Applications Deployment in Multiple PaaS Environments: Requirements, Challenges and Solutions

Rami Sellami[1], Mehdi Ahmed-Nacer[2,3] and Stéphane Mouton[1]
[1]*Software and Services Technologies Department, CETIC, Charleroi, Belgium*
[2]*Computer Science Departement, Institut Mines-Telecom, Telecom SudParis, CNRS UMR Samovar, University of Paris-Saclay, Evry, France*
[3]*University of Sciences and Technology Houari Boumediene, Algiers, Algeria*

Keywords: Cloud Computing, Plateform-as-a-Service (PaaS), Multiple PaaS Deployment, Cloud Services Discovery, Cloud Services Selection, Applications Migration.

Abstract: Cloud computing has recently attracted full attention of many organizations due to its economic, business and technical benefits. Indeed, we observe that the proliferation of offers by cloud providers raises several challenges. One of these innovative challenges is applications deployment in multiple PaaS providers. In fact, developers need to provision components of the same application across multiple PaaS depending on their related requirements and PaaS capabilities. They will not only have to deploy their applications, but they will also have to consider migrating services from one PaaS to another, and to manage distributed applications spanning multiple environments. In this paper, we present and discuss the requirements of applications deployment in multiple PaaS providers and we analyze current state of the art.

1 INTRODUCTION

Cloud Computing, a relatively recent term, has become nowadays a buzzword in the Web applications world. It is defined by the National Institute of Standards and Technology (Mell and Grance, 2009) as a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, etc.) that can be quickly provisioned in an elastic manner that allows to automatically scale up or down in line with demand. In addition, these resources are released with minimal management effort or service provider interaction. Cloud Computing is characterized by its economic model referred to as "pay-as-you-go" model. This latter allows users to consume computing resources as needed.

Services in the Cloud Computing are basically delivered under three well discussed layers namely the Infrastructure as a Service (IaaS), the Platform as a Service (PaaS) and the Software as a Service (SaaS). We present these layers below:

- **IaaS**: Consumers are able to access Cloud resources on demand. These resources can be of type virtual machines, storage and networks. The provider is responsible for installing, transparently managing and maintaining these resources.
- **PaaS**: Consumers are able to develop, deploy and manage their applications into the Cloud using libraries, editors and services offered by the provider. It supports provisioning, managing and maintaining the Infrastructure resources.
- **SaaS**: Consumers are able to use running applications on an IaaS or a PaaS through an interface. They are not responsible for managing or maintaining the provisioned Cloud resources.

Over the past years, a plethora of commercial solutions and research projects have been sought to improve security, scalability, elasticity and interoperability in the IaaS level. Nevertheless, very few works deal with the PaaS and SaaS levels even if they are as important as the IaaS level. Hence, we propose to focus in this document on the PaaS level. A PaaS provider is supposed to support the whole requirements of an application during its life cycle while maintaining Cloud features (e.g. scalability, elasticity, high availability, etc.). However, applications requirements change frequently. Thus, it seems illusory to find a single PaaS provider that efficiently supports various applications with different requirements in terms of applications life cycle management.

Sellami, R., Ahmed-Nacer, M. and Mouton, S.
Applications Deployment in Multiple PaaS Environments: Requirements, Challenges and Solutions.
DOI: 10.5220/0006370906640671
In *Proceedings of the 7th International Conference on Cloud Computing and Services Science (CLOSER 2017)*, pages 636-643
ISBN: 978-989-758-243-1

This forces an application to migrate to another PaaS provider. Moreover, we can find an application with very specific requirements that a single PaaS provider does not have the necessary capabilities to support it.

To circumvent this obstacle, an application may be deployed in multiple PaaS providers and benefits from various PaaS providers capabilities at the same time. For instance, an application can use the Redis database service from DEIS and the middleware Tomcat of Microsoft Azure. In this paper, we focus on existing solutions of the state-of-the-art supporting applications deployment in multiple PaaS providers. More precisely, our contributions are (i) describing different scenarios related to the way applications are deployed in a single or multiple PaaS providers, (ii) defining a set of requirements in terms of applications deployment in a single or multiple PaaS provider(s), and (iii) analyzing and classifying the reviewed works.

