Cloud Description Ontology for Service Discovery and Selection

Molka Rekik[1], Khouloud Boukadi[1] and Hanêne Ben-Abdallah[1,2]
[1]*Mir@cl Laboratory, Sfax University, Sfax, Tunisia*
[2]*King Abdulaziz University, Jeddah, K.S.A.*

Keywords: Ontology, Cloud, Services, Functional Properties, Non-functional Properties, Discovery, Selection.

Abstract: Cloud computing is a model delivery of infrastructure, platform and software services over the net. The lack of standardization of heterogeneous Cloud service descriptions makes service discovery and selection very complex tasks for Cloud users. To ease this complexity, it is crucial to describe the various Cloud service pertinent information in a homogeneous model. To provide for such model, this paper offers to contributions. First, it proposes a Cloud description ontology that integrates service descriptions obtained from heterogeneous sources. The ontology model accounts for the functional and non-functional properties, attributes and relations of infrastructure, platform and software services in order to facilitate the discovery and selection of suitable Cloud services. Second, this paper presents a qualitative and quantitative evaluation of the proposed ontology. For the qualitative evaluation, we use a reasoner to identify defects in the ontology, whilst for the quantitative evaluation we compare our search results with those obtained through a web search engine.

1 INTRODUCTION

Cloud computing technology can be viewed as a set of software services (SaaS), platform services (PaaS) and infrastructure services (IaaS) offered to users over the net. It has the advantage of offering users scalable and elastic service capabilities, at a reduced cost, speeding up the deployment of their businesses. Indeed, in the SaaS model, a user gets the provider applications that are hosted and run in cloud infrastructure; in the PaaS model, the user gets the platform that is running in the cloud infrastructure for creating or deploying the applications; and in the IaaS model, the user gets the virtual resources in which an execution environment can be deployed to run an application.

A Cloud federation is a new paradigm where a set of Cloud providers are grouped in order to offer a wider range of resources/services and, hence, enlarge their market shares (Rochwerger et al., 2009), (Buyya et al., 2010). However, Cloud federations are hindered by the lack of a standard language for describing providers' offers making it difficult for end-users to discover and select an appropriate provider within a federation. In fact, service providers describe similar services with different manners (different pricing policies, different quality of services, etc.); such heterogeneous descriptions complicates service comparison and selection, and may create interoperability problems among multiple providers (especially when migrating from a provider to another). To overcome the problems stemming from heterogeneous service descriptions within a Cloud federation, we propose to define a common ontology for Cloud service description. This ontology is especially useful when users want to search for domain specific information (Ruthven and Lalmas, 2003), (Hoang and Tjoa, 2006), (Bhogal et al., 2007). It is a means to share common understanding about the structure of Cloud service descriptions while covering all Cloud service layers (IaaS, PaaS and SaaS), and to represent semantically and structurally Cloud providers' information.

It is worth noting that, there exists some industrial standards such as (NIST, 2013), (Forum, 2010), (Behrendt et al., 2011) and research projects such as (Beom et al., 2011), (Grozev and Buyya, 2012), (Moscato et al., 2011), (Youseff et al., 2008), (Zhang et al., 2012) that treat the aforementioned problems; unfortunately, current propositions focus only on the IaaS layer: enabling unified access to existing IaaS services to deal with the heterogeneity of the Clouds. That is, they do not offer solutions to the PaaS and the SaaS layers. In fact, our work is the first to treat the three layers together. It relies on existing standards, Cloud providers' catalogs and some research projects to define an ontology that accounts for a set of functional and non-functional configuration properties. The ontology provides for service selection that:

Rekik M., Boukadi K. and Ben-abdallah H..
Cloud Description Ontology for Service Discovery and Selection.
DOI: 10.5220/0005556400260036
In *Proceedings of the 10th International Conference on Software Engineering and Applications* (ICSOFT-EA-2015), pages 26-36
ISBN: 978-989-758-114-4

in the IaaS layer, uses the characteristics of the virtual machine (VM) (*e.g.*, the CPU type, the memory size, the operating system, etc.); in the PaaS layer, refers to the characteristics of the platform (*e.g.*, programming language, libraries, databases, etc.); and in the SaaS, deals with the application characteristics (*e.g.*, usability, scalability, multi-tenancy, etc.).

The rest of the paper is organized as follows. Section 2 describes works pertinent to ontology for Cloud computing. In Section 3, we present our ontology to describe Cloud services, which we evaluate quantitatively and qualitatively in Section 4. Finally, in Section 5, we summarize the status of the presented work and outline its extensions.

2 ONTOLOGY USAGE IN CLOUD COMPUTING

In order to standardize the representation of Cloud computing services, various works use an ontology. In (Youseff et al., 2008), the authors establish the knowledge domain of the area of Cloud computing and its relevant components. They discuss each layer strengths, limitations as well as dependence on other layers. In addition, they define a general taxonomy that captures the inter-relations between the different Cloud components. This work facilitates the understanding of the Cloud environment and its key concepts. The identified concepts cover the three Cloud layers and they are defined and generated from standards. However, the informal description of the concepts in this work does not provide for its direct application in a selection method. This motivated several propositions to formalize the Cloud concepts in an ontology that facilitates service discovery.

