2014 International Conference on Intelligent Networking and Collaborative Systems

Semantic Representation of Cloud Services: a Case Study for Microsoft Windows Azure

Beniamino Di Martino, Giuseppina Cretella, Antonio Esposito and Raffaele Giulio Sperandeo
Department of Industrial and Information Engineering
Second University of Naples
Aversa, Italy
beniamino.dimartino@unina.it, giuseppina.cretella@unina2.it,
antonio.esposito@unina2.it, raffaelegiulio.sperandeo@studenti.unina2.it

Keywords—*cloud computing; semantic web; semantic representation; ontology; cloud interoperability;*

Abstract—**Starting with the provision of ready-to-use infrastructures, such as storage and compute resources, cloud computing quickly became a flexible, cost-effective and complete environment for a wide range of IT services offered over the Internet. A growing number of cloud providers started to expose their own services to the market, to answer consumer's need, engaging a competition in the attempt to offer the easiest access to resources and the wider catalogue of services. Being integrated with proprietary services and infrastructures, these offerings makes difficult to switch between different underlying technologies, so that the customer is tied to the service provider's strategy. This lack in standards interfaces, service requirements and technologies brings into the cloud the vendor lock-in problem. Due to this complex scenery, a categorization of services to support the choice of the right solution, as well as a semantic and computable description of services that enables the comparison and the mapping between different providers' services, proves to be extremely appealing. In an attempt to take a first important step in this context we propose the semantic description of some cloud services, exposing them in terms of functionalities, parameters exchanged, and collaboration between services. In particular in this paper we present the semantic representation of Microsoft Windows Azure APIs, describing functional and non-functional properties of the services.**

I. Introduction

It is quite difficult to find a single complete definition of cloud computing. Most of them, however, describe cloud computing as the dynamic provisioning of IT resources throughout the Internet, pointing out the following characteristic: on-demand approach, elasticity, resource metering, pay-per-use. Application in cloud computing are built through cloud APIs (Application Programming Interfaces), allowing services access and deployment. Vendors provide their own APIs and interface to access the services they offer in the cloud. The lack of a common API prevents first of all the interoperability between services from different providers, and also locks the cloud user in a provider specific solution. The adoption of cloud solutions from some enterprises is still under discussion because applications built around cloud APIs are costly to construct and to change and there is currently no good path to cloud portability to protect the investment in development. Some steps in the scope of avoiding this vendor lock-in have been taken with the introduction of cross-platform APIs such as Delta cloud [1] and mOSAIC API [2]. A cross-platform API provides a higher level of abstraction than cloud provider based API by taking cloud provider specific cloud API calls and making them generic. The benefits of using a cross-platform based cloud API is the ability to use a single API call, to access or leverage cloud resources on more than one provider's cloud computing platform. This saves a considerable amount of time, reduces complexity of the code rather than implementing multiple cloud provider based cloud APIs but also in this case the user is limited to the cloud platforms supported by the cross-platform API and remains locked in it if the adopted solution is not a standard. On the other hand building services upon reliable standard is a road slowly being undertaken from more and more providers. The application of semantic web technology and meta-data can be applied to cloud computing and can bring advantages at different levels, within and across cloud platform. For example it can provide a machine processable meaning of cloud-based resources and cloud services, enabling automated evaluation, navigation and consumption of clouds and cloud-based resources. In this context exploiting the powerful expressiveness of Web Semantic languages, such as OWL-S [3], to represent cloud services might open new horizons in the cloud services discovery and composition. Building a semantic description of the services makes them linked to abstract and cross-platform concepts, but enables the possibility to reconcile and match different semantic representations and then the mapping between different cloud providers' services. Previous investigations on the application of semantic technologies in cloud computing have been done within the mOSAIC project [2]. This work represents an enrichment and reinforcement of the knowledge base of two mOSAIC components, the Semantic Engine [4] and the Dynamic Discovery Service [5]. The Semantic Engine introduces a high level of abstraction over the cloud APIs and cloud resources, by providing semantic based representation of abstract functionalities and resources, related by properties and constraints, and Application domain level concepts and application patterns. Inference rules representing developer experts' knowledge and reasoners are used to support the cloud application developer in the tasks of discovering the needed functionalities and resources for application development through vendor independent representations of such application components, and representation of generic programming concepts and patterns, including application domain related ones. The Dynamic Discovery Service target is to discover Cloud providers functionalities and resources, compare and align to other provider or agnostic API, thus supporting agnostic and interoperable access to Cloud providers offers.

