Towards a Reference Architecture for Semantically Interoperable Clouds

Nikolaos Loutas[1,3,5], Vassilios Peristeras[1,2], Thanassis Bouras[4], Eleni Kamateri[3,5], Dimitrios Zeginis[3,5], Konstantinos Tarabanis[3,5]

[1]National University of Ireland, Galway, Digital Enterprise Research Institute, Galway, Ireland
fistname.lastname@deri.org
[2]Greek National Center for Public Administration and Decentralization, Greece
[3]Centre for Research and Technology Hellas, Thessaloniki, Greece
[4]UBITECH Research, 429 Messogion Ave., Ag. Paraskevi Square, 15343 Athens, Greece
bouras@ubitech.eu
[5]Information Systems Lab, University of Macedonia, Thessaloniki, Greece
{ekamater, zeginis, kat}@uom.gr

Abstract—**This paper focuses on the emerging problem of semantic interoperability between heterogeneous cooperating Cloud platforms. We try to pave the way towards a Reference Architecture for Semantically Interoperable Clouds (RASIC). To this end, three fundamental and complementary computing paradigms, namely Cloud computing, Service Oriented Architectures (SOA) and lightweight semantics are used as the main building blocks. The open, generic Reference Architecture for Semantically Interoperable Clouds introduces a scalable, reusable and transferable approach for facilitating the design, deployment and execution of resource intensive SOA services on top of semantically interlinked Clouds. In order to support the development of semantically interoperable Cloud systems based on RASIC, the model of a common Cloud API is also specified.**

Keywords-Cloud computing; SOA; semantic; interoperability; reference architecture; common Cloud API.

I. INTRODUCTION

The paradigm of Service Oriented Architecture (SOA) has provided the means to develop highly flexible, modular, reusable and easily extendable applications by allowing the users to develop and to deploy discrete pieces of functionality as services. The derived loose coupling of services is perceived to be an important advancement towards interoperable systems. Web Services became the prevailing SOA implementation paradigm.

Recently another paradigm directly affects SOA related deployments, that of Cloud computing. Cloud computing presents a scalable and affordable approach for users to utilize resources on demand. The main idea behind Cloud computing is that computing will increasingly be delivered as a service, over the Internet, from vast warehouses of shared machines. The concept of Cloud computing entails computing resources and IT services being offered as a service over the Internet. Cloud services generally fall in three categories: Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), and Software-as-a-Service (SaaS) [1].

SOAs can be deployed in a Cloud following a pay-as-you-go approach. The Cloud providers offer the physical hosting environment for SOA services, thus reducing the burden of owning, maintaining and upgrading the vast computational power that is needed for resource intensive applications. Cloud computing borrows and uses the idea of service orientation. In this sense the Cloud computing paradigm remains at its principles and foundations basically SOA compliant, while at the same time it creates a new, promising landscape for a next generation of SOA deployments. Cloud services are usually paid per use and the client can have as much of a service as they need at a certain point of time. Moreover, in Cloud computing all the burden of service management and governance falls on the Cloud provider.

Although the fundamental idea behind Cloud is not new but rather dates back some decades, the means and infrastructure currently available for realizing such a vision are really unique, including the maturity of SOA and Web related standards, as well as the drastically dropping prices and availability of hardware and network access. For these reasons, Cloud computing holds the first place in Gartner's 2010 list of the top strategic technologies [2].

However, there are still serious issues to be discussed and solved. In a recent issue of the Economist very interesting concerns with regards to the future of Cloud computing are found [3, 4]. The competition between the three IT giant companies, namely Google, Microsoft and Apple, on who will take the lead in the Cloud is discussed. This dominance becomes quite problematic if combined with another finding discussed in the articles: the familiar risk of technological lock-in, as the three major competing companies promote their own, mutually incompatible, Cloud standards and formats. The lock-in problem is also recognized by the European Network and Information Security Agency as a high risk that Cloud infrastructures entail [5].

Taking into account the above findings, in such a dynamic, evolving but at the same time monopolized market, semantic interoperability between Cloud providers becomes an important issue.

978-0-7695-4302-4/10 $26.00 © 2010 IEEE
DOI 10.1109/CloudCom.2010.38

Currently, there are no widely accepted semantic interoperability standards for the Cloud. Some providers do not even allow software created by their customers to be moved off their platform. But even if in theory someone can freely move their data and application between different Cloud platforms, in practice the high complexity and switching costs discourage users from doing so.

Companies developing applications should be able to choose between different Cloud platforms and should also be able to switch between providers whenever needed. Therefore, initiatives are needed to ensure that the IT industry does not remain dominated by monopolies of single companies and to support competition in the Cloud market.

