Model-based Approach to Automatic Software Deployment in Cloud

Franklin Magalhães Ribeiro Junior and Tarcísio da Rocha
Departamento de Computação, Universidade Federal de Sergipe (UFS), São Cristóvão, SE, Brazil

Keywords: Software Deployment in Cloud, Model-driven Deployment, Human-Computer Interaction, Cloud Computing.

Abstract: Cloud computing provides resources to reduce software processing costs in IT companies. There are automatic mechanisms to software deployment in cloud providers, however it demands manual coding. In this paper we present a model-based approach to automatic software deployment in cloud environment. We show a brief literature review of existent proposals to automatic software deployment in cloud. We analyzed the proposals, where five used deployment mechanisms based on script or programming language, two proposals based on manual mechanisms and two proposals use a model-based approach to software deployment in cloud, however one is still strongly tied to manual aspects and other complex to modelling. This paper presents a new detailed architecture, a use case and the conceptual view of our model-based approach to automatic software deployment in cloud. This approach aims to reduce the human efforts and time to deploy services in cloud, using UML deployment diagrams as input, in order to deploy it as much as possible on the highest abstraction layer.

1 INTRODUCTION

A way to attend the demand of large amount of data is the resources provided by cloud computing. IT Companies are considering software deployment in cloud providers, because processing distribution and larger storage capacity reduces infrastructure costs.

To implement a software system in a cloud provider requires a reconstruction of existing requirements, because cloud environments have their own architectures (Cala and Watson, 2010). An approach to automatic software deployment makes easier this process. In addition, it reduces human efforts (Cala, 2010).

Aiming investigate solutions referred to services deployment in clouds, we present a brief discussion of related proposals in (Ardagna et al., 2012), (Cala and Watson, 2010), (Chieu et al., 2010), (Juve and Deelman, 2011), (Konstatinou et al., 2009), (Li et al.,2012), (van der Burg, 2009) and (Zhang et al., 2013), to characterize and analyze the software deployment mechanisms in cloud environments, under vision of human effort employed to achieve the deployment.

This paper proposes a model-driven approach to automatic software deployment in cloud, which uses UML diagrams as input, to minimize the impact of human efforts and IT costs. In addition, we describe the architecture and the conceptual view of our approach.

The rest of paper is structured as follows. Section 2 describes the deployment in cloud. We review the related works in Section 3. Section 4 compare and analyze our proposal against the related works. In Section 5 we present our model-based approach to automatic software deployment in cloud, followed by a discussion describing the architecture, a conceptual view of proposal and the benefits to reduce human efforts. Finally, we present the conclusion and future works in Section 6.

2 DEPLOYMENT

Cloud computing is a mix between grid computing, distributed computing and virtualization concepts (Kalagiakos and Karampelas, 2011). Among the existing cloud providers, we can list some, such as Amazon EC2 (Amazon, 2013) and Azure Cloud (Microsoft, 2013).

To deploy and run an application in a cloud provider, services to support the application are necessary. In this context, services could be seen as components.

Magalhães Ribeiro Junior F. and da Rocha T..
Model-based Approach to Automatic Software Deployment in Cloud.
DOI: 10.5220/0004941601510157
In *Proceedings of the 4th International Conference on Cloud Computing and Services Science* (CLOSER-2014), pages 151-157
ISBN: 978-989-758-019-2

The OMG (OMG, 2006) defines the deployment process of components in five steps:

- Installation: which publishes software packages in a control repository;
- Configuration: configure the repository to support applications;
- Planning: provides an implementation plan to decide the part of the application that will run;
- Preparation: decides where software will run;
- Launch: decides which roles will connect the component instances in order to run the entire software.

To deploy software in cloud manually is expensive, because it is necessary to understand the cloud architecture, the security mechanisms (Savu, 2011) and others specific configurations of the cloud (Li et al., 2012).

