2013 27th International Conference on Advanced Information Networking and Applications Workshops

Semantic and Agent Technologies for Cloud Vendor Agnostic Resource Brokering

Alba Amato, Giuseppina Cretella, Beniamino Di Martino and Salvatore Venticinque
Department of Industrial and Information Engineering
Second University of Naples - Aversa, Italy
Email: alba.amato@unina2.it,giuseppina.cretella@unina2.it,beniamino.dimartino@unina.it, salvatore.venticinque@unina2.it

***Abstract*—The selection of Cloud providers, whose offers best fit the requirements of a particular application, is a complex issue due to the heterogeneity of the Cloud services, resources, technology and service levels offered by the several providers.**

In order to support this selection and requirements specification, we have developed, in the context of the mOSAIC project, 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 this paper, we illustrate how the two components, the Semantic Engine and the Cloud Agency, interoperate in order to support the Cloud Application Developer to express the requirements and services/resources in vendor agnostic way, to translate automatically these requirements into a neutral format in order to compare it with providers' offers and to broker the best one according to defined policies.

***Keywords*-Cloud Ontology, Semantic, Agents, Cloud provisioning, brokering**

I. Introduction

The choice of Cloud providers whose offers best fit the requirements of a particular application is a complex issue due to the problem of the heterogeneity of the services in terms of resources, technology and service levels that providers ensure.There are thousands of options which are described using metrics or properties which can change from one provider to another and there are some key factors to consider in terms of characteristics of the service, general terms and conditions of service and service levels that providers ensure. Besides sometimes users are not able to find the most suitable configuration of resources for their applications and to compare it with the several offers of the different providers in order to make an efficient choice taking into account not only the technical requirements, but also the providers rules and conditions. 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 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.

In order to support this selection and requirements specification, we have developed, in the context of the mOSAIC project [15], 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 this paper, we illustrate how the two components, the Semantic Engine and the Cloud Agency, interoperate in order to support the Cloud Application Developer to express the requirements and services/resources in vendor agnostic way, to translate automatically these requirements into a neutral format in order to compare it with providers' offers 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 knowldege is not helpful.

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

II. Related Works

Semantic and agent technology are being applied to the task of automated resource brokering in many areas,

978-0-7695-4952-1/13 $26.00 © 2013 IEEE
DOI 10.1109/WAINA.2013.163

including cloud computing. In this field, the solution provided are commonly oriented towards standardization. Many efforts to standardization have been done, in particular the Open Grid Forum [9] has defined an interface (the Open Cloud Computing Interface: OCCI [5] that offers a uniform access to IaaS resources. The Open Grid Forum uses UML as main formalism to define the interface while in this paper we provide resources' description, compliant with OCCI proposal, based on semantic web formalism, namely the *Ontology Web Language - OWL [11]*.

In [12] and [10] ontologies for Cloud domain are proposed. Instead the ontology proposed in this paper is focused on the infrastructure service layer, and aims at describing neutrally the resources offered by different cloud providers.

Thanks to the rapid diffusion of cloud computing an increasing number of initiatives and projects are currently running aimed at developing search engines for discovering Cloud services. In [6] [7] [8] Cloudle, a notable example of Cloud service discovery system based on matchmaking, is presented. In Cloudle users can identify the Cloud services queried 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. Instead our approach try to take advantages from inference rules trying to overlap the gap between high level requirements and technical requirements.

SLA@SOI [3] 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 [1] 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 [2] 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.

III. Objective and Methodology

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) a Knowledge Base, representing resources and domain concepts and rules by means of Semantic Web Ontologies and inference rules; (b) 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; (c) a Cloud Agency, that compares the different offers of providers with the user proposal and retrieves the best offer. In figure 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)* [4] 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.

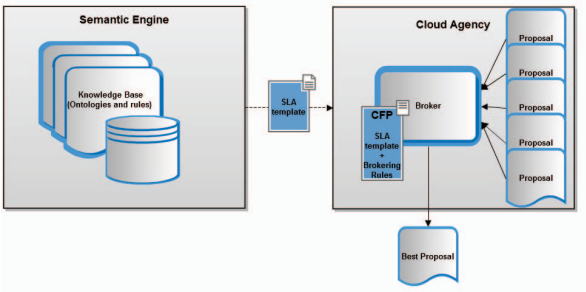

Figure 1. Integrated view of Knowledge Base, Semantic Engine and Cloud Agency

IV. An Ontology for IaaS

Starting from OCCI description of resource interface we offer a uniform way to represent the resources through an ontology. The language chosen for the ontology development is OWL that is the most used languages for ontology definition and is supported by several tools, libraries and reasoners. Our ontology is compliant with the OCCI resource description [5], which is summarized in table I

The ontology we built 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. The class hierarchy, shown in figure 2, is related to the Infrastructural Domain and 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. An example of how

Table I
OCCI RESOURCES ATTRIBUTE SUMMARY

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

a compute resource is represented is shown in figure 3. The vendors' offers of several IAAS cloud provider are represented in this ontology through individuals and their characteristics. 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 of the Semantic Engine and then can be translated in infrastructural requirements by the application of heuristic rules.