Many groups are working on Cloud computing standards and are the main advocates of the need to harmonize the different Cloud APIs and to agree on common standards for Cloud Computing. We have identified the most active groups in the field, namely the CloudAudit[1], the Cloud Security Alliance (CSA)[2], the Distributed Management Task Force (DMTF)[3], the European Telecommunications Standards Institute Technical Committee (ETSI TC CLOUD)[4], the Open Grid Forum (OGF)[5], the Object Management Group (OMG)[6], the Open Cloud Consortium (OCC)[7], the OASIS[8], the Storage Networking Industry Association (SNIA)[9], the Open Group Cloud Work Group[10], the Cloud Computing Interoperability Forum (CCIF)[11], and the Cloud Standards Coordination Working Group[12]. The Cloud manifesto[13], an initiative supported by tenths of companies, including major software and infrastructure vendors such as IBM, SAP, Siemens and Telefonica, argues that Cloud computing should capitalize on open standards.

As we discuss in section II, [6] [7] argue that it is time to create a common Cloud API. Others claim that a broker [8], a site [9] or a semantic repository [10] could be used for semantically interlinking heterogeneous Cloud platforms.

This works proposes a Reference Architecture for Semantically Interoperable Clouds (RASIC) that focuses on resolving the semantic interoperability issues that exist in current Cloud infrastructures and introducing a user-centric approach for applications which are built upon and deployed using Cloud resources. RASIC will facilitate the smooth switching between Cloud providers and will allow the composition and integration of services hosted in different Clouds. To this end, three fundamental and complementary computing paradigms, namely Cloud computing, Service Oriented Architectures and lightweight semantics are combined.

The rest of this paper is structured as follows: Section II discusses the state of the art in the area of semantic interoperability in Cloud computing. Section III presents the Reference Architecture and section IV presents the proposed common Cloud API Model. Finally, section V concludes the paper and discusses our future research directions.

II. STATE OF THE ART

The state of the art in the area of semantic interoperability in Cloud computing is divided into two subsections. In section A we discuss efforts that study semantic interoperability in the Cloud and propose solutions. Several architectural approaches to achieve semantic interoperability between heterogeneous Clouds are presented. Section B focuses on existing Cloud APIs, which are reviewed in order to identify their common elements, hence harmonizing them.

A. Semantic Interoperability in Cloud Computing

The first efforts to scope, study and address semantic interoperability between Clouds that need to exchange information have already appeared.

The most common Cloud computing use cases are defined in [11]. Semantic interoperability conflicts arise at most of the cases, especially in cases of hybrid Clouds. Such a case is the collaboration of private and public services in order to provide more sophisticated and added-value solutions or the case of a Cloud broker responsible for coordination and service provisioning according to the needs of a specific user.

In practice, semantic interoperability conflicts are raised at the IaaS, at the PaaS or at the SaaS level. Foster [12] claims that in the near future we should except multiple standards for all three layers to emerge but only the useful ones will finally survive.

The EU-funded RESERVOIR[14] project develops a layered architecture for federated Cloud providers that cooperate seamlessly to optimize their services and maximize their benefits. The approach allows Small Medium Enterprises (SMEs) to enter the Cloud market. In the development model, infrastructure providers operate RESERVOIR sites and manage the physical resources on which service applications execute [9]. Service providers are the mediators between infrastructure providers and end-users that could be single clients or businesses. After understanding end-user needs, service providers use RESERVOIR sites to establish contracts with the Cloud providers. Service manifests, which formally define this contract and SLA, play a key role in the whole architecture.

SLA@SOI[15], another EU-funded project, envisages "a business-ready service-oriented infrastructure empowering the service economy in a flexible and dependable way". The project tries to create a framework where Cloud services can be traded as economic goods and SLA agreements can be established between customers and service/ business providers, service providers and all the way down to infrastructure

[1] http://www.Cloudaudit.org/
[2] http://www.cloudsecurityalliance.org/
[3] http://www.dmtf.org/home
[4] http://www.etsi.org/WebSite/Technologies/GRID_CLOUD.aspx
[5] http://www.ogf.org/
[6] http://www.omg.org/
[7] http://openCloudconsortium.org/
[8] http://www.oasis-open.org/
[9] http://www.snia.org/home
[10] http://www.opengroup.org/Cloudcomputing/
[11] http://www.Cloudforum.org/
[12] http://Cloud-standards.org/wiki/index.php
[13] http://www.openCloudmanifesto.org/index.htm
[14] http://www.reservoir-fp7.eu/
[15] http://sla-at-soi.eu/

providers. This multi-domain SLA management framework demands monitoring of the services life-cycle.

Metsch et al. [13] developed a framework based on a combination of the previous two models using the Open Cloud Computing Interface (OCCI)[16]. OCCI describes the interfaces between the two Cloud frameworks which allow them to interoperate. Thus, services which intercommunicate can be managed in the SLA@SOI framework and deployed in the RESERVOIR framework. The main contribution of the architecture is that Cloud services can be deployed and managed in an environment which best fits their needs.

Buyya et al. [8] present the idea of a federated Cloud computing environment (InterCloud) that facilitates scalable provisioning of services under variable conditions, which is based on their previous work [14]. The main components of the federated architecture are the client brokering and coordinator services. A client initiates a Cloud broker in order to meet their needs, whereas Cloud Coordinators publish their services to the federation. Cloud Exchange acts as mediator bringing together service providers and customers. It aggregates infrastructure demands from the application brokers and evaluates them by supplying available resources published by the Cloud Coordinators. The whole communication and transaction is assisted by SLA messages.

