approach. The Front-end layer is not limited to providing a pleasant look for the interface, but also to support the users to understand the data and services presented by the interface and efficiently interact with them. Supporting the user in these two key points is what makes an interface "intelligent". Cloud4SOA tackles these challenges by: *i)* providing a metaphor for the interface the user can understand and is familiar with, *ii)* providing the user with contextual information about the data visualized by the interface and *iii)* providing a dynamic interface easily adapted to user needs.

4.1.1. Dashboard Metaphor. In the scope of user interface design the most problematic issue is modeling the interaction [15]. Designing the user-system interaction is a complex process because of the two facets of human-computer interaction: coping with technical constraints as well as human factors [3]. In general, a model of an interaction can be seen as a way to understand and improve the usability of the interface [2, 5].

In our work we adopt the following definition for interaction model [4]: *"An interaction model is a set of principles, rules and properties that guide the design of an interface. It describes how to combine interaction techniques in a meaningful and consistent way and defines the "look and feel" of the interaction from the user's perspective. Properties of the interaction model can be used to evaluate specific interaction designs."*

Since Cloud4SOA interface needs to support user interaction and asynchronous display of events such as monitoring, the dashboard metaphor was chosen. The dashboard metaphor is something users are familiar with since it is vastly adopted (e.g. car dashboards) and offers the following advantages in the Cloud4SOA case:

(i) Loosely coupled widgets represent single instruments on the dashboard, this enables the addition/removal of functionality by adding/removing widgets.

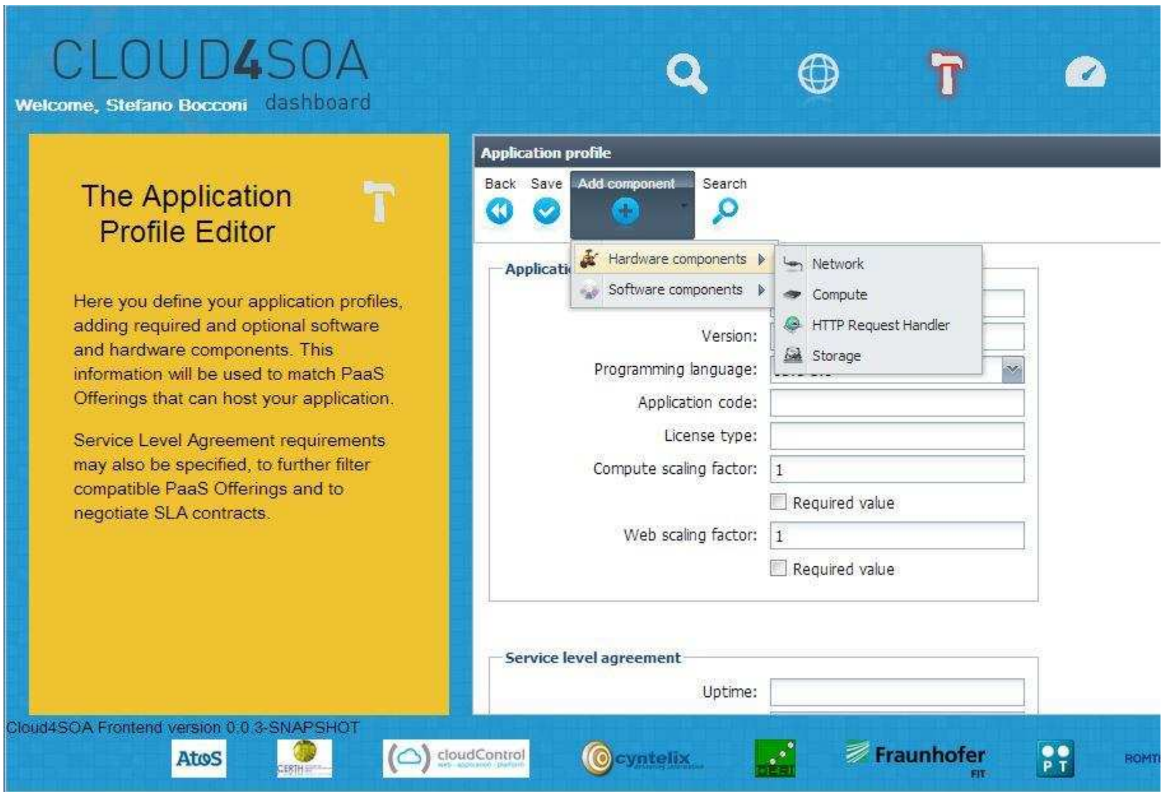
(ii) Widgets can update their display asynchronously and independently of one another, as instruments on a dashboard.

(iii) Widget can be minimized or moved, allowing the user to specify what information should be always on the dashboard.

(iv) Depending on the device used to access the dashboard, some widgets can be hidden e.g. in case of a mobile phone screen, without the need to scroll.

Another capability of the dashboard is the presentation of **contextual information** based on the semantics used to model developers' applications and Cloud offerings. An immediate use for the User Interface is to display the semantic description of the item. This already provides some context to the user when selecting an item. Semantic modelling can also be used in disambiguating terms. The Semantic layer provides mapping between different terms that can be used by the User Interface to suggest to the user that the term he is using has a particular relation to the term the PaaS provider uses, e.g. that a *dyno* is equivalent to n CPUs.

The testing phase of the dashboard metaphor showed that the interface could result confusing to users if there were too many widgets on the same page. Therefore we provided a simplified and more guided dashboard by grouping functionality. This consisted in defining several "places", or sections of the interface based on particular actions the user wants to perform, such as search for a suitable PaaS offering. Inside each place the dashboard metaphor was still valid, but involving less widgets.