Recently the concept of patterns is becoming increasingly common in the cloud, and most of the cloud provider such as Amazon [6] and Windows Azure [7] have proposed their cloud patterns based on their cloud offerings. Patterns have always been an important concept in IT field, describing solution for common problems, with the higher level of abstraction possible. This concept migrated into a cloud based environment, and a new kind of solutions born to face the different variety of problems introduced by the cloud computing. Those solutions involve the combination of cloud computing technologies, crossing the three different cloud service models. Other investigation, not related to specific cloud provider, emerged recently, with the birth of agnostic cloud pattern catalogue which provides generic cloud solution not related to particular vendor [8]. For all the cloud patterns (agnostic and provider dependent) there is a lack of a processable representation [9]. It's reasonable to think that a semantic representation of cloud patterns together with the semantic representation of cloud service can enable a process of discovery that can support a very effective porting of applications to the cloud by following appropriate cloud pattern and by choosing appropriate services from a particular cloud provider or from multiple competitors cloud providers. In this paper we propose the semantic representation of some cloud services offered by Microsoft in their cloud solution *Windows Azure*. This semantic representation is realized using ontologies expressed in OWL [10] to represent agnostic concepts and OWL-S to describe the real cloud provider services. This semantic description exposes the cloud services in terms of functionalities, parameters exchanged, and collaboration between services. We chose windows azure due to its predisposition to the definition of cloud patterns that can be very useful for further progress of this work. The paper is structured as follows: in section II information on existing attempts of making semantic representation of cloud computing are reported; section III briefly presents the Windows Azure platform and the underlying services; section V provides details on how the semantic description of the cloud services are composed; Section VI provides an example of semantic description of a Windows Azure cloud service, namely the *Service Bus*; finally section VII discusses future development and reports some conclusions.

II. Background: The use of semantic representation in cloud computing

As stated in [11], semantic models are helpful in three aspects of cloud computing. The first is functional and non-functional definitions. The ability to define application functionality and quality-of-service details in a platform-agnostic manner can immensely benefit the cloud community. This is particularly important for porting application code horizontally. The second aspect is data modeling. A crucial difficulty developers face is porting data horizontally across clouds. The third aspect is service description enhancement. Cloud providers expose their operations via Web services, but these service interfaces differ between vendors. The operations semantics, however, are similar. Metadata added through annotations pointing to generic operational models would play a key role in consolidating these APIs and enable interoperability among the heterogeneous cloud environments. Our work falls in the first and the third cases: it adds semantic and platform agnostic description to cloud service related to functional and non functional aspects and it enriches the API with meta-data in order to overcome syntactic differences and reconciles similar semantics. Due to the potentialities offered by the application of semantics in this context, several works have been previously proposed and a lot of ontologies related to cloud computing emerged. Darko et al. in [12] try to provide an overview of cloud computing ontologies, their types, applications and focuses. They identified four main categories of cloud computing ontologies according to their scopes: cloud resources and services description, cloud security, cloud interoperability and cloud services discovery and selection. Ontologies oriented to describe cloud resources and services from the point of view of the models belong to the first category. Among these kinds of ontologies, a remarkable work can be found in [13], which is focused on the technologies involved in the cloud phenomenon and describes the different layers of cloud computing, the relationships between them and the users of each cloud layer. Another notable work is presented in [14], in which a formal catalog representation of cloud services was proposed. All the works that fall in the first category focus on the methodology to model service offerings and their corresponding processes rather than on the exposed services. Other studies relevant to our work are the ones which use ontologies to achieve interoperability among different cloud. One ontology in particular, the mOSAIC cloud ontology [15] developed for the mOSAIC platform, has been developed to improve interoperability among existing cloud solutions, platforms and services, both from end-user and developer side. The ontology, developed in OWL [16], can be used for semantic retrieval and composition of cloud services in the mOSAIC project and also maintains compatibility with previous works because it is built upon existing standards. None of the analyzed works focuses on a proper representation of the characteristics and peculiarities of individual cloud services, as in our work. From the automatic reasoning prospective this is an important and not negligible aspect to consider because it enables the discovery of similarity and mapping between different cloud providers services.