The rest of this paper is organized as follows. Section 2 presents the multiple PaaS deployment requirements. In sections 3, 5, and 4, we analyze the state of the art and we present the list of the considered criteria and a synthesis of this analysis. Section 6 provides a conclusion and some open issues.

2 MULTIPLE PaaS DEPLOYMENT REQUIREMENTS

The PaaS is the development, deployment and execution environment of an application in the Cloud. It is devised to provide utmost benefits to the developers and to get rid them of the onerous task of managing applications life cycle. Indeed, it enables high scalability and on demand cloud services provisioning by automating its allocation/ deallocation to optimally support the requirements of an application. In addition, it ensures a high level of availability and reliability thanks to the load balancing and the failover of services capabilities with respect to the application requirements. It is also characterized by the automation in order to ensure an end-to-end support for the applications lifecycle. Last but not least, it allows developers to use multiple operating systems, multiple run-time and frameworks, and multiple databases to collaborate with other developers. Against this background, we can emphasize a first multiple PaaS deployment requirement which is:

- R_0: Ensuring scalability, elasticity and portability

In order to introduce the other requirements, we present two possible use cases of the PaaS level that a developer may encounter. In Figure 1, we showcase the first use case which is applications deployment in a single PaaS. In this case, the application can only allocate the services of this PaaS. In this figure, we illustrate three applications deployed in a single Cloud environment and they are provisioning with three services. Based on this, we stress the problem which is to discover and to decide if the capabilities of a given service are convenient for an application requirements or not. In addition, applications are very dynamic and frequently change its requirements in a Cloud environment. Hence, they are obliged to allocate new services. In most of the cases, an application is forced to migrate to a new Cloud environment. In this context, we identify the following requirements:

- R_1: Choosing a PaaS provider based on Applications requirements
  - R_{11}: Exposing the application requirements
  - R_{12}: Capturing PaaS providers capabilities
  - R_{13}: Matching between application requirements and PaaS providers capabilities
- R_2: Migrating an application from one PaaS to another

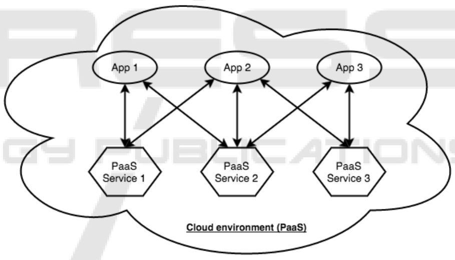

Figure 1: Scenario 1 - Application deployment in a single PaaS provider.

Whereas the second use case matches the applications deployment in multiple PaaS providers (see Figure 2). For instance, the application *App1* uses two services that are *service 1* from the Cloud environment where it is deployed and *service 2* from another Cloud environment. We do not deny that this approach is very interesting since it enables to optimally support applications requirements. It allows taking benefit from the variety of the PaaS offerings. It also provides the developers with more alternatives to satisfy their needs and optimize their applications operation. In addition, it offers a PaaS-independent mechanisms which considerably harmonize and simplify the provisioning procedures of application components across several PaaS providers. Nevertheless, deploying an application in multiple PaaS providers is a cumbersome and requires a painstaking work. Based on this, we add the following requirement:

- R_3: Easy deployment to single or multiple PaaS

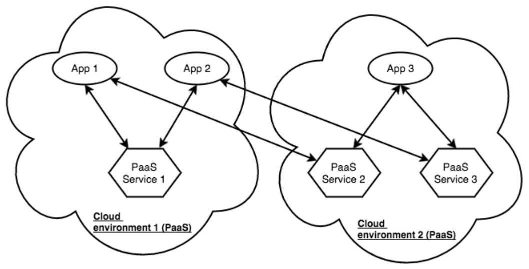

Figure 2: Scenario 2 - Application deployment in multiple PaaS providers.

In the rest of the paper, we will analyze the current state of the art regarding these four requirements. We will mainly focus on the requirements R_1 with its three sub-requirements and R_3. We do not specify a section to the requirement R_0 since we suppose that is implicitly ensured through the other requirements.

3 CHOOSING A PaaS PROVIDER BASED ON APPLICATIONS REQUIREMENTS

Several works deal with the problem of the discovery and selection of services in Cloud environments. Generally, these contributions enable developers to describe their applications requirements using a very specific model (e.g. a manifest, a SLA-based model, a metric-based model, etc.). Then, they propose a set of mechanisms enabling the selection of the most appropriate PaaS provider to deploy the application.

