Variability Management in Infrastructure as a Service: Scenarios in Cloud Deployment Models

Ateeq Khan, Johannes Hintsch, Gunter Saake, and Klaus Turowski
Faculty of Computer Science, University of Magdeburg, Magdeburg, Germany
firstname.lastname@ovgu.de

***Abstract*—Flexible IT landscapes and efficient management of resources are key issues for enterprises. Variability is an important aspect for flexible IT landscape. The main part of the paper discusses variability characteristics in Infrastructure as a Service (IaaS). We show what kind of variability is possible at IaaS. In addition, a case study is provided that shows practical examples of infrastructure variability and how these variability issues can be managed using variability solutions and the software configuration tool Puppet. Software configuration tools enable us to treat infrastructure as code and to meet the varying requirements of customers. In the end, we summarize our paper and provide an outlook for future work.**

I. Introduction

Information technology (IT) systems are supported by IT infrastructures. IT infrastructure consists of software, hardware, data management, networks and telecommunications, and technology services [1]. Adaptation, customization, or variability of IT systems also require changes at the infrastructure level to meet customer requirements and to optimize the usage of resources. The National Institute of Standards and Technology (NIST) defined cloud computing as a model for enabling network access to a shared pool of resources that can be rapidly provisioned with minimal management effort [2]. The cloud model consists of three service models, namely: Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS).

Variability is the ability of a system to extend functionality, modify, customize or configure the system [3]. Variability can occur in these three service models, although variability is cross cutting in nature as discussed in [4]. These variable requirements are discussed in various papers at the application level [5] and at the process level. However, such requirements are also present at the infrastructure level, which is often not discussed. In this paper, we focus on the IaaS service model. In IaaS, customers use the infrastructure resources like servers, storage, network and other basic computing resources over the network for their applications and platforms. The cloud provider controls and manages the infrastructure only and is responsible for maintenance challenges [6]. The customer has the control over operating systems, storage types, database system, platform, deployed application and limited control over networking components, e.g. over the host firewalls [2]. An example of IaaS provider is Amazon, who provides various resources like secure storage service (S3) to store customer data and elastic compute cloud (EC2) for running their application. Customers deploy their platforms and applications on the infrastructure. Customers generate different configurations of servers, operating systems and software according to their requirements. Customers can get information about the location of the servers and where their data is stored.

In this paper, we list some of the variability requirements at IaaS level. We discuss what are the techniques to model the variability and used feature models to model some of the variability requirements. We use Puppet as the software configuration tool, which facilitate us to manage and deploy applications and infrastructure resources. In our example scenarios, we show that how we can provide variability using patterns in different cloud deployment models and in other scenarios in IaaS.

II. Variability in IaaS

Cloud service providers provide standardized offering to save on costs and management efforts. Customers have to choose the best fitting offer from the set of available options from the cloud provider. Often, because of this, varying requirements of customers are not fully addressed or met. Variability in software product lines is well discussed in literature. Variation points and variation models are used to manage variability at design time. At the infrastructure level, variability is only seldom discussed [4]. There are various requirements for IaaS, for example, they concern data security, privacy, geographical location, compliance to laws and regulations, and quantification of benefits of clouds offerings. The following list of variability options is by no means complete, and it only reflects requirements that are recurring or found in literature. Now, we discuss some of these requirements in detail.

A. Cloud deployment model

The cloud deployment model is also a variability option that has to be considered. Customers may choose between a public cloud (e.g. from Amazon, Google) or a community cloud (shared resources between specific community or organizations from a cloud provider), or a private cloud (for internal or partners use managed by organization or by a service provider solely for organization), or a hybrid cloud mixture of these deployment models. It is also possible that a customer chooses a public cloud deployment model, but afterwards, switches from one public cloud provider to the another provider. We discuss this example in detail in our case study.