III. Microsoft Windows Azure platform

Microsoft Azure, formerly Windows Azure, is Microsoft's cloud computing platform and infrastructure. It enables quick build, deployment and management of applications across the global network of Microsoft datacenters, as well as easy application scaling and support to any chosen language, framework or tool for application development. Features and services are exposed using open REST protocols. The selection of services includes the following list [17]:

- Compute: on-demand infrastructure that scales and adapts to changing business needs, including reliable infrastructure for web application deployment and a scalable back-end for mobile solutions (*Virtual Machines*, *cloud Services*, *Web Sites*, *Mobile Services*).
- Data Services: scalable and durable cloud storage, backup and recovery solutions for any data, including cache super-fast access, Hadoop solution and relational databases (*Blob*, *Table*, *Queue and Drive Storage*, *SQ Database*, *Backup*, *Cache*, *HDInsight*, *Hyper-V Recovery Manager*).

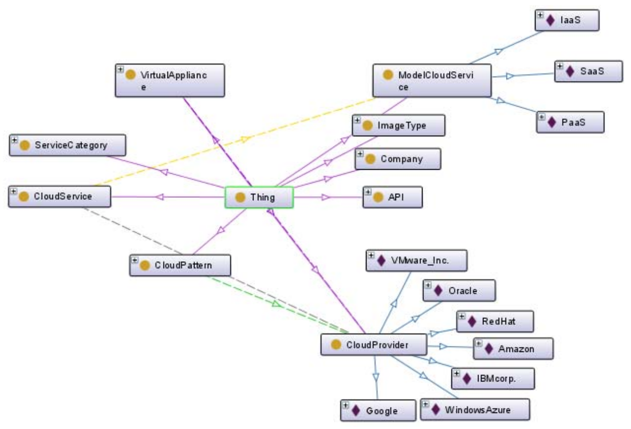


Fig. 1. Overview of the cloud Service Ontology

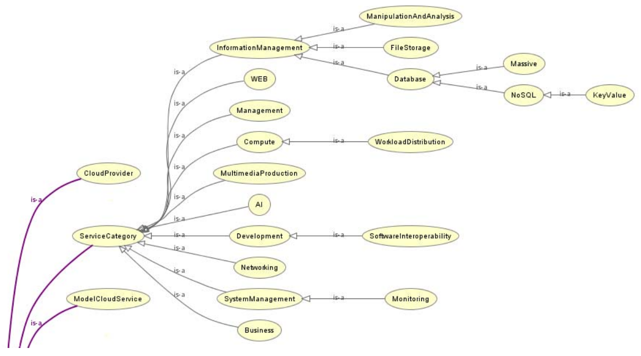
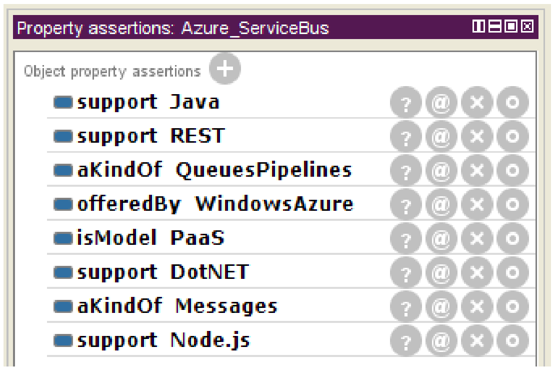


