A Survey on Approaches for Interoperability and Portability of Cloud Computing Services

Kostas Stravoskoufos, Alexandros Preventis, Stelios Sotiriadis and Euripides G.M. Petrakis
Department of Electronic and Computer Engineering, Technical University of Crete, Chania, Greece
{*kgstravo, apreventis, s.sotiriadis, petrakis*}*@intelligence.tuc.gr*

Keywords: Cloud computing, cloud interoperability, cloud portability

Abstract: Over the recent years, the rapid development of Cloud Computing has driven to a large market of cloud services that offer infrastructure, platforms and software to everyday users. Yet, due to the lack of common accepted standards, cloud service providers use different technologies and offer their clients services that are operated by a variety of proprietary APIs. The lack of standardization results in numerous heterogeneities (e.g., heterogeneous service descriptions, message level naming conflicts, data representation conflicts etc.) making the interoperation, collaboration and portability of services a very complex task. In this work we focus on the problems of interoperability and portability in Cloud Computing, we address their differences and we discuss some of the latest research work in this area. Finally we evaluate and point out relationships between the identified solutions.

1 INTRODUCTION

According to the NIST (U.S. Government's National Institute of Standards and Technology) Cloud Computing definition (Mell and Grance, 2009), the Cloud is composed by three Service models: The Infrastructure as a Service (IaaS) model, which is used to manage the underlying hardware in order to provide the consumers with computer resources such as storage, processing power, or network. The Platform as a Service (PaaS) model which virtualizes a computer environment supporting the development and deployment of consumer applications and, finaly, the Software as a Service(SaaS) model which gives consumers the capability to use applications running on the cloud.

The term of interoperability has many definitions in literature (Petcu, 2011) and is often misused to include the term of portability. In order to get a more clear view we focus on the three cloud service models and the different conceptions interoperability refers to in each model. This also applies for applications that are not already ported to the cloud: developers who want to port an application to the cloud must consider all the different options offered by cloud service providers and find the most suitable for their application. On the other hand, in case where a client combines two SaaS (this leads to more complex service composition), interoperability refers to the ability of the services to communicate and be compatible by overcoming message level heterogeneities like naming conflicts or data representation conflicts. the services to communicate and be compatible by overcoming message level heterogeneities like naming conflicts or data representation conflicts.

The NIST makes a clear distinction between interoperability and portability by defining service interoperability, system portability and data portability (Bohn et al., 2011). *Service interoperability* is defined as "the ability of cloud consumers to use their data and services across multiple cloud providers with a unified management interface" while *portability* is further distinguished into *Data Portability* and *System Portability*. Data Portability is defined as "the ability of cloud consumers to copy data objects into or out of a cloud or to use a disk for bulk data transfer" while, System Portability is defined as "the ability to migrate one of a fully stopped virtual machine(VM) instance (that represents a cloud service or an application) or a machine image from one cloud infrastructure to another cloud, or migrate applications and services and their contents from one service provider to another".

The NIST also defines the concept of the "hybrid cloud" which is a composition of two or more clouds that remain as distinct entities but are bound together by standardized or proprietary technology

that enables data and application portability. In general, portability and interoperability between the various cloud service models resort to standards.

In the following sections we present the various approaches for solving the problem. In Section 2 we discuss interoperability and portability and in Section 3 approaches on this. In Section 4 we present a cross comparison study of the discussed approaches to characterize the most crucial requirements for future developments. At last, in Section 5 we highlight the future research steps of this work.

2 CLOUD SERVICE INTEROPERABILITY AND PORTABILITY

Service Interoperability allows customers to use services across multiple clouds using common management API. The general requirement to achieve service interoperability, since world-wide standards haven't been defined yet (Petcu et al., 2013), is the semantic description of services which is vital to create the middleware towards a common management API to override the proprietary ones.

Service Interoperability applies to all three Cloud Computing service models but the meaning of it (and also the requirements) varies in each model. Achieving interoperability in the IaaS model means that clients have the ability to use the infrastructure of different clouds and control them as if they were one. A simple example is when the client has control of several virtual machines that encapsulate the computational resources from different clouds. In PaaS, service interoperability is about enabling the clients to use different APIs, tools, libraries etc. from different platforms ported in different clouds in order to create applications. Finally, in the SaaS model, interoperability refers to enabling cloud applications to exchange messages or data. Below we present some of the latest research work on this subject.

