Cloud Description Ontology for Service Discovery and Selection

Molka Rekik¹, Khouloud Boukadi¹ and Hanène Ben-Abdallah¹²

¹Mir@cl Laboratory, Sfax University, Sfax, Tunisia
²King 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), (Buuya 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:
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 (Youssef 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 protégé® 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.

![Diagram](image1.png)

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 (Moscati 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_Administration 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 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.

1http://protege.stanford.edu/
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 (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 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 NonFunctional 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.
2. The Functional\textsubscript{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\textsubscript{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), Human 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 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\textsubscript{IaaS} Property class: this class includes the Non Functional VM Property and the Non Functional StorageSpace Property subclasses.

(a) The Non Functional\textsubscript{VM} Property class: in this class, we classify the following subclasses (shown in Figure 9):

- VM\textsubscript{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\textsubscript{Scale}Up (the maximum number of VMs provided for one user) and VM\textsubscript{Scale}Down (the minimum number of VMs provided for one user).
- Processing\textsubscript{Speed} class: the processing speed of the CPU.
- Data\textsubscript{Transfer}\textsubscript{Speed} class: the processing speed of the network.
- QoS\textsubscript{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\textsubscript{Tolerance} (the ability to manage auto-
(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](https://aws.amazon.com)
\(^3\) [www.rackspace.com](https://www.rackspace.com)
\(^4\) [www.hpcloud.com](https://www.hpcloud.com)
Table 1: An Extract of IaaS Individuals.

<table>
<thead>
<tr>
<th>Domain</th>
<th>Object Property</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>AmazonEC2</td>
<td>hasIaaSService</td>
<td>VM11, ...</td>
</tr>
<tr>
<td>MicrosoftAzure</td>
<td>hasIaaSService</td>
<td>VM21, ...</td>
</tr>
<tr>
<td>VM11</td>
<td>hasType</td>
<td>XLarge</td>
</tr>
<tr>
<td></td>
<td>hasPricePerHour</td>
<td>0.64$</td>
</tr>
<tr>
<td></td>
<td>hasMemory</td>
<td>16GO</td>
</tr>
<tr>
<td></td>
<td>hasFrequency</td>
<td>2.7GHz</td>
</tr>
<tr>
<td></td>
<td>hasNetwork</td>
<td>50Gbps</td>
</tr>
<tr>
<td></td>
<td>hasLocation</td>
<td>Asia-Pacific</td>
</tr>
<tr>
<td></td>
<td>hasUseCase</td>
<td>General_Use</td>
</tr>
<tr>
<td></td>
<td>hasNonFunctionalVMProperty</td>
<td>Confidentiality, Migration, ...</td>
</tr>
<tr>
<td>VM21</td>
<td>hasType</td>
<td>Small</td>
</tr>
<tr>
<td></td>
<td>hasPricePerHour</td>
<td>0.055$</td>
</tr>
<tr>
<td></td>
<td>hasMemory</td>
<td>2GO</td>
</tr>
<tr>
<td></td>
<td>hasFrequency</td>
<td>2.5GHz</td>
</tr>
<tr>
<td></td>
<td>hasNetwork</td>
<td>20Gbps</td>
</tr>
<tr>
<td></td>
<td>hasLocation</td>
<td>US_West</td>
</tr>
<tr>
<td></td>
<td>hasUseCase</td>
<td>General_Use</td>
</tr>
<tr>
<td></td>
<td>hasNonFunctionalVMProperty</td>
<td>Auto_Scaling, Cloud_Federation, ...</td>
</tr>
</tbody>
</table>

Table 2: An Extract of PaaS Individuals.

