

Semantic Engine and Cloud Agency for Vendor Agnostic Retrieval, Discovery, and Brokering of Cloud Services

Alba Amato, Giuseppina Cretella, Beniamino Di Martino^(✉), Luca Tasquier,
and Salvatore Venticinque

Department of Industrial and Information Engineering, Second University of Naples,
Aversa, Italy

bemiamino.dimartino@unina.it,
{alba.amato,giuseppina.cretella,luca.tasquier,
salvatore.venticinque}@unina2.it

Abstract. Cloud computing is moving from being a testing ground for isolated projects to being a strategic approach of the entire business organization. So the choice among the possible cloud offers, with a strong focus on the choice of services that enable better processes and projects of the business lines, is gaining importance. Nevertheless the heterogeneity of the Cloud services, resources, technology and service levels offered by the several providers make difficult to decide. Besides the inconveniences caused by the “lock-in”, give rise to the need for developers to be able to develop an application regardless of where it is released, structuring and building it in a vendor agnostic way so that it is possible to deploy on the provider that best fits them at the moment. The mOSAIC project aims at designing and developing an innovative open-source API and platform that enables applications to be Cloud providers’ neutral and to negotiate Cloud services as requested by their users, allowing automatic discovery, matchmaking, and thus supporting selection, brokering, interoperability end even composition of Cloud Services among multiple Clouds. In this paper, we illustrate the interoperation of the two components, the Semantic Engine and the Cloud Agency for the agnostic retrieval, discovery and brokering of cloud services. The focus will be put on the way to support the Cloud Application Developer to express the requirements and services/resources in vendor agnostic way and to translate automatically these requirements into a neutral format in order to compare it with the different offers of providers and to broker the best one according to defined policies.

Keywords: Cloud brokering · Multi-agents systems · Cloud ontology · Semantic discovery · Cloud interoperability

1 Introduction

Cloud computing is moving from being a testing ground for isolated projects to being a strategic approach of the entire business organization. So the choice

among the possible cloud offers, with a strong focus on the choice of services that enable better processes and projects of the business lines, is gaining importance. Nevertheless the heterogeneity of the Cloud services, resources, technology and service levels offered by the several providers make it difficult to decide [1, 2]. In fact different vendors have introduced different paradigms and services so leading to clouds that are diverse and vendor-locked, as happened during the early days of the computer hardware industry, when each vendor made and marketed its own version of incompatible computer equipment. Besides the inconveniences caused by the “lock-in”, give rise to the need for developers, to be able to develop an application regardless of where it is released structuring and building it in a vendor agnostic way so that it is possible to deploy on the provider that best fits them at the moment. Even if several efforts have been made to standardize clouds’ important technical aspects, for example from the US National Institute of Standards and Technology (NIST), standardization is still far from reality.

In this scenario, it would be useful to have a way to express the user’s requirements closer to the user logic, translate automatically these requirements into a neutral format in order to compare it with offers of providers and for choosing the best one according to defined policies. A common ontology can help to bridge the gap between application requirements and technical requirements declared by resource providers. In fact semantic can help address clouds key interoperability and portability issues. For example semantic technologies are useful to define an agnostic, machine readable, description of resources to be compared with the vendor offers using a brokering system, that acquire autonomically resources from providers on the basis of SLA evaluation rules.

The mOSAIC project [3] aims at designing and developing an innovative open-source API and platform that enables applications to be Cloud providers’ neutral and to negotiate Cloud services as requested by their users, allowing automatic discovery, matchmaking [4], and thus supporting selection, brokering, interoperability end even composition of Cloud Services among multiple Clouds.

In order to support this selection and requirements specification has been developed:

- a Knowledge Base, representing resources and domain concepts and rules by means of Semantic Web Ontologies and inference rules;
- a support tool, the Semantic Engine, that helps the user to abstract the requirements in vendor independent way starting from application requirements or from specific vendor resources;
- a Cloud Agency, that compares the different offers of providers with the user proposal and retrieves the best offer. The user can also delegate to the Agency the monitoring of resource utilization, the necessary checks of the agreement fulfillment and eventually re-negotiations.

In this paper, we illustrate the interoperation of the two components, the Semantic Engine and the Cloud Agency for the agnostic retrieval, discovery and brokering of cloud services. The focus will be put on the way to support the Cloud Application Developer to express the requirements and services/resources in

vendor agnostic way and to translate automatically these requirements into a neutral format in order to compare it with the different offers of providers and to broker the best one according to defined policies.

The user can choose the known concepts that describe his application or the required resources, utilizing a knowledge base and inference rules managed by the Semantic Engine, which supports him/her to produce a vendor agnostic template of a Service Level Agreement, to be used for negotiating a concrete offer from the available Cloud vendors. The Cloud Agency interacts with the supported providers for retrieving the available offers and brokers the best one(s). The Semantic Engine can further be useful for filtering many proposals, which are optimal according to different criteria, when the user's knowledge is not helpful.

The paper is organized as follows. In Sect. 2 we present the design architecture, in Sects. 3 and 4 we present an ontology supporting the semantic representation of resources and the engine based on it. In Sect. 5 a description of the Cloud Agency and of the utilization of Broker Agents is provided; In Sect. 6 an example is shown. In Sect. 7 we present an overview of works related to semantic representation of cloud resources and multi-cloud resource brokering and negotiation. Conclusions are drawn in Sect. 8.

