ISSN 2278-3091
Volume 8, No.5, September - October 2019
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse61852019.pdf
https://doi.org/10.30534/ijatcse/2019/61852019

Application Portability Pertaining TOSCA in Cloud Computing Environment

Brijesh Pandey[1],Dr. D. K. Yadav[2], Dr. S S. Soam[3]
[1]AKTU,INDIA,brijesh84academics@gmail.com
[2]MNNIT,INDIA,dky@mnnit.ac.in
[3]AKTU,INDIA,sssoam@gmail.com

ABSTRACT

Achieving Application Portability in a cloud computing environment is most demandable in today's scenario. The problem of vendor lock-In issue needs to be handled. There could be two ways of achieving application portability. The first being Intermediation and the other standardization. We have focussed our research on standardization with the help of TOSCA. In this paper, we have discussed all the scenarios, facets and challenges in achieving application portability. We have discussed TOSCA simple profile in YAML common terms, definitions, workflows, etc. The evolution of WS-BPEL from XML has also been discussed in this paper. We have also touched the formal method language PROMELA and tool SPIN for modeling specification and verification of application properties. We have given our model to perform the task of achieving portability of application from one cloud host to another.In this paper, we have taken the example of the online hotel reservation system and discussed its properties.

Key words: Orchestration, Openstack, TOSCA.

1. INTRODUCTION

Cloud computing is the current scenario is the most evolving technology due to its IaaS, PaaS and SaaS services. It offers several benefits such as on-demand virtualized hardware infrastructure, user self-provisioning, pays per use facility and elastic computing.Distributed computing affects monetary exercises which feature its commitment to the world GDP.The wide assortment of cloud administrations has prompted the structures and innovations being licensed by different cloud specialist organizations. This has raised the concern of vendor lock-in where the customer is bound to have the services without its contention. The Incidences where the cloud administrations being closed somewhere around the suppliers or the disclosure of security sickness has emphasized the hazard. The cloud computing like Internet has the significant power to contribute to economic growth and commercial activities but the lack of global standards and interfaces for portability and interoperability has let down the growth.

The Portability in distributed computing setting can be characterized as the capacity of a client to move its application and information starting with one cloud specialist organization then onto the next. Portability in cloud computing can be widely categorized into two forms i.e. Cloud data portability and cloud application portability. The former is the propensity to transfer data easily in an electronic format from one cloud service to another. The latter is to relocate the component or whole application from one cloud service provider to its equivalent another cloud service provider without any significant changes in the application code. The third category of cloud computing portability could be cloud platform portability which is further alienated into two categories i.e. platform source portability and machine image portability.The previous being the reuse of stage segments crosswise over cloud IaaS administrations and non-cloud foundation.The latter being the reuse of packs containing application and information with their supporting stages.

Various Interfaces in the cloud computing environment for portability and interoperability of services are standardized.The semantic web gauges, for example, the Resource depiction structure (RDF) and Web philosophy language (OWL) can be utilized to characterize information and the information models in machine-lucid structure. The Universal data element framework (UDEF) can be used with data models to facilitate the integrated processing of data. There are application interfaces whose enumeration defines the message content, its syntax and the envelope in which it is transported.The Cloud information the board interface (CDMI) is a standard application interface for information stockpiling and recovery applications. There are two conventional principles Topology and Orchestration for cloud application (TOSCA) and Open distributed computing Interface (OCCI) for application the executives interface and stage the board interface.The same standards can be used to define Infrastructure management Interface which exposes the management qualities of Infrastructure services. The Publication Interface covers three areas: functionality, quality of service and condition of the contract. The open data center Alliance (ODCA) is expanding a systematic perspective to standardize the quality of service parameters and conditions of the contract. Acquisition Interface defines a product or cloud service description. OVF or even TOSCA is used for standardizing the acquisition interface. There are interfaces between applications and platforms. The operating system, Programming language, and standard libraries form the standard which will be discussed in the subsequent section. There are Platform interfaces

and Infrastructure interfaces also that hold significant importance which is of utmost importance.