As Gruber (Gruber, 1995), *the ontology provides an explicit specification of a conceptualization*. So, an ontology is a language used for representing formally the available information of a specific domain providing a hierarchy of concepts which can be manipulated by the users.

In (Parhi et al., 2014), the authors propose a semantic-based service description and discovery framework using the multi-agent approach. The framework mainly assists users to discover suitable service on-demand based on a Cloud service description ontology. Adopting a similar approach, the authors in (Ghoneim and Tolba, 2014) propose a framework composed of two main components: controller and cloud ontology blackboard environment. Both components use an ontology to discover the best suited set of responses for the consumer requests. In a similar way, in (Bernstein and Vij, 2010), the authors define an ontology based catalogs that describes features and capabilities of resources offered by Cloud providers, such as CPU, storage, security, and compliance capabilities. The main aim of this work is to provide a federated Cloud environment. Also, the authors in (Beom et al., 2011) propose an ontology based resource management for Cloud Computing. They select the best Cloud resources based on the objective of the Service Level Agreement (SLA). Overall, the aforementioned mentioned ontologies support only a few number of concepts in the Cloud IaaS level. In addition, they do not define the non-functional elements which are important information during the selection process to satisfy user requirements.

In (Zhang et al., 2012), the authors propose the CoCoOn ontology for describing Cloud infrastructure services while detailing the functional and non-functional properties. Using CoCoOn, they implement a service discovery system to search for available infrastructure services. The search method relies on the typical QoS criteria by adopting a Web service approach. In addition, because CoCoOn covers only the IaaS level concepts, the search considers only the virtual machines as Cloud resources.

An interesting initiative in this area is proposed by the mOSAIC project (Moscato et al., 2011). The project analyzes existing standards and proposes an ontology for semantic retrieval and composition of Cloud services. It provides a platform and proposes a set of APIs for solving multiple Clouds interoperability problems. In addition, it treats the functional and non-functional properties. However, the ontology proposed in mOSAIC considers the IaaS Cloud layer and neglects the other Cloud service layers (SaaS and PaaS).

It is worth mentioning that, in all of the above mentioned published works, the authors do not show an evaluation to improve the use of their proposed ontologies.

In the herein presented work, we build upon the above mentioned works to propose a comprehensive Cloud service description ontology that serves as a basis for the discovery and selection of Cloud providers and services in the context of a Cloud federation. More specifically, we propose an ontology that covers both functional and non-functional aspects of Cloud services, and at the three layers of Cloud (IaaS, PaaS and SaaS). In addition, we evaluate our proposed work, first, by using a reasoner to examine its logical or conceptual coherence and, secondly, by comparing the search results it offers with those of Web search engine.

3 CLOUD DESCRIPTION ONTOLOGY

The work presented in this paper is part of an ongoing project where a Cloud service description ontology is the building block for the scheduling of business processes in a Cloud federation environment *(i.e., the task-resource allocation)*. In our previous work (Rekik et al., 2014), we presented a context-based categorization in order to trigger the on-demand resource provisioning across multiple infrastructures providers. We defined a task-related context, user-related context and resource-related context. This paper presents a Cloud service description ontology to model the resource-related context (*i.e.*, the resource-related context is explicitly defined by Cloud providers, it is mainly related to the resource/service's characteristics. Generally, this information models the competitive advantage that Cloud providers may have over one another) in order to facilitate service (resource) selection from a set of available ones.

To construct the ontology, we proceeded in two steps. First, we identified its concepts by analyzing standards and proposals from the literature. Secondly, we followed the design principles, presented on (Gruber, 1995), which are objective criteria for guiding and evaluating ontology designs; they include clarity, minimal encoding bias, extendability, coherence and minimal ontological commitments. In addition to these principles, we followed name standardization which, as suggested in (Cesar et al., 1998), eases the understanding of the ontology. Following these principles, we obtained the concepts shown in Figure 1 and which we detail next. To create the ontology, we used the protègè[1] editor which is the most widely used, freely available, platform independent technology for developing and managing terminologies, ontologies and knowledge bases. It is written in Java and uses Swings to create the complex user interface.

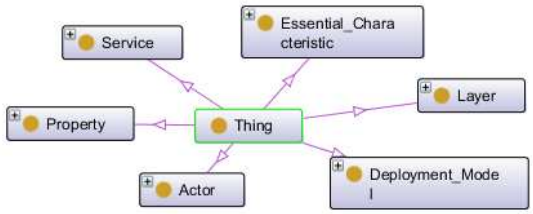

Figure 1: The Top Level Concepts.

3.1 The Actor Class

The Actor class presents the different actors which participate in Cloud systems. We distinguish between:

- According to the mOSAIC (Moscato et al., 2011), the Developer which develops the Cloud services and the Administrator which configures them.
- As NIST (NIST, 2013), the Broker which monitors, negotiates, schedules, etc. these Cloud services.
- As IBM (Behrendt et al., 2011), the Consumer which uses the service and the Provider which offers/has the Cloud service. The NIST (NIST, 2013) classifies the consumer as IaaS, PaaS and SaaS_Consumer.
- According to the NIST (NIST, 2013), the PaaS_Consumer can be an Application_Developer who designs and implements services (application software), an Application_Admistration who configures and monitors the performances, an Application_Deployer who publishes services into the Cloud system and an Application_Tester who runs and tests services. In similar, we classify three types of providers namely IaaS, PaaS and SaaS_Provider.