The Application Profile Editor

Here you define your application profiles, adding required and optional software and hardware components. This information will be used to match PaaS Offerings that can host your application.

Service Level Agreement requirements may also be specified, to further filter compatible PaaS Offerings and to negotiate SLA contracts.

Compute scaling factor: 1

Web scaling factor: 1

Cloud4SOA Frontend version 0.0.3-SNAPSHOT

FIGURE 4.1. *The Application Profile editor, one of the places of the Cloud4SOA User Interface.*

A screenshot of the design can be seen in Fig. 4.1, where the Application Profile editor is shown. Here the user can enter the profile describing the application he or she wants to deploy. The layout is organized as follows: *i)* A header contains a navigation tool bar, *ii)* a central body shows the widgets associated to the place and some instructions of the actions that the user can perform in the place and *iii)* a footer displays static information and status communications.

4.1.2. Dynamic Interface Creation. It can be very time-consuming to redesign an interface layout when fields are added, modified or removed from the data schemas to be displayed. Generating dynamically the interface allows accommodating (semi) automatically the layout when data structures change. A dynamical interface can be designed considering two aspects of the problem: *i)* What items (such as Computing resources) and what properties of each items (such as number of CPUs) should be displayed and *ii)* How to display them, i.e. whether to use a drop-down list, sliders or radio buttons.

Both problems can be tackled by exploiting the semantic modeling. At runtime the interface queries for the objects that are modeled and for their properties. Subsequently rules are defined so that different types of visualization are chosen according to the data that needs to be displayed. An example is to use a slider when values are numerical varying over a range.

This has been implemented in Cloud4SOA by defining a domain specific language[8]. This language allows to define how data is represented in the visual form, including field validation and decoration (labels, tooltips and so on). An example for the field *Programming Language* in the Application Profile editor is the following:

```
field(FieldMetadata.create(PROGRAMMING_LANGUAGE)
  .label(''Programming language'')
  .editType(COMBOBOX)
  .tooltip(''Main programming language for the application'')
  .relatedEntityType(COMBO_PROGRAMMING_LANGUAGE)
);
```

The definition of the form's attributes can be retrieved from the semantic model, which leads to a fully model driven semantic repository editor. An example of this can be seen in the *Add component* item in menu of the Application Profile editor (top-center of Fig. 4.1). The *Network*, *Compute*, *HTTP Request Handler* and *Storage* menu items are subclasses of *Hardware Component* in the Semantic model.

Another typical dynamic interface for semantically modelled data is **faceted browsing**. Faceted browsing allows users to filter dynamically the items in a repository which satisfy the conditions posed on the chosen facets. Faceted browsing shows the information in a continuously updated way while the user selects options/criteria. The user is provided with an immediate response while he refines his query results. The advantages are twofold:

(i) Users can start with specifying few parameters, and decide to add more or revise some of the already chosen ones based on the result list.

(ii) Users do not risk wasting time in specifying all parameters of queries yielding an empty result set, since they receive immediate feedback on what parameter caused the query to return no results.

Cloud4SOA dashboard provides a faceted search widget to browse, filter and see details on existing PaaS offerings, based on faceted criteria selected by the user (see Fig. 4.2). Facet types (columns) and their values correspond to PaaS Offering properties (i.e. ratings, software and hardware components, etc.), which are obtained by inspecting, dynamically, the repository of offerings.

The user can select concrete values (criteria) for the facet types displayed by the widget. The matching PaaS offering result set is continuously updated and incompatible facet values filtered off the facet columns in order to avoid further empty searches. Multiple values within the same facet type are logically connected with the OR operator, while facet types are connected with AND operator.

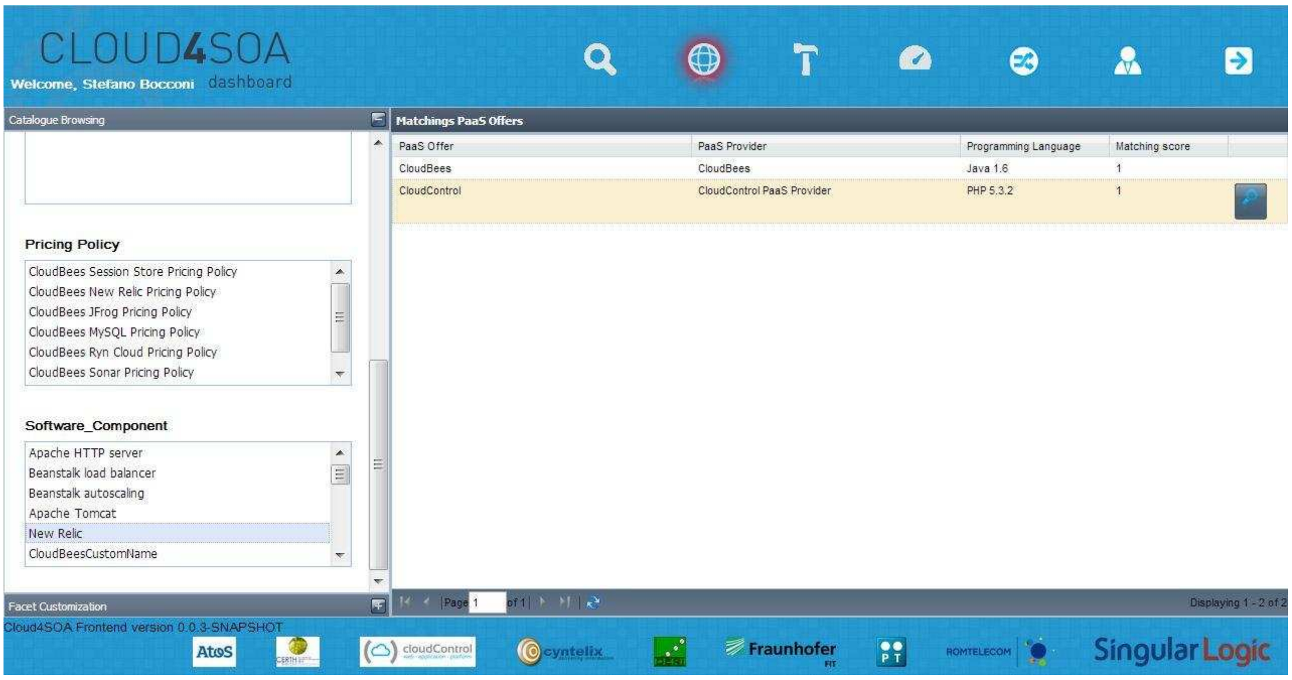
4.2. Repository layer and harmonized API. In order to alleviate the lock-in problem and achieve harmonized interaction with many PaaS offering, two architectural decisions have been taken:

(i) The formulation of one harmonized API which aims to abstract the specificities between various PaaS offerings;

(ii) The introduction of PaaS-specific Adapters to bridge the business logic between the harmonized API and the PaaS offerings.

There are major differences that exist in almost all aspects of PaaS offerings, thus the necessity of an API that is generic enough to readily include a wider variety of situations is evident. The API contains a number of operations that support the management of the Cloud-based applications independent of the specific API of the underlying PaaS offering. An overview of the methods that constitute the API and the supported PaaS offering is depicted at Fig. 4.3. The "Migrate" line indicates the PaaS offerings that are compatible for migration. The

[8]`http://www.martinfowler.com/bliki/DomainSpecificLanguage.html`



CLOUD4SOA dashboard

Welcome, Stefano Bocconi

Catalogue Browsing

Pricing Policy

- CloudBees Session Store Pricing Policy
- CloudBees New Relic Pricing Policy
- CloudBees JFrog Pricing Policy
- CloudBees MySQL Pricing Policy
- CloudBees Ryn Cloud Pricing Policy
- CloudBees Sonar Pricing Policy

Software_Component

- Apache HTTP server
- Beanstalk load balancer
- Beanstalk autoscaling
- Apache Tomcat
- New Relic
- CloudBeesCustomName

Facet Customization

Matchings PaaS Offers

PaaS Offer	PaaS Provider	Programming Language	Matching score
CloudBees	CloudBees	Java 1.6	1
CloudControl	CloudControl PaaS Provider	PHP 5.3.2	1

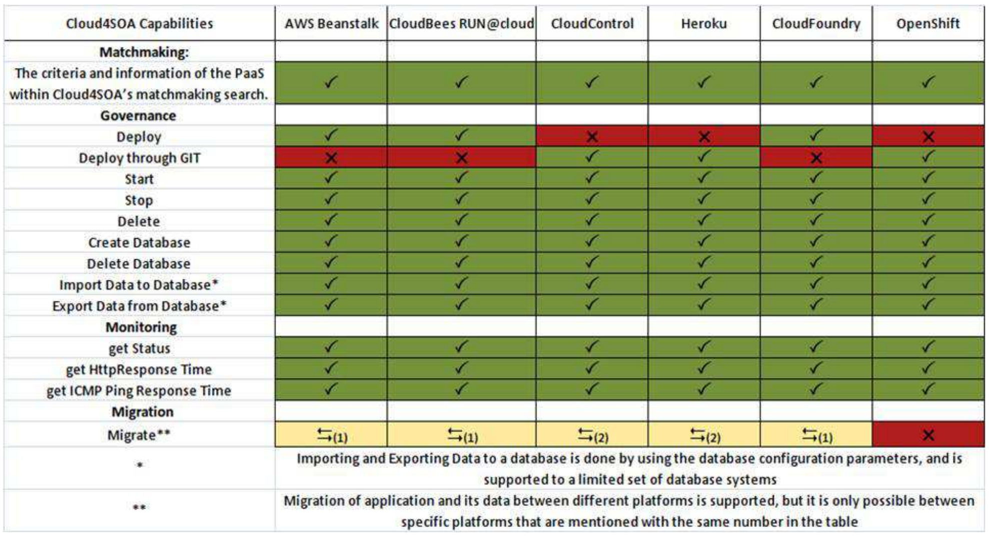
Page 1 of 1

Displaying 1 - 2 of 2

Cloud4SOA Frontend version 0.0.3-SNAPSHOT

FIGURE 4.2. *Faceted Browsing.*

migration is only possible between the platforms that are mentioned with the same number in this line (e.g. AWS Beanstalk, CloudBees and CloudFoundry are compatible; CloudControl and Heroku are compatible too).



Cloud4SOA Capabilities	AWS Beanstalk	CloudBees RUN@cloud	CloudControl	Heroku	CloudFoundry	OpenShift
Matchmaking:						
The criteria and information of the PaaS within Cloud4SOA's matchmaking search.	✓	✓	✓	✓	✓	✓
Governance						
Deploy	✓	✓	✕	✕	✓	✕
Deploy through GIT	✕	✕	✓	✓	✕	✓
Start	✓	✓	✓	✓	✓	✓
Stop	✓	✓	✓	✓	✓	✓
Delete	✓	✓	✓	✓	✓	✓
Create Database	✓	✓	✓	✓	✓	✓
Delete Database	✓	✓	✓	✓	✓	✓
Import Data to Database*	✓	✓	✓	✓	✓	✓
Export Data from Database*	✓	✓	✓	✓	✓	✓
Monitoring						
get Status	✓	✓	✓	✓	✓	✓
get HttpResponse Time	✓	✓	✓	✓	✓	✓
get ICMP Ping Response Time	✓	✓	✓	✓	✓	✓
Migration						
Migrate**	⇆(1)	⇆(1)	⇆(2)	⇆(2)	⇆(1)	✕
*	Importing and Exporting Data to a database is done by using the database configuration parameters, and is supported to a limited set of database systems					
**	Migration of application and its data between different platforms is supported, but it is only possible between specific platforms that are mentioned with the same number in the table					

