Available online at www.sciencedirect.com

ScienceDirect

Procedia Engineering 69 (2014) 1268 – 1275

Procedia Engineering

www.elsevier.com/locate/procedia

24th DAAAM International Symposium on Intelligent Manufacturing and Automation, 2013

A New Cloud Services Portability Platform

Magdalena Kostoska*, Marjan Gusev, Sasko Ristov

Ss. Cyril and Methodius University, Faculty of Information Sciences and Computer Engineering, Rugjer Boshkovik 16, PO Box 393, Skopje, Macedonia

Abstract

This research is motivated by several open research questions about cloud solutions, such as how to wisely choose a cloud host for services and how to change the cloud provider in an easy manner. The ability to move the services from one to another provider in a simpler manner is the definition of the cloud service portability, as an open research area which deserves more attention.
In this paper we propose a new cloud portability service platform and we give an overview of the current trends in the area of cloud service portability, especially analyzing the TOSCA's approach. Our new platform proposal is based on establishing an *adapter model,*which offers the desired cloud portability. The new solution describes the services by a XML TOSCA approach aiming to build a custom Platform as a Service (PaaS) that can accept and exchange installations of any portable application. The scenario for the new approach assumes that the applications are described by appropriate installation requirements and application structure using the TOSCA specificationandthe platform performs the necessary installation according to the specificationto continue with automatic deployment of application/services.

Keywords: cloud computing interoperability; cloud service portability; TOSCA

1. Introduction

Currently, the number of available cloud providersgrows immensely, and it is expected a higher growth rate in near future. Customers are faced with challenges to choose a right cloud platform to migrate their services on, and the open problems are if their applications and data can be exchanged with other solutions, or moved to another

* Corresponding author.
E-mail address: magdalena.kostoska@finki.ukim.mk

provider as an easy task done just by switching a button. This actually defines the problems of interoperability and portability of applications, services and data.

There are no published or fully developed solutions for these problems. The problem is exposed even more due to the lack of large-scale adopted cloud standards. In this situation, while waiting for the emergence of real standards, each vendor tries to impose newly developed technical solutions as a de-facto standard.

The portability of cloud services in each of the layers of the service stack is discussed in a lot of papers from the perspective of standards [1,2] and has even been addressed in open standards [3]. Several solutions that have been proposed include the abstraction-driven approach [4] and CSAL [5], while other approaches focus on exploiting the semantic technology [6, 7, 8, 9].

The efforts to construct cloud standards in this area are represented by the initiative organized by the OASIS (Organization for the Advancement of Structured Information Standards) task force TOSCA (Topology and Orchestration Specification for Cloud Applications) technical committee, that works on the standard for "*interoperable description of application and infrastructure cloud services, the relationships between parts of the service, and the operational behavior of these services (e.g., deploy, patch, shutdown)"*[10]. Although some influential companies like IBM, SAP, Citrix and RedHat are part of this process there is no guarantee that this standard will be accepted by the other vendors [11].

In this paper we present and analyze the TOSCA model and propose a new platform to enhance service portability.

2. Topology and Orchestration Specification for Cloud Applications (TOSCA)

In this section we briefly describe the TOSCA specification. We give overview of the language, the structure and the usage.

OASIS as a not-for-profit, international consortium, aims to develop, converge and adopt open standards for the global information society. The OASIS TOSCA Technical Committee works to enhance the portability of cloud applications and services. Their goal is to enable the interoperable description of application and infrastructure cloud services, the relationships between parts of the service, and the operational behavior of these services [10].

The open standards are used to describe a service and explain how to manage it independently from the supplier who creates the service, as well as from any particular cloud provider (or its technology) that hosts the service. This technical committee has produced a specification (Version 1.0) which explains how to define services in a portable and interoperable manner by a Service Template document. The TOSCA language introduces a grammar for describing service templates by means of Topology Templates and plans [10]. TOSCA utilizes XML Schema 1.0 \footnote{Defined by W3C: XML schema part 1 and 2, Namespaces in XML 1.0 and XML base and WSDL 1.1 [12]. It also contains non-normative references to BPEL 2.0 (Web Services Business Process Execution Language Version 2.0, OASIS Standard) [13], BPMN 2.0 (OMG Business Process Model and Notation Version 2.0) [14], OVF (Open Virtualization Format Specification Version 1.1.0) [15] and XPATH 1.0 (XML Path Language Version 1.0, W3C Recommendation) [16].