Various elements are involved in the portability and interoperability of cloud services. To contemplateapplication portability the model for application is considered which consists of artifacts having instruction sets, metadata and data sets. There are certain dependencies like run time, data services, access management, identity, and encryption. It also includes operating system function, virtualization capabilities, and networking.The objective condition needs to help both the application relics and conditions. The container and its related infrastructure technologies have been holding the importance of application portability. The Dockers containerization platform [1] and Open container Initiative [2] offer a standard approach for the organization of utilization code and related programming stacks. To deploy and manage the set of containers for any given application can be realized through Open source Kubernetes container orchestration tooling [3].

In the subsequent section, we will discuss the various scenarios in which the application portability may occur. The issues need to be tackled in those different scenarios. The various facets of application portability are discussed along with other standardization issues. We will be discussing some technologies, models, and approaches needed for achieving the portability of application in a cloud computing environment.

2. SCENARIOS FOR APPLICATION PORTABILITY IN CLOUD COMPUTING

2.1 Client changes starting with one specialist organization then onto the next

The Application Portability holds the significance of IaaS and PaaS cloud benefits as the application has a place with the client. To move the suit from one cloud service provider to another the various facets of application such as syntactic, instruction, dependencies need to be answered. Ideally, there should not be any changes in the application artifacts but practically some changes are bound to occur. It is the amount and nature of the changes that matter. For PaaS, the application may have dependencies via API and that needs to be matched. For SaaS, well defined and standardized protocols and data formats need to be demanded.

2.2 Multiple Providers providing cloud service to the Customer

Developers often use the same cognizance, tools or technologies to escalate both applications that are used from different service providers.In any case, it might wind up troublesome if the capacities offered by cloud administration very generously. Virtual Machine image formats like OVF and container formats are important components of applicationportability.

2.3 Linkage of one cloud amenity into another cloud amenity by the Customer

The Application Portability does not apply to this scenario because in this there is no movement of data or application from one system to another.

2.4 Customer binding its capabilities with cloud services

The Application portability does not belong to this scenario as well because there is no movement of application from one service provider to another.

2.5 The departure of in premises application code of Customerto cloud services

The Application portability in this scenario suggests that the application code which keeps running on-premises to keep running on PaaS administration with no changes.If the app territory will change, the application code needs to be modified to reckon for the differences.The Chance of switching the on-premises application to an IaaS service is almost negligible. It involves migrating the entire software stack maybe along with the OS.To accomplish this, it should be conceivable to bundle the whole programming stack as a VM picture which shall be imported to cloud administration for execution.

3. FACETS OF THE CLOUD APPLICATION PORTABILITY MODEL

3.1 Instruction

The Target cloud service provider must be able to understand and execute the instructioncontained in the executable antiquities of the application. Programming Languages like C++, Java, and BPEL are universally accepted and hence can be used for the purpose.

3.2Syntactic

The Target cloud service provider must understand and use the format of all application artifacts. If it does not happen then the formats need to be changed. Zip, Tar or Zar can be applied.

3.3 Metadata

The Target cloud service provider mustcomprehend and utilize the metadata that determines the natural conditions for executing the application. The metadata may wish to adapt the capability of the target system. Like YAML, JSON, XML can do the needful.

3.4 Behavior

This facet aims to verify theNon-functional and functional behavior of the solicitation through the test suites. The Application code or the metadata needs to be modified if the ported application bombs components of the test suite.

3.5 Policy

This facet is concerned with the applicable laws, regulations, and policies under which the application is ported.

4. CHALLENGES IN APPLICATION PORTABILITY

The Portability of application in the cloud environment has some major concerns. Some of them can be the programming language, data store, frameworkor platform discrete configuration files and services [4].

All the cloud platforms have some specific programming languages and features that are used to build an application. Like Google App Engine has sustenance to Java but not to Java class libraries braced by Open shift [4]. Some specific APIs are used to provide platform services. A service is defined to be a high-level functionality without going through the details at a lowlevel [5].

