A Semantic-Agent Framework for PaaS Interoperability

Position Paper

Suchismita Hoare
School of Computer Science
University of Hertfordshire
Hatfield, United Kingdom
s.b.hoare@herts.ac.uk

Na Helian and Nathan Baddoo
School of Computer Science
University of Hertfordshire
Hatfield, United Kingdom
n.helian@herts.ac.uk

***Abstract*—Cloud Platform as a Service (PaaS) is poised for a wider adoption by its relevant stakeholders, especially Cloud application developers. Despite this, the service model is still plagued with several adoption inhibitors, one of which is lack of interoperability between proprietary application infrastructure services of public PaaS solutions. Although there is some progress in addressing the general PaaS interoperability issue through various devised solutions focused primarily on API compatibility and platform-agnostic application design models, interoperability specific to differentiated services provided by the existing public PaaS providers and the resultant disparity owing to the offered services' semantics has not been addressed effectively, yet. The literature indicates that this dimension of PaaS interoperability is awaiting evolution in the state-of-the-art.**

This paper proposes the initial system design of a PaaS interoperability (IntPaaS) framework to be developed through the integration of semantic and agent technologies to enable transparent interoperability between incompatible PaaS services. This will involve uniform description through semantic annotation of PaaS provider services utilizing the OWL-S ontology, creating a knowledgebase that enables software agents to automatically search for suitable services to support Cloud-based Greenfield application development. The rest of the paper discusses the identified research problem along with the proposed solution to address the issue.

Keywords—Cloud; PaaS service; interoperability; OWL-S; ontology; semantic; agent; service description; service search; service composition

I. Introduction

Platform as a Service (PaaS), or more specifically application PaaS (aPaaS), is a rapidly growing segment of Cloud Computing, offering application infrastructure services through abstracted development and deployment environments for Cloud application developers. Although developers can benefit immensely from this service model, the inherent heterogeneity in the available PaaS offerings is a major deterrent to its widespread adoption, because it poses various interoperability challenges [1]. PaaS adoption is impacted by, among other factors, a lack of comprehensive services supporting all aspects of Cloud application lifecycle [2] and a lack of interoperability among proprietary APIs and incompatible services of the current public PaaS solutions. Rafique et al. [3] observes PaaS vendors do not support application infrastructure services uniformly. Therefore, even when specific language-based platforms are used, users or developers can encounter interoperability issues owing to differences in underpinning platforms and supported Cloud services like file and data storage, messaging, task queues etc. This hinders simultaneous use and substitutability of available services for the development and deployment of Cloud applications and calls for a focus on PaaS *service/platform* interoperability.

Interoperability has numerous interpretations in the literature and encompasses the aspects of effective collaboration of disparate systems, resources, and services [1]. For PaaS, often misinterpreted as 'portability', it encompasses the software entities of Application, Service and Platform. Among these, *Service interoperability* (the ability of using services across multiple Cloud platforms through a unified management interface) and *Platform interoperability* (the ability of using platform components, either part of an IaaS or a PaaS offer, to interoperate); the two dimensions coinciding when the functionalities of platform components are exposed as services [4], are relevant interoperability dimensions for this study. Also, for the purposes of this paper, *interoperability* can be considered to be synonymous with *integration*, i.e, the ability of diverse services offered by the same or multiple providers to seamlessly interoperate to aid the development and deployment of Cloud applications [1], [5]. This study proposes to focus on 'service/platform' interoperability, assessing the scope of integration of multiple services offered by disparate providers through semantic description of PaaS services, and trying to resolve their incompatibilities to address the interoperability issue.

The rest of the paper is structured as follows: Section 2 presents the background and related work. Section 3 proposes an interoperability framework for PaaS. Section 4 describes the tool and technologies to be used to design and implement the proposed framework. Section 5 discusses future direction of this work and Section 6 concludes.