3.2 The Deployment Model Class

Each provider follows/has a deployment model. By referring to NIST (NIST, 2013), the Deployment_Model class includes Private, Public, Hybrid, Community and Federation sub-classes. In private model, the services are provisioned for exclusive use by a single organization. In community model, the services are provisioned for exclusive use by a specific community of consumers from organizations that have shared concerns. The services are provisioned for open use by the general public in the public model. The hybrid model is a composition of two or more distinct Cloud models (private, community, or public).

In our work, we have extended this class by adding the federation subclass (Figure 2). In the federation model, the services are provisioned by a group of Cloud providers that voluntarily collaborate with each other to exchange resources (Grozev and Buyya, 2012). The federation model has a type and an architecture. The federation type can be vertical which spans multiple levels, or horizontal which takes place on one level of the cloud stack. The federation architecture can be installed with broker (centralized) where there is a central broker that performs and facilitates the resource allocation or without broker (decentralized) where the cloud providers communicate and negotiate directly with each other without mediators like the peer to peer architecture.

[1] http://protege.stanford.edu/

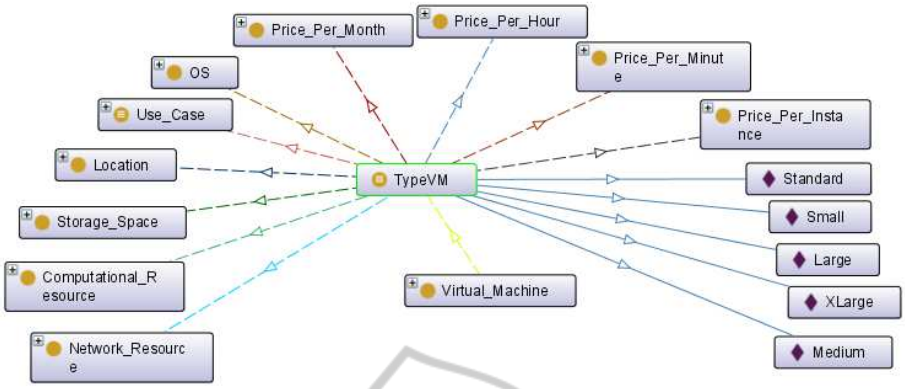

Figure 3: The Virtual Machine's Characteristics.

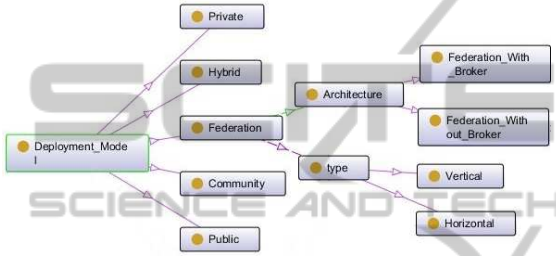

Figure 2: The Deployment Models.

3.3 The Essential Characteristics Class

Each provider has an essential characteristics defined by the NIST standard (NIST, 2013). In fact, in our work, the Essential_Characteristics class includes the following sub-classes.

- On_Demand_Self_Service: The provider offer computing capabilities, such as server, network and storage as needed automatically without requiring human interaction.
- Broad_Network_Access: The resources' capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms.
- Resource_Pooling: The resources are pooled to serve multiple consumers using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand.
- Measured_Service: The Cloud systems automatically leverage a metering capability at some level of abstraction appropriate to the type of service.
- Rapid_Elasticity: The resources' capabilities can be elastically provisioned and released to scale rapidly outward and inward commensurate with demand.

3.4 The Service Class

The NIST (NIST, 2013) defines the services offered by providers as SaaS, PaaS and IaaS. Each provider offers a service that is in reality a virtualized resource. That is, an IaaS provider offers either a virtual machine or a storage space, a PaaS provider offers a platform, whereas a SaaS provider offers a Software or application.

1. IaaS_Service:
   - Virtual_Machine class: By accessing to IaaS providers' catalogs (Cloud, 2013), (Henry, 2009), (Russell and Cohn, 2012), we notice that a VM has a location and a type. Depending to the type, we distinguish the following VM's characteristics a(shown in Figure 3:
   - Network_Resource: The bandwidth is a network resource which has a Delay, Throughput, Protocol and Price_Per_Data_Transfer.
   - Storage_Space: The Memory is the storage space which has Size and Allocation_Strategy.
   - Computational_Resource: The CPU is a computational resource. It has Core(s), Frequency and Architecture.
   - OS: The operating system of a VM.
   - Use_Case: While examining some Cloud providers (Cloud, 2013), (Henry, 2009), we notice that a VM has a use case such as Optimized_Memory, Optimized_Calculation and General_Use.
   - Price_Per_Hour: The VM's allocation price per hour.
   - Price_Per_Instance: The VM's allocation price per instance.

   Just to note here that, first, the relation between the VM, Location and Type classes must be a part of relation. Indeed, wherever Type and Location exist, it is as part of VM, and their pres-

ence implies the presence of the VM. Similarly for Type class and its all sub-classes. To describe a VM, we categorize functional and non-functional properties which are defined in Subsection 3.5 (Figure 4).