Developers can append itsfunctionalities from stage administrations by just fastening to corresponding API [6].The Database and file store arethe twoclasses of data stores that exist on the cloud platform.The database storeaccumulates the organized data whereas the file store stores the data like a secondary disk on the cloud.

Further, thetwo classes of database store SQL and NoSQL exist. SQL, as we know, is the conventional one and is accessible on all the cloud providers' platforms. NoSQL, on the other hand, is a database store which comprises all database system that does not cling to the SQL relational database. NoSQL holds the capability to distribute data among many services [7]. Therefore, standardization or Intermediation can only be the solution for application portability and that remains its biggest challenge.

5. RELATED WORK

Standardization and Intermediation are the only two approaches to improve the application portability of cloud services. Standardization aims to reduce or even eliminate the differences between cloud providers thereby removing the lock-in issues [5] [8] [9]. Intermediation, on the other hand, is to provide a middleware layer that is attuned with multiple cloud providers which can be accessed by the cloud API. Intermediation can further be categorized as an Abstract layer-based approach and a Model-based approach.

In the former approach, the developers divide the application into multiple layers. The lower layer hid the implementation detail exposing the upper layer to access the services. In a scenario where the application desires to be ported is exposed to a diverse implementation with the same interface. The approach is generic but introduces lots of complexity in the application architecture [10]. [10], [11], [12] have employed this technique to achieve application portability according to reference [8]. In the latter approach, model-driven engineering is used to reduce vendor lock-in issues. An application is described at an even higher level of abstraction that is bare by the cloud provider using domain-specific language. The generic transformers are then used to generate code from the model targeting the specific cloud provider. Mod clouds [13] are an example of this approach.

Since our area for interest is standardization, we will explore the approaches and technologies related to it. OASIS holds the priority in this list by introducing TOSCA as an open cloud standardupheld by countless global industry pioneers. Itexplicates the application along with its components,dependencies, relationship, capabilities, and requirements. It sanctions portability and automated management across different multiple providers.In [14] BPMN4TOSCA was anticipated as an area explicit BPMN [15]. It facilitates the demonstrating of the board tact by giving direct access and incorporation to TOSCA topology. Since the functionalities stretched out to the BPMN4TOSCA are not upheld by work process motors along these lines non-standard BPMN is changed into plain BPMN. [16] Presents the evidence of idea for the real compactness highlight of TOSCA on Openstack and Opscode Chef. TOSCA2Chef an execution domain was flourished to robotize the sending of TOSCA based cloud application topologies using Chef and BPEL forms. [17] Presents a brought together conjuring transport and interface to be worn by TOSCA the executive's plans. The administration transport dependent on OpenTOSCA engineering was actualized to bear the cost of a bound together summon interface for TOSCA plans to conjure the activity. A coordinated demonstrating and runtime system was acquainted in [18] with accomplishing a consistent and interoperable arrangement of subjective antiquities. [19] Introduced the procedure displaying idea empowering the combination of provisioning models. The Models depended on expanding basic work process dialects, for example, BPMN and BPEL by method for Declarative Provisioning Activities which encourage demonstrating revelatory provisioning legitimately in the control stream of a work process model. Many research undertakings, for example, [20], [21], [22], [23] and [24] have additionally tended to application compactness in distributed computing in their points of view. A large number of these tasks have not fabricated the TOSCA motor rather changed the TOSCA based application detail into a solitary organization content, for example, YAML and executed it by comparing the board apparatus, for example, CAMP.