The TOSCA specification uses TOSCA xml [17] and xs [18] namespace prefixes, but allows extensibility from other namespaces for attributes and elements, which do not oppose the semantics of the TOSCA namespace [10]. The specification defines a meta model for defining the structure of an IT service and its management. The Topology Template defines the structure of a service. Plans define the process models that are used to create, terminate and manage a service. The major elements defining a service are depicted in Figure 1.

The structure of the services is defined by the Topology Template. This template consists of Node Templates and Relationship Templates. The Node Template describes the required components and their properties, operations and interfaces, while Relationship Template describes connectivity between components (Node Templates) by stating the direction of the relationship (source and target) and can have additional parameters. The purpose of Plans is to interpret the templates and execute appropriate actions. They are defined as process models and they rely on BPMN (primarily) and BPEL. In addition, this specification allows nesting (ex. One Service Template to be part of another).

In order to allow automatic deployment or build of services this specification also defines Artifacts – installable or executional types like scripts, libraries, installation binaries or images etc…

The specification supports the following major use cases:

- Services as Marketable Entities,
- Portability of Service Templates,
- Service Composition and
- Relation to Virtual Images.

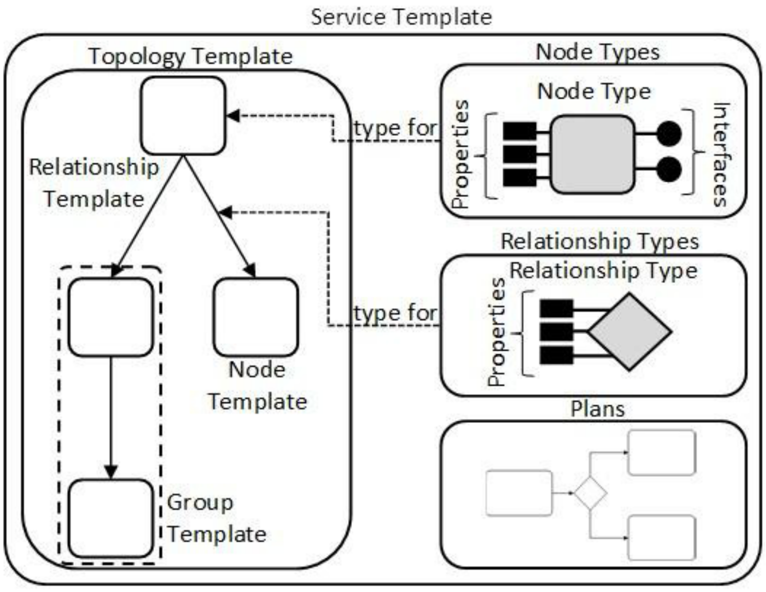

Fig. 1. TOSCA Service Template [10]

3. A new proposed model

This model is based on a specific platform which serves as a PaaS. It uses the TOSCA specification to describe the installation requirements as well as the application/services structure. Its goal is to provide an easy migration of SaaS applications and services from one provider/solution to another via a specific custom platform. It serves as a specific private PaaS platform. The reasons why this type of solution is chosen are the following: PaaS can be deployed on multiple hypervisors and on different types of IaaS; it represents an elastic middleware and adds additional security and control.

This platform in planned to be developed in several phases, which will be further described. The first phase is explained in the following section, while the others are described in Future Research Directions and Work section.

3.1. Platform

This platform is intended to be hosted on either a public or a private cloud and this can be done by using multiple hypervisors. In direction of using open-source and freeware solution, the Linux operating system is used. In the first phase this platform to host multiple applications without dedicating separated allocation for each application. Its intention is to offer a limited list of predefined web servers, programing languages, data and messaging services. Our targets are open-source and freeware solutions in the first experimentation phase.

It's intended to use the TOSCA specification to describe the installation requirements as well as the application/services structure and we will develop and use dedicated tools for processing of these files. These tools assist with the installation of the required tools and software and with the deployments of the specified application/services. Figure 2 depicts the architecture of the platform.