Figure 4: The Virtual Machine Description.

- Storage_Space: A Storage space can be provided with or without a VM. This class covers Distributed_File_System (is a file system that allows access to files across the multiple hosts within the Cloud), Replicated_Relational_Database (is a distributed database which allows the copy of data from a database in one resource to a database in another) and Disk sub-classes. Like the Type (of VM) class, these sub-classes are part of Storage_Space class. The storage space has a Price_Per_Data_Size, Storage_Format, Storage_Size_Min (the minimum storage capacity) and Storage_Size_Max (the maximum storage capacity) (shown in Figure 5).

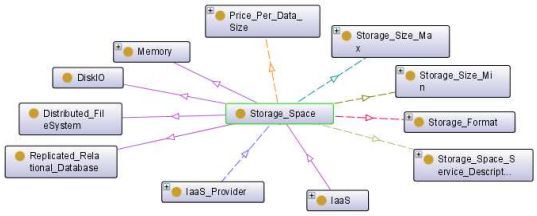

Figure 5: The Storage Space Class.

2. PaaS_Service: The Platform class includes IDE (an integrated development environment), API (an Application Programming Interface), Application_Server (an environment where applications can run), Web_Server (server used to publish websites) and Database (an organized collection of data) subclasses. These all cited sub-classes are part of Platform class. Like the IaaS resource, the PaaS resource has a Price_Per_Hour and PaaS_Service_Description which includes the functional and the non-functional properties. These properties will be detailed in Subsection 3.5.
3. SaaS_Service: The Software has a Price_Per_Hour, Price_Per_License, Price_Per_Instruction, Price_Per_User_Number and a SaaS_Service_Description which includes a functional and non-functional properties which are detailed in Subsection 3.5.

3.5 The Property Class

The Property subclasses contain all elements needed for describing characteristics of Cloud resources. We distinguish the Functional_Property and the Non_Functional_Property subclasses. The first defines the service function (i.e., what a service is supposed to accomplish) while the second specifies the service quality or criteria. For each service provided (IaaS, PaaS and SaaS), we define a set of functional and non-functional properties.

3.5.1 The Functional Property Class

1. The Functional_IaaS_Property class: this class includes the Functional_VM_Property and the Functional_StorageSpace_Property sub-classes.
   (a) The Functional_VM_Property class: in this class, we classify the following subclasses (shown in Figure 6):
   - The Virtualization class: it is a mechanism for operating multiple systems, servers, or applications on the same physical server as a virtual machine. This class covers the Full-Virtualization (virtualization of the OS without any modification) and the Para-Virtualization (virtualization of the OS with modification).
   - VM_Management class: it covers Backup, Recovery and Scheduling. The first consists on creating data images to storage while the second consists in recovering data in disasters or failures. The scheduling is the allocation of cloud resources to consumers.
   - VM_Monitoring class: it detects slow or failing components, such as overloaded or failed VMs.
   - Cloud_Federation class: it allows a set of Cloud providers to federate together.
   - Replication class: it involves sharing information to ensure consistency between redundant resources.
   - Migration class: it consists in moving from the VM overloaded to another underloaded.

   (b) The Functional_StorageSpace_Property class: such as the Functional_VM_Property class, this class covers the Virtualization, StorageSpace_Management, StorageSpace_Monitoring, Cloud_Federation, Replication and Migration sub-classes.

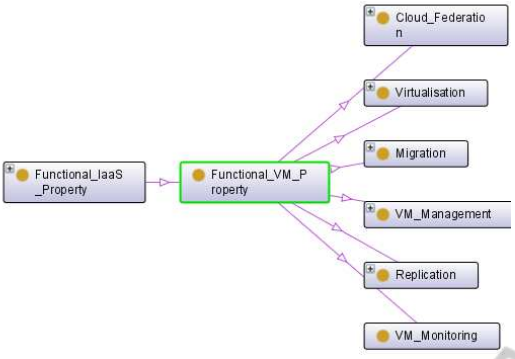

Figure 6: The Virtual Machine Functional Property.

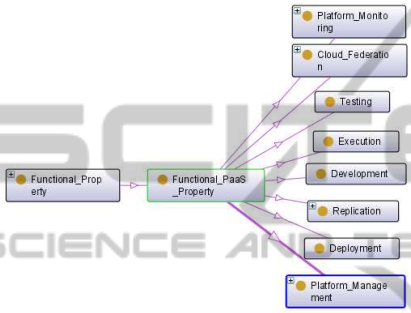

Figure 7: The PaaS Functional Property.

2. The Functional_PaaS_Property class: its subclasses are described in Figure 7. We distinguish the Deployment, the Development, the Testing, the Execution, the Replication, the Cloud_Federation, the Platform_Management which includes the Database and Application_Management and the Platform_Monitoring subclass which includes Application_Monitoring and Database_Monitoring.