Fig. 2. Service Category subclasses


Property assertions: Azure_ServiceBus

Object property assertions

support Java
support REST
aKindOf QueuesPipelines
offeredBy WindowsAzure
isModel PaaS
support DotNET
aKindOf Messages
support Node.js

Fig. 3. Property assertions for the Service Bus service

- Pp Services: fast and flexible application test and development at reduced cost, together with media services for scalable and cost effective media distribution solutions, plus cloud identity services (*Media Services*, *Active Directory*, *Multi-Factor Authentication*, *Service Bus*, *Notification Hubs*, *Biz Talk Services*, *Scheduler*, *Visual Studio On line*, *Automation*, *Azure CAN*).
- Network: creation of virtual private networks or connections between datacenters and infrastructure, completed with a traffic load balancing service (*Express Route*, *Virtual Network*, *Traffic Manager*).

IV. The cloud Service Ontology

This section briefly illustrates the structure of the Cloud Service Ontology (presented in [18]) and provides details of the semantic description of several Microsoft Azure cloud Services. Figure 1 shows a graphical overview of the cloud services ontology, focusing on the main classes.

The *cloud Provider* class lists a subset of cloud providers currently on the market, while the *Cloud Service* class represent a comprehensive list of cloud services offered by the listed vendors. Other minor classes are used to assign services to a specific cloud service model, or the kind of APIs that can be used to access them. Our interest is focused on an all-inclusive arrangement of Microsoft Azure offering, listing the wide range of services made available from Azure, based on a specifically created categorization. For this purpose, the *Service Category* class provides a multi-layered partition of suitable cloud service categories so that services can be classified on the basis of their purpose, thus helping the research of services with similar functionalities. Subclasses composing the *Service Category* class are, for example: the *Business* class for business management services; the *Compute* class, listing the computing resource management tools; the *Development* class, related to the application development and the platform tools; the *Information Management* class, deeply detailed to classify the different solution for storage services. Figure 2 shows a partial overview of the cloud Service Ontology this time focusing on the *ServiceCategory* class.

Individuals of the *cloud Service* class are fittingly linked to individuals of the other listed classes through the use of conveniently created Object Properties, such as: *aKindOf*, linking the service to the category; *offeredBy*, linking the service to the provider; *support*, linking the service to the available APIs and *isModel*, linking the service to the cloud model. Figure 3 shows the example of the Service Bus cloud service, with the above-mentioned properties assertions.

V. The Semantic description of cloud Services

The operational target of this work is the semantic description of Microsoft Azure's cloud services. To achieve such description, since WSDL documents were not available for those services, a semantic annotation was required starting from the REST (*Representational State Transfer*) API which have been deeply analyzed to understand the methods involved in the services usage. To support the semantic description of cloud services, a fitting knowledge base had to be built so to identify conceptual services inputs and outputs, as well as the actual requested parameters for service methods. The information included in the xsd documents with regards to the input and output parameters aren't in fact enough adequate to describe those parameters by a semantic point of view. For this reason an ontology describing the semantic of Windows Azure concepts and relationships has been realized. This ontology classifies methods and related parameters, as described in the following subsection. The semantic description of the real services is performed using OWL-S. OWL-S is an ontology, within the OWL-based framework of the Semantic Web, for describing Semantic Web Services. The service class in OWL-S provides a reference point for a declared semantic service.