6. TOSCA SIMPLE PROFILE IN YAML [25]

The TOSCA simple profile in YAML provides an accessible and concise syntax to enhance the adoption of TOSCA for application portability.This determination guarantees that TOSCA semantics are saved and can be contorted from XML to YAML or the other way around. The TOSCA metamodel initiates the idea of administration formats to investigate cloud outstanding tasks at hand as a topology layout. The topology format is a diagram comprising of hub layouts for demonstrating the parts and relationship format for displaying the connection between those segments. TOSCA serves a sort arrangement of hub types for portraying methods for building up an administration layout and relationship type to depict the conceivable sort of connection. The hub type and relationship type portray the lifecycle activity of an arrangement motor which can refer to the instantiating administration format. These lifecycles activity is sponsored by execution ancient rarities that actualize the genuine

conduct. A coordination motor uses these lifecycle tasks to instantiate a solitary segment at run time and uses the relationship format to create the request for segment instantiation. The TOSCA straightforward profile utilizes the number of base kinds, for example, its hub types and relationship types to be bolstered by the individual part. The template author will not have to define in many cases the type themselves but can use the existing types.

The Individual blends that qualify as implementers of the TOSCA simple profile are as follows:

- TOSCA YAML administration layout is a YAML archive antiquity having administration formats to speak to the cloud application.
- TOSCA processor is a tool to parse Tosca service template.
- TOSCA orchestrator is a processor that interprets TOSCA CSAR to actualize and deploy the narrated application.
- TOSCA generator is a tool to generate TOSCA service template.
- TOSCA Cloud administration file or CSAR is a bundled ancient rarity that contains TOSCA administration layout and different relics usable by TOSCA orchestrator to send an application.

The Commonly used terms in TOSCA simple profile along with their definition are as follows:

- Instance Model:A sent administration is a running example of an administration layout.
- Node Template: A Node template refers to the node semantics such as its properties, attributes, requirements, capabilities and interfaces.
- Relationship Template: It determines the event of the connection between hubs in a layout.
- Service Template (Figure 1): It is utilized to distinguish the topology and organization of IT administration so they can be made to do regarding the executives and arrangements.
- Topology Model: An abstract representation of service and template.
- Topology Template: It includes a lot of hub layout and relationship format together that characterize the topology model of administration.

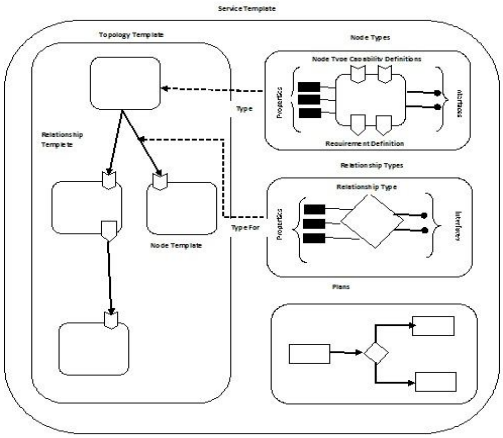

Figure 1: Service Template

7. WS-BPEL

The OASIS developed WS-BPEL for specifying the order of web services invocation. The web service invocation would be either sequential or parallel is determined by the WS-BPEL. A web service business process execution language definition is used to determine the internal business process written in XML. A BPEL represents the behavioral properties of composite web service and dispenses a web service definition language. WS-BPEL is focussed on the business and transforms process to new levels by integration, visualization, monitoring, and optimization. WS-BPEL is rather a concept that technology to automate the business solution.

The Basic concept of WS-BPEL can be exhibited in either Abstracted or Executable form. An Abstract process must be explicitly declared as abstract as it is not intended to execute.However, the executable process is fully specified and is ready to execute. The Abstract process uses two techniques to hide the operational details and they are explicit opaque tokens and Omissions. WS-BPEL permits the evolution of tools, techniques, etc. to enhance the degree of automation thus decreasing the expense of building up cross endeavor computerized business forms.
The Activities of WS-BPEL could be receive, reply, invoke, assign, throw,exit, wait, etc.

8. PROMELA & SPIN REFERRED FROM [26]

Process Meta Language (PROMELA) is a language that describes the concurrent system. SimplePromela Interpreter (SPIN) is a tool for analyzing Promela programs leading to the detection of errors in the system. It detects and validates deadlock, liveness properties, safety properties, etc.
PROMELA is just a specification language rather than a whole programming language. It specifies the abstraction of the system not the whole of the system itself. It emphasizes process synchronization and coordination rather than computation. It uses Non-determinism as an abstraction technique. It is suitable for only software modeling and not for hardware modeling.
The SPIN tool is a simulator for Promela language and verifier for the properties of its Programs.It produces a C program to construct an implementation of a linear temporal logic model checking algorithm for the given model.

