UCC: UML Profile to Cloud Computing Modeling

Using stereotypes and tag values

Ali Kamali
Department of Computer Engineering and IT
Amirkabir University of technology
Tehran, Iran
a.kam@aut.ac.ir

Soheil Mohammadi
Department of Computer Engineering and IT
Amirkabir University of technology
Tehran, Iran
soheil.mohammadi@aut.ac.ir

Ahmad Abdollahzadeh Barforoush
Department of Computer Engineering and IT
Amirkabir University of technology
Tehran, Iran
ahmad@aut.ac.ir

***Abstract*— Today cloud computing has become one of the common technologies that most of the companies want to migrate their legacy systems or deploy their new system to it. Besides modeling the system, software designers need to model the deployment infrastructure, which their system will be deployed on it. In this paper first of all we presented and categorized the requirements of modeling cloud and then illustrated how the software designer can use the advantages of UML's extendibility to model the deployment of cloud computing systems. By using stereotypes and tag values, it is possible to define a meta-model which is suitable for modeling the system deployed in cloud computing. We show that by using the UML profile, it's possible to model the infrastructures and instances of the cloud. Also it's probable to cover all the requirements that the software designer needs to model the cloud computing. It is concluded that it is important to use a standard modeling language to model the cloud, which makes it possible to model and test the whole system with a unified language. The standard languages such as UML reduce the cost required for understanding and designing the cloud computing's model. It's also possible to use this model in MDA (Model Driven Architecture) to understand and test the system's behavior in the cloud computing before deploying it in the cloud.**

Keywords*— UML; Profile; Cloud Computing; Modeling; Meta-model; Software Engineering

I. INTRUDOCUTON

Cloud computing is one of the common technologies that makes the companies to move their legacy systems or their new systems towards it. One of the most common problems that should be solved in developing or migrating these kinds of systems is that software designers and developers may not have direct access to cloud services, so they need to model cloud computing to analysis and test the systems before deploying it on cloud computing.

The followings describe the structure of this paper. First of all, a definition of cloud computing and UML profile is provided and then the requirements that should be met in order to model a system, which will be deployed in cloud computing, is listed. Then, a model is proposed and then a case study is modeled to show that our UML profile meet all the requirements that are needed to model this case study. In the last section a comparison of the proposed model other approaches is provided and coverage of all the requirements in the case study is shown.

A. Cloud Computing

Based on [1, 2, 3] , cloud computing can be defined as a computing model that by virtualizing the existing resources can produces dynamic and manageable virtual resources (e.g., networks, servers, storage, CPU, memory, and services). Cloud computing has three service models (SaaS[1], PaaS[2] and IaaS[3]) and four deployment models (Private cloud, Public cloud, Community cloud and Hybrid cloud). One of the most common service models in cloud computing is IaaS which aims to create a virtual machine with specific resources that the user needs. Also it can be created and modified any time as user requests and enables them to have more control on their services. Most of the cloud providers, like Amazon, provide this kind of service to their users with special tools that can help them to manage their VMs (virtual machines) easily [4].

B. UML Profile for Modeling New Platforms

Modeling is an essential phase of software development projects. This phase of development is performed before the implementation phase. By modeling, the designer and the developer will guarantee that all business needs are complete and correct and all end-user requirements are met. Therefore, they can guarantee the software development project's success. Without modeling, the risk of not meeting all the stakeholder's requirements increase which makes the project a failure, waste of time and budget. [5]

Booch believes that "Modeling is a central part of all the activities that lead up to the deployment of good software. We build models to communicate the desired structure and behavior of our system. We build models to visualize and control the system's architecture. We build models to better understand the system we are building, often exposing opportunities for simplification and reuse. We build models to manage risk." [6]

[1] Software as a Service
[2] Platform as a Service
[3] Infrastructure as a Service

978-1-4799-5359-2/14/$31.00 ©2014 IEEE

In this paper, the Unified Modeling Language (UML) used to model the cloud computing environment. Designers can specify, visualize, and document models of software systems with UML diagrams, including structural, behavioral and interactional diagrams, which describe a software system completely. With any one of UML-based tools, our future application's requirements can be analyzed and a solution to meet them can be designed. Using UML, we can model all types of applications, and prepare them before implementation; regardless of any type of software and hardware platform, operating system, programming language and network. UML 2.0 defines thirteen types of diagrams that model structural, behavioral and interactional characteristics of software systems. [7]

The profile is a generic extension mechanism, which defines undefined features that need to create a new version of a modeling language, or customize an existing model language for a particular platform or domain, such as UML.