Sellami et al. (Sellami et al., 2015) propose an approach to automatically discover Cloud resources in a PaaS environment. Indeed, developers express their requirements in an abstract application manifest. Then, their matching algorithm discovers the PaaS providers capabilities that are exposed in an offer manifest. It computes the distance between the two manifests in order to elect the most appropriate PaaS provider to the application requirements. In their solution, authors mainly focus on the discovery of data services and interest to single PaaS discovery.

Redl et al. (Redl et al., 2012) present an automatic approach to check if the elements of the SLA are valid or not. Doing so, they map each PaaS SLA to an ontology and they define a matching algorithm to compare the SLAs. Results of this matching are analyzed in order to select the most appropriate services provider. Although the idea of selecting Cloud services based on the SLA is interesting, this approach does not enable application requirements description in terms of deployment in one or multiple PaaS providers.

Garg et al. (Garg et al., 2011) present SMICloud which is a framework enabling the comparison between two Cloud providers based on users requirements. In fact, SMICloud computes a set of metrics referred to as Service Measurement Indexes (SMI). The results of these SMIs are used by users in order (1) to compare the different offers of Cloud environments, (2) to classify them using a predefined order, (3) to evaluate their Quality of Service (QoS), and (4) to select the most appropriate one to their requirements. Inspite of the completeness of SMICloud, it is intended to work in the IaaS level.

Wittern et al. (Wittern et al., 2012) propose a model based solution to select services in a Cloud environment in order to express users requirements and services capabilities. For this purpose, they define an algorithm that select services using a set of predefined criteria. Nevertheless, this approach does not support the multiple PaaS discovery.

In the SeaClouds project (Athanasopoulos et al.,), authors propose an open source platform to support applications in a multiple clouds environments. Indeed, one of its key component is the SeaClouds Discoverer which enables to discover available capabilities and add-ons offered by available cloud providers. It allows to declaratively select multiple Cloud services based on the QoS expressed by the user. A matching algorithm is implemented in order to select the Cloud provider corresponding to the QoS required by the user.

Quinton et al. (Quinton et al., 2014; Quinton et al., 2016) propose SALOON which is a platform for automatically selecting and configuring cloud environments. Their solution is based on the functional and non-functional application requirements in order to select the most appropriate cloud environment to an application. The key ingredient of their solution is the use of the Software Product Lines and the Features Models to represent the Cloud variability.

Li et al. (Li et al., 2010; Ang Li, 2010) define an automatic solution to select the most appropriate PaaS provider to an application requirements. This solution is referred to as CloudCmp. It compares the performance and the cost of cloud providers in terms of computing, storage, and networking resources using benchmark tasks. It aims to elect the cloud provider that has the best performance and the less cost. It supports four cloud providers that namely are Amazon AWS, Microsoft Azure, Google AppEngine, and Rackspace CloudServers.

Kang et al. (Kang and Sim, 2011; Kang and Sim, 2016) propose an agent-based solution to discover cloud resources using ontologies. These ontologies are referred to as CO-1 and CO-2, and enable to semantically describe resources and the relationship between each others. The user requirements are defined in the form of classAds. Authors introduce the discovery process using four stages: the selection, the evaluation, the filtering, and the recommendation.

Qu et al. (Qu et al., 2013) propose a model for Cloud services selection by aggregating both (1) the feedback issued from cloud users and (2) the objective performance benchmark testing from a trusted third party. Their solution integrates four key components: The Cloud Selection Service takes into charge a request for services selection reception and preprocessing. The Benchmark Testing Service which designs a set of testing scenarios to evaluate the performances metrics. The User Feedback Management Service that is in charge to collect and manage users feedback. The Assessment Aggregation Service which aggregates the user feedback and the resulting performance values in order to evaluate cloud services scores and weight.

Han et al. (Han et al., 2009) propose a cloud services selection framework based on a recommender system with respect to their requirements. Indeed, the recommender system define ranks to the different services and the user should select the appropriate ones. The rank is evaluated using the QoS and the analysis of resources provided by cloud providers (using quality of virtualization, allocation cost, users feedback, etc.). It is noteworthy that authors propose to select multiple services from multiple cloud providers.

Table 1: Synthesis of the related works about multiple PaaS providers discovery.