9. OUR PROPOSED APPROACH FOR CASE STUDY (ONLINE HOTEL RESERVATION SYSTEM)

We propose our framework for the first scenario described in section II where the customer switches the provider be due to the violation of service level agreement between him and the service provider. We use in our framework the standardization method of achieving application portability and in a manner discarding the problem of vendor lock-in issues. We use TOSCA Simple Profile in YAML which provides a simple, concise, accessible and unambiguous syntax to

enhance the adoption of TOSCA for application portability. To determine the internal behavior of the web service written in XML we use WS-BPEL for its invocation. The files of TOSCA Simple Profile in YAML and WS-BPEL are composed together into an architectural design to produce a cloud service archive i.e. CSAR files. We then propose to formally model this CSAR file using the Promela code and develop some basic safety properties using linear temporal logic. The SPIN Model checker is utilized for verifying the model and its safety properties. Figure 2 shows the architecture for Application Portability.

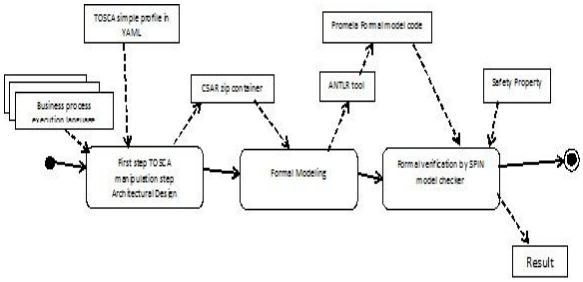

Figure 2: Architecture for Application Portability

We exhibit the case study of an online hotel reservation system. The TOSCA topology consists of six nodes namely Customer, Admin, Reception, Room, Room type and Payment type. To exhibit this we will contemplate the relationship arc called "check Room" between admin and reception. The BPEL of the admin node would invoke the check room operation furnished by the BPEL of the reception node. For each partner link invocation and receive found in BPEL indicated by <invoke> and <receive> commands we generate TOSCA Requirement Type definition and Capability Type definition respectively.

A. Example of BPEL Invocation Command

<"bpel: Import location"="CheckRoomArtifacts.wsdl"
namespace="http://www.example.com/bpel/examples"
importType="http://schemas.xmlsoap.org/wsdl/" />
...
<bpel: invokeinputVariable="reqCheckRoomVar"
"Name"="A2R_CheckRoom"
"Operation"="checkroom"
"OutputVariable"="reqCheckRoomVar"
"PartnerLink"="Reception"
"PortType"="re: Receptioninterface"/>

For all BPEL invocation command the definition of Requirement type, Relationship type and Capability type is generated. We generate the Requirement type name "CheckRoomRequirement" and it's associated Capability type "CheckRoomCapability". Then the Relationship type named "CheckRoomType" is triggered to recognize the source and target relationship and is made ready for being referred to in TOSCA Simple profile template for the admin node.

B. Example of TOSCA Simple Profile Relationship Type, Requirement Type, and Capability Type

<"RequirementType name"="CheckRoomRequirement"
"Requiredcapabilitytype"="tns:
CheckRoomCapability">
<documentation>CheckRoomRequirement</document>
<DerivedFrom typeRef="tns: BPEL
RequirementType"/>
</RequirementType>
...
<"Capabilitytype name"="CheckRoomCapability">
<Documentation>CheckRoomCapability</document>
<DerivedFrom typeRef="tns: BPEL CapabilityType"/>
</CapabilityType>
...
<"Relationshiptype name"="CheckRoomType">
<Documentation>Connected to
BPEL</documentation>
<DerivedFromtypeRef="tns: BPELRelationshipType"/>
<ValidSourcetypeRef="tns:
CheckRoomRequirement"/>
<ValidTargettypeRef="tns: CheckRoomCapability"/>
</RelationshipType>

C. *Example of TOSCA Simple Profile Node Type for BPEL Node*

Here, we offer a generic NodeType called "*BPELContainer*" for any node in the TOSCA Simple profile topology template which is hooked up with BPEL.