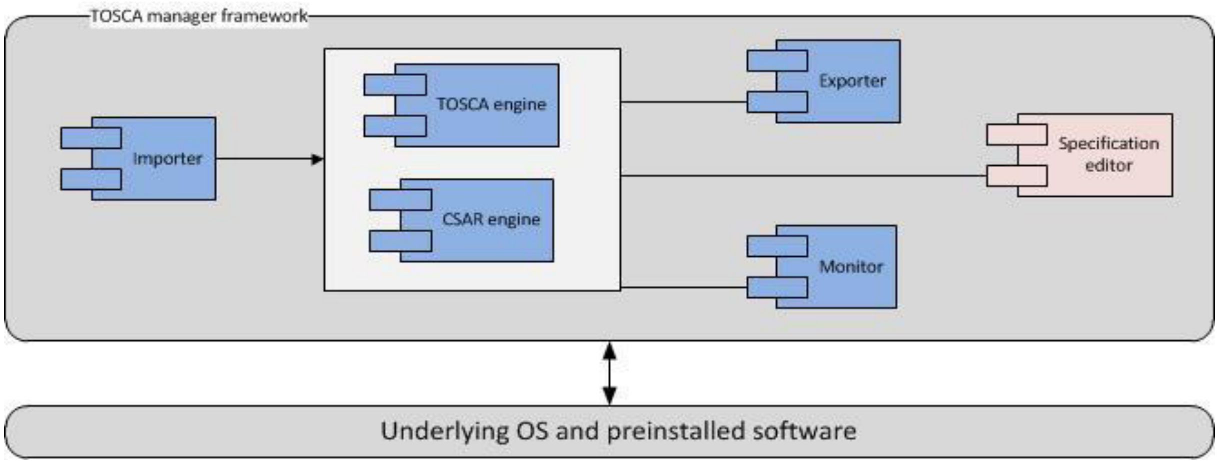

Fig. 2. TOSCA enabled platform.

The central part of the architecture is the TOSCA manager framework. It consists of the following packages:

- Importer – this package is used to import and unpack the archive consisting of specification and artifacts according to given structure, described in the Installation and build requirements section
- TOSCA engine – this engine process the specification and using the artifact performs the deployment
- CSAR (Cloud Service Archive) engine – this engine manages the defined artifacts and enables their processing
- Exporter – the purpose of this package is to create archive for a given application that can be easy transported and deployed on another instance
- Monitor – the purpose of this package is to monitor to and track the work of the TOSCA and CSAR engines
- Specification editor –the purpose of this package is to help the users to create TOSCA specification for a given application

3.2. Installation and build requirements

The installation and build required files in TOSCA are included in the CSAR.It represents anarchive file with at least two directories: the TOSCA-Metadata directory that contains metadata describing the other content of the CSAR and the Definitions directory that contains one or more TOSCA Definitions documents [10]. The various artifacts are described in the TOSCA meta file as name/values pairs. The following code represents an example of TOSCA meta file.

Example of a TOSCA meta file

```
TOSCA-Meta-Version: 1.0
CSAR-Version: 1.0
Created-By: Frank

Name: Service-Template/Payroll.tosca
Content-Type: application/vnd.oasis.tosca.definitions
```

```
Name: Service-Template/PayrollTypes.ste
Content-Type: application/vnd.oasis.tosca.definitions

Name: Plans/AddUser.bpmn
Content-Type: application/vnd.oasis.bpmn

Name: EARs/Payroll.ear
Content-Type: application/vnd.oasis.ear

Name: JARs/Payrolladm.jar
Content-Type: application/vnd.oasis.jar

Name: Python/wsadmin.py
Content-Type: application/vnd.oasis.py
```

The importer imports the CSAR file and unpacks. The structure is checked and the artifacts and definitions are passed to the CSAR and TOSCA engine. In the first phase this platform supports limited types of artifacts.

The concrete executables in TOSCA are represented with Artifact Template, which describes the properties of the executable and defines the location. An example of scripting artifacts follows.

Example of a TOSCA scipting artifact template

```
<ArtifactTemplate id="314"
                    type="ns1:ScriptArtifact">
<Properties>
<ns1:ScriptArtifactProperties
        xmlns:ns1="http://www.example.com/ToscaBaseTypes"
        xmlns="http://www.example.com/ToscaBaseTypes">
<ScriptLanguage>sh</ScriptLanguage>
<PrimaryScript>scripts/ApacheWebServer/configure.sh</PrimaryScript>
</ns1:ScriptArtifactProperties>
</Properties>
<ArtifactReferences>
<ArtifactReference reference="scripts/ApacheWebServer">
<Include pattern="configure.sh"/>
</ArtifactReference>
</ArtifactReferences>
</ArtifactTemplate>
```

3.3. Specification of applications/services structure

To describe the applications/services structure with XML, our proposal is to use the Service Template defined by TOSCA. This template represents a deployment Topology Template. It consists of Node Templates and Relationship Templates that together define the topology model of a service as a directed graph.

