Addressing the Challenge of Application Portability in Cloud Platforms

Fotis Gonidis, Iraklis Paraskakis, Dimitrios Kourtesis

South-East European Research Centre (SEERC),
City College – International Faculty of the University of Sheffield,
24 Proxenou Koromila Street, 54622 Thessaloniki, Greece
{fgonidis, iparaskakis, dkourtesis}@seerc.org

Abstract. Cloud computing is a relatively new paradigm that promises to revolutionize the way IT services are provided. There are multiple benefits that companies can gain from cloud computing. However, there still remain a number of issues to be solved before this new computing paradigm is widely adopted. In this paper we introduce the issues of portability and interoperability and we focus on the cross-platform portability of applications. We present some high level approaches and existing work that try to address this issue. Finally we briefly propose some future research directions towards investigating how to improve the portability of applications across cloud platforms.

Keywords: Cloud Computing, Portability, Interoperability

1 Introduction

Cloud computing is a relatively new paradigm, where computing is offered as a utility, and has the potential to transform a large part of the IT industry [1]. According to the US National Institute of Standards and Technology (NIST), cloud computing is a "model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction" [2].

Due to the fact that cloud computing refers to provisioning of heterogeneous resources, ranging from hardware (storage, processing power) to software (software development platforms, applications), it has been further decomposed into various service layers. According to NIST there are three service layers [3] which are shown in Figure 1. The bottom service layer is the Infrastructure as a Service (IaaS). In IaaS, basic computing resources like storage, processing and networking are provisioned. Amazon is a major IaaS vendor, providing computing resources via virtual machines. The middle service layer is the Platform as a Service (PaaS). In PaaS the consumers are given access to software tools, APIs and programming languages in order to develop and deploy their applications. Example of a PaaS provider is Google, which via Google App Engine provides developers with programming runtime environments, storage options, custom services, deployment capabilities etc. The top

level service layer is the Software as a Service (SaaS). In this service layer, complete and ready-to-be-used software applications are provisioned. SalesForce is a major SaaS provider offering Customer Relationship Management (CRM) Software.

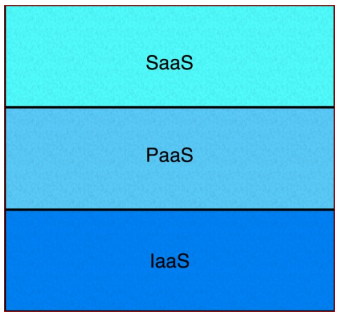

Fig 1. The three service layers of cloud computing

Cloud consumers, whether they are acting on IaaS, PaaS, or SaaS layer can benefit a lot from this computing paradigm. Using cloud computing, consumers have ubiquitous access to their data and applications from all over the world. The only requirement is access to the internet. Consumers no longer need to worry about installing, maintaining and upgrading the software and/or hardware that they use. This burden has now moved entirely to the cloud providers. Another benefit comes from the characteristic of cloud computing to rapidly provision and release resources. Consumers can have the resources that they need available at any given time and pay only for what they use. Therefore they don't need to buy upfront software or hardware that they may never use.

While cloud computing is rapidly emerging and offers several benefits for consumers, there are still a number of issues to be addressed before it becomes widely adopted by companies and organizations. Among others, important issues are portability and interoperability across cloud platforms. As Figure 2 shows, the first refers to the ability to move a service from one cloud provider to another, while the latter one refers to the ability of two services from different clouds to exchange information.

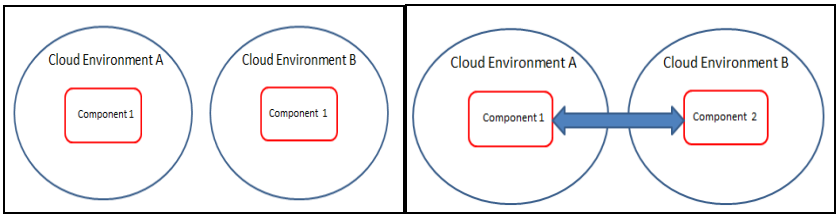

Fig 2. Left: a component is moved from one environment to another (portability). Right: two components of different cloud environments are collaborating (interoperability).