Garcia-Sanchez et al. [15] observed that there is limited work so far on the synthesis of emerging IT trends. The authors explore how the combination of such trends can serve interoperability and cost reduction. Hence, they present a framework, called SITIO, to facilitate access to business and Cloud services from the perspective of a broker, based on a combination of existing technologies, such as SaaS, Semantics, Process Modeling and Cloud computing. The proposed platform provides ontologies in order to semantically describe the available services and facilitate customers locating the desired applications, making use of a user-friendly interface.

Dastjerdi et al. [10] claim that it is difficult for heterogeneous Clouds to enforce semantics of virtual machine descriptions and user's requirements. In order to tackle this problem, end-users can discover the desired virtual machine from a range of providers and dynamically deploy it on different IaaS providers. The approach includes interesting novelties such as: user requirements' conversion to Open Virtualization Format (OVF)[17] to be a standard package format for Cloud development, an advertisement approach for IaaS providers and application of an ontology-based discovery to find the best suited providers.

Sun et. al [16] propose a framework for SaaS integration based on SaaS-DL, tools and runtime components. SaaS-DL is a WSDL extension designed to address customization and non-functional policies, which are required for the integration and are not covered by WSDL.

The architectures discussed in this section influenced the drafting of the architecture presented in this work. For example, RESERVOIR [9] and InterCloud [8] are making use of a broker/ mediator architecture in order to overcome semantic interoperability conflicts. Additionally, [15-16] provide unified ways to describe Cloud services. Hence, the standardization of functional and non-functional features in order to achieving Cloud computing semantic interoperability is recommended. This can be achieved by a common ontology [15] or by WSDL extensions [16]. Furthermore, OVF is a widely supported open standard for specifying, packaging and distributing virtual appliances. However, it does not cover the non-functional requirements needed for Cloud interoperation.

B. Cloud APIs

Semantic interoperability can be achieved by offering a public common API that interfaces all different Cloud platforms, e.g. the use of a driver (i.e. mapping) to manage different Clouds. This section addresses the need for a common API and indentifies the existing efforts to build standardized Cloud APIs.

Harmer et al. [6] highlight the need for generic, Cloud-independent applications. This can be achieved with the development of a simple API which would let users specify their requirements and would offer a set of filters that match requirements with providers' capabilities.

Keahey et al. [7] envision a sky computing platform where users are able to easily compare offerings from different providers and move from one provider to another. In their attempt to investigate the interoperability issue between different IaaS providers, they focus on the main obstacles which can be summarized into: machine-image, contextualization and API-level compatibility obstacles. They stress that there are no widely agreed APIs between different IaaS providers. At the semantic level all implementations use similar functions, but differences exist between the SLAs. Providers should publish a set of benchmarks to give users an idea of which performance factors are relevant.

Cloud providers offer different programming APIs for deploying and managing Cloud services. The Amazon API [17] is widely used not only by Amazon Web Services but also by other open source Cloud management tools, such as Eucalyptus[18], openNebula[19] and Nimbus [20]. The main Cloud elements supported by the Amazon API are: (i) the image that is a predefined way to instantiate a virtual machine; (ii) the instance type that defines the instance characteristics (i.e. CPU, RAM, hard disk) as standard templates; (iii) the storage that denotes the component where the data are stored. Three storage components are offered, namely EBS, S3 and SimpleDB, that differ in the way they store the data and in their usage (e.g. EBS can be used as an attached disk to an image while S3 and SimpleDB are simply used as independent storage services). Finally the API offers the capability to create virtual networks that serve as Virtual Private Clouds (VPC).

[16] http://www.occi-wg.org/doku.php

[17] http://www.dmtf.org/standards/published_documents/DSP0243_1.1.0.pdf

[18] http://www.eucalyptus.com

[19] http://www.opennebula.org/

[20] http://www.nimbusproject.org/

TABLE I. MAIN CONCEPTS OF CLOUD APIS AND STANDARDS

	Amazon	GoGrid	Rackspace	vCloud	Sun Cloud	DeltaCloud	LibCloud	OCCI	OVF	CDMI
Image	X	X	X	X	snapshot	X	X	snapshot	X	-
Instance	X	X	X	vApp	VM	X	X	X	-	-
VM template	Type	-	X	Template	VM template	X	X	-	-	
Storage	EBS, S3, SimpleDB	-	Container	X	volume	-	-	X	disk	X
Network	VPC	IP	IP	X	vNET	-	-	X	X	-
Protocol	REST & SOAP	REST	REST	REST	REST	REST	PYTHON	-	-	-

GoGrid [18] and Rackspace [19] are two popular commercial APIs. They both support the image and instance elements. Rackspace offers standard templates to create instances while at GoGrid the instances are created manually (i.e. CPU, RAM, HD). GoGrid does not support storage, although the user can manage the storage through the Web interface, while Rackspace offers the container as a storage unit that is similar to the notion of folder used by operating systems. Both offer the capability to assign IPs to instances and so create virtual networks.