The service template document has a lot of properties, although most of them are optional. A property that has to be included in each template is *id*, an identifier which will be unique and *TopologyTemplate* which defines the structure (the relation between those components) of the application. The *TopologyTemplate* consists of a *NodeTemplate* which specifies the components and a *RelationshipTemplate,* which specifies the relationships between the components. It can also define tags for description and boundary definitions such as properties and their constraints, requirements, capabilities, policies and interfaces.

The *NodeTemplate* consists of at least the following obligatory properties: id; type, a QName value; and properties and their constraints. It may also define requirements, capabilities, policies and deployment artifacts. The *RelationshipTemplate* must define id and type, as well as source and target elements. It may also describe the relationship constraint type.

The following code represents a part of a service template defined by TOSCA.

Example of a Service Template given by TOSCA

```
<ServiceTemplate id="MyService"
               name="My Service">
<TopologyTemplate>

<NodeTemplate id="MyApplication"
                  name="My Application"
                  type="my:Application">
<Properties>
<ApplicationProperties>
<Owner>Frank</Owner>
<InstanceName>Thomas favorite application</InstanceName>
</ApplicationProperties>
</Properties>
</NodeTemplate>

<NodeTemplate id="MyAppServer"
                  name="My Application Server"
                  type="my:ApplicationServer"
                  minInstances="0"
                  maxInstances="unbounded"/>

<RelationshipTemplate id="MyDeploymentRelationship"
                          type="my:deployedOn">
<SourceElement ref="MyApplication"/>
<TargetElement ref="MyAppServer"/>
</RelationshipTemplate>

</TopologyTemplate>

<Plans>
<Plan id="UpdateApplication"
          planType="http://www.example.com/UpdatePlan"
          planLanguage="http://www.omg.org/spec/BPMN/20100524/MODEL">
<PlanModelReference reference="plans:UpdateApp"/>
</Plan>
</Plans>

</ServiceTemplate>
```

3.4. Usage

The customer provides the requirements for the applications/services, while the required software is automatically installed. Two different scenarios follow:

- the customer gives TOSCA specification that describes the application, the services, the relationships between the services and the source code, so the application is automatically deployed; and
- the customer develops the application on the platform, while also producing the required specification file. Whenever the customer wants to change the cloud provider a new instance of the platform is deployed on the new cloud solution and two scenarios are possible.

Scenario 1 (Relies on direct communication between the old and the new instance): The customer enters the location of the previous (still active) instance on the new instance and pre-installed software, so the migration starts automatically. Firstly, the new instance of the platform takes the defined specifications and installs the necessary software. Afterwards, it reads the application/services structure file and it automatically deploys this solution on the new instance.

Scenario 2 (Relies on communication between the customer and the new instance, using data from the old instance): First, on the old instance, the customer performs export of the current state of the platform and the package of data is downloaded to the customer's local machine. Then the customer connects to the new instance, runs the preinstalled software and uploads the data, which the software will process and will also perform installation and deployment.

4. Future research directions and work

In the first phase the open source cloud solutions (for example, OpenStack or OpenNebula) are used to set the first instances of the described platform and we will try to deploy services on one instance and then to migrate these services to other instances. The services are intended to be deployed on Apache Tomcat web server and will be built by using the Java programming language. In this phase we also plan to use MySQL and PostgreSQL as data services.

In the second phase we plan on adding more types of web servers, while including more languages support and improving the platform with monitor and control tools. Furthermore, we plan on providing the separation of different groups of applications, as well as providing automatic deployment of multiple instances of the same application.

Out future plan is to develop a model for data migration and to enable proprietary software usage.