<table>
<thead>
<tr>
<th>Domain</th>
<th>Object Property</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>GoogleAppEngine</td>
<td>hasPaaSService</td>
<td>Platform11, ...</td>
</tr>
<tr>
<td>CloudFoundry</td>
<td>hasPaaSService</td>
<td>Platform21, ...</td>
</tr>
<tr>
<td>Platform11</td>
<td>hasLanguage</td>
<td>Java</td>
</tr>
<tr>
<td></td>
<td>hasDBMS</td>
<td>Oracle</td>
</tr>
<tr>
<td></td>
<td>hasPricePerHour</td>
<td>0.7$</td>
</tr>
<tr>
<td></td>
<td>hasLocation</td>
<td>Asia-Pacific</td>
</tr>
<tr>
<td></td>
<td>hasNonFunctionalPaaSProperty</td>
<td>Security, Integrity, Fault_Tolerance</td>
</tr>
<tr>
<td>Platform21</td>
<td>hasLanguage</td>
<td>Python</td>
</tr>
<tr>
<td></td>
<td>hasDBMS</td>
<td>MySQL</td>
</tr>
<tr>
<td></td>
<td>hasPricePerHour</td>
<td>0.4$</td>
</tr>
<tr>
<td></td>
<td>hasLocation</td>
<td>Europe</td>
</tr>
<tr>
<td></td>
<td>hasNonFunctionalPaaSProperty</td>
<td>Cloud_Federation, Security, Integrity, Consistency</td>
</tr>
</tbody>
</table>

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 deduced 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.

<table>
<thead>
<tr>
<th>Domain</th>
<th>Object Property</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>SalesForce</td>
<td>hasSaaSService</td>
<td>Software1, ...</td>
</tr>
<tr>
<td>DoliCloud</td>
<td>hasPaaSService</td>
<td>Software21, ...</td>
</tr>
<tr>
<td>Software11</td>
<td>hasCRM</td>
<td>SugarCRM</td>
</tr>
<tr>
<td></td>
<td>hasPricePerHour</td>
<td>2.5$</td>
</tr>
<tr>
<td></td>
<td>hasLocation</td>
<td>Asia_Pacific</td>
</tr>
<tr>
<td></td>
<td>hasNonFunctionalSaaSProperty</td>
<td>Multi_Tenancy, Reusability</td>
</tr>
<tr>
<td>Software21</td>
<td>hasERP</td>
<td>ERP5</td>
</tr>
<tr>
<td></td>
<td>hasPricePerHour</td>
<td>4.5$</td>
</tr>
<tr>
<td></td>
<td>hasLocation</td>
<td>Europe</td>
</tr>
<tr>
<td></td>
<td>hasNonFunctionalSaaSProperty</td>
<td>Multi_Tenancy, Privacy, Availability</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Query</th>
<th>Using the Proposed Ontology</th>
<th>Using the Google Engine</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1</td>
<td>IaaS_Provider and hasService value Virtual_Machine and hasPricePerHour max 1$ and hasMemory min 7GO and hasNonFunctionalVMProperty value “Auto_Scaling”¨¨string</td>
<td>IaaS provider +price per hour&lt;=$1 +auto scaling +memory&gt;=7GO</td>
</tr>
<tr>
<td>Q2</td>
<td>PaaS_Provider and hasService value Platform and hasPricePerHour max 1.5$ and hasDBMS value Oracle and hasLanguage value Java and hasNonFunctionalVMProperty value “Confidentiality”¨¨string</td>
<td>PaaS provider +Oracle+Java +price per hour&lt;=$1.5$ +confidentiality</td>
</tr>
<tr>
<td>Q3</td>
<td>SaaS_Provider and hasService value Software and hasPricePerHour max 4.5$ and hasERP value ERP and hasNonFunctionalVMProperty value “Multi_Tenancy”¨¨string</td>
<td>SaaS provider +ERP5+multi-tenancy +price per hour&lt;=$4.5$</td>
</tr>
</tbody>
</table>

choose the Fact++ reasoner which is an open source, free and available with protégé. 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 Google5 with users’ queries.

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

5http://www.google.com/
Table 5: Query Results with Using Ontology and Google.

<table>
<thead>
<tr>
<th>Query</th>
<th>Ontology Results</th>
<th>Google Results</th>
<th>Correct Google Results</th>
<th>Correct Ontology Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1</td>
<td>10</td>
<td>13000</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>Q2</td>
<td>7</td>
<td>705000</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>Q3</td>
<td>5</td>
<td>1280</td>
<td>0</td>
<td>5</td>
</tr>
</tbody>
</table>

Table 6: Results Correctness with Using Ontology and Google.

<table>
<thead>
<tr>
<th>Query</th>
<th>Google Correctness</th>
<th>Ontology Correctness</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1</td>
<td>0.008%</td>
<td>100%</td>
</tr>
<tr>
<td>Q2</td>
<td>0%</td>
<td>100%</td>
</tr>
<tr>
<td>Q3</td>
<td>0%</td>
<td>100%</td>
</tr>
</tbody>
</table>

Exposure 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