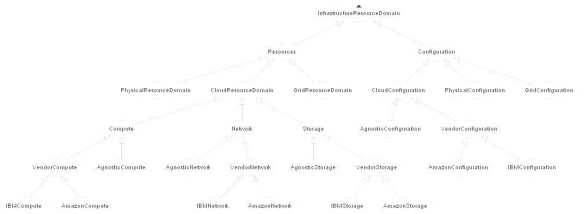

Figure 2. Ontology class hierarchy

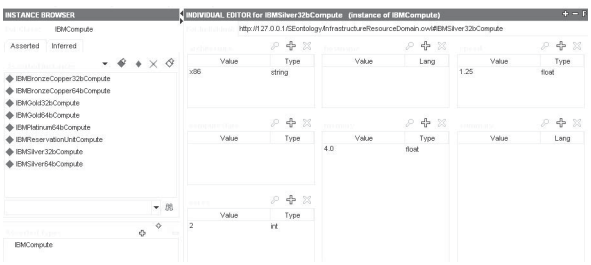

Figure 3. Example of single resource representation

V. SEMANTIC ENGINE

The Semantic Engine [13] 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 concepts, application related concepts, general design patterns and programming functionalities, specific API implementations and Cloud resources. It provides the user with fulltext, synonima augmented, search facility on the knowledge base, with graphical query composition, and inference reasoning facilities, by automatically generating inference rules, in order to select suitable API functionalities, components and Cloud resources suitable for developing the application. 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 [4] 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.

For instance the user can specify the application pattern of her/his application and for this class of applications she/he can specify particular characteristic related to the pattern. For example for a web application the user can specify some parameters like peak visitors per hour, average page views per visitor and average time per visit, and so the Semantic Engine, running heuristic rules defined in the knowledge base or defined by the user, can select the most suitable resource for the pattern described. Listed below is an example of simple rules 1 that links a type of application pattern with infrastructure resources.

Listing 1. Inference Rules

```
@prefix ApplicationPattern: http://127.0.0.1/
    SEontology/ApplicationPattern.owl#
@prefix InfrastructureResourceDomain: http:
    //127.0.0.1/SEontology/
    InfrastructureResourceDomain.owl#

[WebAppRule:
(?x rdf:type
    ApplicationPattern:WebApplication),
(?x ApplicationPattern:hasPeakVisitors ?y),
greaterThan(?y, 500),lessThan(?y, 1000),
(?x ApplicationPattern:maxResponseTime ?r),
greaterThan(?r, 1),lessThan(?r, 3),
(?z rdf:type
    InfrastructureResourceDomain:Compute),
(?z InfrastructureResourceDomain:memory ?k ),
greaterThan(?m, 8),lessThan(?m, 20),
(?z InfrastructureResourceDomain:cores ?k ),
```

```
greaterThan(?m, 8)
-> (?x
    ApplicationDomain:PatternUseInfrastructure
     ?y) ]
```

Running this rules the Semantic Engine can find an offer of IBM Cloud Provides, named Platinum, described in the knowledge base, and produces the part of the CFP listed below 2.

Listing 2. CFP produced from rules

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

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 constrains. 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.

VI. Cloud Agency

Cloud Agency (CA)[4] is a Multi Agent System conceived for provisioning by negotiation, monitoring and reconfiguration of acquired resources. 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. In this paper we will focus our attention on resource provisioning.

The configuration of resources that are necessary to execute the users 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 3 an example of SLA template is shown. It contains service description terms and garantee terms in WS-Agreement. The requested resource is a Virtual Machine configuration with an architecture x86, 2 Cores, 2Gb of available memory and a price not greater than 0.8$.

Listing 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>
    </wasg: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 Figure 4.