II. BACKGROUND AND RELATED WORK

The extant literature highlights the need for reducing interoperability barriers to foster an open market for its players and indicates a research gap in the area of interoperable PaaS solutions requiring simultaneous and/or interchangeable use of multiple PaaS services and capabilities [1], [2] to support the various phases of the Cloud application lifecycle. Di Martino [6] states that a contributing factor to the PaaS interoperability issue is the disparity in the offered services' semantics, i.e., non-uniform representation of services. This calls for a mechanism that establishes standardization of service descriptions leveraging devised domain ontology. This is likely to enable efficient service retrieval based on a semantic match between description of services being sought and description of services offered by PaaS providers. The use of agents (representing service requesters and providers) in this can facilitate automated retrieval of the required services [7]. The following establishes the needed background of this work through an overview of a selection of industry projects and academic research efforts relevant to the research area.

Cloud PaaS interoperability is an active research area that several ongoing research projects both in the industry and academia are trying to address. Some of the recent PaaS interoperability initiatives by standardization groups, industry and the research community include efforts in devising:

i) *Standard APIs* (e.g., Cloud Application Management for Platforms or CAMP) that attempt to homogenize and provide generic platform-agnostic PaaS APIs and *common APIs* (Cloud4SOA and PaaS Manager) that harmonize heterogeneous APIs through abstractions.

ii) *Open-source APIs* (e.g., OpenShift, Cloud Foundry, Heroku and Docker) that aim to become de-facto standards by providing access to source code.

iii) *Agnostic Patterns* (e.g., CloudPatterns.org community) that enable interoperation through detection of similarities among services on the basis of their logical descriptions [4].

iv) *Model-Driven Engineering* (e.g., MODAClouds, Artist and PaaSage projects) that enables platform-agnostic design of Cloud applications.

However, these initiatives are mainly focused on enabling application 'portability' through either facilitating platform-agnostic code or migration of application code and data among platforms. As opposed to this general PaaS interoperability, PaaS 'semantic interoperability' enables PaaS offerings to address semantic incompatibilities (by devising a common vocabulary using ontology) and communicate. The primary semantic interoperability conflicts are specific to the levels of *management* (owing to proprietary API functions), *service* (resulting from proprietary platform services) and *data* [8]. The initiatives aligned to PaaS *semantic interoperability* are discussed here in Table I.

TABLE I. ANALYSIS OF PAAS INTEROPERABILITY/PORTABILITY INITIATIVES

Initiative	Domain	Agent-based	Ontology-based	Scope
Cloud4SOA [9]	PaaS (API)		✓(OWL 2)	Abstractions among PaaS offerings
Androcec et al. [10]	PaaS (API)		✓(OWL)	Resources
Parhi et al. [11]	IaaS (service)	✓	✓(OWL)	Description of Cloud service providers and their attributes
Di Martino et al. [12]	PaaS (service)		✓(OWL-S)	Semantic description (Service Bus queue service of MS Azure)
mOSAIC [6]	IaaS, PaaS (API)	✓	✓(OWL & OWL-S)	Intelligent discovery of Cloud services
Cloudle [13]	IaaS, PaaS, SaaS	✓	✓(no formal ontology language)	Cloud services discovery and similarity reasoning

As can be inferred from the table, PaaS semantic interoperability initiatives can be broadly classified into two primary categories, semantic frameworks and multi-agent systems, as relevant here;

A. Semantic frameworks (ontology-based initiatives)

Some of the ontology-based initiatives and approaches that aim to address the PaaS interoperability challenge through efforts to resolve semantic ambiguities of PaaS APIs include the following:

The Cloud4SOA project aims to devise a distributed reference marketplace architecture consisting of five layers including a Semantic layer as the backbone and a Distributed Repository layer that intermediates between the platform and existing PaaS offerings [14]. The ontology language selected for the implementation of the semantic layer of this reference architecture is an extended Web Ontology Language (OWL 2). To support the developed ontology model, Cloud4SOA provides a harmonized API and provider-specific adapters to bridge their disparities [9].