System portability refers to the ability of cloud services to be deployed on other cloud services of a lower service model (e.g. the ability of a SaaS service to be deployed on different PaaS services). In the IaaS model, portability refers to the ability to migrate and run Virtual Machines along with data and configurations, across different infrastructure providers. In the PaaS model, portability refers to the ability to deploy applications across different platforms (across different virtual machines). Finally portability in the SaaS model refers to the ability of clients to move their data across equivalent SaaS applications. This is also known as data portability and is further analyzed below.

Another case of portability is when an external application needs to be ported in the cloud. In such case, an application description is required along with the description of the platform services. According to (Nagarajan et al., 2006) there are 4 types of semantics describing applications:

1. System semantics are semantics pertaining system characteristics (deployment and load balancing).
2. Data semantics are semantics pertaining to data (typing, storage and manipulation restrictions).
3. Non functional semantics which are semantics pertaining to QoS characteristics (performance, security).
4. Logic and process semantics which semantics pertaining to the core functions of the application (programming language, runtime, exception handling).

The basic requirement to achieve system portability, as in service interoperability, is the description of services. The concept of data portability applies only on the SaaS model and refers to the ability of customers to move their data along equivalent cloud applications hosted on different providers. Since the "vendor lock-in" also affects data portability, the main problem in this task is the variety of data representations used among the different cloud applications. This requires transforming the representation of data which is the output of one cloud application to the representation of data required as input by the target application. Another important problem to deal with is the different import and export functionalities of cloud applications (the mechanisms used to export or import data from and into cloud applications). The description of those functionalities is required to enable migration of data from one application to another (standardization plays an important role in this part).

3 SERVICE DISCOVERY IN CLOUDS

This section describes the analysis of the related approaches in the area of service discovery in cloud systems. The works deal a) with the service description problem (a vital requirement to achieve service Interoperability) and b) with message level heterogeneities. Although messages may be described in the same way (i.e., using SOAP) they may differ in the domain level (Nagarajan et al., 2006). This kind of

Approach	Focus	Key operations
WSDL-S, OWL-S	Approaches that provide the vocabulary for describing services.	Introduce semantics for the description of services (including cloud).
Unified Cloud Interface (UCI)	Standardization for interacting between clouds using an interface.	Common management API for cloud services.
Open Cloud Computing Interface (OCCI)	Standardization approach that proposes a common management API.	API for common management tasks e.g. cloud resource deployment.
OWL-S based Broker	Cloud service description and discovery approach using a broker.	OWL ontology for service discovery, providers need to be compliant.
Cloud Computing Ontology (CoCoOn)	Cloud service description and discovery using a broker.	OWL ontology for service description (IaaS model) and discovery.
Cloud Service Generic Search Engine	Description and discovery of services using a broker.	OWL ontology that uses a service discovery mechanism.
mOSAIC	Service description approach and service discovery and composition using a broker.	OWL-S ontology for service description. Interoperability through a common management API (OCCI).
PSIF	Framework for semantic interoperability conflicts on the PaaS layer.	Description of services for capturing and representing the conflicts.

Table 1: A summary of the cloud interoperability and portability approaches for service discovery

differences is described as heterogeneity at the message level. Dealing with message heterogeneities is very important if we take into consideration all the different proprietary APIs that are used in cloud computing. In Table 1 we present a summary of the cloud interoperability and portability approaches for service discovery.

The **Unified Cloud Interface (UCI)**[1] Project's goal is to create an interface to interact with various cloud API's. The main idea is to create an API to handle all other API's. The UCI abstracts the usage of any cloud API and unifies them in one layer that is agnostic to cloud providers. It uses ontologies and OWL (van Harmelen and McGuinness, 2004) in order resolve any heterogeneities between the cloud based APIs and the existing protocols and standards. OWL is also used for describing the cloud data models and the UCI uses this description to make resources from multiple cloud providers available.

The **Open Cloud Computing Interface (OCCI)**[2] is an API and Protocol that supports different kinds of management tasks (e.g. deployment, monitoring etc.). The API acts as a service front-end to a provider's internal management framework, for controlling the cloud hardware resources and it can also be used as a management API for all kinds of resources (virtualization, networking and storage). It is highly extensible and focuses both on interoperability and portability. OCCI is one of the first approaches towards standardization and was first introduced as a remote management API for IaaS model based services. Since then, with the support of the Open Grid Forum[3] community it evolved to support all three cloud service models. Many academic and industry members currently support OCCI (e.g. OpenStack, GoGrid etc.).