As will be presented in the next section, portability and interoperability affect all three service layers (IaaS/PaaS/SaaS). In this paper we are particularly interested in

the aspect of portability within the context of PaaS, namely the feasibility of a cloud application to be ported across different platforms. Major cloud platform providers like, Google App Engine and Microsoft Azure are currently using proprietary technologies and/or techniques that prevent the portability of cloud applications. This issue plays an important role in the wide adoption of cloud computing as it may discourage consumers from using cloud platform services.

The aim of this paper is to introduce portability and interoperability as concepts, discuss their relationship, and present some high level approaches for addressing the challenge of portability in the context of PaaS. The presented approaches vary from defining common standards, creating APIs, abstract/wrap proprietary ones, to developing open source platforms. Our research goal is to explore how to improve the portability of cloud applications, and the first objective towards this goal is to understand how platform-independence can be enhanced by employing Model Driven Engineering (MDE) and Ontologies.

The rest of the document is organized as follows. In the next section the terms portability and interoperability are introduced and it is explained why they are important issues for the wide adoption of cloud computing. Next, we focus on the issue of portability in the context of the PaaS layer. We discuss high level approaches towards enabling portable cloud applications, followed by an overview of existing work in this field. Finally, in section 4 we present some future research directions.

2 Portability and Interoperability issues in Cloud Computing

Prior to focusing on the aspect of portability in the context of PaaS, it is necessary to introduce the concept of portability in the context of cloud computing and explain how it differs from that of interoperability.

2.1 Portability in cloud computing

NIST refers to portability as the ability "of prospective cloud computing customers to move their data or applications across multiple cloud environments at low cost and minimal disruption" and particularly to system portability as "the ability to migrate a fully-stopped Virtual Machine (VM) instance or a machine image from one provider to another provider" [3]. Two important characteristics of portability can be extracted from this definition. Firstly, the move from one cloud to another should be achieved at the lowest possible cost, effort, and time. Secondly, portability refers to the ability to move any component of any of the three service layers across cloud platforms.

According to Cloud Security Alliance (CSA) [4], there are different portability requirements at the three different cloud service layers. In IaaS, the requirement is to be able to easily port the Virtual Machines (VMs) and data from one vendor to another. For instance, a company or organization operating several VMs in one cloud infrastructure provider should be able to easily move them to another provider. In PaaS, the requirement is to be able to deploy applications across different platforms. For example, if a developer creates and deploys an application on a certain cloud platform, it should be feasible for the application to be ported to a different platform,

with a minimal set of changes, if any. Finally, in SaaS, the requirement, when switching from one software application to another is to be able to extract the data from the first and upload it to the second. For example, if a company uses a CRM application provided by a cloud vendor and decides to switch to a different offering, it is important that all customer data can directly be loaded and processed by the new CRM application.

Consumers, whether they interact with the IaaS, PaaS or SaaS layer need to be able to easily change between cloud providers and be free to choose the one that better serves their needs in terms of quality and/or cost. The ability of consumers to easily migrate from one cloud service provider to another is even more critical in case a cloud provider's operation is unexpectedly terminated. A real example to illustrate this argument is the case of Coghead [5] – an online application development platform supporting the development and hosting of data-driven applications. The company had managed to attract several hundreds of developers before it suddenly announced that it would stop operating, calling all customers to export the data stored in their applications, but not giving them the option to port the applications themselves.

Ensuring portability across cloud providers would eliminate the vendor lock-in problem [6] and would allow consumers to switch between vendors according to their needs. In turn, this would increase consumers' trust towards cloud computing and public cloud services.

2.2 Interoperability in cloud computing

In the IEEE glossary [7], interoperability is defined as "the ability of two or more systems or components to exchange information and to use the information that has been exchanged". According to Petcu [8], one can find several definitions of cloud interoperability in the literature. For example, interoperability has been defined as the ability to "abstract the programmatic differences from one cloud to another", the ability to "translate between the abstractions supported by different clouds", to "flexibly run applications locally or in the cloud or in a combination", or to "use same management tools, server images, software in multiple clouds".