2 Approach and Architecture

A Cloud Application Developer, who intends to develop a cloud based application, would like to express his or her requirements according to the application logic, to make a choice based on what he or she knows and based on high level requirements. In order to support this selection and requirements specification, we have developed:

- a Knowledge Base, representing resources and domain concepts and rules by means of Semantic Web Ontologies and inference rules;
- a support tool, the Semantic Engine, that helps the user to abstract the requirements in vendor independent way starting from application requirements or from specific vendor resources;
- a Cloud Agency, that compares the different offers of providers with the user proposal and retrieves the best offer.

In Fig. 1 the integration and interaction of such components is shown.

The Semantic Engine, based on the ontologies and inference rules representing the Knowledge Base, enables the user in defining his or her requirements in a format suitable for comparison among offers and produces an SLA template that is passed to the Cloud Agency. The Cloud Agency adds the brokering rules so composing the *Call for Proposal (CFP)* [5] that describes the list of resources, which are necessary to run cloud applications. It includes also the negotiation rules to select the best offer among those proposed by providers. After that the Cloud Agency compares each proposal with the user's template and retrieves the best offer.

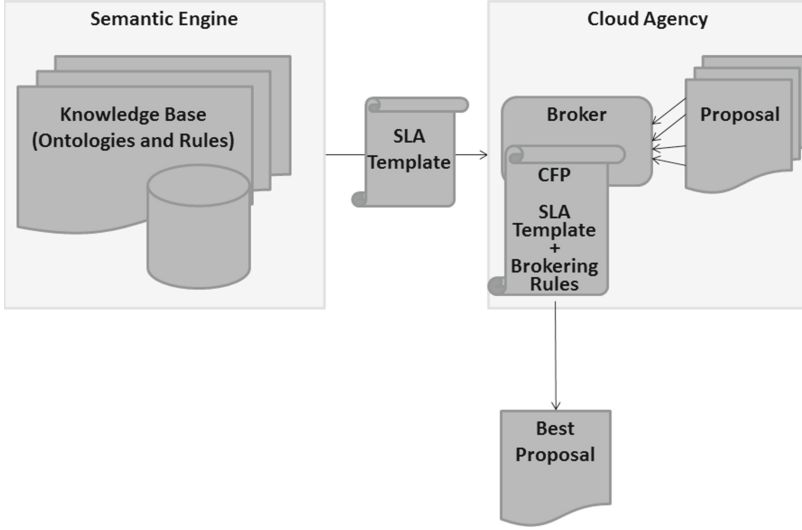


Fig. 1. Integrated view of knowledge base, Semantic Engine and Cloud Agency

3 An Ontology for the Development of Cloud Based Application

The knowledge base developed in order to support the search and discovery of suitable Cloud resources and component is structured into four sub-ontologies and is developed using OWL language [6].

The sub-ontologies have the following purpose.

- The *ApplicationDomain* ontology represents the application and its patterns expressed in the domain terminology of the end user. This level of abstraction contains concepts related to the application domain of applications, as instance data mining, big data application related concepts.
- The *FunctionalDomain* ontology represents functional concepts of both cloud and non cloud domain, such as Cloud functionalities and services but also traditional design and execution patterns for distributed and concurrent application.
- The *InfrastructureResourceDomain* ontology provides concepts and relation useful to describe information related to the resource such as Virtual Machines, storage and network and their composite configurations.
- The *ImplementationDomain* ontology models information related to the concrete APIs of Cloud platforms. In particular this level contains the grounding elements of effective cloud services, that means the elements useful to invoke the implemented functionalities.

The four ontologies are linked together with relationships, for example, the *ApplicationDomain* ontology imports the *FunctionalDomain* in order to relate

application-domain concepts to functional patterns. The *FunctionalDomain* ontology imports the *ImplementationDomain* and the *InfrastructureResourceDomain* ontologies in order to establish semantic relationships for each function needed by the Cloud application with grounding elements as specified above. In such a way, the necessary grounding elements for the Cloud application can be retrieved through selection of domain specific and functional concepts at higher level of abstraction.

It's worthwhile to have a look inside the ontology of the lowest level, which represents the concepts that are actually returned as outputs of the semantic module and are passed to the Cloud Agency. This "Infrastructure Resource Domain ontology" contains the semantic structure to describe the basic resources described by OCCI (Compute, Storage, Network) and an additional concept that is the configuration, that means a composition of single resources. For the representation of this ontology we started from OCCI description of resource interface and we provide a uniform way to represent these information through an ontology. For this reason our ontology is compliant with the OCCI resource description [7].

The class hierarchy of this sub-ontology is shown in Fig. 2. This ontology classifies the resources of type compute, storage and network in vendor resources and agnostic resources. In the vendor resources we collect a series of offers by cloud provider like IBM and Amazon, while in agnostic resources we collect resources and their characteristic not linked with the offers of cloud provider. The link between a configuration element and the resources that compose it are represented through owl object property, while the characteristic of the single resource are defined through owl data property according to the attribute defined in OCCI [8] (Fig. 4). To the standard OCCI attributes we added two parameters for the description, *gpu* and *price*. The vendors' offers of several IAAS cloud provider are represented in this ontology through individuals and their characteristics. Figure 3 reports the list of individuals representing resources from the IBM and Amazon Cloud provider offers already represented in the ontology. Of course this list can be easily enriched. A cloud user accustomed to a particular cloud provider may start from the specific customized solution (for instance the IBM Silver Compute) and translate this solution in vendor independent' terms through the Semantic Engine, then pass this neutral representation to the Cloud Agency to find an equivalent solution that fulfill additional requirements.