Several efforts have been made to develop common standardized APIs. The vCloud API [20] was submitted to the DMTF Open Cloud Standards Incubator [21] as a candidate of common Cloud interface. TCloud, an extension of vCloud, is also submitted to DMTF [22]. The Sun Cloud API is a well structured API proposed by Sun [23]. Both APIs offer the concepts of instance template, storage and network. vCloud offers the image element, while the Sun Cloud does not offer predefined images to instantiate virtual machines, but it allows creating a virtual machine manually and capturing a backup snapshot of it. Finally the Sun Cloud supports the concept of virtual machine while the vCloud offers a more extended concept, the vApp, that is a software solution containing one or more virtual machines.

The DeltaCloud[21], supported by Red Hat, tries to abstract the differences between diverse Clouds. It supports only computational resources (no storage support). LibCloud [22], supported by the Apache Software Foundation, is a client library, only for compute resources, compatible with many Cloud providers. Finally, JCloud[23], DaseinCloud[24] and fog [24] support both compute and storage management, while CloudLoop [25] and SimpleCloud [26] support only storage management.

The OVF is a DMTF standard that describes virtual appliances for deployment across heterogeneous virtualization platforms (i.e. different hypervisor), allowing the users to deploy their virtual appliances at every Cloud provider [25]. A more "Cloud-friendly" extension of OVF has also been proposed offering attributes like elasticity [26]. The main Cloud concepts used by OVF are the image the storage and the network. Note that the notion of instance is not used since OVF describes only the images used for instantiation.

The OCCI API is a specification for remote management of Cloud computing infrastructure, allowing the development of interoperable tools for common tasks including deployment, autonomic scaling and monitoring. This API introduces three fundamental concepts, which are also used by the APIs described at the beginning of this section. These concepts are compute, storage and network.

SNIA has adopted the Cloud Data Management Interface (CDMI) [27] as a Cloud storage architecture standard for data management (i.e. allows applications to create/retrieve/update and delete data elements from the Cloud). The main concept of the CDMI is the container that can be used with OCCI at an integrated Cloud computing environment where CDMI containers can be used by the Virtual Machines in the Cloud Computing environment as virtual disks.

Table I summarizes the main Cloud concepts/elements used by popular Cloud APIs and standards. "X" denotes that the concept appears in the respective API, while "–" denotes that it does not exist. In case the concept appears under a different term, then this is included in the respective cell of Table I. The image is used to instantiate a virtual machine. The VM template is a predefined set of properties that specify the "size" of the instance e.g. CPU, RAM, OS. The storage is a component for permanently storing data. The network describes the ability of the API to define a virtual network and finally the protocol attribute defines the kind of protocol used.

III. REFERENCE ARCHITECTURE FOR SEMANTICALLY INTEROPERABLE CLOUDS

The proposed Reference Architecture will introduce a scalable, reusable, modular, extendable and transferable approach for facilitating the design, deployment and execution of resource intensive SOA services on top of a semantically interoperable Cloud.

The design of RASIC is influenced by related efforts. It can play the role of the mediator between heterogeneous Cloud infrastructures. RASIC capitalizes on IT trends, such as SOA, semantics and Cloud computing as proposed by [16].