Among the features related to software deployment, the main are:

- Coordinators: manage a software stack (Cala and Watson, 2010);
- Software stack: contains the services to support an application;
- Virtual machine (VM): where services run to support the application (Armstrong et al., 2011) and
- Client node: encapsulates the VM (Juve and Deelman, 2011).

3 RELATED WORK

The research in (Li et al., 2012) proposed encapsulation of services in virtual machines (VMs) and the deployment of services in a range of cloud scenarios, such as private cloud, busted cloud, federated cloud and multi cloud brokering. They concluded that the time to deploy depends on the scenario of the cloud.

The investigation in (Juve and Deelman, 2011) evaluated a system called Wrangler. Wrangler sends an XML description of a web service implementation to manage virtual machines provisions and to interact with several cloud providers (such as, Amazon EC2, Eucalyptus and OpenNebula) in order to deploy applications. The proposal used four concepts: clients, coordinator, agents and plugins. They realized that Wrangler makes the deployment process easier.

The research in (Ardagna et al., 2012), the authors presented a model-driven approach for design and execution of applications on multiple clouds called MODAClouds, They proposed architecture is divided in two views, design and run time to software deployment.

Aiming homogeneity access across different physical devices in a hospital that were connected using different networks topologies and complex security policies, such as MRI machines, computers, among others, in (van der Burg, 2009) the authors proposed an architecture which software components are automatically deployed in cloud, using a declarative model creating an extension of Nix.

The authors in (Cala and Watson, 2010) presented a platform for automatic software deployment in Azure Cloud. The work was motivated by the need for execution of an application that analyzes the chemical structure of a drug, the QSAR, used by a multi-agent system, called Discovery Bus. The authors build the deployment in two plans, one to install the services by workers nodes, and other to define spatial and temporal dependences, including library dependencies. They concluded that their platform was successfully applied to OS virtualization levels and can model well the spatial and temporal constrains of deployment process.

The research in (Chieu et al., 2010) presented a virtualization for building and deploying software stack in virtual machines (VM) using XML. They proposed a framework called Vega that provides mechanisms to enable administrators to configure the dependencies between services to achieve deployment in cloud. Also it makes possible to manage hardware and network resource. However, because a limitation of this approach is that it does not deal well with complex cloud scenarios.

To decouple VM from software, (Zhang et al., 2013) developed a framework for deploying applications in cloud, which aims to reduce the time and costs related to deployment. The proposal differs from the others approaches presented so far, because the software would not be pre-installed on a VM image.

The authors in (Zhang et al., 2013) presented the framework and its functionality divided into three stages: (i) in the software preparation, to the customer stores the software in a repository; (ii) in the selection stage, of the customer can select a software in the repository – the software is installed in a VM and this VM is sended and installed on the local machine of the customer, and finally (iii) the software deployment, instead of storing the application in a VM image, it runs on a local machine on the client side without installing it.

In their work, authors in (Konstatinou et al., 2009) presented an architecture that supports model driven software deployment. The technique shows some phases: the virtual solution declarative model (VSM), the model of virtual solution deployment (VSDM) and deployment plan virtual solution (VSDP). However, for each phase, it is necessary human interaction with the proposed system.

4 DEPLOYMENT MECHANISMS

Aiming to discover and analyze the mechanisms used by solutions related to automatic software deployment, we selected some relevant characteristics proposed by related work.

The existing mechanisms for the deployment of services are: manual, script-based, language-based and model-based. In (Talwar et al., 2005) the authors observed that model-based mechanisms deals well with time metric and increases the complexity of the solution (see Figure 1).

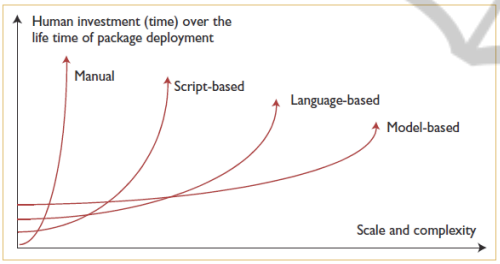

Figure 1: Deployment Mechanisms (Talwar et al., 2005).

