EU Project SeaClouds*
Adaptive Management of Service-based Applications Across Multiple Clouds

Antonio Brogi[1], José Carrasco[2], Javier Cubo[2], Francesco D'Andria[3], Ahmad Ibrahim[1], Ernesto Pimentel[2] and Jacopo Soldani[1]

[1]*Department of Computer Science, University of Pisa, Pisa, Italy*

[2]*Department of Computer Science, University of Malaga, Malaga, Spain*

[3]*ATOS, Barcelona, Spain*

Keywords: Multi-Cloud Deployment, Monitoring, Dynamic Reconfiguration, Service Orchestration.

Abstract: The adaptive management of complex applications deployed across multiple heterogeneous PaaS platforms is one of the problems that have emerged with the cloud revolution. The ongoing EU research project SeaClouds aims at providing seamless adaptive multi-cloud management of complex applications by supporting the distribution, monitoring and migration of application modules over multiple heterogeneous PaaS platforms.
In this paper we present context, motivations and objectives of SeaClouds, its relation with other cloud initiatives, and its initial architecture.

1 INTRODUCTION

Current cloud technologies suffer from a lack of standardization, with different providers offering similar resources in a different manner (M. Armbrust et al., 2010). As a result, cloud developers are often locked in a specific platform environment because it is practically unfeasible for them, due to high complexity and cost, to move their applications from one platform to another (N. Loutas et al, 2011). To take full benefit of the flexibility provided by the cloud, modules of a complex application could be deployed on multiple clouds depending on their characteristics. Thus, in a scenario where a complex application is distributed across different cloud service providers, a solution is needed to manage and orchestrate the distribution of modules in a sound and adaptive way (Parameswaran and Chaddha, 2012).

Consider an application consisting of three modules (A, B, C), each having its own requirements (R_A, R_B, R_C). Suppose that four PaaS platforms are available (α, β, γ, δ), each providing different offers (O_α, O_β, O_γ, O_δ) and based on different technologies. The application developer can deploy the application either on a single PaaS or over multiple PaaS platforms. In the latter case, then she has to figure out the best offer for each module (e.g., $R_A : \{O_\gamma\}$, $R_B : \{O_\beta, O_\gamma\}$, $R_C : \{O_\delta\}$) and perform the deployment accordingly (e.g., $A \rightarrow \gamma$, $B \rightarrow \beta$, $C \rightarrow \delta$). Afterwards, she has to monitor both the single modules and the whole application. In case some requirements are violated (e.g., R_B is no longer satisfied by O_β), she has to manually reconfigure the deployment (e.g., $A \rightarrow \gamma$, $B \rightarrow \gamma$, $C \rightarrow \delta$). This implies that to manage multi-cloud applications, the developer must have knowledge of the different employed PaaS platforms and has to continuously monitor the application. This is a costly and cumbersome process and it would be much better to have some framework doing all the work automatically.

Figure 1: SeaClouds logo.

In this paper, we discuss the ongoing project SeaClouds (Seamless adaptive multi-cloud management of service-based applications) which focuses on the problem of deploying and managing complex multi-component application over heterogeneous clouds in an efficient and adaptive way. SeaClouds works towards giving organizations the capability of "Agility After Deployment" (Figure 1) for cloud-based appli-

*Work supported by EU-FP7-ICT-610531 SeaClouds project (start date: October 2013, end date: March 2016).

758 Brogi A., Carrasco J., Cubo J., D'Andria F., Ibrahim A., Pimentel E. and Soldani J..
EU Project SeaClouds - Adaptive Management of Service-based Applications Across Multiple Clouds.
DOI: 10.5220/0004979507580763
In *Proceedings of the 4th International Conference on Cloud Computing and Services Science* (MultiCloud-2014), pages 758-763
ISBN: 978-989-758-019-2
Copyright © 2014 SCITEPRESS (Science and Technology Publications, Lda.)

cations. Its approach is based on the concept of service orchestration and it is designed to fulfill functional and non-functional properties over the whole application. Applications will be dynamically reconfigured by changing the orchestration of the services when the monitoring detects that such properties are not respected. So, SeaClouds main objective is *the development of a novel platform which performs a seamless adaptive multi-cloud management of service-based applications*. More specifically, the objectives of SeaClouds are:

O_1) Orchestration and adaptation of services distributed over different cloud providers,

O_2) Monitoring and run-time reconfiguration of services distributed over multiple heterogeneous cloud providers,

O_3) Providing unified application management of services distributed over different cloud providers, and

O_4) Compliance with major standards for cloud interoperability.