The work by Androcec et al. [10] proposes a novel PaaS API ontology for a shared understanding of the main concepts of the APIs of Cloud providers. The developed ontology using Web Ontology Language (OWL) defines several classes and categories of operations related to PaaS API resources.

While the two initiatives discussed above focus on semantics of PaaS APIs, they fail to explore the *service* (disparities in service descriptions) level of PaaS semantic interoperability. The literature [6] claims that differences in provider services owing to proprietary terms and semantics and a lack of uniform representation of services is one of the contributing factors to the interoperability issue. The literature [15], [16] also observes that principled definition of PaaS services is a vital requirement for achieving service interoperability among heterogeneous PaaSs to facilitate service discovery and composition. This definition can be achieved through a taxonomy or ontology devised for PaaS services (domain ontology) and annotation of specific service instances on the basis of the domain ontology. This is still an incipient area of research.

Some of the initiatives focused on enabling service/platform semantic interoperability, and therefore directly relevant for this work include:

A work by Parhi et al. [11] presents a semantic-multiagent based service description and discovery prototype to resolve Cloud services' dissimilarities and to assist in the retrieval of appropriate services. The authors outline a search process using multiple agents and mention implementation of the entire system (including negotiation and service composition) as planned future work. However, the initiative is focused on the IaaS layer of the Cloud stack (setting up the object and data properties of Amazon EC2). This study aims to extend the work by adopting the general methodology with suitable modifications to assist PaaS service description and discovery phases, besides implementing PaaS service composition.

A more PaaS-specific effort for service/platform interoperability can be found in a recent initiative by Di Martino et al. [12] which proposes a semantic description of a Microsoft Azure Cloud service (namely the Service Bus messaging service) based on the powerful expressiveness of the Web Ontology Language for Services (OWL-S). The authors comment that this semantic representation of a PaaS provider service is the first step in reconciling differences in cross-platform concepts and in enabling automatic discovery and composition of Cloud services. Also that enriched with proper inference rules and description of a number of providers' services, it has the potential to evolve into a knowledgebase of a framework supporting inter-Cloud interoperability and portability. This is a research direction that has not received much attention yet.

This study intends to build on and extend the last two approaches discussed above to devise a solution for Cloud PaaS service/platform interoperability.

B. *Multi-Agent systems*

The first initiative relevant here is the Open-source API and Platform for multiple Clouds (mOSAIC) that encompasses both IaaS and PaaS layers. The Cloud Agency of this system evidences how agent-based technologies (in conjunction with a semantic model) can alleviate interoperability and portability issues [6]. The Cloudle project [13] represents another multi-agent effort that proposes an agent-based search engine for Cloud service discovery utilizing Cloud ontology.

However, these multi-agent initiatives fail to devise a semantic representation of individual services offered by platforms, which is important from an automatic reasoning perspective that enables mapping between provider-specific services based on the identified similarities. Also, the devised multi-agent systems are either generic or specific to the IaaS service model and/or hybrid deployment model [12], presenting a scope for designing a dedicated multi-agent system (using semantics synergistically) for the PaaS service model.

A categorization of services to support easy discovery, coupled with an abstract semantic and computable description of services enabling automated comparison and mapping (based on machine-processable meaning) between the provider services and their composition [12] has the potential to resolve the interoperability issue. In line with this, the next section proposes a PaaS-specific semantic-agent interoperability model.

III. PROPOSED PAAS INTEROPERABILITY FRAMEWORK: INTPAAS

This section presents the proposed interoperability framework (IntPaaS), which aims to integrate ontology and multiple agents for service description and discovery. This will comprise of a definition of the domain model of the PaaS layer (functional concepts, attributes and relations) and semantic annotation of PaaS services' functionalities (using the OWL-S ontology language) coupled with an agent-based search mechanism (aimed at automatic discovery and composition) to determine similarities and resolve heterogeneities among multiple public aPaaS services.