Criteria / Studied Solutions	R_{11}	R_{12}	R_{13}	**Multiple PaaS**	**Declarative**	**Automatic**
Sellami et al. (Sellami et al., 2015)	+	+	+	−	+	+
Redl et al. (Redl et al., 2012)	−	+	+	−	+	+
SMICloud (Garg et al., 2011)	+	+	+	−	+	+
Wittern et al. (Wittern et al., 2012)	+	+	+	−	+	+
SeaClouds (Athanasopoulos et al.,)	+	+	+	+	+	−
SALOON (Quinton et al., 2016) (Quinton et al., 2014)	+	+	+	+	+	+
Li et al. (Li et al., 2010), (Ang Li, 2010)	+	+	+	−	+	+
Kang et al. (Kang and Sim, 2011), (Kang and Sim, 2016)	+	+	+	−	+	+
Qu et al. (Qu et al., 2013)	+	+	+	−	+	+
Han et al. (Han et al., 2009)	+	+	+	+	+	+

Synthesis

In this section, we present a synthesis about the presented works above. To do so, we fix a set of six criteria on which we rely to evaluate these works. We check whether these works propose a solution to (1) describe application requirements, (2) expose PaaS providers capabilities and (3) apply matching techniques to elect the most suitable PaaS provider to an application. We also verify if these works use a model to automatically and declaratively dicover and select PaaS providers. Finally, we point out that we target applications in multiple PaaS providers. In Table 1, we check for each work, whether it responds or not to our criteria. We use $+$ to denote that the proposed work treats a given criteria. While, we use $-$ to show that the related work does not propose a solution for a given criteria. It is worth noting that we use this notation throughout the upcoming sections to synthesis the studied works.

Against this analysis, we conclude that these works support the requirement R_{11}, R_{12}, and R_{13}. In addition, each solution has its own modeling in order to ensure applications requirements and PaaS providers capabilities description in a declarative way. However, apart from some works, the rest does not support multiple PaaS providers discovery and selection.

4 EASY DEPLOYMENT TO SINGLE OR MULTIPLE PaaS PROVIDERS

Today, we find several solutions enabling application provisioning and deployment in Cloud environments. However, the majority of these works focus on the IaaS level. For instance, we can cite mOSAIC[1], PerfCloud (Mancini et al., 2009), and PaaSage[2]. Indeed, mOSAIC is a european project and proposes a solution enabling the portability and the interoperability of data, services, and applications in different IaaS providers. This project is based on a set of negotiation mechanisms, contracts and ontologies allowing to find Cloud services supporting applications requirements. We also introduce the PaaSage european project which proposes a model driven engineering based solution to model applications requirements and IaaS capabilities and to deploy applications in multiple IaaS providers. Although the completeness and the performance of these solutions, it does not

[1]http://www.mosaic-cloud.eu/

[2]http://www.paasage.eu/

support the PaaS level. Nevertheless, we find some approaches to deploy applications in the PaaS. In fact, we cite Cloud4SOA (D'Andria et al., 2012; Kamateri et al., 2013), PaaSHopper (Paraiso et al., 2012; Walraven et al., 2015), ANEKA (Wei et al., 2011), COAPS (Sellami et al., 2013), SeaClouds (Athanasopoulos et al.,), Nucleus (Kolb and Röck, 2016; Röck and Kolb, 2016), PaaS Manager (Cunha et al., 2014) and Spinnaker [3].

Cloud4SOA (D'Andria et al., 2012; Kamateri et al., 2013) is a FP7 european project enabling to interconnect heterogeneous PaaS providers that share the same technologies. To do so, they use an ontology to cover the heterogeneity between a set of software components and models. Hence they ease the task of developers and enhance platform providers with a number of core capabilities. It is noteworthy that this solution only supports SOA-based applications.

PaaSHopper (Paraiso et al., 2012; Walraven et al., 2015) is a middleware platform enabling to develop and deploy SaaS-based applications in multiple PaaS providers. In fact, developers declaratively describe their application features and requirements in terms of PaaS. Then, PaaSHopper selects the appropriate PaaS providers to execute the application and store its data.

ANEKA (Wei et al., 2011) is a framework for building applications based on users requirements and deploying them on multiple PaaS providers. It supports private and public providers. It includes four key components. Indeed, the Aneka Master and the Aneka Worker taking into charge the deployment process. Whereas the Aneka Management Console and the Aneka Client Libraries take into charge the application development.