If instead the user don't know which are the technical requirement of his/her application, he/she can start specifying high level requirements as the complexity of the algorithms used or functional/design requirements. These requirements may be expressed using concepts contained in the knowledge base and then can be translated in infrastructural requirements by the application of heuristic rules. By following the generic structure of an *ApplicationPattern* that is based on a design pattern or a composition of design pattern, it is possible to semantically describe a whole range of engineering applications. This engineering application can be semantically described by instantiating an *ApplicationPattern* class and all the composing concepts including the *AlgorithmicConcept* and the *Patterns* concepts with specific instances. All these concepts are semantically represented in the Application and Functional Domain Ontologies.

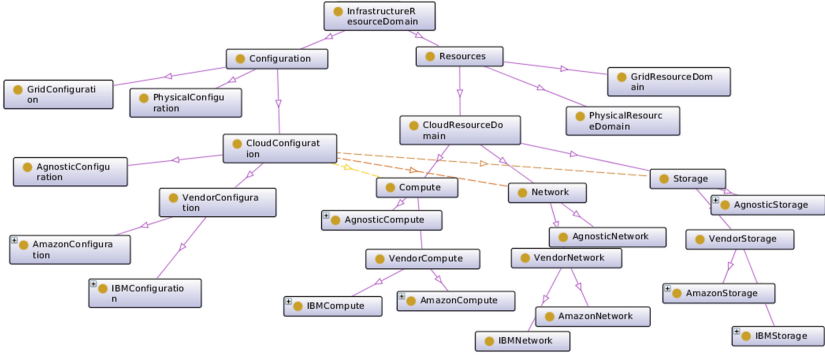


Fig. 2. Infrastructure resource ontology class hierarchy

An important feature of the Semantic Engine is its capability to deduce an appropriate parameterized configuration of the Cloud application and a generic description of needed IaaS resources based on high level requirements. This feature is made possible by the execution of inference rules that extensively use the semantic description of the application, particularly its design pattern and the description of the critical aspects of the application that need elasticity, such as the computational or data complexity of the algorithm. Listing 1.1 shows an example of a rule that provides information on the needed properties of a Virtual Machine that hosts a Web server in a Cloud application. This rule, based on the design logic of the application, in this case a three-tier architectural pattern, and on information about the expected visitors peak, aims to provide information to the developer about the properties of the Virtual Machine that has to be acquired from an IaaS provider.

Listing 1.1. Inference Rules

```
@prefix AP:ApplicationPattern.owl#
@prefix IRD:ResourceDomain.owl#
@prefix FP:FunctionalDomain.owl#

[WebAppRule:
(?x rdf:type AP:WebApplication),
(?x AP:hasDesignLogic ?dl),
(?dl rdf:type FP:Three-tier),
(?x AP:hasPeakVisitors ?y),
swrlb:divide(?k, 100, ?y),
swrlb:add(?k4, 4, ?k)
(?z rdf:type IRD:Compute),
(?z IRD:cores ?k ),
greaterThan(?r, ?k),lessThan(?r, ?k4),
-> (?x AD:PatternUseInfrastructure ?z) ]
```

Amazon Compute	Amazon Storage	Amazon Configuration
<ul style="list-style-type: none"> ◆ AmazonClusterEightXLCompute ◆ AmazonClusterGpuQuadrupleXLCompute ◆ AmazonClusterQuadrupleXLCompute ◆ AmazonHiCpuMediumCompute ◆ AmazonHiCpuXLCompute ◆ AmazonHiMemoryDoubleXLCompute ◆ AmazonHiMemoryQuadrupleXLCompute ◆ AmazonHiMemoryXLCompute ◆ AmazonLargeCompute ◆ AmazonMicroCompute ◆ AmazonSmallCompute ◆ AmazonXLCompute 	<ul style="list-style-type: none"> ◆ Amazon160 ◆ Amazon1690 ◆ Amazon3370 ◆ Amazon350 ◆ Amazon420 ◆ Amazon850 	<ul style="list-style-type: none"> ◆ AmazonClusterEightXL ◆ AmazonClusterGpuQuadrupleXL ◆ AmazonClusterQuadrupleXL ◆ AmazonHiCpuMedium ◆ AmazonHiCpuXL ◆ AmazonHiMemoryDoubleXL ◆ AmazonHiMemoryQuadrupleXL ◆ AmazonHiMemoryXL ◆ AmazonLarge ◆ AmazonMicro ◆ AmazonSmall ◆ AmazonXL
IBM Compute	IBM Storage	IBM Configuration
<ul style="list-style-type: none"> ◆ IBMBronzeCopper32bCompute ◆ IBMBronzeCopper64bCompute ◆ IBMGold32bCompute ◆ IBMGold64bCompute ◆ IBMPlatinum64bCompute ◆ IBMReservationUnitCompute ◆ IBMSilver32bCompute ◆ IBMSilver64bCompute 	<ul style="list-style-type: none"> ◆ IBM1024 ◆ IBM175 ◆ IBM2048 ◆ IBM350 ◆ IBM60 ◆ IBM850 ◆ IBM9600 	<ul style="list-style-type: none"> ◆ IBMBronzeCopper32b ◆ IBMBronzeCopper64b ◆ IBMGold32b ◆ IBMGold64b ◆ IBMPlatinum64b ◆ IBMReservationUnit ◆ IBMSilver32b ◆ IBMSilver64b

Fig. 3. Amazon and IBM resources list

The execution of inference rules on the knowledge base results in a list of the needed IaaS resources for the application. Listing 1.2 presents a part of such description.