[21] http://deltaCloud.org/

[22] http://incubator.apache.org/libCloud/

[23] http://code.google.com/p/jClouds/

[24] http://dasein-Cloud.sourceforge.net/

[25] https://Cloudloop.dev.java.net/

[26] http://www.simpleCloud.org/

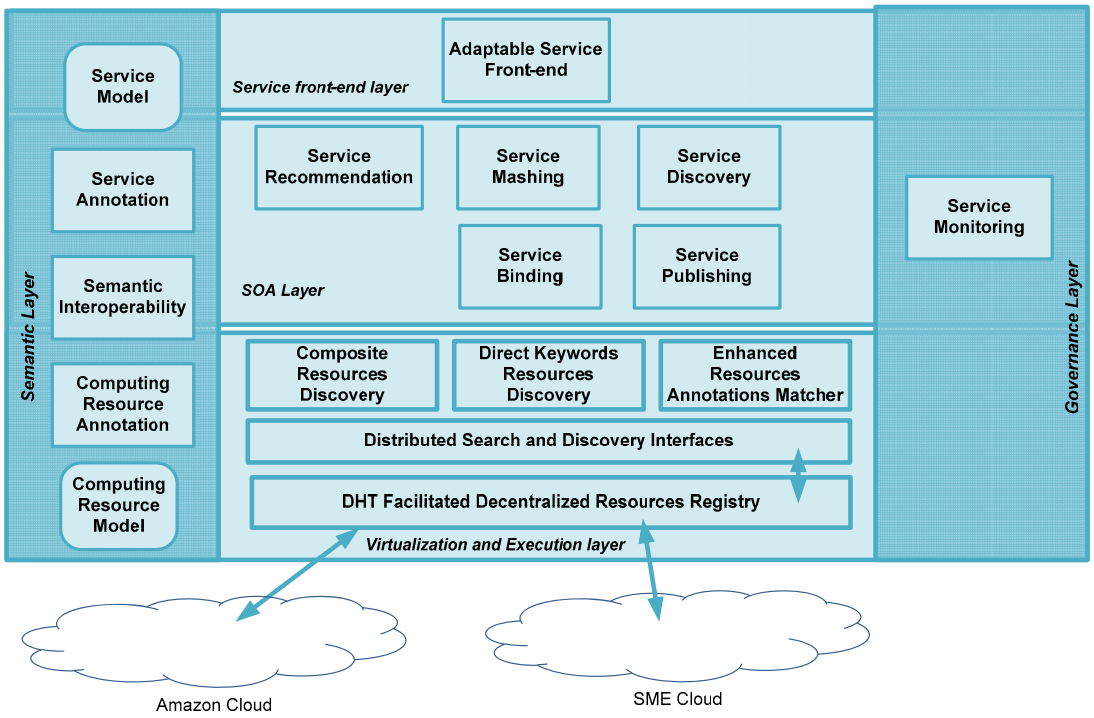

Figure 1. The Reference Architecture for Semantically Interoperable Clouds (RASIC)

The Reference Architecture will rely on open standards and open-source solutions. Moreover, we recognize the importance of service and SLA management and maintenance. However, RASIC, unlike other related efforts, has a clear aim, that of resolving semantic interoperability conflicts in the Cloud in order to facilitate the development of Cloud service-based applications by means of user-centric adaptable interfaces.

The inherited Cloud features allow the on-demand addition of new computing resources, while at the same time the SOA features will allow the development and deployment of rich applications in the form of services. Additionally, the Cloud provides the underlying technical solution for the efficient and highly parallelized execution of resource intensive services. SOA provides the means for users to design and directly deploy the applications of their choice in the form of Web services using intelligent and adaptable service front-ends. SOA also helps to integrate Cloud services backwards into legacy information systems.

Semantics play a catalytic role in the whole process. Lightweight semantics will be used for annotating the Cloud resources. Semantics can be applied at the interface level, the component level, and the data level by utilizing a generic semantic Cloud resource data model. The semantic annotation of resources and services is expected to significantly address the semantic interoperability. Moreover, it facilitates the matching between applications and resources that need to be assigned to it. Semantically annotating Cloud resources also allow us to easily identify clusters of collaborating and/or complementary resources.

Lightweight semantics will also be used in order to annotate the SOA services and the applications deployed by the users on top of the Cloud. The main challenges to be addressed here involve the specification of the appropriate properties such that the service can be deployed, parallelized and efficiently executed in the infrastructure. The rich existing experience and research on the field of service annotation supports this effort.

To provide a flexible Cloud environment where services can allocate resources on demand, existing methods for the adaptive execution and the monitoring of services have to be extended towards dynamic resource allocation mechanisms and service execution. The main challenges to be addressed include the enhancement of existing SLA description models as well as respective negotiation strategies and protocols to enable the annotation of Cloud resources, the development of monitoring and feedback mechanisms to observe the commitments met by an SLA, and the development of adaption strategies to mitigate the effects of possible SLA infringements. All three aspects should be treated both on a design and implementation level.

Special emphasis is given on the definition and development of the adaptable service front-end layer, involving the user in the design process. Our approach here is based on the use of widgets thus enabling the user to manage their Cloud applications in a centralized manner, irrespective of how the Cloud Widgets are distributed across different environments (e.g. mobiles, web browsers, desktop, etc). This service front-end will be supported by rich user interfaces, contextual help and recommendation mechanisms.

RASIC (Fig. 1) comprises of three horizontal layers, namely the Service Front-end layer, the SOA layer, the Virtualization and Execution layer, and two vertical layers, namely the Semantic layer and the Governance layer, that span

across all the horizontal ones. Each of the layers is described in the following sections.

A. The Service Front-end Layer

This layer will offer a set of intuitive (widgetized) service front-ends that will adapt to the user's context. This will allow us to adopt a user centric approach and allow the users to adapt existing Cloud services and create new ones via composition leveraging the paradigm of "mashups".

We should clarify that by users we do not refer to novice end-users but to service developers or people familiar with service-oriented applications design and development.

B. The Semantic Layer

In RASIC, semantic models and technologies will be employed for two main reasons. On the one hand they will resolve semantic interoperability conflicts that are raised when different Cloud platforms exchange data. On the other hand semantics will be used at the SOA layer in order to provide the means for developing intelligent service discovery, mashing and recommendation mechanisms.

The Semantic layer includes the lightweight service and computing resource models, Service and Resource Annotation components, Semantic Interoperability Run-time Engine.

The service model constitutes a simple, open and extendable vocabulary for describing SOA services. It will comprise of a set of terms and their relationships. Similarly the computing resource model will offer a simple, open and extendable approach for describing computing resources. The service and resource models will not be developed from scratch and the reusability of existing efforts will be considered, e.g. existing semantic service models, e.g. [28] or the common Cloud concepts identified during the analysis of Cloud APIs (see common Cloud API described in section IV). Apart from the metadata added by the service providers and Cloud vendors, the service and resource models will also include metadata that emerge bottom-up directly from the usage of the SOA services and the Cloud resources by their end-users [29]. The service and resource models will be formally expressed in ontologies using a standardized ontology language, either OWL or RDF.