The Compatible One Application and Platform Service API (COAPS API) (Sellami et al., 2013) allows human and/or software agents to provision and manage PaaS applications. This API provides a unique layer to interact with any Cloud provider based on manifests. It manages two kinds of resources: (1) the environment which is a set of settings required by developers to host and run their applications, and (2) applications which are computer software or program deployable in a Cloud environment. Recently, COAPS API has been extended to support multiple PaaS providers deployment.

The SeaClouds project (Athanasopoulos et al.,) enables to deploy applications in multiple and heterogeneous Clouds (i.e. IaaS and PaaS). Indeed, it is based on two main components. The first one is referred to as the SeaClouds Planner which generates an orchestration plan with respect to the application requirements and topology. Whereas the second one is the SeaClouds Deployer that executes the deployment plan in one or multiple Clouds provider.

Nucleus (Kolb and Röck, 2016; Röck and Kolb, 2016) is a unified interface enabling applications deployment and management in PaaS providers. It is based on the REST architecture. This interface covers the heterogeneity between the proprietary APIs and models of each PaaS provider.

PaaS Manager (Cunha et al., 2014) is a framework to integrate and aggregate various public PaaS providers based on their similarities. Doing so, this framework includes a unique API in order to reduce the vendor lock-in between PaaS providers and applications developers. Indeed, it enables to create, deploy, start, stop, delete and monitor an application. It includes a REST API to automatically deploy an application in a PaaS provider using Git. It is noteworthy that this solution supports only four PaaS providers and enables single PaaS deployment.

Spinnaker (Spinnaker, 2017) is an open source solution enabling to prepare environment based on multiple clouds and to continuously deploy and manage applications. It involves two key component. The management component to manage resources in the cloud and the deployment management component to construct and manage continuous delivery workflows.

Synthesis

In this section, we present a synthesis of the works that we have analyzed above (see Table 2). Indeed, we investigate whether these solutions enable an easy deployment in multiple PaaS providers. This will allow taking benefit from the variety of the PaaS offerings. It also provides the developers with more alternatives to satisfy their needs and optimize their applications operation. In addition, we check if these solutions are declarative. Indeed, this criteria is ensured using a unified model and generic provisioning mechanisms. PaaS-independent mechanisms will considerably harmonize and simplify the provisioning procedures of applications components across several PaaS. Finally, we are interested in the works that enable automatic provisioning of applications across several PaaS providers. This simplifies the deployment process and alleviates the burden of this task imposed on developers.

Against this analysis, we conclude that the majority of the studied works support multiple PaaS. However, Cloud4SOA and SeaClouds partially meets the second requirement related to the automatic provisioning of applications across several PaaS. Cloud4SOA supports only provisioning of applications that are designed according to SOA specifi-

[3]http://www.spinnaker.io/

Table 2: Synthesis of the related works about easy deployment to multiple PaaS providers

Studied Solutions \ Criteria	Declarative	Automatic	Multiple PaaS
Cloud4SOA (Kamateri et al., 2013; D'Andria et al., 2012)	+	+/−	+
PaaSHopper (Walraven et al., 2015; Paraiso et al., 2012)	−	+	+
ANEKA (Wei et al., 2011)	−	−	+
COAPS API (Sellami et al., 2013)	+	+	+
SeaClouds (Athanasopoulos et al.,)	+	+/−	+
Nucleus (Kolb and Röck, 2016; Röck and Kolb, 2016)	−	+	+
PaaS Manager (Cunha et al., 2014)	−	+	−
Spinnaker	+	+	+

cations and SeaClouds for TOSCA specifications. PaaSHopper, Nucleus and PaaS Manager are designed for supporting multi-PaaS and they meet the automatic deployment through an API and they do not provide any declarative model and/or provisioning operations. Hence, hey do not meet the first requirement. COAPS API and Spinnaker provide an API for application deployment across several PaaS offerings. Both works meets requirements related to the need for declarative model and automatic deployment.