There are three methods to extend UML and define a UML profile:

1. Stereotypes
2. Tag values
3. Constraints

Stereotypes, tag values and constraints are applied to specific model elements, like classes, attributes, operations, etc. A profile includes a collection of stereotypes, tag values and constraints that customize UML.

In this work, UML is extended and a UML profile to model the cloud computing environment using stereotypes and tag values is proposed.

II. Related works

There are some researches, which have attend to model the cloud computing environment with the model driven approach that we proposed two of them. One of the most popular of them is "CloudML" [8], which was proposed by Eirik Brandtzæg. This work is about modeling cloud computing, using xml-based files to use in model configurations of instances that can be transferred to cloud computing command using CloudMLEngine which used by APIs like Jcloud[9], LibCloud[10], DeltaCloud[11] and etc.

Another work, which was done in Europe, is MODAClouds [12] its approach is about designing model driven approach, which contains three abstraction layers: Computation Independent Model (CIM), a Cloud-Provider Independent Model (CPIM) and a Cloud-Provider Specific Model (CPSM). They attend to use a SOA model driven approach and modeling cloud to have the complete model driven approach in designing and publishing service.

All of these works have a different approach from our work and they do not use any visual model or standard modeling languages that can be easy to learn or have strong extendibility facilities. There is another new work recently called "MULTICLAPP" that we have the same approach. It is about presenting a UML profile for modeling multicloud applications, which is focused on the modeling multicloud provider but it doesn't cover all necessary components that should be modeled in order to meet software designer's requirements.

III. Proposed model

In this section, in the first step we list the requirements that the model of the cloud should meet in order to model the system which is deployed on instances in a cloud computing environment. In the next step, we show and explain our meta-model and finally, we display how our case study can be modeled by using this meta-model.

A. Requirements for modeling cloud computing

In cloud computing, we have many items that should be modeled. In this work, we first divide these items into four groups: Design-Time Items, Run-Time Items, Monitoring and Management tools, and accounting, which according to [14, 15, 16] we can define them as follow:

1) Design-time Items: Some items in cloud computing are static and developers deal with them before run-time. They should be prepared before run-time. The list of these items are as follows:

a) Image: An image (disk image) is a computer file which contains contents and structure of a disk volume or a data storage device, such as hard drive, tape drive, floppy disk, optical disk, or USB flash drive. Image file format may be open standards, like ISO image format for optical disk image, or propertiary for particular software applications.

b) Template: A template contains predefined configurations used by consumers to setup cloud services. A template provides necessary technical information to build ready-to-use clouds. Each template contains special configuration details for different cloud infrastructures, with information about servers for particular tasks, such as hosting applications, databases, and websites and so on. The template also includes predefined web services, operating system, databases, security configurations and load balancing.

c) Resources: Like CPU, Memory, Storage and etc.

d) Script: The script is a kind of configuration file that runs after running the instance in order to config or install some kind of features which are required as user mentioned.

2) Run-time Items: Some other items which appear at the run-time so the behavior of them should also be modeled.

a) Instance: The instance is a virtual server running a guest operating system based on the template and image from which the instance was cloned.

3) Monitoring and Management Tools:

a) Instance Descriptions: These kind of tools are for managing and monitoring the instance and can be provided by different interface like command-line, WebUI and API.

b) Cloud Watch: This tool is for managing and controlling the scale of then instances using predefined rules.