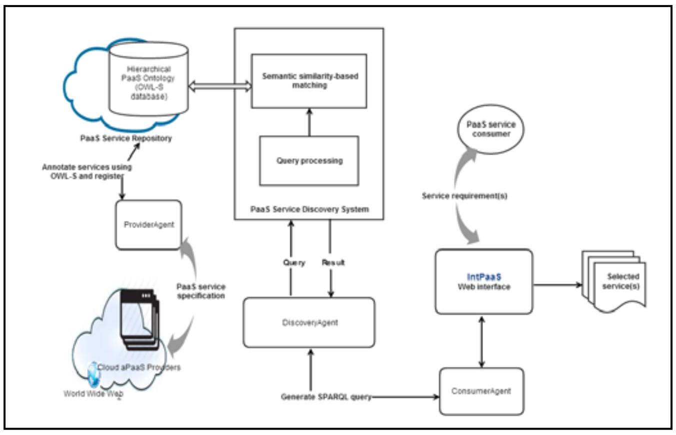

Fig. 1. The IntPaaS semantic-agent Architecture

A. *System Architecture*

The overall architecture of the proposed IntPaaS framework is presented in Fig. 1. The system comprises of various components and software agents assisting the description and discovery of appropriate PaaS services. The general approach to be undertaken towards the development and the primary components of the framework are:

1) *Service Description and Storage Component*

The fundamental element of the proposed semantic-agent architecture for PaaS service discovery and composition is a shared knowledge plane comprised of a semantic model and reasoning rules to govern agent interactions. The two main tasks involved in the development of the semantic model are: i) definition of a PaaS domain ontology (OWL-S classes, properties, etc.) for functional description and systematic classification in a taxonomic hierarchy, and ii) creation of an OWL-S description (consisting of *Service*, *Profile* and *Process* classes) for specific service instances. The semantically annotated service descriptions will be mapped to the domain ontology [17] to resolve semantic dissimilarities. OWL-S, a unified knowledge representation language of the semantic Web, will be leveraged for semantic markup of services as it enables suitable ontological representations to be used by agents [10].

The ProviderAgent plays a central role in this, assisting in the annotation of syntactic service specifications obtained from the PaaS providers into OWL-S ontological (semantic) descriptions and registering the service to a PaaS Service Repository (PSR), an OWL-S Resource Description Framework (RDF) repository.

2) Service Search Component

The first step involved in this is a request generated at the Web interface for a specific service by a PaaS service consumer. Following this, a request for service (RFS) [11] will be generated and passed on to the service matching component of the architecture.

The ConsumerAgent will accept user queries (PaaS service requests) and pass them on to the DiscoveryAgent, maintaining communication with the latter through Agent Communication Language (or ACL) based on SPARQL. The automated matching and retrieval of services will be carried out by the DiscoveryAgent, in conjunction with the PaaS Service Discovery System (PSDS).

3) Service Matching and Retrieval Component

The next step constitutes matching and retrieval of appropriate services to meet user requirements. The registered semantic descriptions (annotated pre-condition and post-condition parameters and service operations) of services will be leveraged and functionality-based similarity analysis will be carried out. This will involve logic reasoning to establish semantic relevance of a provider service to the requested service. The result of the search will be communicated by the DiscoveryAgent to the ConsumerAgent using messages based on SPARQL.

The proposed framework will adopt a layered approach for service search. The two tiers of search include:

i) SPARQL query processing – The DiscoveryAgent will facilitate discovery of the requested PaaS service(s) from the PSDS by processing the received service request(s) to generate a suitable search criteria (in the form of a SPARQL query) for easy processing by reasoners.

ii) Service matching – The PSDS search engine will interface with the PSR, consulting and reasoning (based on semantic similarity) about the semantically annotated services and returning the result (in XML) to the DiscoveryAgent; enabling dynamic and autonomous discovery of suitable PaaS services. The matchmaking mechanism in the PSDS will utilize the service *Profile* of the developed ontology, querying the RDF repository with SPARQL.