5 MIGRATING AN APPLICATION FROM ONE PaaS TO ANOTHER

Applications migration consists in moving already deployed applications from one PaaS provider to another. It is done in two main steps: (1) moving the application data and (2) moving and re-deploying the application itself to the new PaaS provider. In the literature, several studies on cloud migration have been achieved. The majority of existing cloud migration studies are focusing on the provisioning or on the migration of legacy software to the Cloud (e.g. MODAClouds (Ardagna et al., 2012), CloudMIG (Frey and Hasselbring, 2010),etc.). None of these are actually dealing with the migration between PaaS providers. In the rest of this section, we present few works dealing with the PaaS level.

Beslic et al. (Beslic et al., 2013) propose a Model-Driven Architecture and Refactoring based approach to migrate an application between PaaS providers. Their contribution includes three steps: the vendor discovery, the application transformation, and the deployment. However, authors remain only to a high-level concept of the migration process and do not give any details about the implementation level of their approach.

Cloud Motion Framework (CMotion) (Binz et al., 2011) ensures applications migration into and between clouds. It enables to reduce the vendor lock-in for applications already hosted in a given cloud. Based on the application's topology, authors show how adapt existing software to use different PaaS on the Cloud services. Each of the components is migrated while taking into account the relations and dependencies of the components to the others.

ConPaaS (Pierre and Stratan, 2012) is a run-time environment for elastic applications hosting and deployment on multiple PaaS providers. The services are offered through abstractions that hide the real implementations and can be instantiated on multiple clouds. Applications in ConPaaS are composed of services programmed through a common generic Python interface for the service management and a Javascript part to extend the front-end GUI with service-specific information and control.

Migration Assessment Tool (MAT) (Sharma et al., 2013) enables to assess a software application for migration to different popular cloud platforms. This approach is based on (1) analyzing the dependencies of the source code of an application taking into account their technical services and capabilities and (2) evaluating the support for those services and capabilities in the target PaaS platform(s). These capabilities and services are gathered in a set of repositories.

Other existing works use the containerization and virtualization technologies (e.g. Docker, Linux-VServer, OpenVZ, LXC, and Rock) to enable distributed applications deployment across several PaaS (Dua et al., 2014) (Kolb et al., 2015). These approaches consists on packaging the applications components dynamically in a generated service containers. The generated service containers implement the applications components requirements. Then, the containers are pushed to the cloud as standalone applications. Today, various PaaS solutions are based on the container technology and they mainly use Docker coupled with Kubernates. For instance, we can cite Flynn, DEIS, AppFog solutions. Applications in these PaaS solutions are based on the same technologies and standards. Hence, the migration process will be easier.

Synthesis

In this section, we elaborate a synthesis about the discussed works above. Doing so, we fix our criteria based on the classification of Andrikopoulos et al. (Andrikopoulos et al., 2013) that identify four possible types of applications migration. Indeed, we will check what is the type of application migration supported by the considered approaches. First, we have the type I that consists on replacing the application components with new PaaS services. Then, we have the type II which represents the case of partially migration. Indeed, some of the application components are migrated to another PaaS provider. Afterward, we have the type III that consists on migrating the whole software stack of the application to a PaaS provider. This is the classic example of migration to the Cloud, where for example the application is encapsulated in VMs and ran on the Cloud. Finally, we have the type IV which is a complete migration of the application. The application functionality is implemented as a composition of services running on the Cloud. Based on this classification, we showcase in Table 3 the migration methods mentioned above and we associate each one to the appropriate migration type.

Table 3: Synthesis of the related works about migrating an application from one PaaS to another.

Criteria / Studied Solutions	**Type I**	**Type II**	**Type III**	**Type IV**
Beslic et al. (Beslic et al., 2013)	−	+	+	−
CMotion (Binz et al., 2011)	−	+	+	−
ConPaaS (Pierre and Stratan, 2012)	+	−	+	−
MAT (Sharma et al., 2013)	−	+	+	−
Dua et al. (Dua et al., 2014)	−	+	+	+
Jörg et al.(Kolb et al., 2015)	−	+	+	+

To sum up, apart from ConPaaS, all the studied works ensure the **Types II** and **III** of application migration in PaaS levels. However, only ConPaaS take into account the **Type I** since they can replace component by Cloud services. Indeed, it proposes six services for hosting the applications (web and database). Whereas the works published in (Dua et al., 2014) and (Kolb et al., 2015) are considered of **Type IV** since they ensure the migration of the whole application functionality by using the containerization.