In the solutions presented by the related work, we identified the same mechanisms (see Table 1) found in (Talwar et al., 2005).

We perceived that the mechanism presented in (Konstatinou et al., 2009) and (Ardagna et al., 2012) was the approaches that present a model-based software deployment (see Table 1).

We observed that solutions in (Cala and Watson, 2010), (Li et al., 2012) and (Zhang et al., 2013) used the language-based mechanism, which in contrast to the approaches presented in (Chieu et al., 2010); (Juve and Deelman, 2011) and (van der Burg, 2009) it leads better with time investments to the deployment process (see Figure 1). While the code solution of the approaches presented in (Chieu et al., 2010); (Juve and Deelman, 2011) and (van der Burg, 2009) are less complex, they require more time to the deployment process. However, according to (Fazziki et al., 2012) and (Talwar et al., 2005) these aspects have a lower degree of advantages, because they increase the costs with respect to the code and to human efforts.

Table 1: Deployment Mechanisms used in Proposals.

	Deployment Mechanism			
Proposals	Manual	Script-based	Language-based	Model-based
(Ardagna et al., 2012)				X
(van der Burg et al., 2009)	X	X		
(Konstantinou et al., 2009)	X	X		X
(Chieu et al., 2010)		X		
(Cala and Watson, 2010)			X	
(Juve and Deelman, 2011)		X		
(Li et al., 2012)			X	
(Zhang et al., 2013)			X	
Our Approach				**X**

Furthermore, we perceived that approaches mentioned in (Konstantinou et al., 2009) and (van der Burg, 2009) presented semi-automatic mechanisms to the deployment. In other words, in these works there is still the need for manual intervention in the deployment process.

We also perceived that the proposal in (Ardagna et al., 2012) presented a model-based mechanism to deployment, however this approach requires some understanding of the end user about details of the cloud computing structure.

Our proposal consists on a model-based approach to automatic software deployment, aiming to deploy in a higher abstraction layer to reduce the human investment and time as much as possible. Moreover, the use of model-based approach is the best way to increase productivity (Fazziki et al., 2012).

5 MODEL-BASED PROPOSAL TO DEPLOYMENT IN CLOUD

This section explains the architecture and the conceptual view of proposal to automatic software deployment in cloud, moreover, is also presented the use case of proposal. On the use case, we can observe the reduction of human efforts in deployment, because on solution is only used UML deployment diagrams as input.

Our proposal for automatic software deployment does not require additional coding efforts, it only requires as input two UML deployment diagrams: the first one is cloud provider independent (to handle software legacy), that only contains virtual machines

(VMs) and the services dependencies allocated to VMs, the second one has specific aspects related to the cloud, but only the access keys and the cloud provider name. This is possible, because our approach is supported by the 14BIS (SWX Software, 2013) and Chef (Opscode, 2013), that provides a platform to deploy third party software, where the final user of our approach (IT companies that uses the deployment as a service from 14BIS) only uses the software repository (as a general model, to handle the software legacy, being cloud provider independent), the database *url*, the name of cloud provider, the operational system and the access key as inputs.

5.1 Architecture

The proposed architecture is divided into three views (see Figure 2):

- The system view: showing the five modules of the system (*Control*, *Associator*, *Stacker*, *Allocator* and *Preparator*);
- Local view: that corresponds to the user's view, it means, the human-computer interaction (which uses as input passwords and UML deployment diagrams);
- Remote view: which includes the creation of a Chef server instance (Opscode, 2013) in the cloud and the final stage of deployment that is the software installation in the cloud.

The routine of *Associator* module starts when the user input two UML deployment diagrams (specific and general), as well as the hosts and passwords (cloud access keys). Moreover, the *Associator* module interprets UML diagrams to collect information related to UML components (a more detailed concept of each module is described in Section 5.3 and Figure 4).

The *Stacker* module is responsible for creating the software stacks, as previously specified in UML deployment diagram by the user.