FIGURE 4.3. *API methods and PaaS offering compliance*

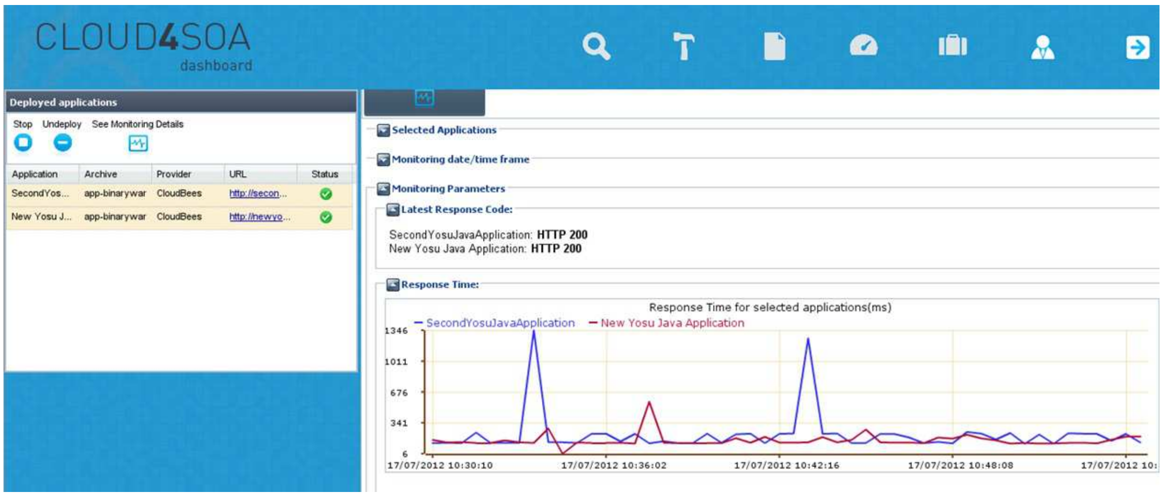
PaaS-specific Adapters are responsible for bridging the business logic between the introduced harmonized API and the APIs of the PaaS offerings by translating the functions of one to another and vice versa. The bridging of various PaaS platforms results on the creation of an overlay network which is responsible for the information fusion between the applications that are hosted at various distributes PaaS providers and the Cloud4SOA central system. Beyond the information fusion, the aforementioned Adapters provide the functionality of deployment, migration, governance and monitoring of an application.

In order to provide high-level functionalities (e.g. matchmaking) there is a need for persistently storing the semantic profiles of developers, PaaS providers and applications. For this reason, Cloud4SOA uses a repository in order to store both semantic and syntactic data. Moreover, there are additional requirements imposed by security and SLA issues.

Regarding the security issues, each developer has to interact seamlessly with every PaaS offering. In most of the cases the communication between the PaaS and the developer is accomplished through the usage of asymmetric cryptography (public-private key infrastructure). Therefore, a specific amount of information has to be persistently stored in the repository. Indicative information include the developer's public key and specific details per each project (version-control-system info).

Beyond security issues, the SLA management imposes many requirements since every deployed application has to be monitored. Therefore, specific data has to be stored at the repository about the monitoring-jobs, monitoring-statistics, SLA violations etc.

4.3. Governance layer - monitoring and SLA management. Cloud providers typically present very diverse architectures providing diverse resource-level metrics used to provide fine-grained Quality of Service (QoS) guarantees. As a consequence, Cloud consumers are not able to compare offerings they are looking for. In order to consider the heterogeneity of diverse Clouds architectures, Cloud4SOA provides a Service Level Agreement (SLA) management layer based on PaaS monitoring functionality that follows a unified and Cloud technology-agnostic approach. This approach defines specific metrics allowing Cloud consumers to monitor the health and performance of their business-critical applications hosted on multiple Clouds environments (see Fig. 4.4 where the Monitoring widget is shown).



Application	Archive	Provider	URL	Status
SecondYos...	app-binarywar	CloudBees	http://secon...	✓
New Yosu J...	app-binarywar	CloudBees	http://newyo...	✓

SecondYosuJavaApplication: **HTTP 200**
New Yosu Java Application: **HTTP 200**

FIGURE 4.4. *Application monitoring*

The monitoring functionality leverages on a range of standardized and unified metrics of different nature (resource/infrastructure level, container level, application level, etc.) that, based on the disparate underlying Cloud providers, allow the runtime monitoring of distributed applications so as to enforce the end-to-end QoS, regardless of where they are deployed across different PaaS.

In the scope of Cloud4SOA several metrics have been defined (Table 4.1) from the Cloud resource as well as the business application perspective, but not all of them have been enforced at runtime since some of them only provide useful information about the status of the application.

In order to provide quality of service guarantees, SLA plays a crucial role. However, in the scope of project, SLA management does not aim at representing a contractual relationship between the cloud consumers and the PaaS providers, nor establishing one. Instead, Cloud4SOA's SLA management describes the functionality of the application that is delivered, the functional and non-functional properties of the negotiated resources, and the duties of each party involved. As a result, its specifies the service level objective (SLO) that must be met in order to fulfill a guarantee as well as policies to pick the right violation level (i.e. warning, medium violation, migration etc.) and consequently suggest the appropriate recovery actions to take.