5. Conclusion

There are no standard solutions for cloud service portability, mainly due to a huge number of various cloud providers on the market. Two approaches are known in technology area to cope with emerging markets and offer. The first approach is using the most successful provider,which will “kill” the other in the concurrent market environment, and the other to establish a highly recognized standard accepted by all market players. TOSCA represents a first step towards building a standard for portable deployment and the migration of existing applications onto a cloud. Unfortunately it is still in its infant development stage and neither represents a standard, nor guaranteesan acceptance by the major cloud providers, such as Google and Amazon (mostly since they are not taking part in the creation of this specification).

Our solution introduces an adapter idea in such a way, not to be a standard, but to present an inter-mediatorservice. The TOSCA specification is used only partially. Although the solution is mainly intended for PaaS, it also offers a XML description of a SaaS application. The final goal is to offer a possibility for a documented service exchange, which solves most of open issues in service portability, service interoperability, service migration onto a cloud and cloud provider independency.

Acknowledgements

This work was partially financed by the Faculty of Computer Science and Engineering at the "Ss.Cyril and Methodius" University.

References

[1] G. Lewis, Role of standards in cloud-computing interoperability, in: 46th Hawaii International Conference on System Sciences (HICSS), 2013, pp. 1652 – 1661.
[2] Z. Zhang, C. Wu, D. W. Cheung, A survey on cloud interoperability: taxonomies, standards, and practice, ACM SIGMETRICS Performance Evaluation Review 40 (4) (2013) 13–22.
[3] F. Galan, A. Sampaio, L. Rodero-Merino, I. Loy, V. Gil, L. M. Vaquero, Service specification in cloud environments based on extensions to open standards, in: Proceedings of the Fourth International ICST Conference on COMmunication System softWAre and middlewaRE, ACM New York, 2009.
[4] A. Ranabahu, E. Maximilien, A. Sheth, K. Thirunarayan, Application portability in cloud computing: An abstraction driven perspective, IEEE Transactions on Services Computing PP (99) (2013) 1.
[5] Z. Hill, M. Humphrey, CSAL: A cloud storage abstraction layer to enable portable cloud applications, in: IEEE Second International Conference on Cloud Computing Technology and Science (CloudCom), 2010, pp. 504 – 511.
[6] N. Loutas, E. Kamateri, K. Tarabanis, A semantic interoperability framework for cloud platform as a service, in: IEEE Third International Conference on Cloud Computing Technology and Science (CloudCom), 2011, pp. 280 – 287.
[7] G. Cretella, B. Di Martino, Towards automatic analysis of cloud vendors APIs for supporting cloud application portability, in: Sixth International Conference on Complex, Intelligent and Software Intensive Systems (CISIS), 2012, pp. 61 – 67.
[8] D. Petcu, How to build a reliable mOSAIC of multiple cloud services, in: Proceedings of the 1st European Workshop on Dependable Cloud Computing, 2012.
[9] F. Moscato, R. Aversa, B. Di Martino, T. Fortis, An analysis of mOSAIC ontology for cloud resources annotation, in: Federated Conference on Computer Science and Information Systems (FedCSIS), 2011, pp. 973 – 980.
[10] OASIS, Topology and orchestration specification for cloud applications version 1.0. (Mar. 2013). URL http://docs.oasis-open.org/tosca/TOSCA/v1.0/cs01/TOSCA-v1.0-cs01.html.
[11] T. Binz, G. Breiter, F. Leyman, T. Spatzier, Portable cloud services using TOSCA, Internet Computing, IEEE 16 (3) (2012) 80–85.
[12] W3C, Web Services Description Language (WSDL) 1.1 (Mar. 2011). URL http://www.w3.org/TR/wsdl
[13] OASIS, Web services business process execution language version 2.0 (Apr. 2007). URL http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html
[14] OMG, Business Process Model and Notation (BPMN) Version 2.0 (Jan. 2011). URL http://www.omg.org/spec/BPMN/2.0/
[15] DFTM, Open virtualization format specification version 1.1.0 (Jan. 2010). URL http://www.dmtf.org/sites/default/files/standards/documents/DSP0243 1.1.0.pdf
[16] WC3, XML Path Language (XPath) Version 1.0 (Nov. 1999). URL http://www.w3.org/TR/1999/REC-xpath-19991116/
[17] OASIS, XML Namespace Document for TOSCA Version 1.0 (Mar. 2013). URL http://docs.oasis-open.org/tosca/ns/2011/12
[18] W3 consortium, XML Schema (Mar. 2013). URL http://www.w3.org/2001/XMLSchema.