<"Definitions id"="NodeDefinitions" name="Node
Definitions"
"TargetNamespace"="http://example/tosca/bpel/hotelres
ervation">
<"NodeType name"="BPELContainer">
<Documentation>BPEL Container</documentation>
<DerivedFromtypeRef="ns1: SwitchYardContainer"/>
...
</NodeType>
...
</Definitions>

D. Example of Service Template for Admin Node and its Partner Links

Next, we generate the service template for the Admin Node and its Partner links.

<Definitions id="HotelReservationDefinitions"
Name="HotelReservationDefinitions"
"TargetNamespace"="http://example/tosca/bpel/hotelres
ervation"
... >
<"ServiceTemplate
id"="HotelReservation""name"="HotelReservationTem
plate">
<TopologyTemplate>

<Partition A >
<"NodeTemplate id"="AdminContainer"
"Name"="Container-Admin" type="ns1:
BPELContainer">
<Properties>
...
</Properties>
<Requirements>
<"Requirement id"="Admin _CheckRoom"

```
"Name"="portType"
"Type"="ns1: CheckRoomRequirement"/>
</Requirements>"
<Capabilities>
...
</Capabilities>
</NodeTemplate>
<"NodeTemplate id"="ReceptionContainer"
"Name"="Container-Reception"
"Type"="ns1: BPELContainer">"
<Properties>
...
</Properties>
<Requirements>
...
</Requirements>
<Capabilities>
<"Capability id"="Reception_CheckRoom"
Name="portType"
Type="ns2: CheckRoomCapability"/>
</Capabilities>
</NodeTemplate>
<! -- Partition B -->
<"RelationshipTemplate id"="A2R_CheckRoom"
"Name"="CheckRoom"
 "type"="ns3: CheckRoomType">
<"SourceElement ref"="Admin_CheckRoom"/>
<"TargetElement ref"="Application_CheckRoom"/>
</RelationshipTemplate>
</TopologyTemplate>
</ServiceTemplate>
</Definitions>
```

Now the Business Process Execution Language along with the standards of TOSCA Simple Profile in YAML are merged to produce the CSAR files. The ANTLR tool can be used to read the CSAR files and convert them into Promela code. The Properties of the Promela code written in linear temporal logic are verified by the SPIN tool. The Model Checker checks for the safety properties in an exhaustive manner.

Let us consider the nodes of admin and reception for checking the mutually exclusive property between the two. The code for the mutually exclusive property can be stated as follows.

E. Code for verifying Mutual Exclusive Property between Admin and Recep

```
# define true 1
# define false 0
# define Adminturn false
# define Recepturn true
bool x, y,t;
proctype Admin ()
{
x=true;
t=Recepturn;
(y==false || t== Adminturn);
/* critical section */
x=false
}
proctype Recep()
{
y=true;
t= Adminturn;
(x==false || t==Recepturn);
/* critical section */
y=false
}
init
{
run Admin (); run Recep ()
}
```

10. CONCLUSION

In this paper, we focussed on application portability in the cloud computing environment to get rid of the vendor lock-in issues. We discussed the scenario, facets, and challenges in achieving application portability in the cloud computing environment. The standardization and intermediation were found to be the two possible solutions in achieving application portability in which we studied for standardization. We discussed the TOSCA Simple Profile in YAML written for standardizing the application in cloud computing. We studied WS-BPEL evolution from XML. Then we realized how the Promela and SPIN can be used for formal modeling and verification of the application properties. We proposed the model for the online hotel reservation system and specified the node for Admin and Reception. We provided the sample of BPEL invocation command along with node relationship type, requirement type, and capability type. We also provided the sample of service template for the Admin node and its partner links. Then we write the code for verifying mutual exclusive property between two nodes.In the future, we will try to model different applications using these languages and tools verifying the properties of it from a different perspective.