3. The Functional_SaaS_Property class: we define the following sub-classes (Figure 8): ERP (is an Enterprise Resource Planning system which automates and tracks a variety of business functions across different departments of an organization), CRM (is a Customer Relationship Management which provides good customer management, opportunities and altogether better customer support), Humain_Capital_Management (it evolves and automates processes such as payroll, performance assessments, recruitment and training), Collaborative_Tools (it allows users to collaborate and to work in groups within and between companies, Content_Management (it is a process enabling company, agency or organization to obtain, organize, store and deliver information essential to its operation in the most efficient manner), Mailing (it is a mail management

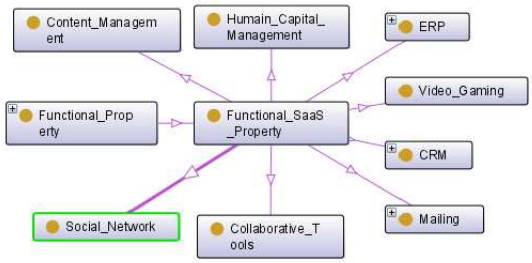

Figure 8: The SaaS Functional Property.

application), Social_Network (a social application that enables users to interact and share data) and Video_Gaming (an interactive application that is used for entertainment, role playing and simulation).

3.5.2 The Non Functional Property Class

Basically, non-functional requirements relate to the qualities of the service (QoS) that cut across user facing features, such as security, reliability, availability, etc. In this work, we present the QoS properties while referring on the ISO-9126 (ISO/IEC, 2001) and non-functional properties which are specific to each service (namely IaaS, PaaS and SaaS).

1. The Non_Functional_IaaS_Property class: this class includes the Non_Functional_VM_Property and the Non_Functional_StorageSpace_Property subclasses.

   (a) The Non_Functional_VM_Property class: in this class, we classify the following subclasses (shown in Figure 9):

   - VM_Auto_Scaling class: In (Alhamad et al., 2010), the elastic provisioning is defined as a concept in which computing resources can be scaled up and down easily by the cloud service provider. According to this definition, we add the following subclasses: VM_Scale_Up (the maximum number of VMs provided for one user) and VM_Scale_Down (the minimum number of VMs provided for one user).
   - Processing_Speed class: the processing speed of the CPU.
   - Data_Transfer_Speed class: the processing speed of the network.
   - QoS_IaaS class: it includes the Integrity (the assurance that the information can be accessed or modified only by those authorized to do), Security, Flexibility (the ability to simply and instantly manage cloud services), Compliance (the services are conform to standards), Fault_Tolerance (the ability to manage auto-

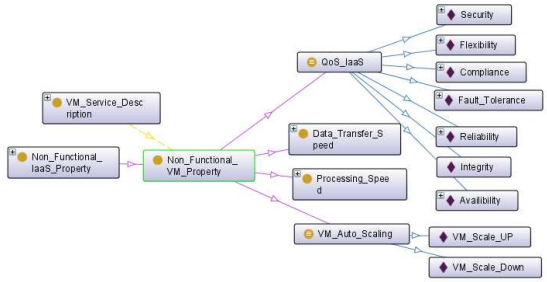

Figure 9: The Non-Functional Virtual Machine Properties.

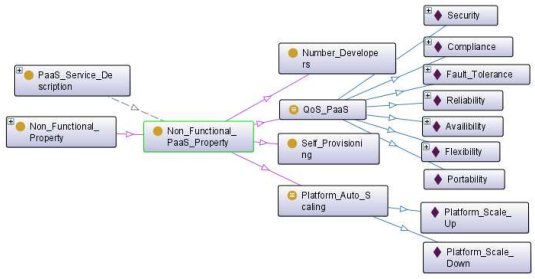

Figure 11: The Non-Functional PaaS Properties.

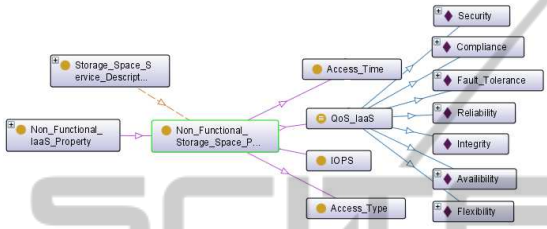

Figure 10: The Non-Functional Storage Space Properties.

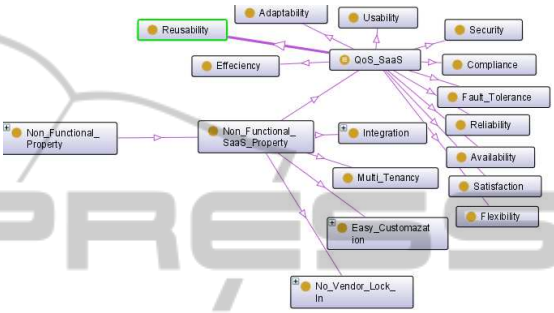

Figure 12: The Non-Functional SaaS Properties.

mated way component failures) and Availability (the services are accessible).

(b) The Non_Functional_StorageSpace_Property class: in this class, we classify the following sub-classes (shown in Figure 10):

- Access_Type class: it indicates the access type of information that can be either read and/or written.
- Access_Time class: it describes the information storage time.
- IOPS class: it includes the input and output operations per second.
- QoS_IaaS class: it includes the Integrity, Security, Flexibility, Compliance, Fault_Tolerance and Availability properties.