Each service instance is composed of three parts: the Profile, the Model and the Grounding. The service profile is used to describe what the service does. This information is primary meant for human reading, and includes the service name and description, limitations on applicability and quality of service, publisher and contact information. The Service concept in OWL-S links the profile, service model and grounding of a given service through the properties *presents, describedBy, and supports*, respectively. The process model describes how a client can interact with the service. This description includes the sets of inputs, outputs, pre-conditions and results of the service execution. The service grounding specifies the details that a client needs to interact with the service, as communication protocols, message formats, port numbers and so on. A grounding is a mapping from the abstract (service profile and model), to the concrete specification, for instance WSDL [19]. There are two ways to represent functionalities of a web service: the first way provides an extensive ontology of functions where each class in the ontology corresponds to a class of homogeneous functionalities; the second way is to provide a generic description of function in terms of the state transformation that it produces (outputs from inputs). OWL-S supports both as the *Service Profile* module provides a high level description of services as a transformation to one state to another but also enables to describe a specific class of capabilities for the service. More precisely, a Service Profile provides two types of information: the first one is a functional description of the Web service in terms of the transformation that the Web service produces, the second one is a set of non-functional properties that specify constraints on the service provided. For our representation we exploit both functional and non-functional properties. The functional properties link the services elements to concepts of the *cloud provider ontology* while the non-functional properties link the service to concepts described in our cloud ontology [18] or other useful ontologies.

A. *The Cloud Provider Ontology*

The *Cloud Provider Ontology* describes with classes and relationships the concepts related to the cloud provider offering, that means all the resources offered by the cloud provider and all the operation available on them, including the parameters used as input and output of the operations. This cloud provider dependent description is annotated semantically with cloud provider independent concepts, thus making the representation agnostic. An excerpt of these agnostic concepts are illustrated in Figure 4

Analyzing the kind of services taken into consideration, namely the compute service (*Virtual Machines*), the storage service (*Blob*) and the queue service (*Service Bus*), the following classes have been identified and included as main classes in the provider concepts ontology:

- Authentication Parameter
- Blob Method
- Blob Parameter
- HTTP Parameter
- Resource Configuration
- Service Bus Method

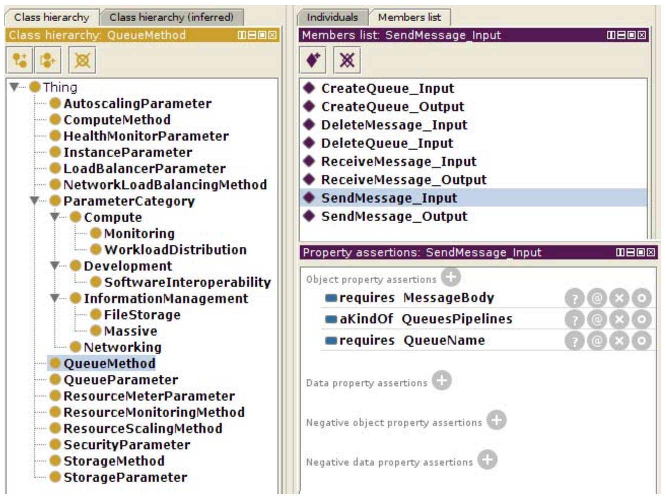


Fig. 4. Cloud provider independent concepts

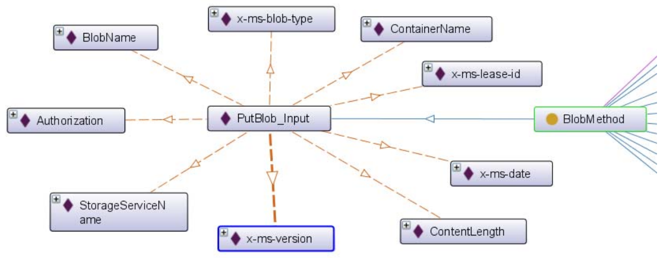
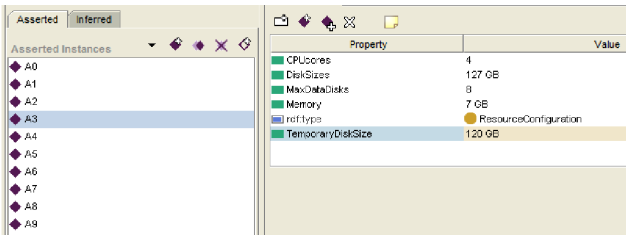


Fig. 5. PutBlob_Input consist property assertion

- Service Bus Parameter
- Service Management Method
- Service Management Parameter
- Virtual Machines Method
- Virtual Machines Parameter