Interoperability affects all three service layers and there are specific requirements in each one of them. In IaaS, interoperability may refer to the ability of a client to seamlessly use infrastructure resources from different vendors through a common management API [9]. For instance, a consumer could be able to perform the same set of operations on VMs from different providers (e.g. start, stop or delete) without creating a different client for each of the providers. In PaaS, developers may need to use tools, libraries or APIs coming from different PaaS providers to create their applications. For example a cloud application may be composed of various cloud services coming from different cloud vendors. Finally, in SaaS, interoperability lies in the ability of different applications to exchange information. For example as Dillon et al. [10] mention, a company might want to outsource the email service to Google and the Human Resource Management (HRM) service to SalesForce. This means that the data format in the e-mail system (e.g. calendar, address book) needs to be compatible with the HRM service.

Lack of interoperability puts a barrier to using and combining solutions from different providers, but also to allowing on-site systems to collaborate and exchange information with cloud services. Enhancing interoperability among cloud providers would enable two ways of collaborations. Firstly, services from various cloud providers could be seamlessly combined to provide best of breed solutions with respect to quality, price, or features. Secondly, companies would be able to "push" to the cloud only part of their services while keeping the most critical ones on–site.

2.3 Relationship between portability and interoperability

In 2.1 and 2.2 portability and interoperability were presented as distinct notions. Nevertheless, it is not uncommon for the term "interoperability" to be used for denoting both concepts. As noted by Petcu [8], authors sometimes define interoperability as the ability to "move applications from one environment to another or run in multiple clouds", or the ability to "move services, processes, workloads, and data between clouds". According to the definition we provided in section 2.1, this is clearly a case of portability. In Dowell et al. [18], portability is referred to as a special kind of interoperability challenge. Seen from this perspective, the inconsistent use of the term "interoperability" to denote both concepts can be justified.

As proposed by Petcu [8], it is perhaps useful to think in terms of vertical and horizontal interoperability. Horizontal interoperability can be defined as the ability of two cloud services of the same service layer (IaaS/PaaS/SaaS) to communicate with each other – a definition consistent with our notion of interoperability as discussed in section 2.2. On the other hand, vertical interoperability can be defined as the ability of a cloud service to be deployed on a cloud service of a lower service layer. For example, allowing a SaaS application to be deployed on various PaaS offerings. This definition is consistent with our notion of portability as discussed in section 2.1, i.e. the ability of an application to be ported across various cloud platforms.

3. Portability in the context of Platform as a Service

As mentioned in section 2.1, there are portability requirements across all three service layers (IaaS/PaaS/SaaS). The scope of this research, however, is to explore the issue of application portability at the level of PaaS.

3.1 Cloud portability issues at platform level

Cloud platforms promise to ease and speed up the application development cycle by offering a complete set of tools for developing, deploying, hosting and maintaining the application. There is presently a broad choice of providers of PaaS, such as Google App Engine, Microsoft Azure, and Force.com, who offer a wide range of services for application development, including file and data storage, messaging, queuing, workload management, analytics etc [20]. The combination of the large number of platform offerings and the variety of cloud services contributes to the

portability challenge, as it makes moving an application across cloud platforms even more complex.

The above statement becomes clearer if we consider that the cloud services, according to each provider, may use different technologies and be offered to clients by various proprietary APIs. For example, Microsoft provides the SQL Azure database for data storage, while Google App Engine provides, among others, the App Engine DataStore. On top of that, application portability may be hindered by the fact that certain cloud services offered by one platform are not available in another. For example, the mailing service offered by Google App Engine may not be offered by any other provider. Thus, the heterogeneity of today's platform offerings contributes significantly to the challenge of application portability.

3.2 General approaches for addressing cloud portability in PaaS

There are some generic approaches and strategies that could be adopted in order to tackle the issues outlined in section 3.1, and eventually ease application portability across platforms.

One obvious approach is the definition of common set of standards for PaaS offerings. The adoption of such standards by all cloud providers would enable developers to create their applications independently of specific platform environments and then deploy them to the cloud platform of their choice. This set of standards could include a standardized API to access the service offered by the platform, standard formats for representing file structures, standard data stores, etc.

Standardization seems to be a very efficient approach to achieve cloud portability. However, for reasons not necessarily related to technology, it is very difficult for all cloud platforms to eventually agree on a common set of standards. All major cloud vendors use proprietary APIs and file formats as a way of locking-in customers to their services. The effort required to re-engineer an application in order for it to be ported to another platform is discouraging customers to move. In addition, a set of common standards would prevent platform providers from offering the special, platform-specific features that allow vendors to differentiate from their competitors.