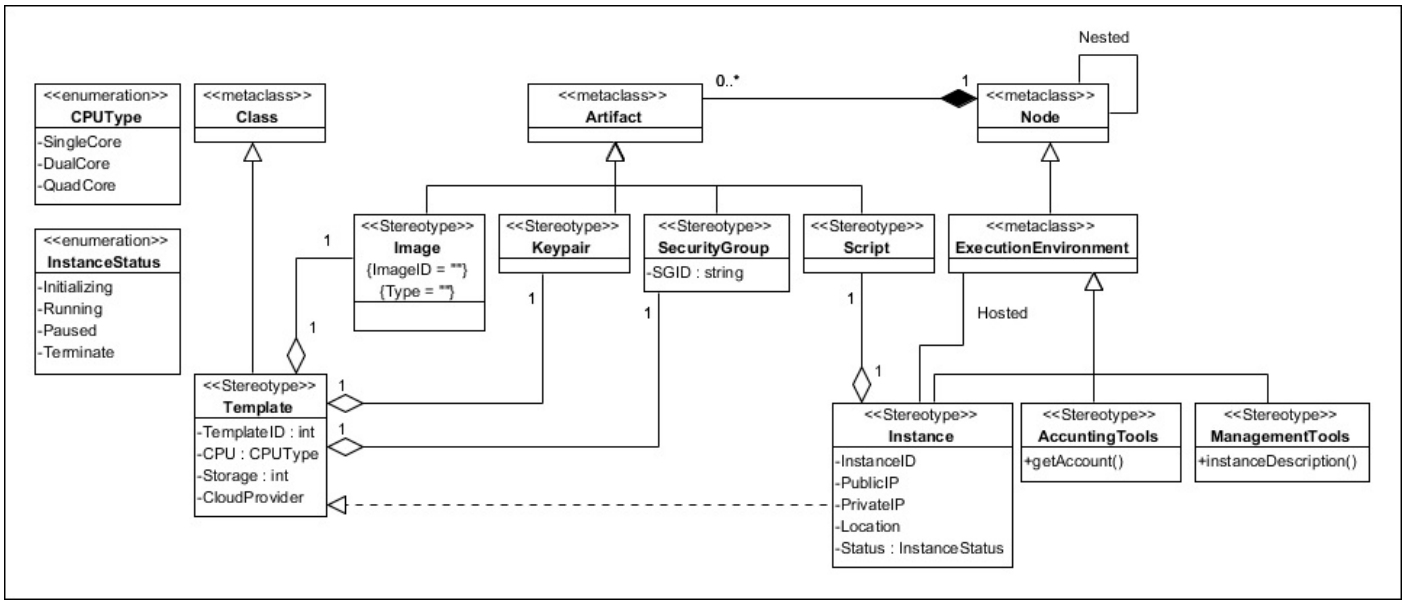

Figure 1 – Our proposed Meta-model

4) Accounts: It's also possible to use different cloud services from different providers called inter-cloud. So the model should have this facility to provide connection to inter-cloud.

a) Key pairs (Username): are some kind of keys to enable users to access their instances.

b) Security Groups: are the list of rules mostly about firewall rules which apply to instances.

B. The Proposed Meta-Model

In this work, we propose a meta-model to deploy an application in a cloud computing environment. We used UML to present this meta-model as an UML profile to deploy an application in a cloud computing environment. This meta-model is represented in Figure 1.

Enumeration class "CPUType" specifies the CPU type, which is used in each instance. The CPU type in each instance can be one of these three types: SingleCore, DualCore and QuadCore. Another enumeration class, which is defined in this meta-model, is "InstanceStatus". This class represents the different states that an instance may have. Each instance may have one of these four states: Initializing, Running, Paused, Terminated.

In this meta-model, a stereotype class, which is named "Template" is defined. This stereotype class represents the "Template" specifications. These specifications are TemplateID (template ID number), CPU (CPU type in a template, which is from "CPUType" enumeration class type), Storage (required storage space) and CloudProvider (cloud provider who deploys the application on its cloud). This class is inherited from metaclass "Class". A template for creating needs an image. Therefore, a stereotype class "Image" should be defined, which has an aggregation relation with "Template" class. Class "Image" has some tag values, such as ImageID (image ID number), Type (image type, that can be one the types "Windows", "Linux", etc.). Class "Image" is inherited from metaclass "Artifact". In addition, another class named "Script" is inherited from metaclass "Artifact". Stereotype class "Script" represents all scripts which are provided by cloud developers to run when a template is initializing. For example, a MS SQL Server 2012 script runs in windows 7 which is installed on a template and to install MS SQL Server 2012 software on it. There are two other stereotype classes, which are inherited from metaclass "Artifact". One of them is stereotype class "Keypair"; and another one is stereotype class "SecurityGroup" with an attribute named SGID (Security Group ID number).