Listing 1.2. Description of a resource necessary for the Cloud application

```

<ws:ServiceDescriptionTerm ws:Name ="Compute" >
  <Compute >
    <cpuSpeed>1.25</cpuSpeed>
    <cpuCores>16</cpuCores>
    <architecture>x86</architecture>
    <memory>16</memory>
  </ Compute >
</ws:ServiceDescriptionTerm >

```

The list of needed resources provided by the Semantic Engine can be used by the Cloud Agency to automatically negotiate resources with a variety of IaaS providers. Once the needed resources are negotiated with the Cloud provider, the application can be deployed by using the list of needed software components and the mOSAIC's deployment tool.

4 Semantic Engine

The Semantic Engine [9] is a prototype tool that supports the user in Cloud Applications' development by discovering cloud APIs functionalities and resources based on semantic technologies. It handles, maintains and exposes to the user in a graphical way the semantic descriptions of application domain

Resource	Attribute	Type	
Compute	architecture	Enum {x86, x64}	■ address
	cores	Integer	■ allocation
	hostname	String	■ architecture
	speed	Float	■ cores
	memory	Float	■ gateway
	state	Enum {active, inactive, suspended}	■ gpu
	summary	String	■ hostname
Storage	size	Float	■ label
	state	Enum {online, offline, backup, snapshot, resize, degraded}	■ memory
	summary	String	■ persistent
Network	vlan	String	■ price
	label	String	■ size
	state	Enum {active,inactive}	■ speed
	address	String	■ state
	gateway	String	■ status
	allocation	Enum {dynamic, static}	■ summary
	summary	String	■ vlan

Fig. 4. List of object property to define resources compliant with OCCI

concepts, application related concepts, general design patterns and programming functionalities, specific API implementations and Cloud resources. In other words it exposes graphically the knowledge base presented in Sect. 3. In order to achieve its mission, the Semantic Engine (based on the ontology levels already described) introduces a high level of abstraction over a range of domain concepts from the engineering discipline [10], generic application patterns as well as details on existing Cloud APIs and IaaS providers. By implementing an additional layer of abstraction, this tool overcomes syntactical differences of existing Cloud APIs, so that it is possible to explore application design patterns independently from the target API. The Semantic Engine fully exploits the expressivity of the OWL DL language specie to relate entities with properties and constraints.

It allows for reuse of the semantic description of the application to be developed, performed by the user during the query phase, by allowing for the definition of application patterns, stored in the knowledge base, and reused in future searches.

In this section we describe how the user can create an agnostic description of resources guided by the Semantic Engine. To produce the CFP part related to resource list the user can use three different options. The first one (the simplest) is to fill the fields suggested by the tool for the particular resource selected. The second one is to select a cloud vendor customized resource configuration and from this obtain an agnostic description. The third one is to specify the user requirements referring to the application he or she intends to develop like information related to the workload or design and functional pattern.

In particular for this third usage mode, the developer, who is a domain specialist may use the Semantic Engine to:

- search for domain concepts related to the application domain, for example the information retrieval, e.g. KWIC (Key Word In Context) index and find the concept of an application for KWIC system based on a specific model;
- investigate the key requirements of the application, e.g. find out that a critical computationally intensive part of the application;
- analyse the Cloud application pattern and eventually associate algorithms or functional patterns to the *ApplicationPattern*;
- identify key software components, such as message queues and storage components, necessary for the Cloud application, as represented in terminology of the appropriate programming model and the associated software platform;
- identify the workflow between the components;
- draw a detailed design of the necessary software components of the application and the information and data flow among them;
- query the Semantic Engine to retrieve the number of needed software components for the task at hand;
- use the Semantic Engine to prepare a list of required resources (i.e. Virtual Machines) for the application, which can be used by the mOSAIC's Cloud Agency for negotiation of optimal offers;
- analyse a list of proposed IaaS providers suitable for the application;
- finally, provide a descriptor of the Cloud application, which can be used by the software platform to start the execution of all the necessary software components i.e. to launch the Cloud application.

In addition to the list of resources and their characteristic, the Semantic engine provides also a way to support the definition of some constraints. The definition of these constraints can be driven by heuristic rules that define the parameters to take into account while developing a certain kind of application and by user constraints. For example the user can express constraints like the maximum price, or the need to have at least a certain value for a resource's parameter.

5 Cloud Agency

Cloud Agency (CA) [11] is a Multi Agent System conceived for provisioning by negotiation, monitoring and reconfiguration of acquired resources (Fig. 5). Using Cloud Agency, the user can negotiate the needed resources in order to run his applications. The user can also delegate to the Agency the monitoring of resource utilization, the necessary checks of the agreement fulfillment and eventually re-negotiations. Cloud Agency will supplement the common management functionalities which are currently provided by IaaS Private and Public infrastructure with new advanced services, by implementing transparent layer to IaaS Private and Public Cloud services. Cloud Agency will support the Cloud user in two different scenarios. In the *Deployment* scenario Cloud Agency supports the discovering and provisioning of the available resources needed to run Cloud applications. In this case the user is negotiating, by the Cloud Agency, the resources it needs to run his/her applications. In order to propose to the user