The Service Annotation component will facilitate the semi-automatic semantic annotation of the SOA services, while the Resource Annotation component will facilitate the semantic annotation of resources.

The Semantic Interoperability Run-time Engine will allow the Cloud vendors to create mappings in a formal language, e.g. RDF or OWL, between their platforms data models and the Reference Architecture's models. Thus, semantic interoperability between different platforms will be achieved. Afterwards, these mappings will be used for resolving semantic interoperability problems at run-time, e.g. conflicts that might be raised when data are exchanged. At this point reasoning techniques will be employed, in order to perform the semantic matching between equivalent concepts in different ontologies, e.g.[30, 31].

C. The SOA Layer

The SOA layer offers a toolbox (accessible through the adaptable service front-end) in order to design, develop and deploy resource intensive services which utilize the Cloud resources. The SOA layer comprises of the Service Discover, Service Mashing and Service Recommendation components.

The Service Discovery component capitalizes on the search mechanisms offered by the Virtualization and Execution layer and employs lightweight semantic models and techniques in order to find available SOA services, taking care of the binding of the SOA service by retrieving binding parameters from the Decentralized Resources Registry of the Virtualization and Execution layer.

The Service Mashing component utilizes lightweight semantic models and techniques implements a simple and lightweight approach for mashing services. It capitalizes on the publishing mechanisms offered by the Virtualization and Execution layer in order to publish the composite services in the Decentralized Resources Registry.

The Service Recommendation component offers to the users suggestions of related service. The degree of relation between two services is computed based on the similarity of their semantic descriptions as well as on latent relations that emerge from the usage of the services.

All three software components rely heavily on the lightweight semantic service and resource models of the Semantic layer.

D. The Governance Layer

Cloud governance involves applying policies to the use of Cloud services. Cloud platforms, like the services in a SOA, are predominantly accessed using Web service APIs, and so they might be expected to come under the same heading as SOA governance. At the very least, you can reuse the principles behind SOA governance, and especially the lifecycle management of services, referring to the ability to control and track changes to services (i.e. individually-tailored customizations and extensions), and to place controls over who can change a service. Once this facility is in place, an organization can determine who created a service, who changed it, and when the changes were made.

The envisaged technological basis will be a combination of registry and repository tools. According to insights from recent research efforts around the themes of Service Oriented Architecture Governance and Web Service Management [29], and according to several reviews of the state of the software vendor market (e.g. [32] [33] [34]), it appears that integrated registry and repository platforms represent the most effective approach for introducing governance in a service-oriented infrastructure.

E. The Virtualization and Execution Layer

In Cloud infrastructures, it becomes possible to have more or fewer nodes running at any given point in time. To distribute applications and requests across these nodes efficiently it is important to measure each nodes load on a per application basis and develop algorithms to enable the Cloud to decide how to distribute the applications, or when to add or remove

nodes. Distribution of requests and applications are two completely different tasks. Applications have to be distributed across nodes, to enable them to answer requests in the first place. Then on a per request basis it has to be decided which node should answer which request. The Cloud Infrastructure Maintenance Layer, the Virtual Execution Box and the Decentralized Services Registry Infrastructure are involved here.

The Cloud Infrastructure Maintenance Layer includes the different nodes and all the software needed to execute applications on these nodes. Each node is an instance of a predefined virtual machine image consisting of: (a) an operating system enabled to run on top of a virtualization layer, (b) all software necessary to execute applications, and (c) a reporting application with the purpose to report the status of the node and to wait for tasks to process. This application running on each node reporting the status to the controlling application is the first part of a series of applications needed to control the Cloud. The second is the controlling application that receives each nodes status and is able to decide based on this status information how to distribute the applications and requests and also when to create new nodes or when to stop ones not needed anymore. This application should run on the Cloud platform itself to benefit from the scalability and high availability the Cloud platform offers. The Cloud Infrastructure Maintenance Layer is going to support the load measurement process

The Virtual Execution Box defines what and how applications can be executed on the Cloud platform. Applications often have a multitude of dependencies starting with the operating system, executable formats, compilers and ending with middleware or libraries for specific functions. The Cloud Virtual Execution Box has to include some of these dependencies and/or provide a way for developers to add more themselves. Moreover, different programming languages need to be supported to give application developers a choice. This forces the platform to handle the same problems in multiple ways therefore heightening the complexity by a multitude. The measurement of application load is important not just for scalability reasons important, but also for the quality of service. Security is also a major concern here.

RASIC proposes a semantically-enhanced Services Registry Infrastructure, which will index service descriptions of services offered by different Cloud vendors.

Finally, services and resources discovery involves:

- the *direct resources discovery* module that is based on the exact keyword match functionality;
- the *ontology-enhanced services and resources discovery* module that provides semantic level service matching; and
- the *composite discovery* module that tries to compose several services and resources, in case a single service cannot fulfil the query requested.