The *AuthenticationParameter* class is used to list the parameters that services can require for authorization purposes, since, for example, requests towards storage services must be authenticated, unless the request is for a blob or container resource made available for public access. The *BlobMethod* class lists a set of inputs and outputs for each method provided by the Blob services (such as *CreateContainer*, *PutBlob*, *LeaseBlob* and so on) linked through the "*consists*" object property to the actual inputs or output parameters, some of which are listed in the *BlobParameter* class. For example, the *PutBlob_Input* individual of the *BlobMethod* class sees typical storage method parameters such as *BlobName* and *ContainerName*, as well as the *Authorization* parameter, and others provided in the *HTTP_Parameter* and *ServiceManagementParameter* classes (*x-ms-date* and *StorageServiceName* respectively), as shown in Figure 5.

The *HTTP_Parameter* class contains parameters closely related to the typical HTTP request or response header, like



A0
A1
A2
A3
A4
A5
A6
A7
A8
A9

Property	Value
CPUcores	4
DiskSizes	127 GB
MaxDataDisks	8
Memory	7 GB
rdf:type	ResourceConfiguration
TemporaryDiskSize	120 GB

Fig. 6. Virtual Machine sizes

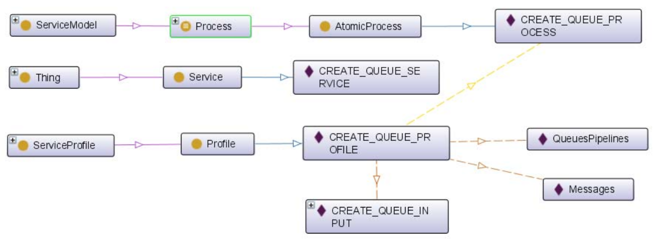


Fig. 7. CreateQueue OWL-S components

ContentLength and *ContentType*. The *ServiceBusMethod* class lists the methods' inputs and outputs for the Service Bus cloud service, once again linked to the actual input or output parameters, listed in the *ServiceBusParameter* class. Since is our purpose to give a full description of the semantical representation for one of the Microsoft Azure's cloud services, and the Service Bus has been chosen as case study, these two classes will be later described more accurately. The *ServiceManagementMethod* class lists provider-specific management methods, required for accounting purposes, such as *CreatecloudService*, *CreateStorageAccount* and *ListLocations* methods. Related parameters are listed in the *ServiceManagementParameter* class. Finally, the set of *VirtualMachinesMethod* is listed, together with the list of parameters in the *VirtualMachinesParameter* class. In particular, the *RoleSize* parameter is, in turn, linked to the individuals of the *ResourceConfiguration* class through the "*hasValue*" Object Property. The *ResourceConfiguration* class lists the available sizes and options for the virtual machine-based compute resources, providing different choices for the *RoleSize* parameter. In addition, different Data Properties provide deployment information such as number of *CPUcores*, *Memory* capability, *DiskSizes*, *TemporaryDiskSize* and *MaxDataDisks* number, 1 TB each. Role Sizes are specified as A0 (formerly "extra small") to A9, as shown in Figure 6, though not all the physical hosts in Azure data center may support larger virtual machine sizes, like those from A5-A9, mainly used for compute intensive workloads.

VI. The OWL-S description of Service

Once the cloud provider ontology is defined, it is possible to enrich the API of the service with the semantic concepts represented. In the following we will illustrate this description with the example of the Service Bus. Service Bus is a messaging infrastructure providing a channel for cloud application and services connection. It offers a simply FIFO (*First In First Out*) approach to message delivery, solving challenges of communication between applications and the outside world, as well as providing a notification mechanism. Two different kinds of message-oriented-middleware technologies are offered: a *Queues* mechanism and the *Topics/Subscriptions* approach. Queues offers FIFO message delivery to one or more competing consumers, so that messages are received and processed by the receiver in the temporal order in which they are added to the queue. Each message is received and processed by only one message consumer. On the other hand, topics and subscriptions provide a one-to-many form of communication, following a "*publish and subscribe*" pattern. Each published message is made available to each subscription registered with the topic, though messages can be delivered to one or more associated subscriptions, depending on filter rules, so that messages with specific characteristic are processed in specific ways, and subscriptions can find messages with desired properties. The following service methods, referring to both the messaging approaches, have been semantically described:

- Create Namespace
- Create Queue
- Create Rule
- Create Subscription
- Create Topic
- Delete Queue
- Delete Subscription
- Delete Topic
- Read Delete Message
- Read Delete Message from Subscription
- Send Message
- Send Message to Topic

The OWL-S description of each method consists of three parts according to the OWL-S specification. In concrete, concerning the Create Queue method, we have declared a CREATE_QUEUE_SERVICE, described by a CREATE_QUEUE_PROCESS and which presents a CREATE_QUEUE_PROFILE as shown in Figure 7.

The Grounding section links to a specifically created WSDL document, with actual bindings and parameters of the method. The Profile section provides functional and non-functional description of the method. In the profile the connection with the cloud Service ontology is realized through the *serviceCategory* property, placing the method in the Service Bus related category (*Messages* and *QueuesPipelines*). The Process section instead provides connections with the cloud Provider ontology, since the individual CREATE_QUEUE_INPUT is linked to a *CreateQueue_Input* individual belonging to the *ServiceBusMethod* class of the cloud Provider ontology, which, in turn, consists of the actual method's parameter, listed in the *ServiceBusParameter* class, as derived from the documentation and described in the WSDL document. The *CreateQueue_Input* individual, for example, contains the following parameters:

- *ContentType*, belonging to the *HTTP_Parameter* class;

- *MaxQueueSizeInMegaBytes*, specifying the maximum queue size in megabytes. Attempts to enqueue a message that causes the queue to exceed this value will fail;
- *QueuePath*, referring to the name or path to the queue;
- *ServiceNamespace*, needed as a scoping container for addressing Service Bus resources within the application;
- *SubscriptionID*, referring to the Azure accounting to access clouds service resources;
- *x-ms-version*, belonging to the *ServiceManagementParameter* class.

Other examples of the Service Bus parameters, included in the cloud Provider ontology and related to other methods are: *TopicPath*, *TopicMessageTimeout* and *MaxTopicSizeInMegaBytes* for the Service Bus Topics; *TopicSubscriptionName* for the Subscriptions associated to the Topics; *MessageBody* referring to the message payload.

VII. Conclusion and Future works

In the last years cloud computing platforms grew in popularity, but the fear of vendor lock in remains one of biggest barrier to widespread adoption of cloud technology. The cloud provider platforms are integrated with proprietary services and infrastructures and then make difficult to switch between different underlying technologies, so that the customer is tied to the service provider's strategy. Due to this complex scenery, a categorization of services to support the choice of the right solution, as well as a semantic and computable description of services that enables the comparison and the mapping between different providers' services, proves to be extremely appealing. The application of semantic web technology and meta-data to cloud computing can bring advantages in providing a machine processable meaning of cloud-based resources and cloud services, enabling automated evaluation navigation and consumption of clouds and cloud-based resources. In this paper we propose a semantic description of Windows Azure cloud services that includes an ontology that represents the cloud resources and operations in Windows terms, an ontology to annotate with agnostic terms the Windows Azure ontology and a semantic description for the cloud services offered by Windows Azure through OWL-S. This description is only a starting point to enable automatic advanced discovery and composition of cloud services, and can be seen, enriched with proper inference rules and description of a certain number of providers' cloud services, as the knowledge base of a framework to support inter-cloud interoperability and portability.

Acknowledgment

The research leading to these results has received funding from the European Communitys Seventh Framework Programme (FP7/2007-2013) under grant agreement n 256910 (mOSAIC Project) and has been supported by CoSSMic (Collaborating Smart Solar-powered Micro-grids - FP7-ICT-608806) and PRIST 2009, Fruizione assistita e context aware di siti archeologici complessi mediante dispositivi mobili.