The final step, i.e., service composition, will involve an interoperability evaluation. A prototype (to be developed using Java) will be implemented to map to the services of some prominent public PaaS solutions to assess their semantic interoperability. The framework will be evaluated through the composition of relevant retrieved services for the development of a Cloud application, establishing the scope of using other similar services interchangeably in the process.

IV. Implementation – Tools and Technologies

The tools and technologies to be used for the design and implementation of the proposed framework include:

A. Experiment Platform

Some of the available options for this are Cloud9 IDE, CodeRun Studio, Eclipse Orion. However, the Java language is not supported in these IDEs. The online tools that support Java for coding in the Cloud include Ideone and eXo Cloud IDE. As Ideone is more of a pastebin (supporting compilation and debugging) that does not permit the creation of projects [18], the experimental platform chosen for the project is the eXo Cloud IDE.

B. Ontology Language

OWL-S, an upper ontology which is based on the W3C standard OWL ontology and used to describe the semantics of services, will be leveraged to create the ontology for the proposed framework. OWL-S has the dual function of introducing ontology to describe the concepts of services' domain and generic concepts for describing the services themselves and the way they relate to the domain ontology via Input, Output, Precondition and Effects. The expressive semantics of this ontology for services enables semantically rich descriptions facilitating automated machine reasoning over services and domain descriptions, enabling intelligent discovery, invocation and composition of services. The ontology is constituted of three main classes, with a *Service* class providing a point of reference for these classes: i) *Profile*, describing the functionality of a service (through the functional properties of *hasParameter*, *hasInput*, *hasOutput*, *hasPrecondition* and *hasResult* along with non-functional attributes such as serviceName, serviceCategory, textDescription, etc.) that assists in advertising and discovering services; ii) *Process* (atomic or composite), informing how the service works, providing information that enables service-seeking agents to perform a deeper analysis of the services for their composition; and iii) *Grounding*, specifying how the service is activated (through details on port numbers, communication protocols, message formats, etc.), thereby mapping an 'abstract' grounding (service profile and process) to a 'concrete' grounding (service API) [19], [12]. These classes together facilitate the use of services by agents.

C. Ontology Editor

The ontology for the framework will be implemented using Protégé Desktop 4.3.0 (build 304) ontology editor (an OWL-specific IDE), along with Java Virtual Machine (JVM)1. However, an issue is that OWL-S, which makes use of OWL Full constructs, is not natively supported by Protégé. Although this makes the use of OWL-S in Protégé difficult, the issue can be addressed through the use of the OWL-S editor (Build 24)2 as a Protégé plug-in. GraphViz3 graph visualization software will be used to represent structural information of the developed ontology.

D. *Other APIs and tools*

Other APIs and tools to be used for development of the framework include:

i) **JADE 3.4 agent system** – The Java Agent Development Framework, which assists in the implementation of multi-agent systems4, will be used for developing the agents. The configuration of the JADE-based system will be controlled via the Web interface of the proposed framework.

ii) **Apache Jena 2.5.5 Semantic Web framework**5 – This API will be used to facilitate interconnection of the ProviderAgent and the OWL-S knowledge base.

iii) **SPARQL 1.1 Query Language**6 – SPARQL, which is recommended by W3C as the standard Query Language for OWL (and is supported by many tools including Protégé and Jena), will be leveraged to query RDF data stored in the PSR. The tool to assist in this is ARQ, a SPARQL 1.1 compliant engine supported by Jena.

These tools and APIs selected for implementation of the architecture are the most prominent ones for the intended purposes and are Java-compliant.