IV. COMMON CLOUD API

After the literature review discussed in section II.B, we observed that all APIs use similar concepts with similar properties and actions. Hence, common semantics apparently exist. However, the different names and structures used make it difficult to seamlessly move from one Cloud provider to another, as the change of the API may require application refactoring or data transformation. Thus need for a common standardized Cloud API is obvious. This section introduces a model for such a common Cloud API. The concepts and operations included in the proposed model emerge from the review of the related literature presented in section II.B. The common concepts shared by the APIs of section II.B are: *image*, *instance*, *storage* and *network*.

Additionally, most of the APIs offer common operations for the main Cloud concepts. They all offer operations for creating, deleting, starting, stopping and restarting the instance. The main operation assuming the network is the creation of the network. For the storage the operations used are the creation, deletion of a storage unit and the storage of data. Finally for the image the operations provided are the creation and deletion.

Based on the above, a model for a common Cloud API is specified. The main concept of the model is the instance. The API should support the following operation in order to manage the instance: create, delete, start, stop and restart. In order to create the instance an image is used. The operations that need to be supported for managing the image are the creation and the deletion. The instance stores permanently data at a storage component. The API should support the following operation in order to manage the storage: create, delete and store. Finally the instances can be connected to one or more networks.

The common Cloud API model is visualized in Figure 2 using a UML class diagram.

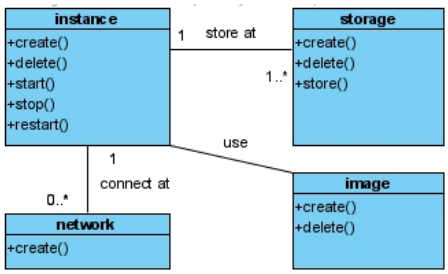

Figure 2. Common Cloud API Model

V. CONCLUSIONS AND FUTURE RESEARCH

Addressing Semantic interoperability in Cloud computing is a major concern. Towards this direction, this work proposed in a semantically-enhanced open Reference Architecture (RASIC) and a model for a common Cloud API. RASIC comprises of three horizontal layers, namely the Service Front-end layer, the SOA layer, the Virtualization and Execution layer, and two vertical layers, namely the Semantic layer and the Governance layer, that span across all the horizontal ones.

RASIC and the development of a standardized common Cloud API can significantly contribute to the reduction of the switching costs between different Cloud vendors and to the elimination of the vendor lock problem. The latter is expected to encourage the entrance of smaller and specialized

companies, apart from the big Cloud vendors, in the Cloud industry.

As part of our future work, we plan to use RASIC as a blueprint in order to develop tools and solutions that will support real-life scenarios of Cloud-based service-oriented applications in the Telecom and Collaborative Work Environment fields in the context of the Cloud4SOA FP7 ICT project. In these scenarios services from different Cloud providers will be composed seamlessly (as semantic conflicts will be resolved) using intuitive, adaptable service front-ends. The common Cloud API specified in section IV, will also contribute towards this directions. Thus, the completeness, the usefulness and the usability of RASIC and of the common Cloud API will be validated and evaluated.

ACKNOWLEDGMENT

This work is supported in part by Science Foundation Ireland under grant SFI/08/CE/I1380 (Líon 2) and in part by the Cloud4SOA FP7 ICT Project under grant 257953.

REFERENCES

[1] P. Mell and T. Grance, "The NIST Definition of Cloud Computing, National Institute of Standards and Technology " in DOI: http://csrc.nist.gov/groups/SNS/Cloud-computing/Cloud-def-v15.doc, 2009.

[2] Gartner, "Gartner Identifies the Top 10 Strategic Technologies for 2010 " in DOI: http://www.gartner.com/ it/page.jsp?id=1210613, 2009.

[3] Economist, "Battle of the Clouds," in DOI: http://www.economist.com/ opinion/displaystory.cfm?story_id=14644393, 2009.

[4] Economist, "Battle of the Clouds," in DOI: http://www.economist.com/displaystory.cfm?story_id=14637206, 2009.

[5] ENISA, "Cloud Computing-Benefits, risks and recommendations for information security," in DOI: http://www.enisa.europa.eu/act/ rm/files/deliverables/Cloud-computing-risk-assessment/, November 2009 (White paper).

[6] T. Harmer, P. Wright, C. Cunningham, and R. Perrott, "Provider-Independent Use of the Cloud," in *Proceedings of the 15th International Euro-Par Conference on Parallel Processing (Euro-Par '09),* Delft, The Netherlands, 2009, pp. 454-465.

[7] K. Keahey, M. Tsugawa, A. Matsunaga, and J. Fortes, "Sky Computing," *IEEE Internet Computing,* vol. 13, pp. 43-51, September/October 2009.

[8] R. Buyya, R. Ranjan, and R. N. Calheiros, "InterCloud: Utility-Oriented Federation of Cloud Computing Environments for Scaling of Application Services," in *Proceedings of the 10th International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP 2010)*, Busan, South Korea, 2010.

[9] B. Rochwerger, D. Breitgand, E. Levy, A. Galis, N. K., 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,* October 2008.

[10] A. V. Dastjerdi, S. G. H. Tabatabaei, and R. Buyya, "An Effective Architecture for Automated Appliance Management System Applying Ontology-Based Cloud Discovery," in *Proceedings of the 10th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid 2010),* Melbourne, Australia, 2010.