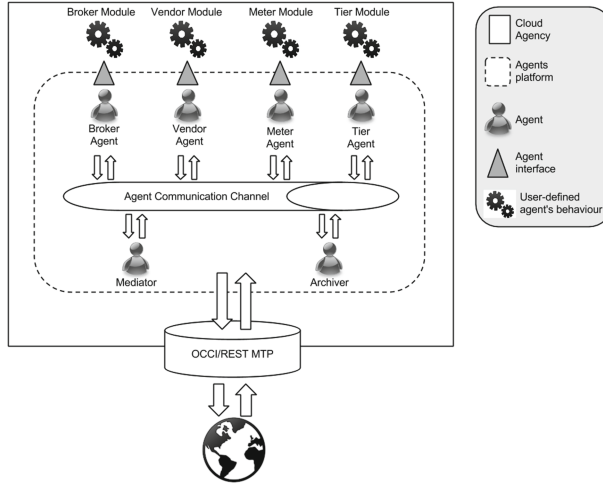


Fig. 5. Cloud Agency architecture

the best offer of resources, that fits his requirements at best, the Cloud Agency will use a Brokering Module that chooses among the available offers the best one. Furthermore for configuration and start of resources it will provide management facilities. In the *Execution* scenario it allows to monitor and eventually to reconfigure Cloud resources according the changed requirements of the Cloud Application. More specifically, during the execution Cloud Agency allows the user for the Monitoring of the infrastructure in terms of resource utilization and for the definition of some strategy of autonomic reconfiguration. Reconfiguration can use management facilities by stopping, starting, moving instances, but it could ask for provisioning of additional resources. By going more in details, Cloud Agency exposes four main services:

- The *Provisioning Service* allows the user to discover, acquire and set up resources for deployment of his/her applications. The result of provisioning is a set of Cloud resources that are described, together with the offered service levels, in a Service Level Agreement (SLA).
- The *Management Service* is used both for deployment and for execution. In fact it is needed to configure and start resources before starting the application, and it is necessary to start/stop/migrate and reconfigure in general the resource dynamically during its utilization.
- The *Monitoring Service* is used to take under control Cloud resources in terms of performance indexes and QoS parameters. This service is implemented by the using of dedicated agents that act as probes on the selected resources.
- The *Reconfiguration Service* is in charge of reconfiguring the Cloud infrastructure when some critical events occur, such as saturation or under-utilization of a resource, SLA violation and so on.

Cloud Agency provides asynchronous APIs in order to access the Cloud Agency services. To address this issue Use Cases are designed in terms of Service

Requests, Events and Callbacks. Access to Cloud Agency services will be enabled by *HTTP RESTful interface*. Asynchronous requests are used to ask the Cloud Agency for something to be executed. For example to start a Negotiation, to accept or to refuse an SLA, to change a Policy, etc. They are not-blocking invocations. Execution is started remotely, but the client can continue to run. Completion or failures of requests are notified at client side. Clients are in charge to handle incoming events. Synchronous requests are available to get information. For example clients can ask for reading an SLA, the status of a negotiation, to get the list of vendors, or the list of resources. Queries are synchronous, they return immediately the response if it is available, an exception otherwise. An OCCI compliant Message Transfer Protocol (OCCI-MTP) allows the communication between the client and the Cloud Agency [12,13]. By using this interface, clients can start new provisioning transactions in order to broker the Cloud resources.

The configuration of the resources that are necessary to execute the user's application produced by Semantic Engine and expressed in terms of SLA template may be complemented by the user with other information. In particular the SLA template can include desired service levels and other terms of service like contract duration, data location and billing frequency. In listing 1.3 an example of SLA template is shown. It contains service description terms and guarantee terms in WS-Agreement. The requested resource is a Virtual Machine configuration with an architecture x86, 2 Cores, 2 Gb of available memory and a price not greater than 0.8\$.

Listing 1.3. Service Description Term and Guarantee Term

```
<ws:ServiceDescriptionTerm ws:Name='Compute' >
    <Compute>
        <cpuCores>2</cpuCores>
        <architecture>x86</architecture>
        <memory> 2GB </memory>
    </Compute>
</ws:ServiceDescriptionTerm>
[...]
```

```
<wsag:GuaranteeTerm wsag:Name='Availability'>
    <wsag:Variables>
        <wsag:Variable wsag:Name="Price"
            wsag:Metric="price/hour" />
        <wsag:ServiceLevelObjective> 0.8 </
            wsag:ServiceLevelObjective>
    </wsag:Variables>
    [...]
</wsag:GuaranteeTerm>
```

The SLA template is part of the Call for Proposal (CfP). The last part of the CfP is a set of brokering rules. Examples of brokering rules are the best price, the greatest number of cores, the best accredited provider or the minimum accepted availability. The provisioning service provided by Cloud Agency implements an extension of the Contract Net Interaction Protocol [14]. The CfP is submitted

to Cloud Agency that returns one or a number of different solutions, which can be optimal according to different criteria. The sequence diagram that describes the interaction among agents for resource provisioning is shown in Fig. 6.

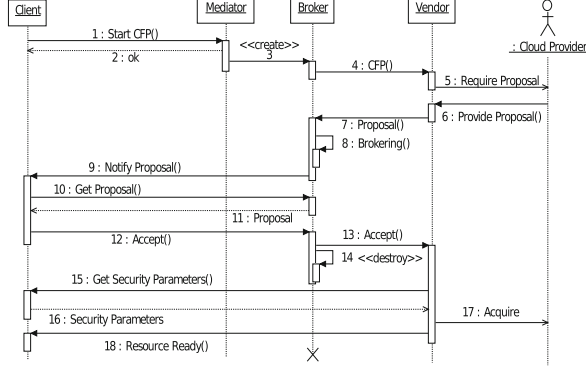


Fig. 6. Interaction among agents for resource provisioning

For each received CFP Cloud Agency creates a broker that searches for vendors that can offer resources with the required QoS (Quality of services). Cloud Vendors neither implement negotiation services, nor provide descriptions of their SLA in machine-readable language. We address these issues by Vendor Agents, which wrap the services of each Cloud provider and return, for each SLA template received from the broker, the available proposal that accomplishes at the best the claimed requirements. The broker collects a number of proposals described in a uniform way and chooses the best one(s) according to the brokering rules. If the user accepts one among the received proposals an SLA is agreed and the offered resources are allocated.