V. Future Direction

This study proposes to establish separate semantic descriptions of services to establish a knowledgebase enabling semantic matching of heterogeneous application infrastructure services of prominent public aPaaS providers. A list of important terms derived from some of the existing PaaS services will be used to represent the classes of the ontology and will then be organized into a hierarchical taxonomy to represent the PaaS domain ontology. The base concepts of the domain ontology will be leveraged for semantic annotation of PaaS service instances, to resolve their incompatibilities and facilitate opportunistic use of suitable differentiated PaaS provider services by multiple agents. Service matchmaking will involve the integration of a semantic algorithm that operates on service functionality.

The work by Di Martino et al. [12] has devised a semantic description (using OWL-S ontology) for the *Service Bus* service of Windows Azure platform. However, the work does not compare it with similar services provided by some of the other existing PaaS provider, and thus fail to extend it to explore the interoperability dimension. This study proposes to explore the opportunity of establishing interoperability between the offered services of some of the public aPaaS providers through synergistic use of the devised domain ontology for PaaS and agents that are able to reason over and access functionally similar services from the knowledgebase.

1 http://protegewiki.stanford.edu/wiki/P4_3_Release_Announcement
2 http://projects.semwebcentral.org/projects/owlseditor/
3 http://www.graphviz.org/
4 http://jade.tilab.com/
5 https://jena.apache.org/
6 https://www.w3.org/TR/2013/REC-sparql11-query-20130321/

VI. Conclusion

Cloud application PaaS has the potential to lead to a new paradigm of Cloud application development. In spite of this, its wider adoption has been hindered by a lack of interoperability between provider services coupled with the unavailability of comprehensive services supporting the Cloud application lifecycle. Interoperability is an important concern and an active area of research, with various initiatives in the industry and academia trying to address it. However, the proposed solutions focused on API standardization, Model-driven Engineering, Semantic Framework, Multi-agent system, etc. have failed to resolve this issue. In view of the fact that PaaS service/platform interoperability is an area awaiting further progress, this work proposes an interoperability architecture based on semantic (encompassing PaaS services' description using OWL-S) and multi-agent (encompassing population of semantic service repository, service search and composition) technologies as the first step to an effective solution. The synergistic union of these two technologies is, to the best of our knowledge, a novel approach in the PaaS service/platform interoperability space. The study is ongoing and will be extended through the development and testing of a proof-of-concept implementation in the future to investigate the technical challenges involved in enabling this interoperability.

Acknowledgment

Our thanks to IEEE eXpress Conference Publishing for allowing us to modify the template they have developed.

References

[1] E. Kamateri, N. Loutas, D. Zeginis, J. Ahtes, F. D'Andria, S. Bocconi, P. Gouvas, G. Ledakis, F. Ravagli, O. Lobunets, and K. A. Tarabanis, "Cloud4SOA: A semantic-interoperability paas solution for multi-cloud platform management and portability," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8135, Berlin Heidelberg: Springer, 2013, pp. 64–78.

[2] M. Pezzini and B. J. Lheureux. (2011, Mar 07). Integration Platform as a Service: Moving Integration to the Cloud. *Gartner, Inc.* [Online]. Available: https://www.gartner.com/doc/1575414/integration-platform-service-moving-integration. (URL)

[3] A. Rafique, S. Walraven, B. Lagaisse, T. Desair, and W. Joosen, "Towards portability and interoperability support in middleware for hybrid clouds," in *Conf. 2014 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS),* Toronto, ON, 2014, pp. 7–12.

[4] B. Di Martino, G. Cretella, A. Esposito, "Cloud Portability and Interoperability," in *Cloud Portability and Interoperability: Issues and Current Trends*, 1st ed. Springer International Publishing, 2015, ch. 1, sec. 1.2, pp. 01-14.

[5] V. K. Dileep and R.V. Sivabalan., "Challenges in Achieving Interoperability in Cloud Computing," *Aust. J. Basic & Appl. Sci.*, vol. 9, no. 16, pp. 36-43, 2015.