978-1-5090-4588-4/17/$31.00 ©2017 IEEE

B. Technical reasons

At the infrastructure level, variability can occur in terms of servers, operating system selection, storage system types, database systems, storage locations, network facilities, hardware virtualization, software configuration, and legal issues. As in cloud computing, servers or resources are dynamically added and removed depending on the workload and needs. From provider perspective, there are two ways to meet scalability or elasticity demands of customers, namely horizontal scaling or vertical scaling. In horizontal scaling, for example, the number of virtual machines is increased or decreased. In vertical scaling, the capacity of resources is increased (e.g. adding more CPUs, memory, disk or bandwidth). Customers may require dedicated infrastructure resources such as virtual machines with specific guest operating systems from a specific cloud provider. The virtual machine requirements vary depending on the user needs. Virtual machine attributes that may vary include, for example, the no. of virtual CPU, main memory, or size of secondary storage. Customers may have different infrastructure demands based on their usage patterns (calculated from historical data or from industry segments) that the customer can specify. For example, in case of sporting events, the usage of resources is increased on an event day as compared to a normal day. Such usage pattern can help organizations to decide or choose cloud providers for a specific day and better describe their SLAs (service level agreements) to cloud providers. Similarly, various hardware variability from the perspective of storage technology is possible which we show in Fig. 1. In [7], a technique is presented to store data in the cloud in a cost effective manner. For data transfer between the customer and cloud provider, network variability is also possible depending on specific network requirements. The customer can choose from a virtual private network, dedicated fiber optics lines, virtual LAN, or the normal network between cloud provider and customer. The cloud provider and their partners offer various services for customers. Software defined networks (SDNs) are used to increase the flexibility and to adapt networks for new traffic patterns. Cloud infrastructure and networks should be adaptable and configurable to meet the QoS requirements. Hypervisor and other network vendors are introducing SDN capabilities in their offerings [8].

C. Physical location of infrastructure

The geographical location of the cloud can be determining for the customer's usage experience in cloud, e.g. in case of network latency or lag, quality, and availability. If the cloud is far from its users, then content latency issues may arise and result in an unsatisfactory user experience [9], [10]. Provider fulfill customer's requests from nearby resources or using content delivery networks for better performance, optimized delivery and to avoid latency in response. This could also be a reason to move from one cloud provider to another cloud provider due to lower costs or to avoid latency issues. We discuss this example in detail in our case study.

D. Environmental perspective

Environmental factors can also be a reason for variability in choosing cloud providers. The term green cloud computing [11] means choosing an environmentally friendly cloud provider or a cloud provider who is carbon neutral or has low carbon emissions. Organizations are keen on reducing their carbon footprint in order to attract positive public attention and to avoid penalties in case of exceeding limits of allowed carbon footprints. A customer may prefer a cloud provider who uses renewable energy, or located in a region where energy prices are low, or generally the temperature is low to save on energy costs (surveys show that cooling costs and energy costs are the major spending for a cloud data center [11], [12]).

III. Variability modeling

There are numerous modeling languages, approaches, and tools generally used to model the system landscape or infrastructure resources. The list of modeling languages, approaches, and tools includes but not limited to IT landscape model, IT system model, BPMN and their extensions, unified modeling language (deployment diagrams), architecture model, IT modeling language [13], and system modeling language. For example, in deployment diagram nodes are used to define the infrastructure resources. In [13], a domain-specific IT modeling language (ITML) is developed to model the system landscape. The domain specific language is part of a multi-perspective enterprise modeling to cover various perspectives, e.g. strategical perspective, organizational perspective, and information system perspective. These modeling perspective are covered by the meta models [14]. ITML focuses on providing graphical modeling means users with different technical knowledge in order to achieve a better business/IT alignment. However, these modeling techniques are used when designing the system for first time. All such approaches fall short of providing hints how to model the changing requirements or how to model variability requirements. The main activity in software product line engineering (SPLE) is to define common and variable artifacts in product lines. Variability management is a key concern in SPLE. Variability modeling techniques from SPLE can be used to model the variability in SOC and afterwards tools and scripts used/developed to convert the selected options into needed infrastructure configurations. We use feature models [15] to model the variability from the customer and provider perspective at infrastructure level in our case study. The feature model is used to captures the common and variable requirements. Some of the variability options, discussed in this paper, are depicted as a feature diagram in Fig. 1 (the notation is quite simple and standard in the product line community: boxes illustrate features; features connected by empty circles are optional, features connected with filled circles are mandatory, and arcs are used to describe alternatives). The variability requirements can be provided by using patterns or scripts. It is also possible to develop a plug-in supporting feature diagram. Change in feature model will be reflected in the infrastructure or will generate scripts to meet the demands.