REFERENCES

1. Docker: **Containerization Platform Available on**https://www.docker.com.
2. Linux Foundation: **Open Container Initiative Available on** https://www.opencontainer.org.
3. Kubernetes: **Container Orchestration Available On** https://Kubernetes.io.
4. Yup, Max, Cao J, LuJ, and **Application Mobility in pervasive computing: A Survey. Pervasive and Mobile Computing** 2012, 2013; 9; 2-17. https://doi.org/10.1016/j.pmcj.2012.07.009
5. GonodisF, Paraskakis I, and Simons AJH Kourtesis D. **Cloud application portability: An Initial view** BCI'13; September 19-21, 2013. ACM.978-1-4503-1851-8/13/09.

6. Peteu D, Macariu G, Panica S, Craciun C. **Portable Cloud application from theory to practice. Future Computer Generation System** 2012; 29(2013): 1417-1430 https://doi.org/10.1016/j.future.2012.01.009.
7. Binz T, Breiter G, LeymanF, Spatzier T. **Portable cloud services using TOSCA.IEEE Internet computing** vol.16 no.3 pp 80-85. May-June 2012. DOI:10.1109/MIC.2012.43 http://doi.ieeecomputersociety.org/10.1109/MIC 2012.43.
8. D. Peteu, **Portability, and Interoperability between clouds: challenges and case study springer** 2011. https://doi.org/10.1007/978-3-642-24755-2_6
9. N. Levitt's**cloud computing really ready for primetime?"** computer, vol 42 pp 15-20, 2009. https://doi.org/10.1109/MC.2009.20
10. Z. Hill and M. Humprey**," CSAL: A Cloud storage Abstraction layer to enable portable cloud application."** IEEE 2010.
11. **Apache Foundation "Jclouds."** 2011. Jclouds.apache.org.
12. **"Libcloud",** 2009.libcloud.apache.org.
13. E.A.N. da silva, V.G. dasilva, D. Lucrdio and R.P. de **MattoS Fortes" Towards a model-driven approach for promoting cloud PaaS Portability",** IEEE 2013. https://doi.org/10.1109/CLEI.2013.6670667
14. Kopp, O; Binj, T, T; Breitenbucher, U; and Laymann, F (2012) BPMN4TOSCA: **A Domain-specific language model management plans for composite application,** pages 38-52. Springer Berlin Heidelberg.
15. OMG (2011) **Business Process Model and Notation (BPMN 2.0)** http://www.omg.org/spec/BPMN/2.0/. As Accessed on 15-2-17.
16. Katsaros, G; Menzel M Lenk A. Revelant JR; Skip R and EberhardtJ. (2014). **Cloud Application Portability with TOSCA, Chef, and OpenStack. In Proceedings of the 2014 IEEE International Conference on Cloud,** Engineering I2CE'14 pages 295-302, Washington DC USA IEEE Computer Society. https://doi.org/10.1109/IC2E.2014.27
17. Wettinger J, Binz T, Breitenbucher U; Kopp O; Leymann F and Zimmermann, M (2014). **Unified invocation of scripts and services for provisioning, deployment, and management of cloud applications based on TOSCA.** In Proceedings of the 4th International conference on cloud computing and services pages 559-568.
18. Wettinger, J; Breitenbucher, Kopp O and LeymannF (2016). **Streamlining DeVos automation for cloud application using TOSCA as a standardized metamodel.** Future generation computer system 56: 317-332. https://doi.org/10.1016/j.future.2015.07.017
19. Breitenbucher U, BinzT, Kopp O; Leymann F and Wettinger, J (2015**). A Model concept to integrate declarative and imperative cloud application provisioning technologies**. In Proceedings of the 5th International conference on cloud computing and services pages 487- 496.
20. Menychats A, Kenstanteli K, Alonso J,OrneEchevarria L, Gorronogoitia J, Kousiouris G, Santzaridon C, Bruneliere H, Pellens B, Stuer P Stran B O, Senakova T, and Varvarizon.TA (2014). **Software modernization and cloudification using the artist migration methodology and framework scalable Computing: Practice and Experience**, 15(2). https://doi.org/10.12694/scpe.v15i2.980
21. Brogi, A; Carrasco, J; Cubo J; Nitto, E; Duran F; Fazzolari, M; Ibrahim. A;Rossini, **A (2016) Cloud application Modeling and Execution Language (CAMEL) and the PaaSageworkflow,** pages 437-439. Springer International publishing cham.
22. Ferry, N; Almeida, M, and Solberg, A (2017). **The MODA Clouds Model-Driven Development** pages 23-33. Springer International Publishing, cham. https://doi.org/10.1007/978-3-319-46031-4_3
23. Bassiliades, N; Symeonidis, M; Meditskos, G; Koutopoulos, E, Gouvas P, and Vlahavas, I (2017). **A Semantic recommendation algorithm for the passport platform as a service marketplace. Expert systems with the application**, 67:203-227. https://doi.org/10.1016/j.eswa.2016.09.032
24. **OASIS TOSCA Profile in YAML** Version 1.2.
25. http://www.Spinroot.com.
26. 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
27. P. Mell and T. Grance, **The NIST definition of cloud computing‖,** National Institute of Standards and Technology, Gaithersburg, 2011. https://doi.org/10.6028/NIST.SP.800-145
28. 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.
29. O. T. TC. (2016) **Topology and orchestration specification for cloud applications version 1.0.** [Online]. Available: http://docs.oasisopen.org/tosca/TOSCA/v1.0/csprd01/TOS CA-v1.0-csprd01.pdf.
30. G. Juve and E. Deelman, **Automating application deployment in infrastructure clouds, in Cloud Computing Technology**