The rest of the paper is organized as follows. Section 2 will position SeaClouds with respect to other cloud initiatives and it will describe its main challenges. Section 3 will present the SeaClouds approach along with its initial architecture. Finally, Section 4 will draw some concluding remarks.

2 SeaClouds IN CONTEXT

This section describes how SeaClouds, with the purpose of achieving its main goal, will advance the state of the art with respect to O_1, O_2 and O_3, and will contribute to obtain O_4. Figure 2 illustrates how SeaClouds intends to relate to the previously mentioned initiatives and standards.

Orchestration and Adaptation in the Cloud. O_1 will be addressed by developing adaptive orchestrators of cloud-based application modules. Orchestrators are widely used in the service-oriented computing paradigm (A. Brogi, J. Camara, C. Canal, J, Cubo and E. Pimentel, 2007; C. Canal and Salaun, 2008; J. Cámara, J.A. Martín, G. Salaün, J. Cubo, M. Ouederni, C. Canal and E. Pimentel, 2009; H. Nezhad, G.Y. Xu and B. Benatallah, 2010; J. Cubo and E. Pimentel, 2011), mainly focusing on behavioural and context-aware adaptation of services, by coordinating the interactions between different services. Several approaches exist that target formal adaptation of orchestrated services, but, to the best of our knowledge, only few (middleware-based) attempts have been proposed to face challenges such as heterogeneity of cloud platforms and migration to different cloud providers (e.g., (Guillén et al., 2013)).

Challenges in Orchestration and Adaptation for the Cloud. SeaClouds will address the following challenges in order to extend service-oriented approaches to the cloud: (i) Adaptation contracts need to take into account cloud provider characteristics and Service Level Agreements (SLA), (ii) Violations of Quality of Service (QoS) properties need to be monitored across different cloud platforms, and (iii) Dynamic architecture reconfiguration might involve migrating some components of the application to other cloud providers at runtime. The latter two challenges are discussed further in the following sections.

Monitoring of Multi-cloud Services. The EU FP7 Cloud4SOA project (www.cloud4soa.eu) provides an open source interoperable framework for application developers and PaaS providers. Cloud4SOA aims at supporting developers in deploying and monitoring their application with the ultimate objective of reducing the risk of vendor lock-in. The monitoring is based on unified metrics, but Cloud4SOA monitors each application separately and it is not able to aggregate monitoring results of multi-component applications.

Several Commercial and Open Source Initiatives Target Monitoring of Cloud Applications. Often these initiatives address only particular platforms, for example Appsecute (www.appsecute.com) monitors only (open-source) CloudFoundry-based platforms. More platform-independent technologies are available for the IaaS level, since the latter has undergone a stronger harmonization effort. Deltacloud (deltacloud.apache.org) encapsulates the native API cloud provider to enable management of resources in different IaaS platforms, such as Amazon EC2. Rightscale (www.rightscale.com) supports monitoring several public (e.g., Amazon Web Services, Google Cloud Platform, Rackspace, Windows Azure) and private IaaS clouds (e.g., CloudStack, Eucalyptus, OpenStack). Truly platform-independent monitoring solutions exist, the most known being NewRelic (www.newrelic.com). NewRelic achieves platform-independency by requiring each provider to implement a monitoring component and to integrate it in the offered cloud platform. On the one hand, this approach yields the best results from a monitoring point of view. On the other hand, it forces providers to invest quite some resources in order to implement the monitoring.