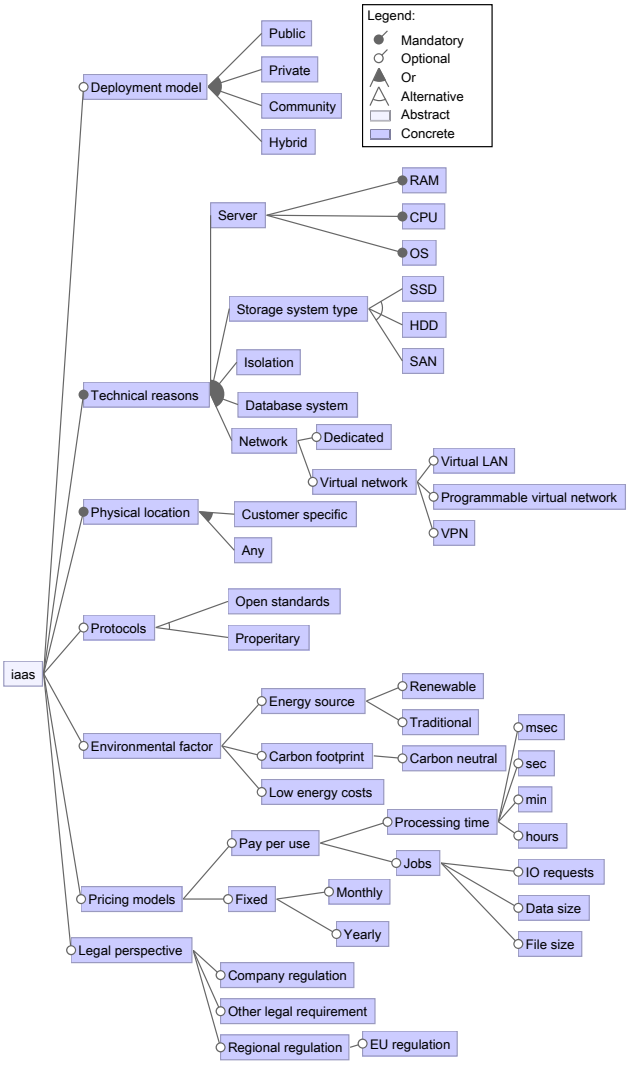

Fig. 1. IaaS variability options

IV. CASE STUDY

We show example scenarios to manage variability at the infrastructure level using the configuration management tool Puppet. The example scenarios enhances our case study of SaaS sports application [16] where we evaluate patterns in a case study. We use our case study on various cloud provider offerings with variability options and used scripts to configure it.

A. Example scenario 1

In our first scenario, a cloud service provider operates a SaaS sports application for different sports organizations. They use customized content management systems (CMS) to advertise sports events and enable interaction with the organizations' fan bases. For a customer organization, the sales department of the provider had initially come to an estimate of requests per hour that justified the operation of the CMS in the provider's private cloud data center. However, actual usage soon surpassed the estimate. As capacities in the data center of the provider could not meet this new demand, it was decided to move the system to a public cloud provider that would be able to provide the required resources.

In this scenario, we show how variability requirement (technical reasons: vertical scaling or scaling up) can be met on an operational level. The provider uses the IaaS platform OpenStack for managing its virtual infrastructure resources. Together with the customer, they had decided to move the respective resources to Amazon Web Service (AWS). Therefore, we show how to move a virtual machine from a private OpenStack installation to the public AWS cloud. Several options exist for moving a virtual machine from one provider to another. In this example, one option is to spawn a virtual machine at the target provider, configure it, migrate the data, reroute the traffic, and terminate the original virtual machine. This scenario will be detailed in the following. For configuration, the configuration management tool Puppet is used, as it is described as more mature [17] compared to its often named competitors [18].

An overview of Puppet[1] is provided in Fig. 2. In an ongoing process, four steps are executed: First, the Puppet agent that is installed on the host to be configured sends facts to the Puppet master. These facts include information such as the installed operating system or disk usage. In the second step, the master sends a catalog to the agent. This is a compiled version of the configuration option specified in the site.pp. It contains configuration instructions for all managed hosts. In Fig. 2, an Apache configuration module is used to set up a virtual host answering requests to the specified domain name on port 80 by serving from the directory /var/www. The configuration, based on the catalog, is performed in the third step by the agent. When the configuration is finished, the agent sends status information to the master in the fourth step.