[11] Cloud Computing Use Case Discussion Group, "Cloud Computing Use Cases V.4," in DOI:http://Cloud-computing-use-cases.googlegroups.com /web/Cloud_Computing_Use_Cases_Whitepaper-4_0.pdf, 2010 (White Paper).

[12] J. Foster, "Cloud Computing Standards, Dream vs. Reality " *in Trend Cloud Security Blog,* 2009.

[13] T. Metsch, A. Edmonds, and V. Bayon, "Using Cloud Standards for Interoperability of Cloud Frameworks," 2010 (Technical Report).

[14] R. Buyya, C.-S. Yeo, and S. Venugopal, "Market-Oriented Cloud Computing: Vision, Hype, and Reality for Delivering IT Services as Computing Utilities," in *Proceedings of the 2008 10th IEEE International Conference on High Performance Computing and Communications (HPCC '08),* 2008, pp. 5-13.

[15] F. Garcia-Sanchez, E. Fernandez-Breis, R. Valencia-Garcia, E. Jimenez, J. Gomez, J. Torres-Nino, and D. Martinez-Maqueda, "Adding Semantics to Software-as-a-Service and Cloud Computing," *WSEAS TRANSACTIONS on COMPUTERS,* vol. 9, 2010.

[16] W. Sun, K. Zhang, S.-K. Chen, X. Zhang, and H. Liang, "Software as a Service: An Integration Perspective," in *Proceedings of the 5th international conference on Service-Oriented Computing,* Vienna, Austria, 2007.

[17] Amazon, "Amazon Elastic Compute Cloud API," in DOI: http://docs. amazonwebservices.Scom/AWSEC2/latest/APIReference/, 2010.

[18] GoGrid, "GoGrid API," in DOI: http://wiki.gogrid.com/wiki/index.php/ API, 2010.

[19] Rackspace, "Cloud Servers Developer Guide API v 1.0," in DOI: http://docs.rackspaceCloud.com/servers/api/v1.0/cs-devguide-20091015. pdf, 2009 (beta).

[20] VMware, "vCloud API Specification v 0.9," in DOI: http://communities.vmware.com/servlet/JiveServlet/previewBody/12464 -102-1-13008/vCloud _API_Specification.pdf, 2009 (techical note).

[21] DMTF, "DMTF Open Cloud Standards Incubator," in DOI: http://www.dmtf.org/about/Cloud-incubator, 2010.

[22] Telefonica, "Telefónica's TCloud API Specification," in DOI: http://www.tid.es/files/doc/apis/TCloud_API_Spec_v0.9.pdf, 2010.

[23] SunMicrosystems, "The Sun Cloud API," in DOI: http://kenai.com/ projects/sunCloudapis/pages/Home, 2009.

[24] W. Beary, "Fog Cloud," in DOI: http://github.com/geemus/fog, 2010.

[25] DMTF, "Open Virtualization Format Specification," in DOI: http:// www.dmtf.org/standards/published_documents/DSP0243_1.0.0.pdf., 2009.

[26] F. Galan, A. Sampaio, L. Rodero-Merino, I. Loy, V. Gil, and L. M. Vaquero, "Service specification in Cloud environments based on extensions to open standards," *in Proceedings of the Fourth International ICST Conference on COMmunication System softWAre and middlewaRE (COMSWARE '09)*, New York, NY, USA, 2009, pp. 1-12.

[27] SNIA, "Cloud Data Management Interface version 1.0," in DOI: http://www.snia.org/tech_activities/standards/curr_standards/cdmi/CDM I_SNIA_Architecture_v1.0.pdf, 2010 (SNIA Technical Position).

[28] J. Farrell and H. Lausen, "Semantic Annotations for WSDL and XML Schema (W3C Recommendation)," in DOI: http://www.w3.org/TR/ sawsdl/, 2007.

[29] D. Kourtesis, I. Paraskakis, and A. J. H. Simons, "Semantic Web Technologies in Support of Service Oriented Architecture Governance," *In Proceedings of the 4th South East European Doctoral Student Conference*, 2009.

[30] K. Gomadam, A. Ranabahu, L. Ramaswamy, K. Verma, and A. P. Sheth, "Mediatability: Estimating the Degree of Human Involvement in XML Schema Mediation," *In Proceedings of the 2nd IEEE International Conference on Semantic Computing*, Santa Clara, CA, USA, 2008.

[31] A. Mocan, F. Facca, N. Loutas, V. Peristeras, S. Goudos, and K. Tarabanis, "Solving Semantic Interoperability Conflicts in Cross-Border E-Government Services," *Int. J. Semantic Web Inf. Syst*., vol. 5, pp. 1-47, 2009.

[32] Hewlett-Packard, "Transforming Business: Optimizing the Business Outcomes of SOA," 2008 (White Paper).

[33] Oracle Corporation, "SOA Governance: Framework and Best Practices," 2007 (White Paper).

[34] Sun Microsystems Inc, "Effective SOA Deployment Using an SOA Registry-Repository," 2005 (White Paper).