6 CONCLUSION

In this paper, we have defined and classified requirements of application deployment in multiple PaaS providers through two possible scenarios. We have identified four requirements (i.e. $R_1, R_2, R_3, and R_4$). Regarding each requirement, we have analyzed the current state of the art and stressed the strenghts and the limitations of each solution. As a result, we summarized the different solutions in three tables. In fact, we have defined for each table a set of criteria taking into account the existing lock-in in the PaaS market and developers expectations (i.e. to easily and seamlessly deploy and manage their applications in multiple PaaS providers).

We are currently working to address part of these requirements. Indeed, we are focusing on proposing an automatic and declarative solution (1) to discover and to select services, and (2) to deploy applications in multiple PaaS providers.

ACKNOWLEDGMENT

This work has been funded by the Belgian research project MoDePaaS identified by the convention number 1610385.

REFERENCES

Andrikopoulos, V., Binz, T., Leymann, F., and Strauch, S. (2013). How to adapt applications for the cloud environment. *Computing*, 95(6):493–535.

Ang Li, Xiaowei Yang, S. K. M. Z. (2010). Cloudcmp: Shopping for a cloud made easy. In *2nd USENIX Workshop on Hot Topics in Cloud Computing*. USENIX.

Ardagna, D. et al. (2012). Modaclouds: A model-driven approach for the design and execution of applications on multiple clouds. In *Proceedings of the 4th international workshop on modeling in software engineering*, pages 50–56.

Athanasopoulos, D. et al. Seaclouds: Agile management of complex applications across multiple heterogeneous clouds. In *STAF Projects Showcase*, pages 54–61.

Beslic, A., Bendraou, R., Sopenal, J., and Rigolet, J. (2013). Towards a solution avoiding vendor lock-in to enable migration between cloud platforms. In *Proceedings of the 2nd International Workshop on Model-Driven Engineering for High Performance and CLoud computing co-located with 16th International Conference on Model Driven Engineering Languages and Systems MODELS'13, Miami, Florida, USA, September 29*, pages 5–14.

Binz, T., Leymann, F., and Schumm, D. (2011). Cmotion: A framework for migration of applications into and between clouds. In *IEEE International Conference on Service-Oriented Computing and Applications, SOCA'11, Irvine, CA, USA, December 12-14*, pages 1–4.

Cunha, D., Neves, P., and de Sousa, P. N. M. (2014). Paas manager: A platform-as-a-service aggregation framework. *Comput. Sci. Inf. Syst.*, 11(4):1209–1228.

D'Andria, F., Bocconi, S., Cruz, J. G., Ahtes, J., and Zeginis, D. (2012). Cloud4soa: Multi-cloud application management across paas offerings. In *14th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, SYNASC'12,* *Timisoara, Romania, September 26-29*, pages 407–414.

Dua, R., Raja, A. R., and Kakadia, D. (2014). Virtualization vs containerization to support paas. In *The IEEE International Conference on Cloud Engineering, IC2E'14*, pages 610–614.

Frey, S. and Hasselbring, W. (2010). Model-based migration of legacy software systems to scalable and resource-efficient cloud-based applications: The cloudmig approach.

Garg, S. K., Versteeg, S., and Buyya, R. (2011). Smicloud: A framework for comparing and ranking cloud services. In *IEEE 4th International Conference on Utility and Cloud Computing, UCC'11, Melbourne, Australia, December 5-8*, pages 210–218.

Han, S.-M., Hassan, M. M., Yoon, C.-W., and Huh, E.-N. (2009). Efficient service recommendation system for cloud computing market. In *Proceedings of the 2Nd International Conference on Interaction Sciences: Information Technology, Culture and Human*, ICIS '09, pages 839–845. ACM.

Kamateri, E. et al. (2013). Cloud4soa: A semantic-interoperability paas solution for multi-cloud platform management and portability. In *Service-Oriented and Cloud Computing - Second European Conference, ESOCC, Málaga, Spain, September 11-13*, pages 64–78.

Kang, J. and Sim, K. M. (2011). Towards agents and ontology for cloud service discovery. In *2011 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, CyberC'11, Beijing, China, October 10-12*, pages 483–490.

Kang, J. and Sim, K. M. (2016). Ontology-enhanced agent-based cloud service discovery. *IJCC*, 5(1/2):144–171.