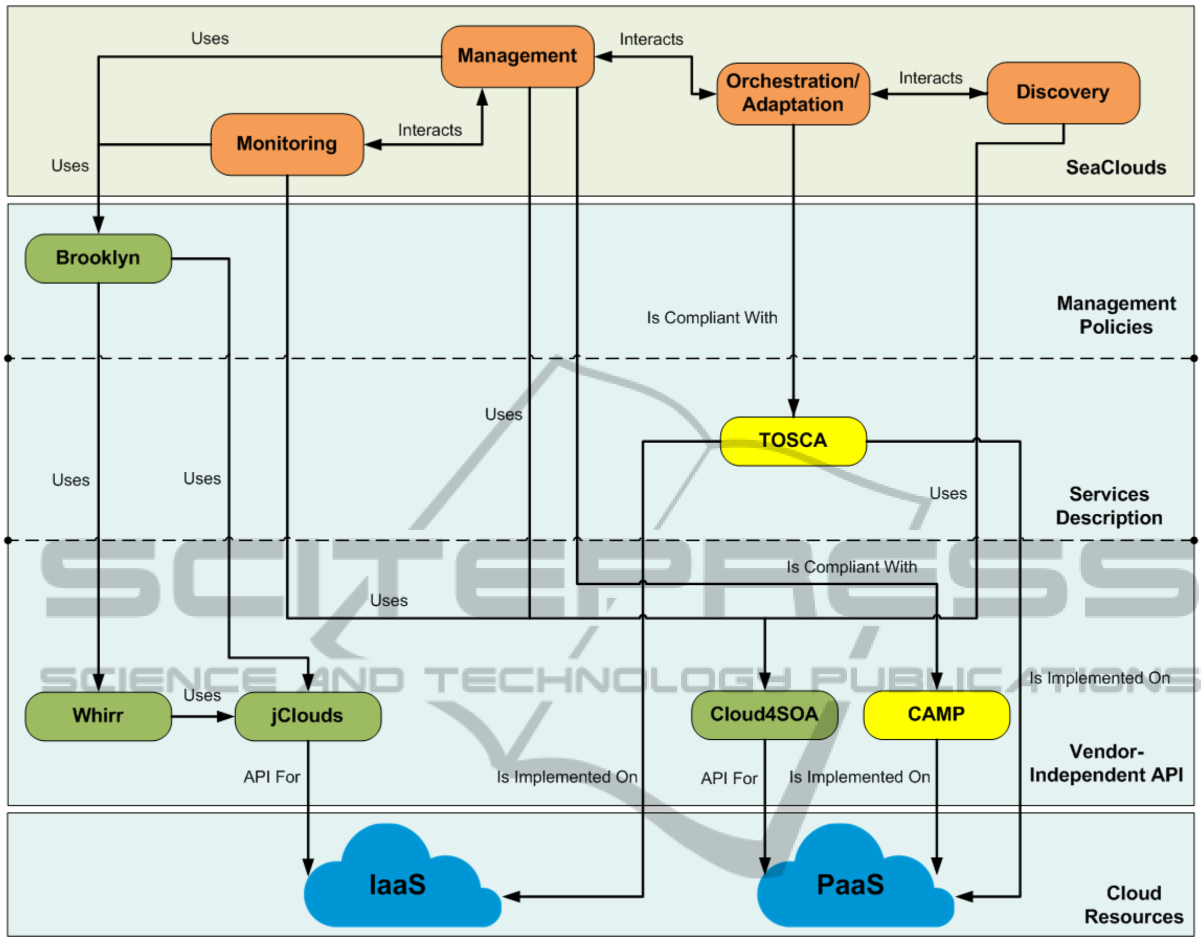

Figure 2: Positioning SeaClouds.

Challenges in Monitoring of Services on Multiple Clouds. In order to address O_2, SeaClouds monitoring will use and enhance existing monitoring functionalities for the PaaS and IaaS levels: (i) With respect to the IaaS level, SeaClouds will simply reuse what is available (e.g., Deltacloud), and (ii) With respect to the PaaS level, SeaClouds aims at augmenting the set of metrics currently available from Cloud4SOA (response time and up-time). For both the IaaS as the PaaS level, SeaClouds aims at coordinating, monitoring and aggregating monitoring information at the single service level to fulfill the purposes of whole orchestration. Thus, SeaClouds aims at: (i) monitoring each application component, and (ii) combining and aggregating the above mentioned data to highlight possible problems and their impact.

Unified Management of Multi-cloud Applications. Brooklyn (www.brooklyn.io) is an open source, policy-driven control plane for distributed applications delivered by CloudSoft (a member of the SeaClouds consortium). It enables single-click deployment of applications across machines, locations and clouds. Then it continuously optimizes running applications to ensure ongoing compliance with policies. Brooklyn uses two open source tools to operate on cloud resources, Apache Whirr (whirr.apache.org) and jClouds (www.jclouds.org), that support several IaaS providers. The already mentioned Cloud4SOA project also offers deployment and lifecycle management functionality using a harmonized API layer to encapsulate the providers' APIs. Furthermore, the previously mentioned initiatives (RightScale, Deltacloud for the IaaS level, and Appsecute, CloudFoundry for the PaaS level) all offer management functionality, with the same limitations as discussed while explaining monitoring.