Another approach towards achieving portability between platforms is intermediation. That is, introducing an intermediate layer that decouples application development from specific platform APIs and supported formats. In this case developers create their applications using an intermediate API which is platform agnostic and can "hide" or "wrap" the proprietary APIs of particular vendors. The intermediate layer prevents developers from being bound to specific programming languages, file formats or data stores. For example an application could be developed in a language-independent manner, and later on, through model transformations, be translated into the particular programming language supported by a PaaS provider (such as Java, Python or C#), or the database query language particular to a platform (e.g. Microsoft SQL or MySQL).

In this case, of introducing an intermediate layer for decoupling application development from specific platforms, no consensus by platforms vendors is required. However, the challenging part here is to develop the translation rules and the model transformations between the intermediate layer and each platform vendor specifically.

3.3 Existing works addressing cloud portability at platform level

There are several existing works that try to address the issue of portability across cloud platforms, by adopting one or combining the two approaches presented in section 3.2.

1) *mOSAIC.* mOSAIC is a framework that promises to ease application portability across platforms by providing a set of APIs that are independent of vendors [12], [13]. At design-time, developers are using these APIs to create applications that consist of multiple cloud components, each one performing a certain function. A cloud component can, for example, be a Java application. At this point the application is not bound to any specific platform. Then, at runtime, the mOSAIC platform decomposes the application into the various cloud components and deploys each one on the cloud platform that provides the best implementation for the cloud component's functionality. Communication between components deployed in different platforms is achieved via cloud based message queues technologies [8]. The code for connecting the application components to a concrete platform provider is generated by mOSAIC. Therefore developers can focus on developing their applications in a platform-neutral manner, and later on, they can decide on which cloud provider they wish to deploy them. The mOSAIC API acts as an intermediary layer between the developers and the actual cloud platforms, and developers do not have to use proprietary APIs of the target could platforms. Therefore mOSAIC could be classified as an intermediation approach that tries to decouple application development from particular platform technologies.

   An example application, as described in Petcu et al [13], could be a check out service for buying products online. The application could be split into four core operations: a) retrieving user payment details and product list, b) calculating the total amount to be charged to the user's credit card, c) charging the credit card by contacting the bank, d) saving the transaction details. At development time, each of these operations is assigned to a cloud component, and during run-time each component can be deployed on a platform that best performs the operation.

2) *Open Cloud Computing Interface (OCCI):* OCCI is a set of specifications that allow the development of tools for performing common cloud tasks like deployment, autonomic scaling and monitoring across different cloud service providers. It offers an API that is supported by various cloud computing stacks, like Open Eucalyptus, OpenNebula, and OpenStack. Therefore, OCCI could be classified as a standardization approach. It should be mentioned that OCCI started as an API for managing cloud infrastructure. However, as it is stated in its official website, the Open Cloud Computing Interface will eventually also serve other layers in addition to IaaS, i.e. PaaS and SaaS [19].

3) *PaaS Semantic Interoperability Framework (PSIF).* The PSIF framework proposed by Loutas et al. [14] aims at modelling semantic interoperability conflicts that may occur during migration or deployment of an application on a cloud platform. The framework is structured according to 3 dimensions, (i) the different architectural entities in a PaaS environment, (ii) the type of semantics of a PaaS entity's description i.e. functional, non-functional and execution

semantics, (iii) and the level at which semantic conflicts occur, i.e. the level of the information model and the level of data. Semantic conflicts are identified and classified according to these 3 dimensions.

Loutas et al. [15], provide two examples of the framework's operation. In the first example, an application is ported from one platform to another. A conflict arises when the application is trying to connect to a database, because the two platforms use different function calls (e.g. "connect a db" vs. "insert a db"). This semantic conflict occurs due to differences in the definitions of the management interfaces of the two platforms and specifically due to the way their functional semantics are modelled. The conflict is raised at the data level, since it is caused by different naming of the same functionality. Another semantic conflict may occur due to differences in the modelling of the PaaS offerings. For example, one provider uses a field "programming language" to describe both the language and the version, e.g. Java 1.6, while another platform offering uses two different fields. In this case the same term has different meaning in each platform. The conflict is raised due to differences in the semantic models of the PaaS offerings and specifically to the way the non-functional semantics are modelled. The conflict occurs at the information model level, since it is caused by different logical representation of the same information. In a similar way, other semantic conflicts which may occur while moving applications across platforms are classified.