Another stereotype class, which is defined in this meta-model, is class "Instance". An instance represents an execution environment in the cloud computing environment; so, this class is inherited from metaclass "ExecutionEnvironment". A template is installed in each instance. Therefore, there is a realization relation between class "Instance" and class "Template". An instance has some specifications, which is defined as attributes of class "Instance". These specifications are InstanceID (instance ID number), PublicIP (IP which used to access through the internet), PrivateIP (IP which used to access through the local network), Location (geographical location of instance) and Status (instance status at any time, which is from "InstanceStatus" enumeration class type). In addition, there are two other stereotype classes, which are inherited from metaclass "ExecutionEnvironment". Stereotype class "AccountingTools" with some functions, such as 'getAccount()'; and stereotype class "ManagementTools" with some functions, such as 'instanceDescription()'.

C. Application of the proposed model to a case-study

For more explanation of our model, we used the common bank management's system case study which is deployed in

cloud computing environment, based on our proposed meta-model as showed in Figure 2.

According to the model, we create two images to build the needed templates. One of those images relates to the Linux operating system and another one is related to the Windows operating system. Using the Linux image, we build a template for the bank database that is named "BankDatabase". The image has some specifications, such as:

- TemplateID = TID_12
- CPU = DualCore
- RAM = 6 GB
- Storage = 100 TB
- CloudProvider = CP_1

In addition, we use the Windows image to build two templates: "BankApplication" and "BankInterface". The specifications of "BankApplication" are:

- TemplateID = TID_7
- CPU = QuadCore
- RAM = 8 GB
- Storage = 1 TB
- CloudProvider = CP_2

And the specifications of "BankInterface" are:

- TemplateID = TID_71
- CPU = DualCore
- RAM = 4 GB
- Storage = 500 GB
- CloudProvider = CP_2