2. The Non_Functional_PaaS_Property class: this class includes Number_Developers, Language_Of_Development, Platform_Type, Platform_Architecture and QoS_PaaS which includes the Portability (the usability of the same application in different environments), Security, Flexibility (the ability to adjust in needs), Compliance, Fault_Tolerance (the ability to continue when a resource failure occurs) and Availability properties (Figure 11).
3. The Non_Functional_SaaS_Property class: referring to (KPMG, 2013), the Multi_Tenancy (multiple users can simultaneously access the same service), the Integration (integrate the data of users in the SaaS), the No_Vendor_Lock_In, the Easy_Customization (each user can easily customize applications to fit their business processes without affecting the common infrastructure), the No_Large_Up_Front_Investment (allowing customers to avoid the large initial investment in IT infrastructure), the Contract_Length (the period that the software product is legally available for a client) and the QoS_SaaS which includes the Efficiency (the ability to accomplish a job with an optimal performance), Satisfaction, Usability, Adaptability (the ability to adjust in needs), Security, Flexibility, Compliance, Fault_Tolerance and Availability properties (Figure 12).

3.6 Instances Creation

In this subsection, we associate real world instances to the proposed concepts by referring to a set of Cloud providers' catalogs available on the Internet (such as Amazon EC2[2], Rackspace[3], HP Cloud[4], etc.). We present an extract of instances related to the IaaS, PaaS and SaaS services while focusing on some of functional and non-functional properties (Tables 1, 2, 3).

4 ONTOLOGY EVALUATION

Once constructed, the ontology needed to be evalu-

[2]aws.amazon.com
[3]www.rackspace.com
[4]www.hpcloud.com

Table 1: An Extract of IaaS Individuals.

Domain	Object Property	Range
AmazonEC2	hasIaaSService	VM11, ...
MicrosoftAzure	hasIaaSService	VM21, ...
...	...	...
VM11	hasType	XLarge
	hasPricePerHour	0.64$
	hasMemory	16GO
	hasFrequency	2.7GHz
	hasNetwork	50Gbps
	hasLocation	Asia_Pacific
	hasUseCase	General_Use
	hasNonFunctionalVMProperty	Confidentiality, Migration, ...
VM21	hasType	Small
	hasPricePerHour	0.055$
	hasMemory	2GO
	hasFrequency	2.5GHz
	hasNetwork	20Gbps
	hasLocation	US_West
	hasUseCase	General_Use
	hasNonFunctionalVMProperty	Auto_Scaling, Cloud_Federation, ...
...	...	...

Table 2: An Extract of PaaS Individuals.

Domain	Object Property	Range
GoogleAppEngine	hasPaaSService	Platform11, ...
CloudFoundry	hasPaaSService	Platform21, ...
...	...	...
Platform11	hasLanguage	Java
	hasDBMS	Oracle
	hasPricePerHour	0.7$
	hasLocation	Asia_Pacific
	hasNonFunctionalPaaSProperty	Security, Integrity, Fault_Tolerance
Platform21	hasLanguage	Python
	hasDBMS	MySQL
	hasPricePerHour	0.4$
	hasLocation	Europe
	hasNonFunctionalPaaSProperty	Cloud_Federation, Security, Integrity, Consistency
...	...	...

ated in terms of its qualitative and quantitative aspects. The qualitative evaluation treats the logical or conceptual coherence while the quantitative evaluation is based on some structural criteria (Pradel et al., 2012).

1. Qualitative Evaluation: Such evaluation tends to produce a set of properties that reflect the respect for quality criteria. In our work, we are referring to the following criteria defined by Staab et al. in (Staab and Studer, 2004) to improve the qualitative evaluation of our proposed ontology:
   - Duplication: checks if the elements that can be deducted needn't to be explicitly mentioned,
   - Disjunction: checks if the class is the conjunction of disjoint classes,
   - Consistency: checks if the definition of a class does not lead to a contradiction.

Pellet (Evren et al., 2007), Fact++ (Tsarkov and Horrocks, 2006) and HermiT (Shearer et al., 2008) are reasoners which allow the evaluation of class hierarchy, object property hierarchy, data property hierarchy, class assertions, object property assertions and same individuals. In this work, to validate our ontology, we

Table 3: An Extract of SaaS Individuals.

Domain	Object Property	Range
SalesForce	hasSaaSService	Software11, ...
DoliCloud	hasPaaSService	Software21, ...
...	...	...
Software11	hasCRM hasPricePerHour hasLocation hasNonFunctionalSaaSProperty	SugarCRM 2.5$ Asia_Pacific Multi_Tenancy, Reusability
Software21	hasERP hasPricePerHour hasLocation hasNonFunctionalSaaSProperty	ERP5 4.5$ Europe Multi_Tenancy, Privacy, Availability
...	...	...

Table 4: Users' Query Examples with Using Ontology and Google.