6 A Concrete Use Case

In order to show how the brokering process takes place and the two components (Semantic Engine and Cloud Agency) interact, we present in this section a simple example involving real cloud providers. Please consider that, even if the proposals reported in this example are real, the final result of evaluation may have completely different results with little changes in offerings, that continuously happen in the cloud environment. Let us assume a user (Cloud Application Developer) looking for a Virtual Machine with (i) specific CPU architecture and a fixed amount of memory, (ii) the maximum number of cores, (iii) brokering the best price among the proposals which satisfy (i) and (ii). The user can identify and express in agnostic way her/his requirements with the help of Semantic Engine, by means of the graphical facility shown in Fig. 7, to express the resources' requirements and then to automatically translate them into the SLA template.

Table 1. Available Instance Types And Prices

Offer	Amazon EC2	Windows Azure
xsmall	N/A	CPU Cores: Shared, Memory: 768 MB, Disk Space Web: 20 GB, Disk Space VM Role: 20 GB, Bandwidth: 5, Cost/Hour: \$ 0.04
small	CPU: 1 EC2 Compute Unit (1 virtual core with 1 EC2 Compute Unit), Memory: 1.7 GB, Disk: 160 GB, Platform:32-bit or 64-bit, I/O Performance: Moderate, Cost/Hour Linux/UNIX Usage: \$ 0.09, Cost/Hour Windows Usage \$ 0.115	CPU Cores: 1, Memory: 1.75 GB, Disk Space Web: 230 GB, Disk Space VM Role: 165 GB, Bandwidth: 100, Cost/Hour: \$ 0.12
medium	CPU: 2 EC2 Compute Unit (1 virtual core with 2 EC2 Compute Unit), Memory: 3.75 GB, Disk: 410 GB, Platform:32-bit or 64-bit, I/O Performance: Moderate Cost/Hour Linux/UNIX Usage: \$ 0.180, Cost/Hour Windows Usage \$ 0.230	CPU Cores: 2, Memory: 3.5 GB, Disk Space Web: 500 GB, Disk Space VM Role: 340 GB, Bandwidth: 200, Cost/Hour: \$ 0.24
large	CPU: 4 EC2 Compute Unit (2 virtual core with 2 EC2 Compute Unit), Memory: 7.5 GB, Disk: 850 GB, Platform: 64-bit, I/O Performance: High Cost/Hour Linux/UNIX Usage: \$ 0.360, Cost/Hour Windows Usage \$ 0.460	CPU Cores: 4, Memory: 7 GB, Disk Space Web: 1 TB, Disk Space VM Role: 850 GB, Bandwidth: 400, Cost/Hour: \$ 0.48
xlarge	CPU: 8 EC2 Compute Unit (4 virtual core with 2 EC2 Compute Unit), Memory: 15 GB, Disk: 1690 GB, Platform: 64-bit, I/O Performance: High Cost/Hour Linux/UNIX Usage: \$ 0.720, Cost/Hour Windows Usage \$ 0.920	CPU Cores: 8, Memory: 14 GB, Disk Space Web: 2 TB, Disk Space VM Role: 1890 GB, Bandwidth: 800, Cost/Hour: \$ 0.96

In the example we assume that the user requests a VM with at least 2 GB memory, CPU Intel architecture, the maximum number of cores and that she/he wants to broker a best offer that does not exceed 0.8\$ per hour. The Cloud Agency adds the brokering rules to the SLA template produced by Semantic Engine, asks to vendors for available offers, brokers the best one and allows to close the transaction. Table 1 summarizes some of the available offers of the Amazon EC2 and Microsoft Azure cloud providers. Each cloud provider has an

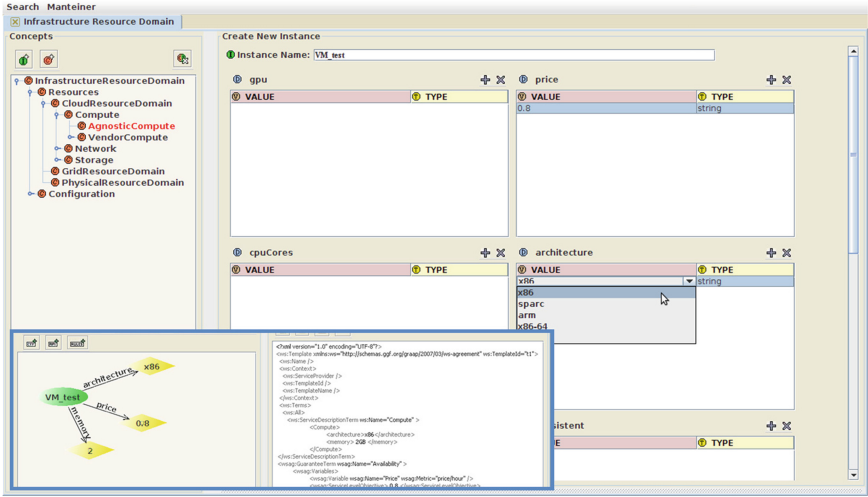


Fig. 7. SLA template graphical composition

offer consisting of several Virtual Machine configurations, which are different in cpu cores, available memory and price.