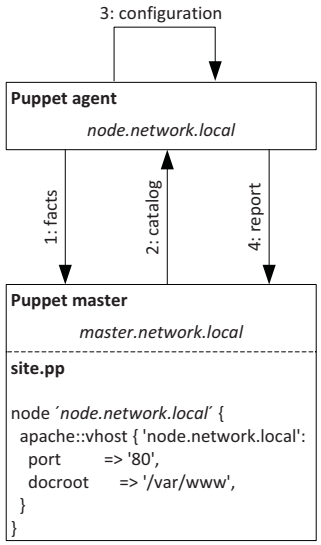

Fig. 2. Overview of the typical master/agent Puppet setup

This approach is implemented using Copy and Adapt pat-

[1]https://docs.puppet.com/

tern [16]. In the following, the virtual machine (VM) in the public cloud is called public VM and that in the private cloud is called private VM. A simple architecture is used for the CMS. The VM contains an application and database server. The application server executes the CMS and the CMS uses the database to store its data. Both the application and database server are configured by Puppet. For the CMS, a database, a database user, as well as a virtual host pointing to the CMS's source files are configured. Following steps are followed to implement the scenario: 1. spawning the VM at AWS using the AWS command line interface[2] 2. A private hostname of the public VM is added to the Puppet master's site configuration file (site.pp) along with the configuration modules that are needed to configure the host as required. The configuration agent on the public VM will then contact the configuration master and receive its configuration instructions. The agent configures the software on the VM as instructed. The configuration modules are sourced from a module repository[3]. 3. The data migration is then achieved by dumping data from the CMS's database into a file on the private VM, copying the dump file to the public VM, and importing the dump to the database in the public VM. 4. Then, the entry in the domain name service (DNS) of the cloud service provider is changed from the floating IP connected with the private VM to the floating IP connected with the public IP. Floating IPs are used to attach a publicly accessible IP address to IaaS instances (VMs). 5. The VM is terminated.

Although, this scenario enables moving the VM from one provider to the other it causes service downtime. The downtime of the service equals the time of the data migration plus that of switching DNS names. During this time, write requests cannot be served because they would cause data inconsistency. Downtime could be minimized by live migration of the data as well as by load balancing. These measures would have to be implemented at the platform and software level though.
However, it is also possible that instead of terminating the private VM, some of the customers requests are fulfilled from private cloud and other requests are forwarded to public cloud. In such case, we need data replication between both virtual machines, technique described in following scenario.

B. Example scenario 2

The second scenario concerns the variability of the physical location of infrastructure. Here, the cloud service provider needs to provide access to its customers in a different geographical location. In order to decrease latency, the provider decides to spawn an instance in a similar fashion as described in the first scenario with AWS. However, this time he chooses a data center in closer proximity to its customer's customers. The setup is depicted in Fig. 3. The public VM at the distant geographical location is set up in the same way as described in the first example. However, a database replication must additionally be set up. In this use case, we used WordPress as

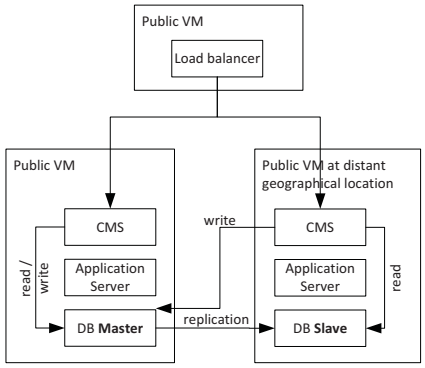

Fig. 3. Replication between database master and slave

a CMS and MySQL as a database system. Web applications, such as the discussed CMS, often have more read accesses than write accesses [19]. Therefore, the distant public VM accesses the database master for write access. All written data is replicated to the DB slave. A hardware or software load balancer can be used to forward the requests to public VMs using load balancing algorithms or rules to meet the requirements. In our scenario, we use load balancer and routing pattern [16] to forward the requests from the specific geographical location to a particular public VM.
In order to transmit this data securely (write and replication), a secure route between the VM at the distant geographical location and the public VM needs to be established. Different options exist for this. One is to add a VPN between the original public VM and the public VM at the distant geographical location. OpenVPN can be used for this. Furthermore, Puppet also provides a module for OpenVPN [4]. In order to set up such a connection, the following steps would needed to be taken: first, the VPN needs to be set up on both VMs. The original VM would be the server and the distant VM the client. Second, this would result in an added virtual network interface in a different virtual address space that is shared between the two VMs. Third, using ipTables, the network traffic of write transactions and replications would needed to be rerouted over virtual network interface. This traffic can be recognized by destination and port.