The *Allocator* accesses the cloud and creates a Chef server (Opscode, 2013) instance in the cloud.

The *Preparator* module receives an acknowledgment from the *Allocator* module that the server instance was created in the cloud. So *Preparator*, prepares a client with a virtual machine (VM), it installs the operating system on the VM, allocates the services (the software stack) on VM and finally deploys the application in the cloud. Thereafter, the application can be executed by the provider.

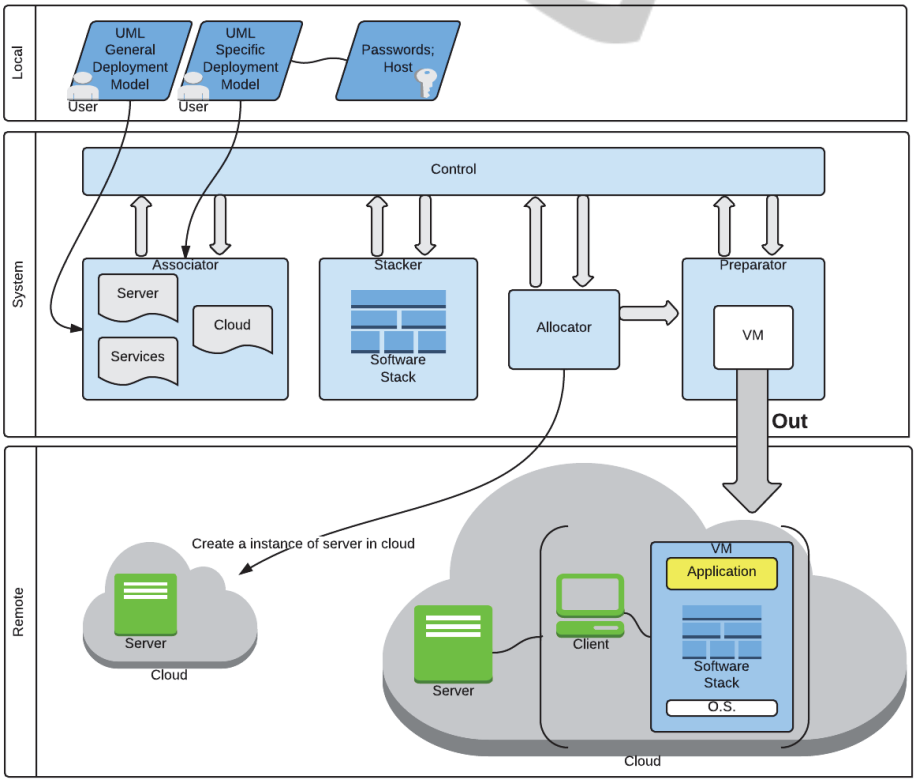

Figure 2: Proposed architecture.

The *Control* block manages the other four modules, acting as an intermediary in the communication between modules and is responsible for:

- Send *Associator* services to the *Stacker* and the list of dependency relationships between services (software stack), from *Stacker* to *Preparator* module;
- Enable communication between *Allocator* and the cloud;
- Inform to *Allocator* about creating the server instance in the cloud;
- Inform *Preparator* which operating system must be installed on the VM;
- Inform *Preparator* which software stack must be allocated on its VM;
- Manage the transmission in order to inform and perform tasks between the blocks.

5.2 Reduction of Human Efforts

The main advantage of the proposal is to focus on reducing the costs of time and human efforts to deployment. The reduction of this metrics values (time and human investment) may come from the deployment mechanism through the use of UML deployment diagrams as input (without coding). In addition, it is known that the use of model-driven development to generate code increases software quality, reduces costs and possible coding errors (Fazziki et al., 2012).

The use case (see Figure 3) aims to show the reduction of human efforts in the deployment process. In use case, it can be observed that the workload is summarized in creating UML deployment diagrams and implementation of services. The user would have just to use as input two deployment models, a general (for VMs and services) and other specific (corresponding to aspects of chosen cloud).