Having modelled in detail the fundamental PaaS entities in a particular PaaS offering, a semantic layer will be implemented to provide a PaaS Offering Model and an Application Model for the common description of available PaaS offerings [15]. PaaS providers will be able to publish their offerings based on these common models. By letting providers adopt a common model for their offerings the application portability across the platforms will be enhanced. According to our understanding and the available literature on the PSIF, we could classify this work as an approach of defining a set of common standards.

4) *SimpleCloud.* SimpleCloud is an API that allows developers to use storage services independently of particular cloud platforms. Among others, it offers two key services: (i) *File Storage Service* and (ii) *Document Storage Service.* The File Storage Service allows for performing operations on files such as storing, reading, deleting, copying, storing metadata, etc. It does so by providing so-called "storage service adapters" that allow developers to access storage services from Amazon, Microsoft Azure, Rackspace and others, using the same application code. The Document Storage Service abstracts the interfaces of all major document databases, again allowing developers to access different providers through a single API. SimpleCloud is supported by IBM, Microsoft, Rackspace, GoGrid, and several other cloud service providers. By offering an API for storing data which abstracts/hides all proprietary ones, SimpleCloud can be considered as an intermediate layer for decoupling applications from directly accessing the storage mechanisms of specific platforms.

4. Future research directions

This research work is still at an initial stage and is carried out in the context of the first author's doctoral research. As it was already mentioned in section 3, the research scope is to explore and propose a framework for improving the cross-platform development and deployment of cloud applications.

Today, there is a wide range of available platform offerings that a cloud developer can choose from. Each platform may have unique characteristics and use certain technologies, tools, APIs etc. Moreover the provided services may vary significantly across platforms. Therefore, it is not feasible to achieve a single generic solution that could enable applications to be ported across all available cloud platforms without any modifications. A realistic approach would be to focus on a certain set of platform offerings, services, technologies, as in the previously mentioned cases of SimpleCloud, which supports PHP and focuses on abstracting cloud storage mechanisms, or mOSAIC, whose scope is to support Java and Python applications.

Therefore the major research question at this stage of our work is to define and map the area within the context of PaaS that this research will focus on. In order to answer this question there are several sub-questions that need to be addressed first. Firstly, a survey of the available platform offerings needs to be performed. Afterwards, these platforms need to be examined and analyzed in order to extract the platform components and characteristics of each one of them. Then it would be feasible to examine what are the specific conflicts, and at which level they occur while porting an application across platforms (e.g. an obvious one is the programming language while moving from Microsoft Azure to Google App Engine, since the first primarily supports C# while the latter supports Java and Python). After forming a clear idea about the various platform offerings and the characteristics of the cloud platform, we will able to narrow down our research effort to a specific set of platform vendors and platform characteristics that are prone to conflicts when porting an application. At the same time, an extensive survey of related work in the field should be performed. An analysis of this work should be followed to identify their exact contribution and also their limitations. This will improve our understanding of the field and enable us to better define our research direction and focus.

In section 3.2 we presented two generic approaches for enhancing application portability. The first one involved the definition of a common set of standards that need to be adopted by platform vendors in order to provide uniform services. This approach however, presupposes the agreement of major platform providers and organizations, and is therefore beyond the scope of this research work. Thus, our intention is to explore the second approach which proposes an intermediary layer for decoupling the development of applications from specific proprietary technologies. This approach introduces the notion of platform-independence, where the application is initially developed in a platform-agnostic way. A research area and domain of technology which is deemed highly probable to have a positive contribution towards achieving platform independence is Model Driven Engineering (MDE).

MDE allows developers to create applications independently of a target platform, by first creating a platform-agnostic model. This intermediate model "hides" specific characteristics of platforms that could cause conflicts when deploying an application. The intermediate model allows developers to avoid binding applications to particular

programming languages or to a specific database query languages. Subsequently, by performing automated model transformations, the intermediate models can be translated into platform-specific ones. The key characteristic of MDE, which is to allow developers to work with abstract models, has several benefits. According to Esparza-Peidro and Munoz-Escoi [21], it allows developers to focus on their applications and ignore minor details linked to specific platforms. They can avoid low-level and error-prone platform features and can more efficiently communicate with each other. Therefore the overall design and implementation process can be dramatically improved.