Table 4.1
Cloud PaaS metrics

Metric	**Description**	**App. status**	**Cloud performance**
CPU load	The amount of computational work that the application performs	✓	
Memory load	The amount of memory consumed by the application	✓	
HTTP response code	Includes custom status messages to understand the health of the application, but also the performance of the Cloud	✓	✓
Application and DB Response Time	Time that measures the efficiency and speed with which servers deliver requested web content to end users.	✓	✓
Application container Response Time	The elapsed time between the end of an inquiry or demand on a Cloud system and the beginning of a response.		✓
Cloud Response Time	The time the Cloud needs to process and forward to the application the incoming call.		✓

To deal with a SLA in an automatic way (i.e. to have mechanisms that automate the SLA setting up, monitoring and enforcement) the SLA itself has to be expressed through Cloud4SOA in a formalized way using an SLA specification language. Several languages for SLA specification have been reviewed (e.g. WSOL, WSLA, WSML etc.) but the WS-Agreement specification [1] was selected as the most appropriate:

(i) WS-Agreement offers a protocol to be followed for the negotiation process and a common understanding (i.e. language) of the objects the negotiation is about. Thus, it enables the automatic creation of SLAs.

(ii) The outcome of a successful negotiation with WS-Agreement is a SLA with binding character for both parties to deliver a reliable service to the end-user.

(iii) WS-Agreement is a novel but well-accepted standard for creating and enforcing SLAs in distributed environments as well as monitoring their resources properties.

Cloud4SOA provides a RESTful implementation of the WS-Agreement standard. On top of the implementation the governance layer offers three main functionalities (Fig. 4.5) that enable users negotiate and enforce SLA, as well as recover from a SLA violations:

(i) **SLA Negotiation**. Allows the automatic negotiations on behalf of PaaS providers, based on the semantic description of offerings and the QoS requirements specified by the Application Developer.

(ii) **SLA Enforcement**. Supervise that all the agreements reached in a SLA agreement are respected (i.e. measurements are within the thresholds established in SLA agreement for QoS metrics).

(iii) **SLA violation recovery**. Whenever the execution of the business application does not satisfy the SLA (i.e. breaches of the agreement occurs), the most appropriate recovery action (e.g. warning messages, stop or migration of the application) is suggested based on the policies defined by the software developer.

The SLA violations leveraged by the Cloud4SOA system in order to provide an enhanced rating mechanism which is responsible for rating negatively a PaaS offering whenever a violation occurs.

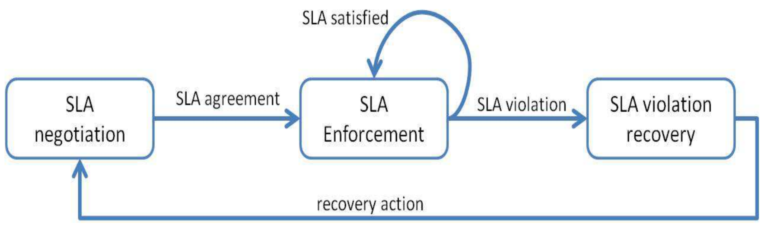


Figure 4.5. *SLA negotiation - enforcement - recovery process*

5. A hybrid multi-Cloud scenario. This section presents a hybrid multi-Cloud scenario that shows the way to manage multiple Cloud environments of varying natures (i.e. public, private, hybrid). The hybrid cloud model is used to manage both public and private clouds using a single platform (i.e. the Cloud4SOA platform). Specifically, we describe the way to integrate a private PaaS into Cloud4SOA and describe a set of scenarios that have been simulated to evaluate the usefulness and usability of the adopted approach. The scenarios contain a distributed deployment into more that one platforms and cloud bursting.

5.1. Integrating a private PaaS to Cloud4SOA. In order to enable hybrid deployment we have installed an instance of CloudFoundry[9], an open-source private PaaS solution, and integrated it with Cloud4SOA. With this integration, any Cloud user can easily manage a hybrid deployment approach using the Cloud4SOA platform. The process for integrating a PaaS provider (Fig. 5.1) is described in detail at the following paragraphs.

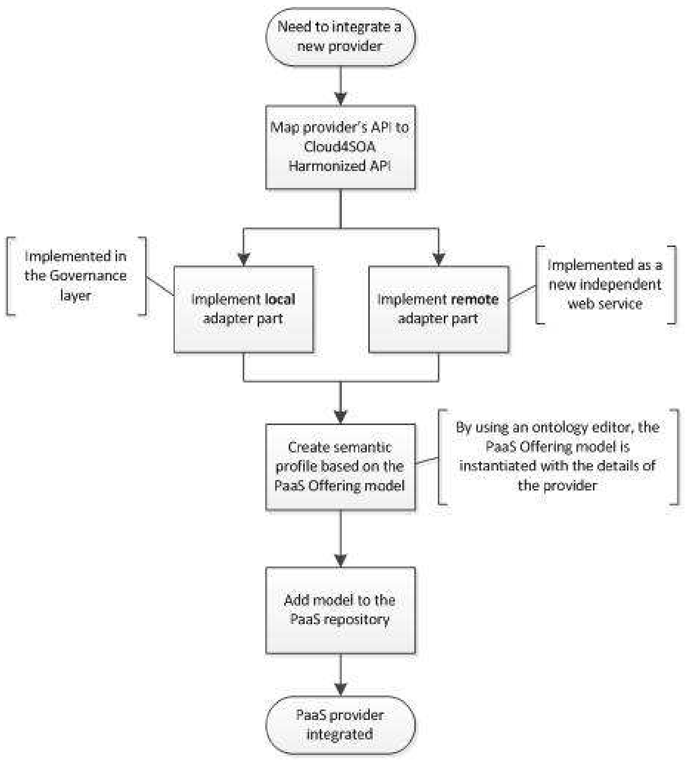

Need to integrate a new provider
Map provider's API to Cloud4SOA Harmonized API
Implemented in the Governance layer
Implement local adapter part
Implement remote adapter part
Implemented as a new independent web service
Create semantic profile based on the PaaS Offering model
By using an ontology editor, the PaaS Offering model is instantiated with the details of the provider
Add model to the PaaS repository
PaaS provider integrated