OWL-S (Martin et al., 2004) is an OWL ontology that provides the vocabulary for describing services. It enables automatic service discovery, composition and invocation. The main OWL-S ontology consists of three sub-ontologies, the "ServiceProfile", the "ServiceModel" and the "ServiceGrounding". The service profile is an upper level description of the service. It is mostly used for describing what the service does, in order to be discovered by agents and determine if it fits their needs.

The service model gives a detailed description of how the service works. It describes a) the semantic content of the service's inputs and outputs b) the conditions that have to be satisfied under which the service can be performed c) step by step processes that lead to the outcome of the service and d) the result of the service, meaning the effects of the service in the world (i.e., change in the balance of an account). Service grounding describes how a service can be accessed. It specifies the communication protocols, the message formats and other specific details such as the port that is used for accessing the service. OWL-S has such a structure that can be used with WSDL (Christensen et al., 2001) by extending the existing bindings of the later.

The **Cloud Computing Ontology (CoCoOn)** (Zhang et al., 2012) is an OWL ontology for de-

[1] http://code.google.com/p/unifiedcloud/

[2] http://occi-wg.org/

[3] http://www.gridforum.org/

scribing cloud infrastructure services. Although they currently focus on the IaaS model, CoCoOn creators claim that the ontology will be extended in the future to discribe the PaaS and SaaS models. The ontology proposed in this work consists of: a) functional Cloud service configurations information parameters and b) non-functional service configuration parameters.

In the functional cloud service configuration information part, the concepts defining the IaaS model are defined as a taxonomy. For example, the IaaS concept is divided into three subclasses: Compute, Network and Storage, which in turn are further analyzed and categorized. In the non-functional cloud service configuration information part, properties of the ontology concepts are described. A distinction is made here between properties of Cloud resources that are known at design time (non-functional properties) like the provider or the deployment model and attributes that can only be recorded after at least one execution cycle of a Cloud service (QoS attributes) like durability or performance. Based on the ontology describing the services, a service discovery system has been implemented that uses the CoCoOn Ontology to search for available infrastructure services. The ontology has been populated with well-known cloud providers (Amazon, Microsoft Azure, GoGrid, etc).

OWL-S Cloud Broker (Ngan and Kanagasabai, 2012) is an OWL-S based semantic cloud service brokering system that enables dynamic service discovery. The broker uses the OWL-S ontology to describe the services and SWRL [4] to apply complex constraints on them. The user describes the desired service with a set of preferences and constraints that the service should satisfy. The preferences are represented by OWL entities that are inserted into the system's ontology. The broker uses reasoning techniques to match the inputs and outputs of the requested service to the inputs and outputs of services available. The services that do match the user's preferences are checked for satisfying also the constraints of the preference. In the final stage, the services that have passed the previous steps are ranked using a scoring function.

The **Cloud Service Generic Search Engine** (henceforth referred to as CSGSE) (Nagireddi and Mishra, 2013) is an approach towards cloud service discovery. The work is based on an OWL ontology which provides the description of services and their attributes. Also it includes a search engine for services discovery in the ontology directory. The search engine uses the SPARQL query language [5] to retrieve information from the ontology and to match services based on user requests. Cloud services must be registered in an ontology registry and are available to the search mechanism.

mOSAIC Project's (Moscato et al., 2011) vision is to provide a platform that will enable interoperability between different cloud services, portability of cloud services in different platforms, automatic service discovery and composition and management of Service Level Agreement. It provides a set of APIs that enable developers to build vendor independent applications that consist of multiple cloud components. Each of these components perform a simple operation. On run-time, the applications are decomposed into these components and the mOSAIC platform automatically will decide which cloud implementation is better for each component to be deployed on. The mOSAIC API serves as an intermediate layer between the developers and the cloud platforms. It lets the developers deploy their applications without getting involved with APIs.

The **PaaS semantic interoperability framework (PSIF)** (Loutas et al., 2011) is a framework used to capture and represent semantic interoperability conflicts on the PaaS layer. Semantic interoperability conflicts may arise a) during the migration of an application from one PaaS to another, b) during the deployment of an application to a PaaS or, c) on the message exchange level between two PaaS systems. PSIF is structured according to three dimensions. Firstly, the fundamental PaaS entities (i.e., PaaS system, PaaS offering, management interface, software component, IaaS system and application) it is used to determine which entities are involved in the conflict and secondly, the types of semantics on the PaaS layer (functional, non-functional and execution) it is used to identify the semantic conflict. Thirdly, the levels where semantic conflicts occur (either on the information model or the data).