Vendor Agents of Amazon and Azure have to answer to the broker with their proposal that best fits the user's requirements. In this case Amazon VA will exclude the small offer because of its memory. Three offers remain, but the most powerful machine, compliant with the fixed price is the xlarge. Azure VA will exclude xsmall and small offers because of the memory requirement. Furthermore its xlarge offer is too much expensive. The selected offer eventually is the large one. Finally the broker will select the best price, i.e. Azure's offer. The presented example represents a basic application of a methodology, which is currently been developed, and in which we are considering not only price constraints but also factors like the capacity for each provider, the service levels that providers ensure and the trustworthiness of the provided measured using user's feedback and benchmarking report.

7 Related Works

Semantic and agent technology are being applied to the task of automated resource brokering in many areas, including cloud computing. In this field, the solutions provided are commonly oriented towards standardization.

The cloud service landscape is growing up very rapidly and there are different aspects of this evolution that need to be systematized in a formal way. A good means that can allow overcoming the limits related to heterogeneity of terms used by Cloud vendors are surely ontologies. Indeed for this reason a lot of ontologies related to cloud computing emerged. Darko et al. in [15] try to provide an overview of Cloud Computing ontologies, their types, applications

and focuses. They identified four main categories of cloud computing ontologies according to their scopes: Cloud resources and services description, Cloud security, Cloud interoperability and Cloud services discovery and selection. Among the classified ontologies, relevant to our work are ontologies used to discover and select the best Cloud service alternative. In [16] is presented a notable example of Cloud service discovery system based on matchmaking. In the presented system the users can identify the Cloud services required by means of three kinds of requirements: functional requirement (like programming language for PaaS service type), technical requirement (like CPU clock or RAM for IaaS service type) and cost requirement (like max price) as input parameters. In addition to this work, our ontology takes into account additional kind of requirements, such as the application category which is not considered in any other works present in the literature. In particular our approach promotes the Cloud agnostic principles of application development and covers both the design and application deployment part.

SLA@SOI [17] is the main project which aims at offering an open source based SLA management framework that will provide benefits of predictability, transparency and automation in an arbitrary service-oriented infrastructure, being compliant with the OCCI standard. SLA@SOI results are extremely interesting and offer a clear starting basis for the SLA provisioning and management in complex architectures.

In [18] an architecture is presented for a federated Cloud computing environment named InterCloud to support the scaling of applications across multiple vendor Clouds using a Cloud Broker for mediating between service consumers and Cloud coordinators for an allocation of resources that meets QoS needs of users.

Sim [19] proposes an extension of the alternate offers protocol that supports multiple complex negotiation activities in interrelated markets between user agents and broker agents, and between broker agents and provider agents.

In [20] is presented an architectural design of a framework capable of powering the brokerage based cloud services that is currently being developed in the scope of OPTIMIS, an EU FP7 project. In this model a broker is used to serve the needs of several different models. In particular it is used to ensure data confidentiality and integrity to service customers, to match the requirements of cloud consumer with the service provided by the provider, to negotiate with service consumers over SLAs, to maintain performance check on these SLA's and take actions against SLA violation, to effectively deploy services provided by the cloud provider to the customer, to manage the API so that provider does not learn anything about the identity of the service consumer, to securely transfer customer's data to the cloud, to enforce access control decisions uniformly across multiple clouds, to scale resources during load and provide effective staging and pooling services, to securely map identity and access management systems of the cloud provider and consumer, to analyze and take appropriate actions against risks, to handle cloud burst situations effectively. OPTIMIS introduces the problem and the architectural design, but we have not knowledge about an implementation or algorithms to achieve the brokering.

Tordsson [21] explores the heterogeneity of cloud providers, each one with a different infrastructure offer and pricing policy, in a cloud brokering approach that optimizes placement of virtual infrastructures across multiple clouds and also abstracts the deployment and management of infrastructure components in these clouds. Besides he presents a scheduling algorithm for cross-site deployment of applications. However he presents a fine grained interoperability of cloud services by way of a cloud API that do not takes into account the different implementation models for the virtual machine manager (VMM) that are at the base of each of the cloud providers infrastructure.

8 Conclusion

The support for brokering of service level agreement is a weakness in cloud market nowadays. The increasing number of Cloud providers, the lack of interoperability and the heterogeneity in current public Cloud platforms, leads to the need of innovative mechanisms to find the most appropriate Cloud resource configuration as easy and automated as possible. In this paper, which includes results of the mOSAIC project, we have shown how it is possible to build a complex brokering system, that is independent from the cloud provider technologies and allows the user to broker the best cloud service, that is compliant with his requirements. The proposed solution adopts two collaborative modules. The Semantic Engine, whose aim is to create an agnostic description of resource based on users' service requirements and a brokering system, the Cloud Agency, whose aim is to acquire autonomically resources from providers on the basis of SLA evaluation rules finding the most suitable Cloud provider that satisfy users' requirements. Recently we have investigated the chance of using a scalable broker as a service solution. We presented a prototype implementation and provided preliminary performance figures [22]. In future work we aim at improving the proposed solution, investigating mechanisms for dynamic filtering of the proposals.