Challenges in Unified Application Management of Services Distributed over Different Cloud Providers. SeaClouds will use existing management functionalities. Specifically, (i) SeaClouds discovery functionality may use and extend existing matchmaking functionalities to match application requirements with

PaaS offerings, and (ii) SeaClouds management will use a REST harmonized API for the deployment, management and monitoring of simple cloud-based applications across different and heterogeneous cloud PaaS offerings. SeaClouds intends to use Brooklyn's policy-driven functionality to integrate support for IaaS providers. Moreover, Brooklyn's approach to policy modeling and enforcing can provide guidance for SeaClouds orchestration/adaptation and management functionality. On the other hand, Brooklyn only targets the IaaS level and provides no support for orchestration. Beyond what Brooklyn provides, SeaClouds will therefore extend policy-driven functionality to the PaaS level and also add support for adaptation and orchestration. Due to a common partner in both initiatives (CloudSoft), Brooklyn can also benefit from integrating SeaClouds functionalities, especially regarding the integration of adaptation techniques in supported policies.

Standards for Cloud Interoperability. CAMP (Cloud Application Management for Platforms) (OASIS, 2012a) aims at defining harmonized APIs, models, mechanisms and protocols for the self-service management (provisioning, monitoring and control) of applications in a PaaS, independently of the cloud provider. However, CAMP is only a protocol specification, so it needs to be implemented by parties adopting the protocol.

The OASIS TOSCA (Topology and Orchestration Specification for Cloud Applications) (OASIS, 2012b) aims at enabling the interoperable description of application and infrastructure cloud services, the relationships between parts of the service, and the operational behaviour of these services, independently from the cloud provider. By increasing service and application portability in a vendor-neutral ecosystem, TOSCA aims at enabling portable deployment to any compliant cloud, smoother migration of existing applications to the cloud, as well as dynamic, multi-cloud provider applications.

Challenges in Standards for Cloud Interoperability. SeaClouds intends to actively contribute to the standardization effort of CAMP (OASIS, 2012a) both by exploiting CAMP-compliant interfaces provided by PaaS providers, and by contributing review proposals that will possibly emerge while specifying properties of SeaClouds orchestrations, adaptations and monitoring. SeaClouds will exploit the TOSCA specification to drive the design of the model for specifying cloud service orchestrations. In doing so, SeaClouds might actively contribute to the standardization effort of TOSCA, by contributing review proposals that will emerge while trying to devise TOSCA-compliant instances of the SeaClouds service orchestration model. On the other hand, SeaClouds will also focus on developing functionalities that are deliberately out of the scope of TOSCA (OASIS, 2012b) to solve issues about policies for the dynamic management of service orchestrations. Although the currently available implementations of TOSCA and CAMP do not yet support the management of complex applications over multiple clouds, SeaClouds will work towards building such management on top of them.

3 THE SeaClouds APPROACH

SeaClouds aims at homogenizing the management of different cloud providers and at supporting sound and scalable orchestrations of services across them. Moreover, systems developed with SeaClouds will inherently support the evolution of their constituent services, so as to easily and dynamically cope up with needed changes. The development, monitoring and reconfiguration via SeaClouds will include a unified management service, where services can be deployed, replicated, and administered by means of standard and harmonized APIs.

In order to obtain its main goals (viz., the development of the platform and the definition, implementation and validation of the use cases), SeaClouds will build a foundation to allow **"Agility After Deployment"** by providing necessary tools and a framework for *Modelling*, *Planning* and *Controlling* cloud applications.

Figure 3 illustrates SeaClouds initial architecture. The **Planner** module will implement SeaClouds planning policy to orchestrate the multi-cloud deployment of application modules. The Planner will take the input data provided by SeaClouds users, who will specify the application modules to be deployed, as well as desired QoS properties of the whole application and/or QoS and technology requirements for individual application modules. The Planner will determine the best offer for each module (by using the **Discovery API**) and will determine how to orchestrate the modules across the selected cloud platforms (viz., *orchestration specification*).