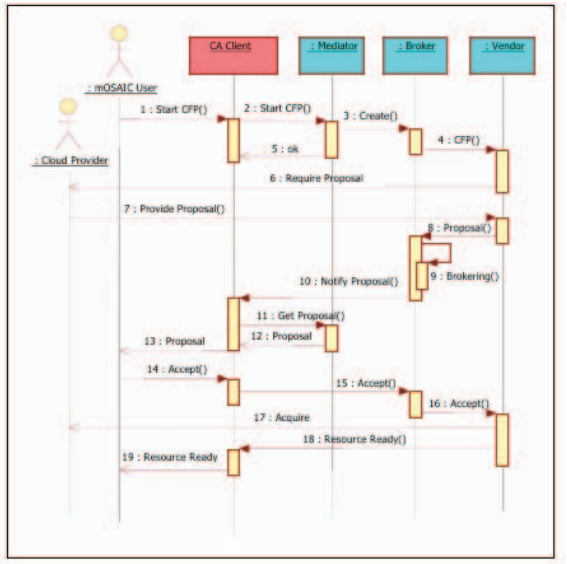

Figure 4. Interaction among agents for resource provisioning

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 an 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.

Table II
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

VII. CASE STUDY

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 happens 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. 5, to express the resources' requirements and then to automatically translate them into the SLA template.

Figure 5. SLA template graphical composition

In the example we assume that the user requests a VM with at least 2GB 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, brokes the best one and allows to close the transaction. Table II summarizes some of the available offers of the Amazon EC2 and Microsoft Azure cloud providers. Each cloud provider has an 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.

VIII. CONCLUSION

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 we described the architecture design of a Cloud service composed by 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. In future work we will investigate mechanisms for dynamic filtering the proposals and more complex models of SLA negotation. The brokering methodology is currently being developed, and we are considering factors like the capacity for each provider, the service levels that providers ensure and the trustworthiness of the provided measures using user's feedback and benchmarking reports.

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] Buyya, R., Ranjan,R.,Calheiros R.N.: InterCloud: Utility-Oriented Federation of Cloud Computing Environments for Scaling of Application Services, In: ICA3PP 2010, pp. 13-31.

[2] Sim, K. M.,Towards Complex Negotiation for Cloud Economy, 5th International Conference on Advances in Grid and Pervasive Computing (GPC 2010),pp. 395-406.

[3] sla-at-soi.eu

[4] Venticinque, S., European Research Activities in Cloud Computing, Cambridge Scholars, January 2012, ch. Agent Based Services for Negotiation, Monitoring and Reconfiguration of Cloud Resources, pp. 178202.

[5] T. Metsch and A. Edmonds: Open Cloud Computing Interface - Infrastructure, GFD-P-R.184, April 2011. Available: http://ogf.org/documents/GFD.184.pdf

[6] T. Han and K. M. Sim, Cloudle: An ontology-enhanced cloud service search engine in WISE Workshops, ser. Lecture Notes in Computer Science, D. K. W. Chiu, L. Bellatreche, H. Sasaki, H. fung Leung, S.-C. Cheung, H. Hu, and J. Shao, Eds., vol. 6724. Springer, 2010, pp. 416-427.

[7] Kang, J., Sim, K.M.: Cloudle: A Multi-criteria cloud service search engine. To appear in the Proceedings of the 2010 IEEE Asia-Pacific Services Computing Conference, Hangzhou, China, December 6-10 (2010).

[8] T. Han, K. M. Sim: An Ontology-enhanced Cloud Service Discovery System. Proceedings of the International Multi-Conference of Engineers and Computer Scientists 2010 Vol I, March 2010.

[9] Open Grid Forum: Open Cloud Computing Interface (OCCI). http://forge.ogf.org/sf/projects/occi-wg

[10] Youseff, L., Butrico, M., and Da Silva, D. (2008): Toward a unified ontology of cloud computing. In Grid Computing Environments Workshop, 2008. GCE 08, pages 110.

[11] Deborah L. McGuinness and Frank van Harmelen. Owl web ontology language overview. http://www.w3.org/TR/2004/REC-owl-features-20040210/, 2004.

[12] Francesco Moscato, Rocco Aversa, Beniamino Di Martino, Dana Petcu, Massimiliano Rak, and Salvatore Venticinque, "'An Ontology for the Cloud in mOSAIC"', in: Lizhe Wang, Rajiv Ranjan, Jinjun Chen, Boualem Benatallah (Eds.), Cloud computing: methodology, system, and applications, Chapter , pp. 467-487, CRC press - Taylor e Francis group, 2011 (ISBN 978-1-4398-5641-3).

[13] Beniamino Di Martino, Giuseppina Cretella, "Towards a Semantic Engine for Cloud Applications Development Support", Proc. of *CISIS-2012: The Sixth International Conference on Complex, Intelligent, and Software Intensive Systems*, July 4-6th 2012, Palermo, Italy, IEEE CS Press, 2012.

[14] FIPA TC Communication. Fipa contract net interaction protocol, 2002. Available at http://www.fipa.org.

[15] mOSAIC. The mOSAIC Project. Available at http://www.mosaic-cloud.eu/.