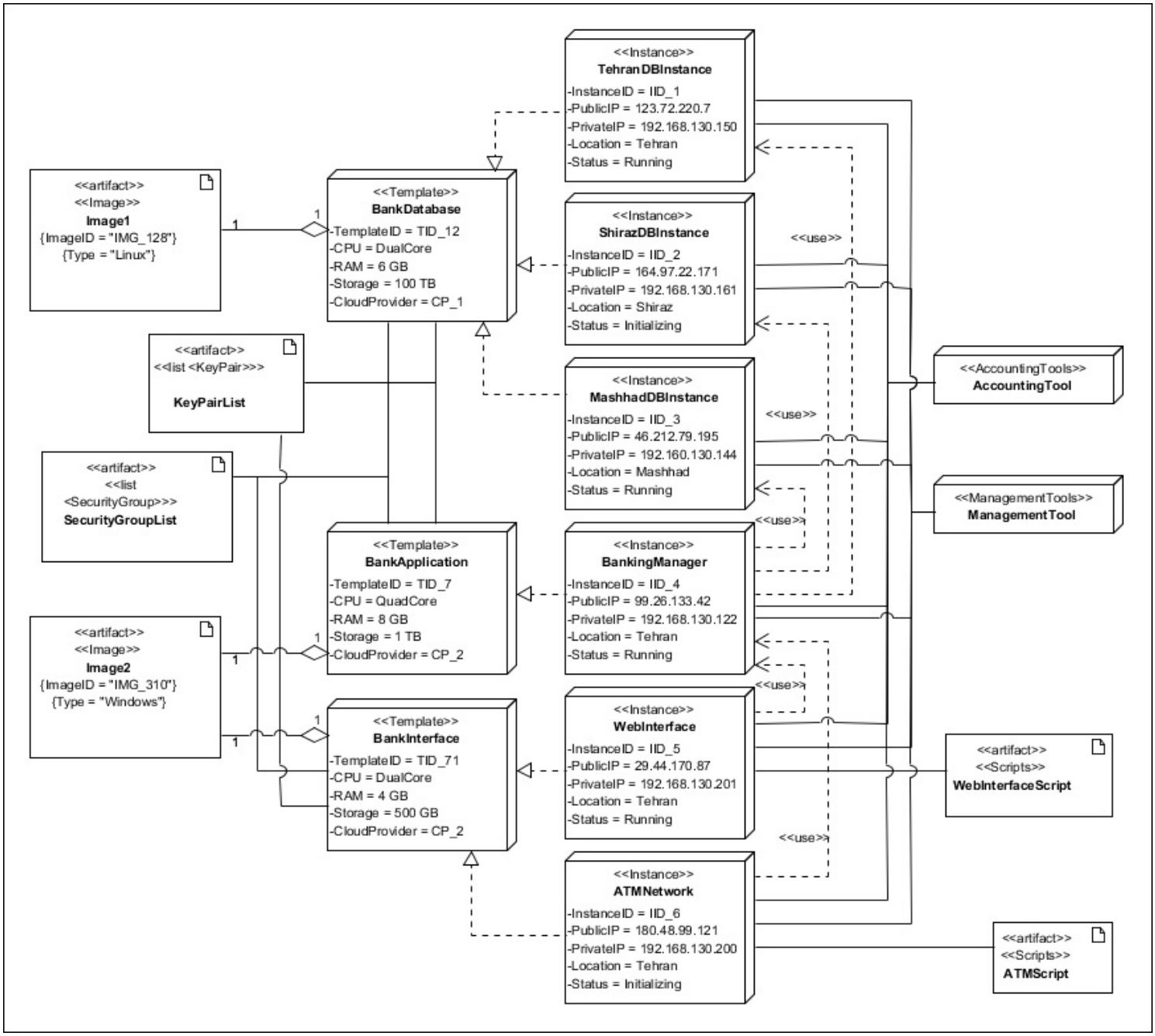

Figure 2 - Apply the proposed model to case-study

Now, based on proposed meta-model and provided descriptions in previous sections, we can use these templates and define instances. In this case study, we assume that the database is distributed in three locations: Tehran, Mashhad and Shiraz. As a result, we need to define three instances to deploy the database in the cloud computing environment. Each instance is used as a platform to place a part of bank database in; one instance in Tehran cloud, another one on Mashhad cloud, and third on Shiraz cloud. The specifications of each instance, as follows:

- TehranDBInstance:
  - InstnaceID = IID_1
  - PublicIP = 123.72.220.7
  - PrivateIP = 192.168.130.150
  - Location = Tehran
  - Status = Running
- ShirazDBInstance:
  - InstnaceID = IID_2
  - PublicIP = 164.97.22.171
  - PrivateIP = 192.168.130.161
  - Location = Shiraz
  - Status = Initializing
- MashhadDBInstance:
  - InstnaceID = IID_3
  - PublicIP = 46.212.79.195
  - PrivateIP = 192.168.130.144
  - Location = Mashhad
  - Status = Running

After defining the instances related to the database, we should define an instance of bank application on cloud computing environment. For this purpose, we use "BankApplication" template and create "BankingManager" instance. The specifications of this instance are:

- BankingManager:
  - InstanceID = IID_4
  - PublicIP = 99.26.133.42
  - PrivateIP = 192.168.130.122
  - Location = Tehran
  - Status = Running

This instance uses distributed database instances; so, there is a "use" relation between "BankingManager" instance and three instances related to distributed database.

The third type of instances that should be defined in this system, are interface instances, which are used by bank customers to connect banking systems. Interface instances use "BankInterface" template. We define two different interface instances for this banking system:

1. Web-based interface (Internet banking)
2. ATM Network

The specifications of these instance are as follows:

- WebInterface:
  - InstanceID = IID_5
  - PublicIP = 29.44.170.87
  - PrivateIP = 192.168.130.201
  - Location = Tehran
  - Status = Running
- ATMNetwork:
  - InstanceID = IID_6
  - PublicIP = 180.48.99.121
  - PrivateIP = 192.168.130.200
  - Location = Tehran
  - Status = Initializing

There is a "use" relation between these interface instances and "BankingManager" instance.