The **Controller** module will implement the multi-cloud deployment of application modules and the monitoring policy. In particular, the **Multi-Cloud Deployer** component will input the *orchestration specification* generated by the Planner, and deploy (by exploiting the **Multi-Cloud Deployment API**) the application modules on the specified clouds. The **Monitor** component will be in charge of monitoring (by exploiting the **Monitoring API**) that application

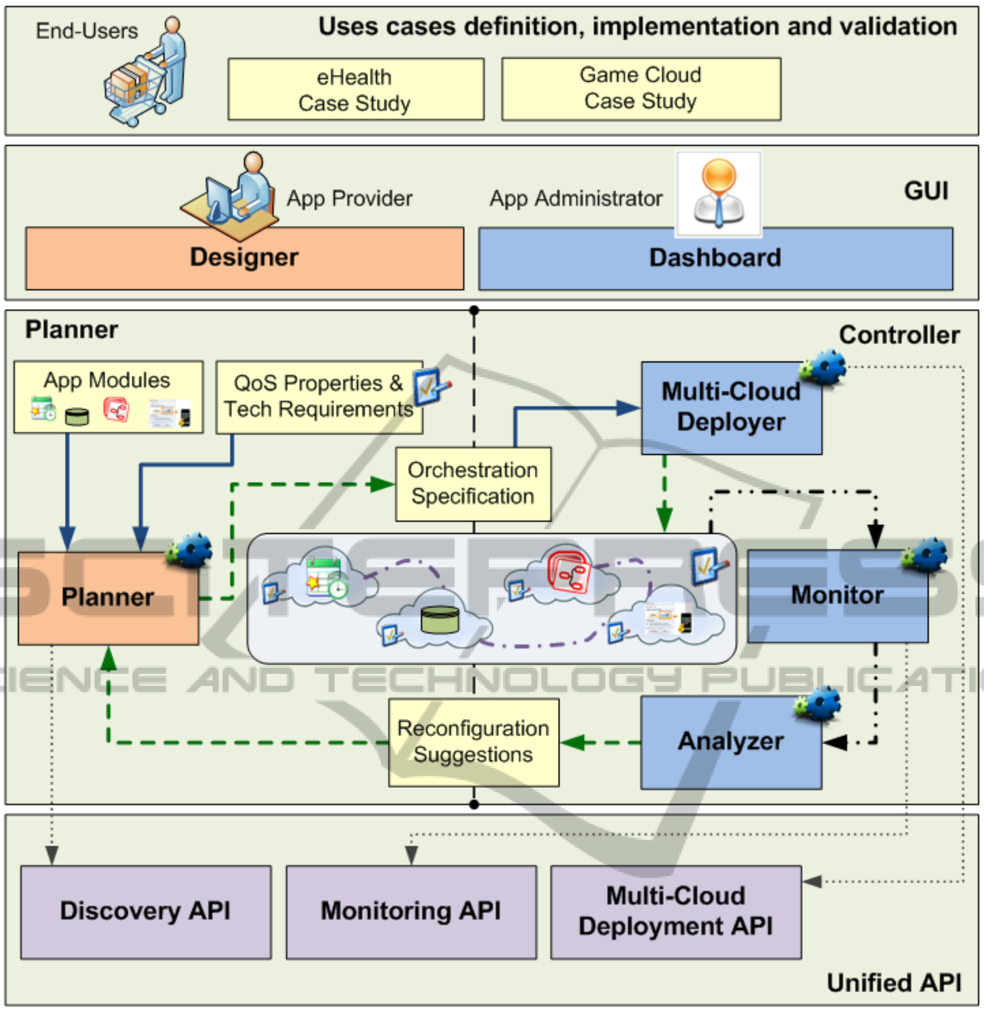

Figure 3: SeaClouds initial architecture.

(module) requirements are not violated. If the monitoring component detects (QoS or technological) requirement violations, then it will trigger the **Analyzer** component. The latter will generate (if needed) *reconfiguration suggestions* and pass them to the Planner in order to trigger the generation of a new orchestration plan. Users will exploit the Graphical User Interface (**GUI**) module to interact with SeaClouds. The GUI module will contain a **Designer** component to allow users to provide SeaClouds input data (viz., application modules, QoS and technological requirements). The GUI will also contain a **Dashboard** component to let users easily manage and visualize the state of the multi-cloud deployed applications.