Query	Using the Proposed Ontology	Using the Google Engine
Q1	IaaS_Provider and hasService value Virtual_Machine and hasPricePerHour max 1$ and hasMemory min 7GO and hasNonFunctionalVMProperty value "Auto_Scaling" ^^string	IaaS provider +price per hour<=1$ +auto scaling +memory>=7GO
Q2	PaaS_Provider and hasService value Platform and hasPricePerHour max 1.5$ and hasDBMS value Oracle and hasLanguage value Java and hasNonFunctionalVMProperty value "Confidentiality" ^^string	PaaS provider +Oracle+Java +price per hour<=1.5$ +confidentiality
Q3	SaaS_Provider and hasService value Software and hasPricePerHour max 4.5$ and hasERP value ERP and hasNonFunctionalVMProperty value "Multi_Tenancy" ^^string	SaaS provider +ERP5+multi-tenancy +price per hour<=4.5$

choose the Fact++ reasoner which is an open source, free and available with protègè. So, while applying the reasoner, we remark there exist inconsistency errors. So, we mustn't define the Non_Functional_IaaS_Property, the Non_Functional_PaaS_Property and the Non_Functional_SaaS_Property classes as disjoint classes in order to obtain a coherent and valid Cloud description ontology. Since there exist QoS individuals which are ISO properties. These properties describe the non-functional IaaS services, the non-functional PaaS services and the non-functional SaaS services.

2. Quantitative Evaluation: The majority of existing works, evaluate their proposed ontologies while comparing them with a reference ontology or with a domain corpus which includes a set of ontologies and providing a set of similarity measures. However, in our case, there do not exists a standard or open ontology to refer to it to it comparing with ours. So, we present users with our ontology and ask them to give their Cloud service queries in order to evaluate its pertinence. Indeed, we have interviewed three types of users. The first one is an IaaS user who asks for IaaS providers which offer a virtual machine that has a price per hour at maximum 1$, a memory capacity at minimum equal to 7GO and guarantee the auto-scaling non-functional property (query Q1). The second is a PaaS user who requests PaaS providers which offer a platform that has a price per hour at maximum 1.5$, an Oracle database, the Java as a language of development and guarantee the confidentiality non-functional property (query Q2). The third user is a SaaS user who asks for SaaS providers which specify an ERP5 as software, a location at Europe zone and offer at least the multi-tenancy non functional requirement (query Q3). Then, we compare the results obtained while interrogating the ontology and the web search engine Google[5] with users' queries.

Table 4 describes the queries written in DL-Query language (Horridge and Drummond, 2008) which is a plugin for protègè to make it possible for ex-

[5]http://www.google.com/

Table 5: Query Results with Using Ontology and Google.

Query	Ontology Results	Google Results	Correct Google Results	Correct Ontology Results
Q1	10	13000	1	10
Q2	7	705000	0	7
Q3	5	1280	0	5

Table 6: Results Correctness with Using Ontology and Google.

Query	Google Correctness	Ontology Correctness
Q1	0.008%	100%
Q2	0%	100%
Q3	0%	100%

posure users' queries using the proposed ontology. We take also these examples to find results obtained by the Google web search engine (*i.e.*, without using ontology). When interrogating the ontology and the Google web search engine with the users' queries in order to select appropriate Cloud service provider, we remark that the results carried out using the Google are not precise and not adequate. Indeed, we find redundant results and parasite links (i.e., links for blogs, publicity pages, documents, providers' catalogs, etc.). In opposition, while integrating our proposed ontology, we obtain quickly and precise results which satisfies all users (i.e., all results are correct or precise signifies that all founded Cloud providers offer all functional and non-functional user's requirements).

Table 5 presents the correctness of both our ontology and the Google web engine while discovering services for the Cloud users. As shown, the ontology has a high value of correctness which make it a relevant and pertinent way for users to select the appropriate services.

5 CONCLUSION & FUTURE WORKS

In this work, we proposed a comprehensive ontology for Cloud service description that covers the three layers of Cloud models, namely IaaS, PaaS and SaaS. Our proposed ontology treats the functional and the non-functional properties of services. In addition, it improves the interoperability problem among multiples Cloud infrastructures, platforms and applications within a Cloud federation environment. Furthermore, it helps users to discover and select appropriate Cloud services and that is proved by interrogating it with some examples of users' queries. The ontology presented in this paper is mainly based on standards, literature study, Cloud providers catalogs, analysis and comparisons of existing Cloud Computing services ontologies and taxonomies.

As a future work, we aim to improve the evaluation of the proposed ontology. First, we plan to work on populating the ontology with more individuals from real Cloud systems to increase the number of instances (since the evaluation depends on the introduced population). Second, we investigate to interrogate more number of users' requests and consequently to determine the precision, the recall and the inference ability of the ontology. Third, we intend to compare our proposed ontology with another of the same field. As another perspective, we envisage to extend our ontology to capture the dependency between services across the three Cloud models (namely, IaaS, PaaS and SaaS).

REFERENCES

Alhamad, M., Dillon, T., and Chang, E. (2010). Conceptual sla framework for cloud computing. In *4th IEEE International Conference on Digital Ecosystems and Technologies*, DEST'10, pages 606–610. IEEE.

Behrendt, M., Glaser, B., Kopp, P., Diekmann, R., Breiter, G., Pappe, S., Kreger, H., and Arsanjani, A. (2011). Introduction and architecture overview ibm cloud computing reference architecture 2.0. IBM.

Beom, M. Y., Ho, J. S., and Sik, L. J. (2011). Ontology-based resource management for cloud computing. In *Proceedings of the Third International Conference on Intelligent Information and Database Systems*, ACIIDS'11, pages 343–352, Berlin, Heidelberg. Springer-Verlag.