Another option is to add a dedicated network connection between the data center of the cloud provider and IaaS provider who hosts the public VM at the distant geographical location. Amazon provides such a service with AWS Direct Connect [5]. The network connection is not part of the internet and dedicated to the user of the connection. To increase security, additional network encryption could be enabled.

Other variability characteristics can be achieved in a similar fashion as described in example scenario 1 using service variability patterns. The selection of patterns depends on the variability requirement and evaluation criteria as discussed in [16]. Management of large system landscapes can also

[2]https://aws.amazon.com/cli/
[3]https://forge.puppet.com/
[4]https://forge.puppet.com/luxflux/openvpn
[5]https://aws.amazon.com/directconnect/?nc2=h_m1

be achieved by configuring virtual machines as templates (variant pattern in [16]) that are reused depending on the demands. Other scenarios are not described in detail due to space limitation in this paper.

V. Related work

Numerous approaches and solutions exist to manage the infrastructure. These approaches are used to deploy and manage infrastructure resources like virtual machines, platforms, and networks. In [20], authors uses configuration management as multi-cloud enabler to avoid vendor lock-in. They also discuss some of the differences of IaaS providers. In [21], authors propose a network hypervisor service to configure software defined networks from multiple providers and to deploy new services according to customer demands in network. They also highlight the challenges of low level APIs and lack of standardization between SDN providers.

Other infrastructure provision approaches focus on better resource allocation in a heterogeneous cloud environments [22], [23]. In [24], authors discussed the packaging and deploying SaaS applications with variability descriptors for multi-tenant environments. In [25], [26], authors provide a mechanism to deal with the different QoS requirements (availability and performance) in a cloud environment. Appropriate combination of infrastructure resources is selected to handle the variation in QoS at infrastructure level.

In contrast, we provide a list of variability possibilities at infrastructure level in form of a feature model and discuss how we can model variability requirements. We also provide a case study to show how we can use Infrastructure as Code using software configuration tool Puppet.

VI. Conclusion and outlook

The paper serves as an overview to list numerous variability requirements at the infrastructure layer. We shows that what kind of variability is possible at the infrastructure level and how it can be managed. The paper also discuss how we can model variability by using variability modeling techniques from software product lines and to meet varying and changing customer demands at IaaS. We use software configuration tool Puppet in our example scenarios to show variability requirements in various cloud deployment models.

However, there are still few areas which should be addressed in future work. A graphical plug-in can be developed, e.g. showing variability options in form of feature models as discussed in Section III. A user selects the options according to the requirements, Puppet scripts are generated based on the options and executed on the environment.

References

[1] K. C. Laudon and J. P. Laudon, *Essentials of Management Information Systems*, 10th ed. Pearson Education, Inc., NJ, USA: 2013.

[2] P. Mell and T. Grance, "The NIST definition of cloud computing," National Institute of Standards and Technology, Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology Gaithersburg, Tech. Rep. 800-145, Sep. 2011.

[3] M. Svahnberg, J. van Gurp, and J. Bosch, "A taxonomy of variability realization techniques," *Software - Practice and Experience*, vol. 35, no. 8, pp. 705–754, 2005.

[4] A. Khan, C. Kästner, V. Köppen, and G. Saake, "The Pervasive Nature of Variability in SOC," in *9th Intl. Conf. on Frontiers of Information Technology (FIT'11), Islamabad, Pakistan*.IEEE, Dec. 2011, pp. 69–74.

[5] C.-P. Bezemer and A. Zaidman, "Multi-tenant SaaS applications: maintenance dream or nightmare?" in *Proc. of the Joint ERCIM Workshop on EVOL and IWPSE*, NY, USA: ACM, 2010, pp. 88–92.

[6] I. Neamtiu and T. Dumitraş, "Cloud software upgrades: Challenges and opportunities," in *Intl. Workshop on Maintenance and Evolution of Service-Oriented and Cloud-Based Sys.*. IEEE, Sep. 2011, pp. 1–10.