and Science (Cloud-Com), 2011 IEEE Third International Conference on, 2011, pp. 658665.

31. O. C. T. Members (2016**) Cloud application management for platforms**, version 1.0. [Online]. Available: https://www.oasisopen.org/committees/download.php/472 78/CAMP-v1.0.pdf.
32. OpenTOSCA. (2016) **Opentosca initiative.** [Online]. Available: Http: //www.iaas.uni-stuttgart.de/OpenTOSCA/
33. OpenTOSCA. (2016) **Opentosca ecosystem**. [Online]. Available: http://les.opentosca.de/v1/
34. Openstack. (2016) **Heat - OpenStack Orchestration. [Online].** Available: https://wiki.openstack.org/wiki/Heat
35. AparnaVijaya, Pritam Dash, and Neelanarayanan V., **Migration of Enterprise Software Applications to Multiple Clouds: A Feature-Based Approach, Lecture Notes on Software Engineering, Vol. 3, No. 2, 2015** https://doi.org/10.7763/LNSE.2015.V3.174
36. 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.
37. 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,** 2011, Albuquerque, New Mexico, pp. 258-263
38. S. Charrington, **"Don't Pass on PaaS in 2010,"** (ebizo), [online]2010, http://www.ebizq.net/topics/cloud_computing/features/122 79.html? page=2.
39. 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. https://doi.org/10.1109/CloudCom.2011.45
40. N. Loutas, E. Kamateri, and K. Tarabanis, **—Cloud Semantic Interoperability Framework‖,** Cloud4SOA, D1.2, 2011.
41. O. T. TC. (2016, Nov.) **Topology and orchestration specification for cloud applications** version 1.0. [Online]. Available: http://docs.oasisopen.org/tosca/TOSCA/v1.0/csprd01/TOS CA-v1.0-csprd01.pdf
42. Daniel Oliveira and Eduardo Ogasawara. Article: **Is Cloud Computing the Solution for Brazilian Researchers?** International Journal of Computer Applications 6(8):19–23, September 2010. https://doi.org/10.5120/1096-1432
43. ISO/IEC 19941 **Cloud computing - Interoperability and Portability** https://www.iso.org/standard/66639.html