Bernstein, D. and Vij, D. (2010). Intercloud directory and exchange protocol detail using xmpp and rdf. In *Proceedings of the 6th World Congress on Services*, pages 431–438.

Bhogal, J., Macfarlane, A., and Smith, P. (2007). A review of ontology based query expansion. *Inf. Process. Manage*, 43(4):866–886.

Buyya, R., Ranjan, R., and Calheiros, N. (2010). 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 - Volume Part I, ICA3PP'10, pages 13–31. Springer-Verlag.

Cesar, A. J., Asuncion, G., Lozano, T. A., and Andrade, P. H. S. (1998). Onto2 agent an ontology based www broker to select ontologies. In *Proceedings of the Workshop on Applications of Ontologies and Problem solving Methods at the 13th European Conference on Artificial Intelligence*.

Cloud, A. E. C. (2013). Amazon ec2. http://aws.amazon.com/fr/ec2/.

Evren, S., Bijan, P., Cuenca, G. B., Aditya, K., and Yarden, K. (2007). Pellet: A practical owl-dl reasoner. *Web Semantique*, pages 51–53.

Forum, O. G. (2010). Open cloud computing interface (occi). http://forge.ogf.org/sf/projects/occi-wg.

Ghoneim, A. and Tolba, A. (2014). Cobe framework: Cloud ontology blackboard environment for enhancing discovery behavior. *International Journal on Cloud Computing: Services and Architecture*, 4(4):29–36.

Grozev, N. and Buyya, R. (2012). Inter-cloud architectures and application brokering: Taxonomy and survey. *Software-Practice and Experience*, 44:369–390.

Gruber, T. R. (1995). Toward principles for the design of ontologies used for knowledge sharing. *International Journal of HumanComputer Studies Special issue: the role of formal ontology in the information technology*, 43:907–928.

Henry, L. (2009). *Introducing Windows Azure*. Apress, Berkely, CA, USA.

Hoang, H. and Tjoa, A. (2006). The state of the art of ontology-based query systems a comparison of existing approaches. In *In Proc. of ICOCI06*.

Horridge, M. and Drummond, N. (2008). Dlquery. http://protegewiki.stanford.edu/wiki/DLQuery.

ISO/IEC (2001). *ISO/IEC 9126. Software engineering – Product quality*. ISO/IEC.

KPMG (2013). The cloud changing the business ecosystem. http://www.kpmg.com/IN/en/Issues And Insights/Thought Leadership/The Cloud Changing the Business Ecosystem.pdf.

Moscato, F., Aversa, R., and Martino, B. D. (2011). An analysis of mosaic ontolog y for cloud resources annotation. In *Proceedings of the Federated Conference on Computer Science and Information Systems*, FedCSIS'11, pages 973–980.

NIST (2013). *Nist Cloud Computing Standards Roadmap*. CreateSpace Independent Publishing Platform.

Parhi, M., Pattanayak, B., and Patra, M. (2014). A multi-agent-based framework for cloud service description and discovery using ontology. In *InProceedings of Intelligent Computing, Communication and Devices*, volume 1 of *ICCD 2014*, pages 337–348. Springer India.

Pradel, C., Hernandez, N., Kamel, M., and Rothenburger, B. (2012). Une ontologie du cinéma pour évaluer les applications du web sémantique. In *In Atelier Ontologies et Jeux de Données pour évaluer le web sémantique*, IC'2012.

Rekik, M., Boukadi, K., and Ben-Abdallah, H. (2014). A context based scheduling approach for adaptive business process in the cloud. In *IEEE 7th International Conference on Cloud Computing*, CLOUD'2014, pages 948–951. IEEE.

Rochwerger, B., Breitgand, D., Levy, E., Galis, A., Nagin, K., Llorente, M., andY. Wolfsthal, R. M., Elmroth, E., Caceres, J., Ben-Yehuda, M., Emmerich, W., and Galan, F. (2009). The reservoir model and architecture for open federated cloud computing. *IBM J. Res. Dev.*, 53(4):535–545.

Russell, J. and Cohn, R. (2012). *Gogrid*. Book on Demand.

Ruthven, I. and Lalmas, M. (2003). A survey on the use of relevance feedback for information access systems. *Knowl. Eng. Rev.*, 18(2):95–145.

Shearer, R., Motik, B., and Horrocks, I. (2008). Hermit: A highly-efficient owl reasoner. In *Proc. of the 5th Int. Workshop on OWL: Experiences and Directions*.

Staab, S. and Studer, R., editors (2004). *Handbook on Ontologies*. International Handbooks on Information Systems. Springer.

Tsarkov, D. and Horrocks, I. (2006). Fact++ description logic reasoner: System description. In *Proceedings of the International Joint Confererence on Automated Reasoning*, pages 292–297.

Youseff, L., Butrico, M., and Silva, D. D. (2008). Towards a unified ontology of cloud computing. In *in Proceedings of the Grid Computing Environments Workshop*, GCE08, pages 1–10.

Zhang, M., Ranjan, R., Haller, A., Georgakopoulos, D., Menzel, M., and Nepal, S. (2012). An ontology based system for cloud infrastructure services discovery. *CoRR*.