Acknowledgments. The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007–2013) under grant agreement n 256910 (mOSAIC Project).

References

1. Amato, A., Venticinque, S.: Multi-objective decision support for brokering of cloud SLA. In: 27th International Conference on Advanced Information Networking and Applications Workshops, WAINA 2013, pp. 1241–1246 (2013)
2. Amato, A., Di Martino, B., Venticinque, S.: Evaluation and brokering of service level agreements for negotiation of cloud infrastructures. In: 7th International Conference for Internet Technology and Secured Transactions, ICITST 2012, London, United Kingdom, pp. 144–149 (2012)
3. mOSAIC. The mOSAIC Project. <http://www.mosaic-cloud.eu/>

4. Cretella, G., Di Martino, B.: Semantic and matchmaking technologies for discovering, mapping and aligning cloud providers services. In: Proceedings of the 15th International Conference on Information Integration and Web-based Applications and Services (iiWAS 2013), pp. 380–384 (2013)
5. Venticinque, S.: European research activities in cloud computing. In: Agent Based Services for Negotiation, Monitoring and Reconfiguration of Cloud Resources, pp. 178–202. Cambridge Scholars, January 2012
6. McGuinness, D.L., van Harmelen, F.: Owl web ontology language overview (2004). <http://www.w3.org/TR/2004/REC-owl-features-20040210/>
7. Open Grid Forum: Open Cloud Computing Interface (OCCI). <http://forge.ogf.org/sf/projects/occi-wg>
8. Metsch, T., Edmonds, A.: Open Cloud Computing Interface - Infrastructure, GFD-P-R.184, April 2011. <http://ogf.org/documents/GFD.184.pdf>
9. Di Martino, B., Cretella, G.: Towards a semantic engine for cloud applications development support. In: Proceedings of CISIS-2012: The Sixth International Conference on Complex, Intelligent, and Software Intensive Systems, July 4–6th 2012, Palermo, Italy. IEEE CS Press (2012)
10. Cretella, G., Di Martino, B., Stankovski, V.: Using the mosaics semantic engine to design and develop civil engineering cloud applications. In: Proceedings of 14th International Conference on Information Integration and Web-based Applications and Services (iiWAS 2012), p. 9. ACM (2012)
11. Venticinque, S., Tasquier, L., Di Martino, B.: Agents based cloud computing interface for resource provisioning and management. In: 2012 Sixth International Conference on Complex, Intelligent and Software Intensive Systems (CISIS), pp. 249–256, 4–6 July 2012
12. Amato, A., Tasquier, L., Copie, A.: Vendor agents for iaas cloud interoperability. In: Fortino, G., Badica, C., Malgeri, M., Unland, R. (eds.) IDC 2012. SCI, vol. 446, pp. 271–280. Springer, Heidelberg (2012)
13. Venticinque, S., Amato, A., Di Martino, B.: An OCCI compliant interface for IAAS provisioning and monitoring. In: CLOSER 2012 - Proceedings of the 2nd International Conference on Cloud Computing and Services Science, pp. 163–166 (2012)
14. Fipa, TC Communication. Fipa contract net interaction protocol (2002). <http://www.fipa.org>
15. Androcec, D., Vreck, N., Seva, J.: Cloud computing ontologies: a systematic review. In: MOPAS 2012, The Third International Conference on Models and Ontology-based Design of Protocols, Architectures and Services, pp. 9–14 (2012)
16. Han, T., Sim, K.M.: An ontology-enhanced cloud service discovery system. In: Proceedings of the International MultiConference of Engineers and Computer Scientists, vol. 1, pp. 17–19 (2010)
17. SLA@SOI, sla-at-soi.eu
18. Buyya, Rajkumar, Ranjan, Rajiv, Calheiros, Rodrigo N.: InterCloud: utility-oriented federation of cloud computing environments for scaling of application services. In: Hsu, Ching-Hsien, Yang, Laurence T., Park, Jong Hyuk, Yeo, Sang-Soo (eds.) ICA3PP 2010, Part I. LNCS, vol. 6081, pp. 13–31. Springer, Heidelberg (2010)
19. Sim, K.M.: Towards complex negotiation for cloud economy. In: Bellavista, P., Chang, R.-S., Chao, H.-C., Lin, S.-F., Sloot, P.M.A. (eds.) GPC 2010. LNCS, vol. 6104, pp. 395–406. Springer, Heidelberg (2010)

20. Nair, S.K., Porwal, S., Dimitrakos, T., Ferrer, A.J., Tordsson, J., Sharif, T., Sheridan, C., Rajarajan, M., Khan, A.U.: Towards secure cloud bursting, brokerage and aggregation. In: Proceedings of the 2010 Eighth IEEE European Conference on Web Services, pp. 189–196 (2010)
21. Tordsson, J., Montero, R.S., Moreno-Vozmediano, R., Llorente, I.M.: Cloud brokering mechanisms for optimized placement of virtual machines across multiple providers. *Future Gener. Comput. Syst.* **28**(2), 358–367 (2012). ISSN: 0167–739X
22. Amato, A., Di Martino, B., Venticinque, S.: Cloud Brokering as a Service. In: 3PGCIC 2013, pp. 9–16 (2013)

Intelligent Cloud Computing

First International Conference, ICC 2014, Muscat,

Oman, February 24-26, 2014, Revised Selected Papers

al Saidi, A.; Fleischer, R.; Maamar, Z.; Rana, O.F. (Eds.)

2015, X, 169 p. 66 illus., Softcover

ISBN: 978-3-319-19847-7