4 SERVICE DISCOVERY FOR PORTING APPLICATIONS IN CLOUDS

This section focus on the critical analysis of the related approaches for cloud service discovery. Table 2 demonstrates a cross comparison of the cloud service interoperability and portability approaches in order to define the most prominent. Based on the comparison we conclude that mOSAIC uses OWL-S and OCCI standards as well as provides the most complete ontology for service description. In addition, it is compliant with most of the cloud providers. The CoCoOn and OWL-S also present ontologies, yet there is still

[4] http://www.w3.org/Submission/SWRL/

[5] http://www.w3.org/TR/rdf-sparql-query/

place for improvement. Their difference is that CoCoOn is compliant with most common vendors (and we consider it as an advantage), while OWL-S requires vendors to be compliant with it. Finally, CGSE ontology is non detailed and requires from vendors to register their services to the broker (thus no support for most common cloud services).

In Figure 1, we demonstrate the portability and interoperability in the case of porting a service from a legacy system to a cloud. Here portability refers to the porting action while interoperability refers to the translation mechanism for matching purposes. The assumption is that the service to be ported requires to give an input to a broker that interprets and translates the service description. This has been characterized as a vital requirement by (OWL-S, CSGSE, OWL-S Broker and mOSAIC) for service acknowledgment. The broker includes the interoperability functionality in order to interpret and translate the request and match it with available ontological descriptions.

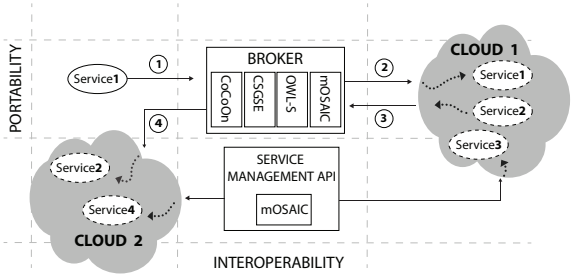

Figure 1: The interoperability and portability in cloud service discovery

We highlight portability for the SaaS, PaaS and IaaS cases:

- The SaaS user/developer a) decides the service(s) to be ported, b) the broker interprets and translates the service(s) description based on ontologies, c) the broker selects a cloud instrance that matches the service description, and d) the developer uses the service(s).
- The PaaS user/developer a) decides the service(s) to be ported, b) the broker interprets and translates the service(s) description based on ontologies, c) the broker selects a cloud node for instantiation, d) the broker selects a cloud instance that matches the service(s) description and build the platform, and e) the developer accesses the service(s).
- The IaaS user/developer a) decides the service(s) to be ported, b) the broker interprets and translates the service(s) description based on ontologies, c) the broker selects the cloud node that matches the required computational resources, d) the developer accesses and utilizes the node(s).

Figure 1 shows that a service (Service1) could be ported to a cloud (CLOUD1), as shown in steps 1 and 2, through a broker (as implemented from the literature review approaches) based on semantically translation of the service requirements, described as Service Level Agreement (SLA). The broker operation includes the matching process (match SLA description with available ones based on ontologies). The interoperability is demonstrated in steps 3 and 4, where a service (Service2 from CLOUD1) is moved to another cloud (CLOUD2). The service management API is the common control point for different clouds (CLOUD1 and CLOUD2) integrated based on OCCI standards.

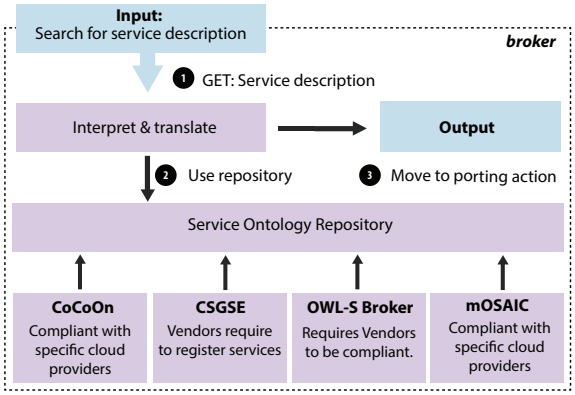

Figure 2: The interoperability and portability cloud service discovery broker

Figure 2 demonstrates the functionality of the service broker. The first step is to get the service description and interpret and translate the descriptions. The second step is to use the repository where all four approaches add as input their ontologies. For example, CoCoOn is compliant with service descriptions of specific cloud providers (e.g. Microsoft Azure). After the matching, the process moves to step 3 that is the porting action.