Figure 5.1. *Flowchart showing the process followed to integrate a new PaaS provider to Cloud4SOA*

The first step consists on analyzing the API of the provider and designing the necessary wiring to implement each of the methods in the harmonized API (see Fig. 4.3). In the CloudFoundry integration this step was one of the toughest ones most difficult since it is still young and it lacks a extensive set of documentation. This required to a directly examine direct examination of the code of the platform and to get involved in the developer's community in order to resolve all doubts.

Once the API is analyzed, the adapter needs to be implemented. The adapter is divided into two parts: a local one that runs along Cloud4SOA and a remote one that runs in the provider along applications deployed on it. The implementation of the local part is included in the Governance layer of the system while the remote part is an independent component that is automatically deployed to the provider each time an application is deployed. CloudFoundry offers a set of open-source libraries to interact with the API. We used these libraries to implement the adapter since they provide a reliable way to interact with CloudFoundry which contributes to the overall robustness of the adapter.

Once the adapter has been implemented and tested, the semantic profile of the provider has to be created and stored at the semantic repository. The semantic profile enables the description of functionalities offered (e.g. programming languages, software components, hardware components etc) as well as SLAs and Service quality terms.

[9]`http://www.Cloudfoundry.com/`

5.2. The hybrid scenarios. In order to evaluate the adequacy and usefulness of the proposed approach into a hybrid deployment we have identified two scenarios. These scenarios have been simulated using a complex modular framework provided by Portugal Telecom Inovacao (PTIN). The framework is formed by a set of pervasive services which can intelligently take proactive actions to select, rate and deliver suitable content (e.g. photos, videos, etc.) based on user's context information. The framework offers three basic services:

(i) The Context Enabler is a SOAP web service that subscribes and receives context information about a specific user.

(ii) The Group Manager Enabler is a rule engine responsible for identifying, creating and managing groups of consumers based on their context information.

(iii) The Content Selection Enabler is a rule engine that aims at rating and selecting content (e.g. photos, videos, etc.) for users based on their group(s).

Since the framework is based on several autonomous services, it is possible to exploit the distribution of them across several public/private platforms and test their behavior in each environment and, thus, it is adequate to simulate the defined hybrid deployment scenarios, which are described below.

Distributed deployment. In this scenario an application composed by two autonomous parts is distributed between a private Cloud and a public Cloud. On one hand, the part deployed privately may contain sensitive or confidential data that requires to be stored in the same country as the company and secured by using a set of custom policies. On the other hand, the part deployed in the public Cloud may require a higher level of reliability or performance. By using Cloud4SOA, a developer can manage the deployment and governance of both parts from the same place and compare their performance in a graphical way by using the monitoring tools provided by the system.

We have used two of the pervasive services from the PTIN framework to simulate the scenario. First, the Group Manager Enabler is deployed in a public Cloud, CloudBees in this case, since the rule engine can be highly demanding in terms of performance. Then, the Context Enabler is deployed in the private CloudFoundry instance as it can contain sensitive users' information. We assume that the third service (the Content Selection Enabler) is deployed in another PaaS and that the Context Broker runs locally on the PTIN infrastructure.

Bursting on demand. In this scenario all parts of an application are deployed in the same private Cloud and are governed by a SLA between the developer and the provider. After some time, Cloud4SOA detects a violation on an SLA term and triggers a notification about the issue. The developer, then, driven by this warning, decides to burst the critical part of the application to a public Cloud able to fulfill the SLA. By using Cloud4SOA, a developer can be notified about such SLA violations and decide the actions to take. Moreover, if he or she decides to burst a part of an application, he or she can perform the migration from the same Cloud4SOA system. Figure 5.2 shows the scenario in a graphical way.

In this scenario, the two services (the Group Manager Enabler and the Context Enabler) are deployed in the private CloudFoundry[10] instance and after a warning from Cloud4SOA stating that some performance-related SLA terms are being violated for the Group Manager Enabler, it bursts to a public Cloud.

5.3. Discussion and evaluation. An evaluation of the performance and usefulness of Cloud4SOA has been conducted based on the two hybrid scenarios deployed. The evaluation shows that Cloud4SOA can provide benefits to user willing to adopt a hybrid approach, since it allows deploying, governing and monitoring both parts of a hybrid Cloud from the same single place. Moreover, it provides useful information and notifications about the performance of the applications and the fulfillment of SLAs and allows bursting an application in case of SLA violation.

Moreover, we analyzed the overhead imposed by the Cloud4SOA system compared to the usage of the provider's APIs or tools directly when executing operations such as deploy, start, stop or undeploy. Table 5.1 shows the comparison of the time that Cloud4SOA takes to execute some basic deployment and governance operations. The values shown are the average of the results obtained from repeating each operation 10 times. From these results we see that there is an overhead in time imposed by the system, but, in general, it is acceptable taking into account the added-value provided.