A most relevant benefit in relation to our particular research focus is the abstract nature of models, which has the potential to improve cross-platform development and deployment of applications by decoupling the development from specific platform technology. For example, we could think of an application that requires the use of a noSQL database. OpenShift [23], the PaaS offering from Red Hat, supports, among others, mongoDB. Google App Engine [24] on the other hand provides a noSQL database called App Engine Datastore. A conflict is raised when the application developer needs to move the application from one platform to another. MDE, in this case, could allow the developer to model the database as an abstract model. Later on, through automatic model transformations, the abstract model could be mapped on a specific database implementation according to the target platform. Therefore we intend to explore what the contribution of using MDE techniques could be in creating cloud applications that will be more independent of the targeted platforms.

On top of using MDE, in this research we are particularly interested in exploring whether involving ontologies could contribute to improving cloud portability at the level of PaaS. Ontologies, according to Gruber [22], are a "formal explicit specification of a shared conecptualization". In other words, ontologies can provide a common vocabulary and understanding over a particularly domain. Specifically in PaaS layer, the use of a common acceptable ontology can lead to more uniform offerings from PaaS providers` side. Therefore application portability can be improved. The PSIF, presented in 3.3, is inspired from this notion and aspires to provide such an ontological model for PaaS offerings. This is one approach of how ontologies could contribute to the issue of cloud application portability. Another interesting research direction to explore is how ontologies can be combined with MDE and what the potential benefits of such a combination could be in the field of application portability.

Gasevic et al. [16], as well as Happel and Seedorf [17], support the view that combining ontologies with model-driven engineering during software application development has many benefits. Ontologies provide support for logical inference and if combined with software models, such as UML, they can enable reasoning over the models used in model-driven engineering. The ability to perform reasoning and inferring new knowledge from software models motivate us to explore the benefits of combining MDE and ontologies.

In the area of cloud platforms, Ranabahu and Seth exploit the combination of MDE and ontologies. MDE is used in order to abstract platform specific details while ontologies enrich the models with more information. For example, in a demo application they define a component to perform a certain action that needs to be

secured with the "ssl" protocol. In order to add this piece of information, they semantically annotate the model using an available security profile ontology.

Therefore it is obvious that it is not the first time that ontologies are used to tackle the issue of portability in cloud applications. However to the best of our knowledge, the combination of MDE and ontologies for enabling cloud portability at PaaS model is an area that has not yet been thoroughly explored. This fact further motivates our research direction.

5. Conclusions

In this paper, we introduced the challenges of portability and interoperability in cloud computing and provided a brief overview of portability and interoperability issues specific to each of the three cloud service layers (IaaS, PaaS, and SaaS). The focus of our ongoing research work is to address the challenge of application portability in the context of PaaS, i.e. exploring ways to enable applications to be deployed across various cloud application platforms. In this context, we gave a high level description of the factors that may have an impact on application portability, such as proprietary APIs and differences in the functionality across different platform providers. We have also described the two high level strategies that can be employed to address the challenge of portability: standardization and intermediation. Standardization addresses cross-platform portability through the adoption of common standards by cloud providers. Alternatively, intermediation enables developers to create applications independently of a specific platform and then bind them to particular target platforms through some form of automatic translation. In the next steps of our research we will focus on exploring how to improve application portability by employing an intermediation strategy that combines elements of model-driven engineering and ontological modelling.