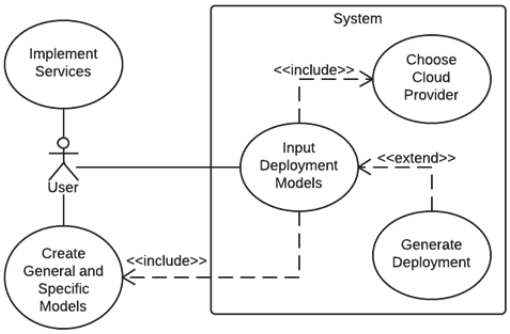

Figure 3: Use case.

5.3 Conceptual View

To observe the proposal main aspects, we have developed a conceptual view (see Figure 4), using the Amazon EC2 (Amazon, 2013) as the cloud provider, the Chef (Opscode, 2013) as server management and StarUML tool (Plastic Software, 2013) to create UML diagrams. The conceptual view is composed of four modules mentioned in the proposed architecture see Subsection 5.1), where each of these modules corresponds to a state in a state machine (see Figure 4).

5.3.1 Associator Module

In the conceptual view, the state that represents the *Associator* module is responsible to interpret the meta-language generated by the UML deployment diagram (created by user in StarUML tool). After the analysis of generated meta-language (see Figure 4), we observed that the Virtual Machine node is represented as a array element *OwnedViews[],* which has a *type = UMLNodeView*, an identifier *guid* and the coordinates as integer values by the *Left, Top, Width and Height* attributes (see Figure 4).

We perceived that the dependency relationship between the artifacts *Service [n]* and *Service [n-1]* is identified by *type = UMLDependencyView* and that this relationship has two attributes named by *head* and *tail,* which corresponds to the identifiers of UML artifacts: *Service[n]* and *Service [n-1],* where *Service[n]* depends on *Service[n-1].*Moreover, the *guid* assigned to *Service [n]* corresponds to the ID attribute *head* of the dependency relationship (see Figure 4).

5.3.2 Stacker Module

The *Stacker* module identifies which components (UML artifacts) are inside the UML node envelopment, in other words, which services belong to the node and stores them in lists of artifacts (software stacks corresponding to the respective VM node). For it be possible, it is necessary to compare the coordinates of each node with each artifact, however, this technique complexity would be $O(m*n)$, where m is the number of nodes and n, the number of artifacts.

To reduce the complexity of the algorithm $O(m*n)$, we developed an algorithm (see Figure 4) of complexity $O((n + m) + (m * \log m) + (n * \log n))$, which presents a lower complexity compared to $O(n*m)$, but only if $n >= 4$ and $m >= 3$. However, this solution uses the algorithm complexity $O(m* n)$ *if* $m<3$ and $n<4$.