Integrating CloudFoundry with Cloud4SOA has also been useful to learn more about the complexity of adding a new provider to the system. The ultimate goal is to enable an easy to follow process that can be performed by providers willing to join Cloud4SOA themselves. The process is quite easy however work is still required to enhance this experience and to reduce the amount of intervention required.

[10]http://www.Cloudfoundry.com/

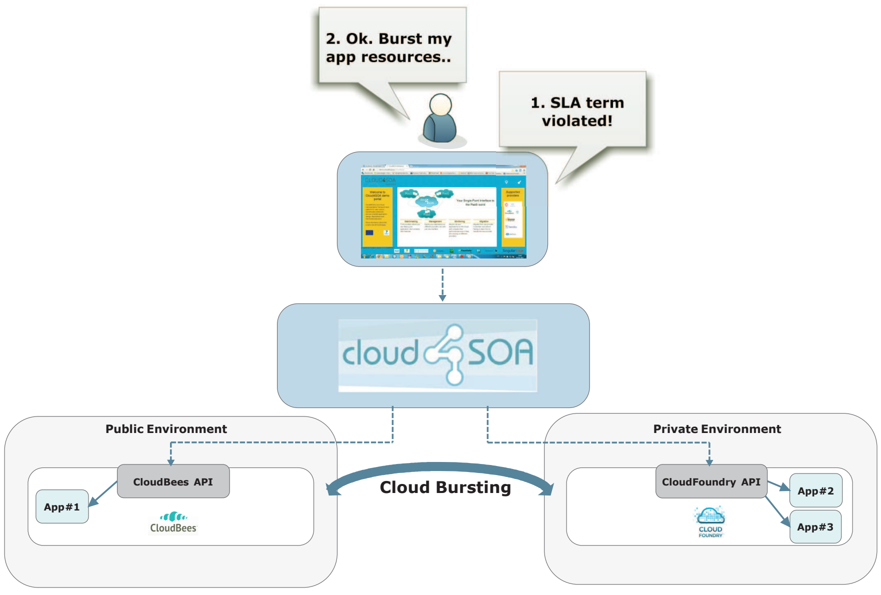


FIGURE 5.2. *Bursting on demand scenario representation*

TABLE 5.1
Comparison of Performance between Cloud4SOA and the PaaS API

Operation	Cloud4SOA	Provider API
Deploy (CloudFoundry)	42 sec.	28 sec.
Deploy (CloudBees)	155 sec.	130 sec.
Stop (CloudFoundry)	11 sec.	5 sec.
Undeploy(CloudFoundry)	15 sec.	6 sec.

Regarding the adapter, the experience has been more positive with respect to the remote adapter than with the local one. As discussed in Sect. 5.1, the remote adapter has been implemented as a brand new independent component and did not require any additional wiring code in the Governance layer. Moreover, Cloud4SOA provides a Generic Remote Adapter template that greatly helps in implementing the adapter.

With respect to the local adapter, the implementation requires a better understanding of the Cloud4SOA architecture and code, especially on the Governance layer. Moreover it supposes a threat for the integrity of the platform since a change in the code of this layer may affect the functioning of the whole platform. In order to overcome this, it should be possible to encapsulate the local part of adapters as independent projects in a similar way to the remote part.

6. Conclusions. Currently, the PaaS market is still quite young, chaotic and highly fragmented, dominated by a few providers which use and promote incompatible standards and formats. This introduces adoption barriers due to the lock-in issues that prevent the application and data portability and the semantic interoperability between heterogeneous Cloud PaaS offerings. Additionally, there is a need by software developers not only to migrate applications from one Cloud platform to another but also to manage and monitor distributed applications spanning multiple public and/or private Clouds.

In this work we present the Cloud4SOA solution for multi-Cloud application management, monitoring and migration; all of which are completely independent of a particular vendor. In this way developers are not just aided in the form of a tool, but are provided a great opportunity for Cloud software development in general, alleviating the lock-in that has long been attached to platform adoption.

The evaluation of Cloud4SOA, based on the two hybrid scenarios, showed that there is an overhead imposed by the platform but it is considered acceptable taking into account the added-value provided since it allows deploying, governing and monitoring of applications across diverse heterogeneous public and private Clouds.

Concluding the main contributions of Cloud4SOA can be summarized in the following:

(i) It adopts a user-centric approach for the UI design in order to offer friendly and intuitive interfaces tailored to the user's needs.

(ii) It solves the interoperability issues between heterogeneous Cloud PaaS environments by offering a harmonized API.

(iii) It offers a multi-PaaS application management and monitoring solution that enables the deployment of applications at public, private or hybrid multi-Cloud environments.

Acknowledgments. This work is partially funded by the European Commission in the context of the Cloud4SOA project, under the contract No. 257953 within the Seventh Framework Programme of the European Community. The authors would like to thank Esen Kunt and Stefano Travelli for their contribution to the design/implementation of the User Interface and David Goncalves Cunha from PTIN for providing the framework used for the hybrid scenarios.

REFERENCES

[1] A. Andrieux, , K. Czajkowski, A. Dan, K. Keahey, H. Ludwig, T. Nakata, J. Pruyne, J. Rofrano, S. Tuecke, and M. Xu, *Web services agreement specification (ws-agreement)*, Specification from the Open Grid Forum (OGF), (2007). available at: http://www.ogf.org/documents/GFD.107.pdf.

[2] Y. Anokwa, G. Borriello, T. Pering, and R. Want, *A User Interaction Model for NFC Enabled Applications*, in Proceedings of the Fifth IEEE International Conference on Pervasive Computing and Communications Workshops, Washington, DC, USA, 2007, IEEE Computer Society, pp. 357—-361.