5 CONCLUSIONS

This work presented an analysis of the related works that aim to service interoperability and portability on cloud systems. By summarizing the different approaches we concluded to key features such as a) the use of semantics in the description of services and ontologies as the most suitable tool for this task (e.g. CoCoOn, mOSAIC etc.), b) the definition of standards in the description of services (e.g. OCCI) and c) the use of broker to bridge gap of translating services between systems.

	CSGSE	OWL-S Broker	mOSAIC
CoCoOn	CSGSE ontology in not as detailed compared to CoCoOn. Also, CoCoOn is compliant with specific providers, yet CGSE requires vendors to register in a broker.	OWL-S Broker is considered as a more detailed ontology. It uses OWL-S standard. Yet, OWL-S requires compliant providers with its ontology.	Compared with CoCoOn, mOSAIC is more detailed ontology in the level of descitpion of services. It uses the OWL-S standard and OCCI as a service front-end to a providers internal management framework.
CSGSE		OWL-S Broker is a more specialized ontology as it uses standards as OWL-S and SWRL. Also, CSGSE uses databases for storage thus it looses the benefits of semantics.	mOSAIC has a more specialized ontology on the level of service description for IaaS. Also it uses OWL-S, SPARQL unambiguous queries, as an advantage compared with CGSE that uses database for storage. It uses also OCCI.
OWL-S Broker			mOSAIC uses OWL-S, SPARQL and OCCI. In the level of ontology repository, OWL-S Broker requires vendors to be compliant with ontologies, yet mOSAIC is compliant with most providers (e.g., MS Azure).

Table 2: A cross correlation study of the cloud interoperability and portability approaches

ACKNOWLEDGMENTS

The research leading to these results has received funding from the European Unions Seventh Framework Programme (FP7/2007-2013) under grant agreement no 604691 (project FI-STAR).

REFERENCES

Bohn, R. B., Messina, J., Liu, F., Tong, J., and Mao, J. (2011). Nist cloud computing reference architecture. In *Proceedings of the 2011 IEEE World Congress on Services*, SERVICES '11, pages 594–596, Washington, DC, USA. IEEE Computer Society.

Christensen, E., Curbera, F., Meredith, G., and Weerawarana, S. (2001). Web Service Definition Language (WSDL). W3C recommendation, W3C. http://www.w3.org/TR/wsdl.

Loutas, N., Kamateri, E., and Tarabanis, K. A. (2011). A Semantic Interoperability Framework for Cloud Platform as a Service. In *CloudCom*, pages 280–287.

Martin, D. L., Paolucci, M., McIlraith, S. A., Burstein, M. H., McDermott, D. V., McGuinness, D. L., Parsia, B., Payne, T. R., Sabou, M., Solanki, M., Srinivasan, N., and Sycara, K. P. (2004). Bringing Semantics to Web Services: The OWL-S Approach. In *SWSWPC*, pages 26–42.

Mell, P. and Grance, T. (2009). The NIST Definition of Cloud Computing. Technical report.

Moscato, F., Aversa, R., Martino, B. D., Fortis, T.-F., and Munteanu, V. I. (2011). An Analysis of mOSAIC ontology for Cloud Resources annotation. In *FedCSIS*, pages 973–980.

Nagarajan, M., Verma, K., Sheth, A., Miller, J., and Lathem, J. (2006). Semantic Interoperability of Web Services - Challenges and Experiences. In *ICWS '06: Proceedings of the IEEE International Conference on Web Services (ICWS'06)*, pages 373–382, Washington, DC, USA. IEEE Computer Society.

Nagireddi, V. and Mishra, S. (2013). An ontology based cloud service generic search engine. In *Computer Science Education (ICCSE), 2013 8th International Conference on*, pages 335–340.

Ngan, L. D. and Kanagasabai, R. (2012). Owl-s based semantic cloud service broker. In *ICWS*, pages 560–567.

Petcu, D. (2011). Portability and interoperability between clouds: challenges and case study. In *Proceedings of the 4th European conference on Towards a service-based internet*, ServiceWave'11, pages 62–74, Berlin, Heidelberg. Springer-Verlag.

Petcu, D., Macariu, G., Panica, S., and Crciun, C. (2013). Portable Cloud applications - From theory to practice. *Future Gener. Comput. Syst.*, 29(6):1417–1430.

van Harmelen, F. and McGuinness, D. (2004). OWL web ontology language overview. W3C recommendation, W3C. http://www.w3.org/TR/2004/REC-owl-features-20040210/.

Zhang, M., Ranjan, R., Haller, A., Georgakopoulos, D., Menzel, M., and Nepal, S. (2012). An ontology based system for cloud infrastructure services discovery. *CoRR*, abs/1212.0156.