Kolb, S., Lenhard, J., and Wirtz, G. (2015). Application migration effort in the cloud - the case of cloud platforms. In *8th IEEE International Conference on Cloud Computing, CLOUD'15, New York City, NY, USA, June 27 - July 2*, pages 41–48.

Kolb, S. and Röck, C. (2016). Unified cloud application management. In *IEEE World Congress on Services, SERVICES'16, San Francisco, CA, USA, June 27 - July 2*, pages 1–8.

Li, A., Yang, X., Kandula, S., and Zhang, M. (2010). Cloudcmp: comparing public cloud providers. In *Proceedings of the 10th ACM SIGCOMM Internet Measurement Conference, IMC'10, Melbourne, Australia - November 1-3*, pages 1–14.

Mancini, E. P., Rak, M., and Villano, U. (2009). Perfcloud: GRID services for performance-oriented development of cloud computing applications. In *18th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises, WETICE'09, Groningen, The Netherlands, 29 June - 1 July*, pages 201–206.

Mell, P. and Grance, T. (2009). The NIST definition of cloud computing. *National Institute of Standards and Technology*, 53(6):50.

Paraiso, F., Haderer, N., Merle, P., Rouvoy, R., and Seinturier, L. (2012). A federated multi-cloud paas infrastructure. In *2012 IEEE Fifth International Conference on Cloud Computing, Honolulu, HI, USA, June 24-29*, pages 392–399.

Pierre, G. and Stratan, C. (2012). Conpaas: a platform for hosting elastic cloud applications. *IEEE Internet Computing*, 16(5):88–92.

Qu, L., Wang, Y., and Orgun, M. A. (2013). Cloud service selection based on the aggregation of user feedback and quantitative performance assessment. In *The IEEE International Conference on Services Computing, Santa Clara, CA, USA, June 28 - July 3*, pages 152–159.

Quinton, C., Romero, D., and Duchien, L. (2014). Automated selection and configuration of cloud environments using software product lines principles. In *The IEEE 7th International Conference on Cloud Computing, Anchorage, AK, USA, June 27 - July 2*, pages 144–151.

Quinton, C., Romero, D., and Duchien, L. (2016). SALOON: a platform for selecting and configuring cloud environments. *Softw., Pract. Exper.*, 46(1):55–78.

Redl, C., Breskovic, I., Brandic, I., and Dustdar, S. (2012). Automatic SLA matching and provider selection in grid and cloud computing markets. In *13th ACM/IEEE International Conference on Grid Computing, GRID'12, Beijing, China, September 20-23*, pages 85–94.

Röck, C. and Kolb, S. (2016). Nucleus – unified deployment and management for platform as a service. Technical report, University of Bamberg.

Sellami, M., Yangui, S., Mohamed, M., and Tata, S. (2013). Paas-independent provisioning and management of applications in the cloud. In *2013 IEEE Sixth International Conference on Cloud Computing, Santa Clara, CA, USA, June 28 - July 3*, pages 693–700.

Sellami, R., Vedrine, M., Bhiri, S., and Defude, B. (2015). Automating resources discovery for multiple data stores cloud applications. In *CLOSER'15 - Proceedings of the 5th International Conference on Cloud Computing and Services Science, Lisbon, Portugal, 20-22 May*, pages 397–405.

Sharma, V. S., Sengupta, S., and Nagasamudram, S. (2013). Mat: A migration assessment toolkit for paas clouds. In *The IEEE Sixth International Conference on Cloud Computing, Santa Clara, CA, USA, June 28 - July 3*, pages 794–801.

Spinnaker (2017). Spinnaker website. http://www.spinnaker.io/.

Walraven, S., Landuyt, D. V., Rafique, A., Lagaisse, B., and Joosen, W. (2015). Paashopper: Policy-driven middleware for multi-paas environments. *J. Internet Services and Applications*, 6(1):2:1–2:14.

Wei, Y., Sukumar, K., Vecchiola, C., Karunamoorthy, D., and Buyya, R. (2011). Aneka cloud application platform and its integration with windows azure. *CoRR*, abs/1103.2590.

Wittern, E., Kuhlenkamp, J., and Menzel, M. (2012). Cloud service selection based on variability modeling. In *Service-Oriented Computing - 10th International Conference, ICSOC'12, Shanghai, China, November 12-15*, pages 127–141.