[3] M. Beaudouin-Lafon, *An overview of human-computer interaction*, Biochimie, 75 (1993), pp. 321–329.

[4] M. Beaudouin-Lafon, *Instrumental interaction: an interaction model for designing post-wimp user interfaces*, in Proceedings of the SIGCHI conference on Human Factors in Computing Systems, CHI '00, New York, NY, USA, 2000, ACM, pp. 446–453.

[5] D. Benyon and D. Murray, *Applying user modeling to human-computer interaction design*, Artificial Intelligence Review, 7 (1993), pp. 199–225.

[6] E. Bernard, J. Cachopo, B. Ciciani, D. Didona, F. Giannone, M. Little, S. Peluso, F. Quaglia, L. Rodrigues, P. Romano, and V. A. Ziparo, *Cloudtm d2.1: Architecture draft*, (2011). available at: http://www.gsd.inesc-id.pt/ romanop/files/deliverables/D2_1.pdf.

[7] V. D. Cunsolo, S. Distefano, A. Puliafito, and M. Scarpa, *Cloud@home: Bridging the gap between volunteer and cloud computing*, Proceedings of the 5th International Conference on Intelligent Computing (ICIC 2009), (2009).

[8] F. D'Andria, S. Bocconi, J. G. Cruz, J. Ahtes, and D. Zeginis, *Cloud4soa: Multi-cloud application management across paas offerings*, Management of resources and services in Cloud and Sky computing (MICAS) 1st Workshop, organized in conjunction with SYNASC 2012, Timisoara, Romania, 2012.

[9] B. di Martino, D. Petcu, R. Cossu, P. Goncalves, T. Mahr, and M. Loichate, *Building a mosaic of clouds*, in Proceedings of the 16th European Conference on Parallel and Distributed Computing (Euro-Par 2010), Ischia-Naples, Italy, 2010.

[10] Economist, *Battle of the clouds*, (2009). available at http://www.economist.com/opinion/displaystory.cfm?story_id=14644393

[11] ———, *Clash of the clouds*, (2009). available at: http://www.economist.com/displaystory.cfm?story_id=14637206

[12] Gartnet, *Gartner identifies the top 10 strategic technology trends for 2013*, (2013). available at: http://www.gartner.com/newsroom/id/2209615.

[13] P. Harsh, Y. Jegou, R. Cascella, and C. Morin, *Contrail virtual execution platform challenges in being part of a cloud federation*, In Proceedings of the 4th European conference on Towards a service-based internet (ServiceWave'11), Berlin, Heidelberg, 2011, pp. 50 – 61.

[14] R. Jimenez-Peris, M. Patino-Martinez, K. Magoutis, A. Bilas, and I. Brondino, *Cumulonimbo: A highly-scalable transaction processing platform as a service*, ERCIM News 89, Special Issue on Big Data, (2012). available at: http://ercim-news.ercim.eu/en89/special/cumulonimbo-a-highly-scalable-transaction-processing-platform-as-a-service.

[15] G. Lee, C. M. Eastman, T. Taunk, and C.-H. Ho, *Usability principles and best practices for the user interface design of complex 3D architectural design and engineering tools*, International Journal Human-Computer Studies, 68 (2010), pp. 90–104.

[16] N. Loutas, ed., *D1.1 Requirements Analysis Report*, 2011. available at: http://www.cloud4soa.eu/sites/default/files/Cloud4SOA %20D1.1%20Requirements%20Analysis.pdf.

[17] N. Loutas, ed., *D1.3 Cloud4SOA Reference Architecture*, 2011. available at: http://www.cloud4soa.eu/sites/default/files/D1.3 %20Cloud4SOA%20Reference%20Architecture.pdf.

[18] N. Loutas, E. Kamateri, and K. Tarabanis, *A semantic interoperability framework for cloud platform as a service*, In Proceedings of 3rd IEEE International Conference on Cloud Computing Technology and Science, Athens, Greece, 2011, pp. 280 – 287.

[19] N. Loutas, V. Peristeras, T. Bouras, E. Kamateri, D. Zeginis, and K. A. Tarabanis:, *Towards a reference architecture for semantically interoperable clouds.*, CloudCom 2010, (2010).

[20] NIST, *A nist definition of cloud computing*, (2011). available at: http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf.

[21] E. Oliveros, S. Garcia, S. Strauch, T. Binz, A. Spriestersbach, K. Stanoevska, S. Carrie, and J. Niemoeller, *4caast value proposition*, white paper, (2011).

[22] B. Rochwerger, D. Breitgand, E. Levy, A. Galis, K. Nagin, I. Llorente, R. Montero, Y. Wolfsthal, E. Elmroth,

J. Caceres, M. Ben-Yehuda, W. Emmerich, and F. Galan, *The reservoir model and architecture for open federated cloud computing*, IBM Systems Journal, (2008).

[23] A. Sheth and A. RanabahuS, *Semantic modeling for cloud computing, part i and ii*, IEEE Internet Computing Magazine, 14 (2010), pp. 81 – 83.

[24] W. Theilman, R. Yahyapour, and J. Butler, *Multi-level sla management for service-oriented infrastructures*, Proceedings of the 1st European Conference on Towards a Service-Based Internet (ServiceWave '08), Madrid, Spain, 2008.

[25] H. Zhang, G. Jiang, K. Yoshihira, H. Chen, and A. Saxena, *Intelligent workload factoring for a hybrid cloud computing model*, in Proceedings of the 2009 Congress on Services - I, SERVICES '09, Washington, DC, USA, 2009, IEEE Computer Society, pp. 701–708.

Edited by: Marc Eduard Frîncu
Received: Mar 15, 2013
Accepted: Mar 29, 2013