References

1. M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R.H. Katz, A. Konwinski, G. Lee, D.A. Patterson, A. Rabkin, I. Stoica and Matei Zaharia, "Above the Clouds: A Berkeley View of Cloud Computing", UC Berkeley Reliable Adaptive Distributed Systems Laboratory, 2009
2. P. Mell and T. Grance, "The NIST definition of cloud computing", National Institute of Standards and Technology, Gaithersburg, 2011.
3. Fang Liu, Jin Tong, Jian Mao, Robert Bohn, John Messina, Lee Badger and Dawn Leaf, "NIST Cloud Computing Reference Architecture", National Institute of Standards and Technology, Gaithersburg, 2011.
4. Cloud Security Alliance, "Security Guidance for Critical Areas of Focus in Cloud Computing V2.1", Cloud Security Alliance, 2009.
5. J. Kincaid, "Coghead Grinds To A Halt, Heads To The Deadpool", (techcrunch), [online] 2009, http://techcrunch.com/2009/02/18/coghead-grinds-to-a-halt-heads-to-the-deadpool/ (Accessed: 2012)

6. Neal Leavitt, “Is Cloud Computing Really Ready for Prime Time?” ,*Computer*, vol. 42, no. 1, pp. 15–20, 2009.
7. “IEEE Standard Glossary”, [online], http://www.ieee.org/education_careers/education/standards/standards_glossary.html, (Accessed: 2012).
8. D. Petcu, “Portability and interoperability between clouds: challenges and case study”, in *Proceedings of the 4th European conference on towards a service-based internet*, Poznan, 2011, pp. 62–74.
9. DMTF,“Interoperable Clouds A White Paper from the Open Cloud Standards Incubator”,Distributed Management Task Force Inc., Portland, DSP-IS0101, Nov. 11, 2009.
10. T. Dillon, Chen Wu, and E. Chang, “Cloud Computing: Issues and Challenges”, in *2010 24th IEEE International Conference on Advanced Information Networking and Applications (AINA)*, Perth, 2010, pp. 27–33.
11. B. P. Rimal, Eunmi Choi, and I. Lumb, “A Taxonomy and Survey of Cloud Computing Systems”, in *Fifth International Joint Conference on INC, IMS and IDC, 2009. NCM ’09*, Seoul, 2009, pp. 44–51.
12. F. Moscato, R. Aversa, B. D. Martino, T.-F. Fortis, and V. Munteanu, “An Analysis of mOSAIC ontology for Cloud Resources annotation”, in *FedCSIS*, 2011, Szczecin Poland pp. 973–980.
13. D. Petcu, G. Macariu, S. Panica, and C. Crăciun, “Portable Cloud applications—From theory to practice”, *Future Generation Computer Systems*, 2012.
14. N. Loutas, E. Kamateri, and K. Tarabanis, “A Semantic Interoperability Framework for Cloud Platform as a Service”, in *2011 IEEE Third International Conference on Cloud Computing Technology and Science (CloudCom)*, Athens, 2011, pp. 280–287.
15. N. Loutas, E. Kamateri, and K. Tarabanis, “Cloud Semantic Interoperability Framework”, Cloud4SOA, D1.2, 2011.
16. D. Gasevic and N. Kaviani, “Ontologies and software engineering.” in *Handbook on Ontologies*. Springer Berlin Heidelberg, 2009, pp. 593-617
17. H.J Happel, S. Seedorf, “Applications of ontologies in software engineering”, in *Proc. 2nd InternationalWorkshop on SemanticWeb Enabled Software Engineering (SWESE 2006)*, Athens, USA, 2006
18. S Dowell, A. Barreto III, J.B Michel, and M.T. Shing, “Cloud to Cloud Interoperability”, *in Proceedings of the 2011 6th International Conference on Systems Engineering*, Jun 27-30 2011, Albuquerque, New Mexico, pp. 258-263
19. Open Cloud Computing Interface, OCCI, [online], http://occi-wg.org/ (Accessed: 2012)
20. S. Charrington, "Don't Pass on PaaS in 2010," (ebizo), [online] 2010, http://www.ebizq.net/topics/cloud_computing/features/12279.html?page=2, (Accessed: 2012)
21. J. Esparza-Peidro, F.D. Munoz-Escoı, “Towards the Next Generation of Model-Driven Cloud Platforms”, Instituto Universitario Mixto Tecnologico de Informatica, Valencia, TR-ITI-SIDI-2011/001, 2011
22. T. Gruber,“A translation approach to portable ontologies.” Knowledge Acquisition, vol.5, no.2, pp.199-220, 1993
23. OpenShift, [online], https://openshift.redhat.com/app/ (Accessed: 2012)
24. Google App Engine, [online], https://developers.google.com/appengine/ (Accessed: 2012)