References

[1] D. Cloud, "Delta cloud-many clouds. one api. no problem."

[2] B. Di Martino, D. Petcu, R. Cossu, P. Goncalves, T. Máhr, and M. Loichate, "Building a mosaic of clouds," in *Euro-Par 2010 Parallel Processing Workshops*. Springer, 2011, pp. 571–578.

[3] D. Martin, M. Burstein, J. Hobbs, O. Lassila, D. McDermott, S. McIlraith, S. Narayanan, M. Paolucci, B. Parsia, T. Payne *et al.*, "Owl-s: Semantic markup for web services," *W3C member submission*, vol. 22, pp. 2007–04, 2004.

[4] B. D. Martino and G. Cretella, "Towards a semantic engine for cloud applications development support," in *Proceedings of Sixth International Conference on Complex, Intelligent, and Softeare Intensive Systems*. IEEE CS Press, 2012, pp. 198–203.

[5] G. Cretella and B. Di Martino, "Semantic and matchmaking technologies for discovering, mapping and aligning cloud providers's services," in *Proceedings of the 15th International Conference on Information Integration and Web-based Applications and Services (iiWAS2013)*, 2013, pp. 380–384.

[6] "Aws cloud patterns," http://en.clouddesignpattern.org, [Online; accessed 22-June-2014].

[7] "Azure cloud patterns," http://msdn.microsoft.com/en-us/library/dn568099.aspx, [Online; accessed 22-June-2014].

[8] C. Fehling and R. Retter, "Cloud computing patterns," http://cloudcomputingpatterns. org note =, 2011.

[9] B. Di Martino, G. Cretella, and A. Esposito, "Semantic and agnostic representation of cloud patterns for cloud interoperability and portability," in *Proceedings of the 5th IEEE International Conference on Cloud Computing Technology and Science (CloudCom2013)*, 2013, pp. 182–187.

[10] D. L. McGuinness, F. Van Harmelen *et al.*, "Owl web ontology language overview," *W3C recommendation*, vol. 10, no. 10, p. 2004, 2004.

[11] A. Sheth and A. Ranabahu, "Semantic modeling for cloud computing, part 2," *Internet Computing, IEEE*, vol. 14, no. 4, pp. 81–84, 2010.

[12] D. Androcec, N. Vrcek, and J. Seva, "Cloud computing ontologies: A systematic review," in *MOPAS 2012, The Third International Conference on Models and Ontology-based Design of Protocols, Architectures and Services*, 2012, pp. 9–14.

[13] L. Youseff, M. Butrico, and D. Da Silva, "Toward a unified ontology of cloud computing," in *Grid Computing Environments Workshop, 2008. GCE'08*. IEEE, 2008, pp. 1–10.

[14] Y. Deng, M. Head, A. Kochut, J. Munson, A. Sailer, and H. Shaikh, "Introducing semantics to cloud services catalogs," in *Services Computing (SCC), 2011 IEEE International Conference on*, July 2011, pp. 24–31.

[15] F. Moscato, R. Aversa, B. Di Martino, T. Fortis, and V. Munteanu, "An analysis of mosaic ontology for cloud resources annotation," in *Computer Science and Information Systems (FedCSIS), 2011 Federated Conference on*. IEEE, 2011, pp. 973–980.

[16] S. Bechhofer, "Owl: Web ontology language," in *Encyclopedia of Database Systems*. Springer, 2009, pp. 2008–2009.

[17] "Microsoft windows azure," http://azure.microsoft.com/, [Online; accessed 22-June-2014].

[18] B. Di Martino, G. Cretella, and A. Esposito, "Towards an unified owl ontology of cloud vendors appliances and services at paas and saas level," in *Proceedings of the 8th International Conference on Computational Intelligence in Security for Information Systems (CISIS2014)*, 2014.

[19] E. Christensen, F. Curbera, G. Meredith, S. Weerawarana *et al.*, "Web services description language (wsdl) 1.1," 2001.