4 CONCLUSIONS AND PERSPECTIVES

In this paper, we have presented context, motivations and objectives of the SeaClouds project (www.seaclouds-project.eu). The project aims at providing an open source framework to address the problem of deploying, managing and reconfiguring complex applications over multiple and heterogeneous clouds. The SeaClouds approach works towards achieving “Agility After Deployment” by tackling the problem from the service orchestration perspective. A complex application, which consists of modules and (technological and QoS) requirements, is provided as input to the SeaClouds planner. The latter generates an orchestration by assigning (groups of) modules to different PaaS platforms. The orchestration is then deployed and monitored according to standard metrics. If requirements are violated, then the Sea-

Clouds analyzer will generate reconfiguration information, which leverages the creation of a different orchestration of the application.

The expected results of the SeaClouds project can be summarized as follows:

- *Orchestration of Cloud Services*. High-level orchestration specifications will be used to describe the topology and behavior of the application distributed across different platforms.
- *Monitoring Process*. SeaClouds will monitor QoS requirements to determine whether/which modules should be migrated.
- *Traceability of QoS Violations*. Monitoring will have to keep track of possible requirement violations and their causes. This is needed for the generation of reconfiguration suggestions.
- *Reconfiguration Mechanism*. SeaClouds will feature flexible reconfiguration by supporting the migration of application modules across different PaaS platforms.
- *Unified Management API for Cloud Platforms*. SeaClouds will facilitate the access and the administration of both public and private cloud providers by providing multi-cloud management tools as well as offering cloud providers and consumers a REST-based approach to cloud-based application management.
- *Dashboard for Administration of Services Distributed across Multiple PaaS Platforms*. SeaClouds will provide a graphical interface to visualize and manage the current configuration of the deployed services.
- *Contribute to the Development of Cloud Standards*. SeaClouds will provide feedback to CAMP by proposing new functionalities and implementing a CAMP-compliant interface. It will also contribute to the standardization effort of TOSCA, by employing TOSCA as an orchestration modeling language.

REFERENCES

A. Brogi, J. Camara, C. Canal, J, Cubo and E. Pimentel (2007). Dynamic contextual adaptation. In *Proc. of Fifth International Workshop on the Foundations of Coordination Languages and Software Architectures 2006 (FOCLASA'06), vol. 175, no. 2*, pages 81–95. Elsevier.

C. Canal, P. P. and Salaun, G. (2008). Model-Based Adaptation of Behavioural Mismatching Components. *IEEE Transactions on Software Engineering*, 34(4):546–563.

Guillén, J., Miranda, J., Murillo, J. M., and Canal, C. (2013). A service-oriented framework for developing cross cloud migratable software. *Journal of Systems and Software*, 86(9):2294–2308.

H. Nezhad, G.Y. Xu and B. Benatallah (2010). Protocol-aware matching of web service interfaces for adapter development. In *Proc. of 19th International Conference on World Wide Web 2010 (WWW'10)*, pages 731–740. ACM.

J. Cámara, J.A. Martín, G. Salaün, J. Cubo, M. Ouederni, C. Canal and E. Pimentel (2009). Itaca: An integrated toolbox for the automatic composition and adaptation of web services. In *Proc. of International Conference on Software Engineering 2009 (ICSE'09)*, pages 627–630. IEEE Computer Society Press.

J. Cubo and E. Pimentel (2011). Damasco: A framework for the automatic composition of component-based and service-oriented architectures. In *I. Crnkovic, V. Gruhn, M. Book (editors), European Conference on Software Architecture 2011 (ECSA'11), LNCS 6903*, pages 388–404. Springer-Verlag.

M. Armbrust et al. (2010). A view of cloud computing. *Commun. ACM*, (4).

N. Loutas et al (2011). D1.1 Requirements Analysis Report. Cloud4SOA Project Deliverable, http://www.cloud4soa.eu/.

OASIS (2012a). CAMP 1.0 (Cloud Application Management for Platforms), Version 1.0. http://docs.oasis-open.org/camp/camp-spec/v1.1/camp-spec-v1.1.html/.

OASIS (2012b). TOSCA 1.0 (Topology and Orchestration Specification for Cloud Applications), Version 1.0. http://docs.oasis-open.org/tosca/TOSCA/v1.0/TOSCA-v1.0.pdf.

Parameswaran, A. and Chaddha, A. (2012). Cloud Interoperability and Standardization. *SETLabs Briefings - Infosys*, 7(7):19–26.