[6] B. Di Martino, "Applications Portability and Services Interoperability among Multiple Clouds," *IEEE Cloud Computing*, vol. 1, no. 1, pp. 74–77, May. 2014.

[7] S. Bromuri, V. Urovi, M. Morge, K. Stathis, and F. Toni, “A multi-agent system for service discovery, selection and negotiation,” in *Proc. Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, Dubrovnik, Croatia, 2009, vol. 2, pp. 1423–1424.

[8] N. Loutas, E. Kamateri, K. Tarabanis, and D’Andria, F., (2011). D1.2 Cloud4SOA Cloud Semantic Interoperability Framework. [Online]. Available: http://www.cloud4soa.com/sites/default/files/D1.2_Cloud4SOA%20Cloud%20Semantic%20Interoperability%20Framework.pdf. (URL)

[9] A. Corradi, L. Foschini, A. Pernafini, F. Bosi, V. Laudizio, and M. Seralessandri, “Cloud PaaS Brokering in Action: The Cloud4SOA Management Infrastructure,” in *Conf. 2015 IEEE 82nd Vehicular Technology Conference (VTC2015-Fall)*, Boston, MA, 2015, pp. 1–7.

[10] D. Androcec, N. Vrcek, and P. Kungas, “Service-Level Interoperability Issues of Platform as a Service,” in *Congr. 2015 IEEE World Congress on Services*, 2015, pp. 349–356.

[11] M. Parhi, B. K. Pattanayak, and M. R. Patra, “A Multi-agent-Based Framework for Cloud Service Description and Discovery Using Ontology,” in *Advances in Intelligent Systems and Computing*, vol. 308, India: Springer, 2015, pp. 337–348.

[12] B. Di Martino, G. Cretella, A. Esposito, and R. G. Sperandeo, “Semantic Representation of Cloud Services: A Case Study for Microsoft Windows Azure,” in *Conf. 2014 International Conference on Intelligent Networking and Collaborative Systems*, Salerno, Italy, 2014, pp. 647–652.

[13] K. M. Sim, “Agent-Based Cloud Computing,” *IEEE Trans. Services Computing*, vol. 5, no. 4, pp. 564–577, Nov. 2012.

[14] F. D'Andria, S. Bocconi, J. G. Cruz, J. Ahtes, and D. Zeginis, “Cloud4SOA: Multi-cloud Application Management Across PaaS Offerings,” in *Symp. 2012 14th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*, Timisoara, Romania, 2012, pp. 407–414.

[15] K. Stravoskoufos, A. Preventis, S. Sotiriadis, and E. G. M. Petrakis (2014) “A survey on approaches for interoperability and portability of cloud computing services,” 2014, pp. 112–117.

[16] F. Paraiso, N. Haderer, P. Merle, R. Rouvoy, and L. Seinturier, “A Federated Multi-cloud PaaS Infrastructure,” in *Conf. 2012 IEEE Fifth International Conference on Cloud Computing*, Honolulu, HI, 2012, pp. 392–399.

[17] D. Elenius. (2005). The OWL-S Editor – A Domain-Specific Extension to Protégé [Online]. Available: http://protege.stanford.edu/conference/2005/submissions/abstracts/accepted-abstract-elenius.pdf. (URL)

[18] L. M. Gadhikar, L. Mohan, M. Chaudhari, P. Sawant, and Y. Bhusara, “Browser based IDE to code in the cloud,” in *Advances in Intelligent Systems and Computing*, vol. 203, S. Patnaik, P. Tripathy and S. Naik, Ed. Berlin Heidelberg: Springer, 2013, pp. 59–69.

[19] D. Elenius. (2004). Modeling Services with Protégé [Online]. Available: http://protege.stanford.edu/conference/2004/posters/Elenius.pdf. (URL)