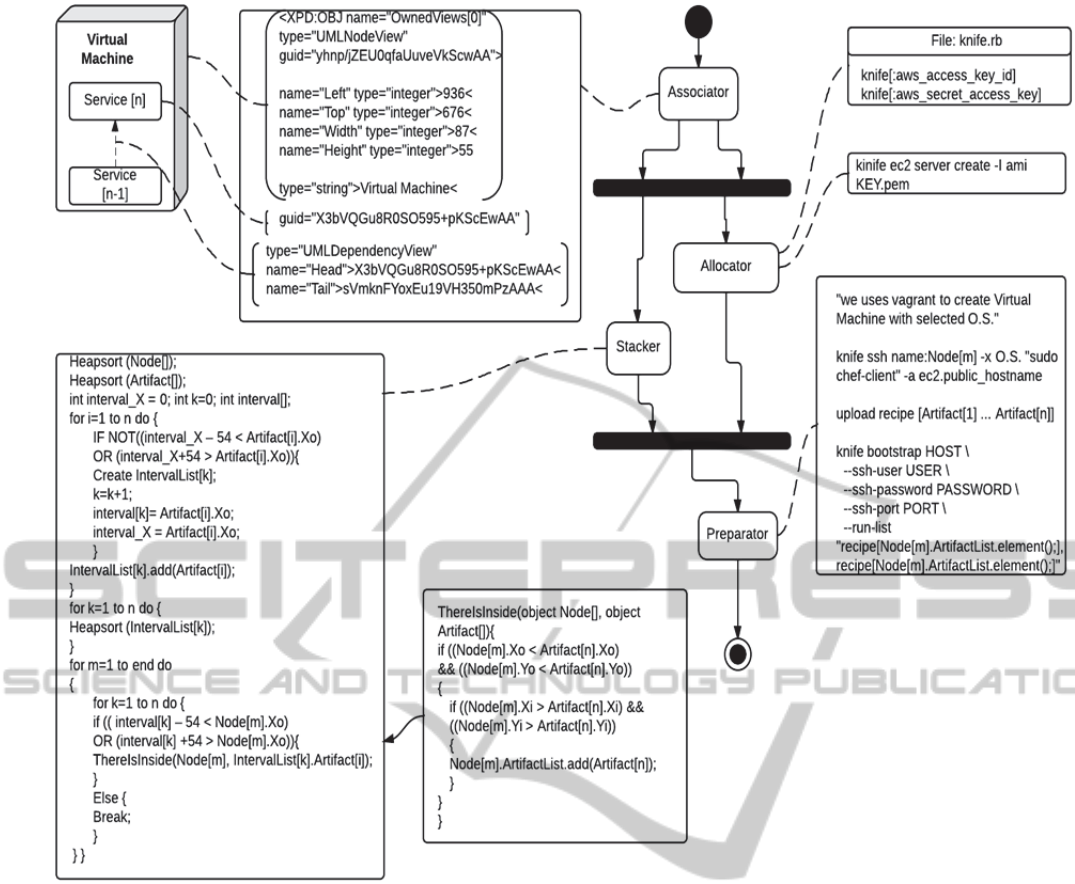

Figure 4: Conceptual view.

5.3.3 Allocator Module

The *Allocator* module, on this example, using the Chef (Opscode, 2013) as a tool for creating a server instance in the Amazon EC2 cloud (Amazon, 2013). In it is used access keys and passwords, written in the file called knife.rb, which is required by EC2 provider (see figure 4).

5.3.4 Preparator Module

The *Preparator* module is responsible for creating the virtual machine and for the installation of selected operating system in the VM. After it, this module creates a client node and allocates the virtual machine to this client.

The solution uploads all services (elements of *ArtifactList*) to the VM with the command *"upload recipe [Artifact[n-1]]…[Artifact [n]]"*. Finally it *bootstrap* services related to VM node (see Figure 4), with command *"recipe [Node[m].ListArtifact.element();]"* (see Figure 2).

6 CONCLUSIONS

In this study, we presented an approach to automatic software deployment in the cloud. Through a literature review, we identified and analyzed the properties of existing solutions. In the proposed approaches, we found four software deployment mechanisms, such as: manual, script-based, language-based and model-based.

We perceived that two researches (Konstatinou et al., 2009) and (Ardagna et al., 2012) presented a model-based proposal, but first one with strongly manual aspects and second to expert deployment users.

In this research we proposed an automatic model-based approach to software deployment in cloud to reduce the human efforts and time, using only UML diagrams as input. Furthermore, we described the architecture of our proposal, which is divided into three visions and five modules, as well as a conceptual view of the proposal was also described.

As future work, we will verify and analyze human investment and time in the company *SWX Softwares* about deployment in cloud (using our proposal against traditional proposals) to evaluate our approach.

ACKNOWLEDGEMENTS

Thanks to *CAPES* for supporting this work and *SWX Softwares* for the partnership.

REFERENCES

Amazon, 2013. Elastic Compute Cloud (EC2), Available in: http://aws.amazon.com/ec2. Access in 10, June, 2013.