During the "WebInterface" instance initializing, a script, named "WebInterfaceScript" runs. Also, during the "ATMNetwork" instance initializing, a script, named "ATMScript" runs.

All of these instances are controlled and managed with both "ManagementTool" and "AccountingTool".

IV. RESULTS

First of all, we list and categorize the requirements that should be met in order to model cloud, as follows:

1) Design-times Items

R1)Images

R2) Template

R3) Resources

R4) Script

2) Run-time Items

R5) Instance

3) Monitoring and Management Tools:

R6) Instance Descriptions

R7) Cloud Watch

4) Accounts

R8) Key pairs (Username)

R9) Security Groups

As Table 1 shows our work meets all 9 requirements that we presented in previous sections and have some advantages like using a standard language which reduced the cost required for understanding and designing the cloud computing model. It is also possible to use this model in MDA (Model Driven Architecture) to understand and test the system's behavior in the cloud computing before deployment in the cloud.

Table 1 - Comparing our work with related works

	UCC	MULTI-CLAPP	CloudML	MODA-Clouds
Meet Requirements	9 out of 9	4 out of 9	7 out of 9	7 out of 9
Standard Language	yes	yes	no	no
Extendable	yes	yes	yes	yes
Ease of Use	yes	yes	no	no

V. CONLUSTION AND FUTURE WORKS

This work introduces and categorizes the requirements which are needed to model the cloud computing environment. Then it proposes the UCC to model the cloud computing by extending the unified modeling language (UML). This model has the same approach as The MULTICLAPP project, but MULTICLAPP does not meet some of the requirements (Image, Template, Script, Instance Description , Security Groups) which are needed to model the cloud, but the proposed model meets all the requirements and also uses the standard language which makes it is easy to use and understand.

As future works we intent to combine this model with service oriented modeling languages such as SOAML to get the unified modeling language for the whole system that includes services which are deployed in the cloud computing environment. Also with this language it is also possible to create a model driven approach for service deployment in cloud computing infrastructures.

REFERENCES

[1] Peter Mell, Timothy Grance, "The NIST definition of cloud computing", NIST special publication 800, no. 145 (2011): 7.

[2] I. Foster et al., " Cloud computing and grid computing 360-degree compared. " pp. 1-10.

[3] B. A. Sosinsky, " Cloud computing bible": Wiley, 2011.

[4] "Amazon Web Services (AWS) " http://aws.amazon.com/

[5] "Introduction To OMG's Unified Modeling Language™ (UML®)", Internet: http://www.omg.org/gettingstarted/what_is_uml.htm, Feb. 27, 2014 [Apr. 28, 2014].

[6] Grady Booch, James Rumbaugh, and Ivar Jacobson. "Unified Modeling Language User Guide", 2nd Edition, Addison-Wesley Professional, 2005.

[7] "OMG Unified Modeling Language (OMG UML) Infrastructure, v2.4.1" http://www.omg.org/spec/UML/2.4.1/Infrastructure/PDF

[8] Brandtzæg, Eirik. "CloudML: A DSL for model-based realization of applications in the cloud." (2012).

[9] "Apache jclouds " http://jclouds.apache.org/

[10] "Apache Libcloud " https://libcloud.apache.org/

[11] "Apache Deltacloud " https://deltacloud.apache.org/

[12] Ardagna, Danilo, et al. " Modaclouds: A model-driven approach for the design and execution of applications on multiple clouds." Modeling in Software Engineering (MISE), 2012 ICSE Workshop on. IEEE, 2012

[13] Guillén, Joaquín, et al. "A UML Profile for Modeling Multicloud Applications." Service-Oriented and Cloud Computing. Springer Berlin Heidelberg, 2013. 180-187.

[14] "Eucalyptus Documentation" https://www.eucalyptus.com/docs/

[15] "OpenNebula Documentation " http://docs.opennebula.org

[16] "Openstack Documentation " http://docs.openstack.org/