Ardagna, D.; Di Nitto, E.; Mohagheghi, P.; Mosser, S.; Ballagny, C.; D'Andria, F.; Casale, G.; Matthews, P.; Nechifor, C.-S.; Petcu, D.; Gericke, A.; Sheridan, C., "MODAClouds: A model-driven approach for the design and execution of applications on multiple Clouds," *Modeling in Software Engineering (MISE), 2012 ICSE Workshop on* , pp.50,56, 2-3 June 2012.

Armstrong, D.; Djemame, K.; Nair, S.; Tordsson, J.; Ziegler, W., 2011. "Towards a Contextualization Solution for Cloud Platform Services," Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on, pp.328,331.

Cala J., 2010. Adaptative Deployment of Component-based Applications in Distributed Systems. A dissertation for the degree of Doctor of Philosophy. University of Science and Technology, Krakrów, Poland.

Cala J., Watson P., 2010. Automatic Software Deployment in the Azure Cloud. Distributed Applications and Interoperable Systems Lecture Notes in Computer Science Volume 6115, pp 155-168.

Chieu T.; Karve, A.; Mohindra, A.; Segal, A., 2010. "Simplifying solution deployment on a Cloud through composite appliances," Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010 IEEE International Symposium., pp.1,5, 19-23.

Fazziki, A.E.; Lakhrissi, H.; Yetognon, K.; Sadgal, M., 2012. "A Service Oriented Information System: A Model Driven Approach," *Signal Image Technology and Internet Based Systems (SITIS), 2012 Eighth International Conference on*., pp.466,473, 25-29.

Juve, G.; Deelman, E., 2011. "Automating Application Deployment in Infrastructure Clouds," Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on, pp.658,665.

Kalagiakos, P.; Karampelas, P., 2011. "Cloud Computing learning," Application of Information and Communication Technologies (AICT), 2011 5th International Conference on , vol., no., pp.1,4, 12-14.

Konstantinou A. V., Eilam T., Kalantar M., Totok A. A., Arnold W., Snible E., 2009. An architecture for virtual solution composition and deployment in infrastructure clouds, Proceedings of the 3rd international workshop on Virtualization technologies in distributed computing.

Li W., Svard P., Tordsson J., and Elmroth E., 2012. A General Approach to Service Deployment in Cloud Environments. 2012 Second International Conference on Cloud and Green Computing, UmeaUniversity, Sweden.

Microsoft Corporation. 2013. Azure Cloud, Available in: http://www.windowsazure.com/en-us/. Accessed: 10, June, 2013.

Object Management Group, 2006. Inc.: Deployment and Configuration of Component-based Distributed Applications Specification, Version 4.0.

Opscode, 2013. Chef. Available in: http://www.opscode.com/chef/. Access in 09, May, 2013.

Plastic Software, 2013. StarUML. The Open Source UML/MDA Platform, Available in: http://staruml.sourceforge.net/en/ Access in: 08, August, 2013.

Savu, L., 2011. "Cloud Computing: Deployment Models, Delivery Models, Risks and Research Challenges," Computer and Management (CAMAN), 2011 International Conference on , vol., no., pp.1,4, 19-21.

SWX Softwares. “14BIS - Software como Produto na Nuvem” Avaliable in: http://swx.com.br/ and http://ciomarket.com.br/ . Access in: May, 2013.

Talwar V., Milojicic D., Wu Q., Pu C., Yan W., and Jung G., 2005. Approaches for Service Deployment. IEEE Internet Computing, 9(2):70–80.

van der Burg, S.; de Jonge, M.; Dolstra, E.; Visser, E., 2009. "Software deployment in a dynamic cloud: From device to service orientation in a hospital environment," Software Engineering Challenges of Cloud Computing, 2009. ICSE Workshop on, vol., no., pp.61,66, 23-23.

Zhang Y., Li Y., Zheng W., 2013. Automatic software deployment using user-level virtualization for cloud-computing, Future Generation Computer Systems, Volume 29, Issue 1, Pages 323-329.