[7] S. Agarwala, D. Jadav, and L. a. Bathen, "iCostale: Adaptive Cost Optimization for Storage Clouds," in *4th Intl. Conf. on Cloud Computing*. IEEE, Jul. 2011, pp. 436–443.

[8] P. A. Jelgersma, J. F. E. Lauzan, J. Hall, G. Lidbetter, P. Purswani, B. Scharinger, and H. Tardieu., "Future Networks: journey 2018 the 3rd digital revolution agility and fragility," http://atos.net/en-us/home/we-are/news/press-release/2015/pr-2015_01_28_02.html, Jan. 2015.

[9] A. Bora, "Cyber security challenges in using cloud computing in the electric utility industry," Pacific Northwest National Laboratory, Richland, Washington 99352, Tech. Rep. PNNL-21724, Sep. 2012.

[10] A. Khajeh-Hosseini, I. Sommerville, J. Bogaerts, and P. Teregowda, "Decision support tools for cloud migration in the enterprise," in *Intl. Conf. on Cloud Computing (CLOUD)*. IEEE, 2011, pp. 541–548.

[11] S. K. Garg and R. Buyya, "Green cloud computing and environmental sustainability," *Harnessing Green IT: Principles and Practices*, pp. 315–340, 2012.

[12] C. Bash and G. Forman, "Cool Job Allocation: Measuring the Power Savings of Placing Jobs at Cooling-Efficient Locations in the Data Center." in *USENIX Annual Technical Conf.*, vol. 138, 2007, p. 140.

[13] U. Frank, D. Heise, H. Kattenstroth, D. Ferguson, E. Hadar, and M. Waschke, "ITML: a domain-specific modeling language for supporting business driven it management," in *Proc. of 9th Workshop on Domain-Specific Modeling*. Helsinki Business School, 2009, pp. 28–35.

[14] U. Frank, "Multi-perspective enterprise modeling (memo) conceptual framework and modeling languages," in *Proc. of the 35th Annual Hawaii Intl. Conf. on System Sciences (HICSS)*. IEEE, 2002, pp. 1258–1267.

[15] K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak, and A. S. Peterson, "Feature-Oriented Domain Analysis (FODA) Feasibility Study," Software Engineering Institute, Carnegie Mellon University, Tech. Rep. CMU/SEI-90-TR-21, Nov. 1990.

[16] A. Khan, C. Kästner, V. Köppen, and G. Saake, "Service Variability Patterns," in *Advances in Conceptual Modeling. Recent Developments and New Directions*, ser. LNCS, Springer, Nov. 2011, pp. 130–140.

[17] A. Schaefer, M. Reichenbach, and D. Fey, "Continuous Integration and Automation for Devops," in *IAENG Transactions on Engineering Technologies*, ser. LNEE, Springer, 2013, vol. 170, pp. 345–358.

[18] J. Hintsch, C. Görling, and K. Turowski, "A Review of the Literature on Configuration Management Tools," in *Proc. of Conf-IRM*, no. 71. Association for Information Systems (AIS), May 2016.

[19] *MySQL 5.5 Reference Manual: Using Replication for Scale-Out*, MySQL, [Online]. Available: https://dev.mysql.com/doc/refman/5.5/en/replication-solutions-scaleout.html

[20] B. Vanbrabant and W. Joosen, "Configuration management as a multi-cloud enabler," in *Proc. of the 2nd Intl. Workshop on CrossCloud Systems*, ser. CCB '14. NY, USA: ACM, 2014, pp. 1:1–1:3.

[21] S. Huang, J. Griffioen, and K. L. Calvert, "Network hypervisors: Enhancing SDN infrastructure," *Computer Communications*, vol. 46, pp. 87 – 96, 2014.

[22] J. Dejun, G. Pierre, and C.-H. Chi, "Resource provisioning of web applications in heterogeneous clouds," in *Proc. of 2nd USENIX Conf. on Web application development*. USENIX Association, 2011, pp. 5–5.

[23] L. Gunho and R. H. Katz, "Heterogeneity-Aware Resource Allocation and Scheduling in the Cloud," in *Proc. of the 3rd USENIX Workshop on Hot Topics in Cloud Computing*. USENIX Association, 2011.

[24] R. Mietzner, F. Leymann, and M. P. Papazoglou, "Defining Composite Configurable SaaS Application Packages Using SCA, Variability Descriptors and Multi-tenancy Patterns," in *ICIW*. 2008, pp. 156–161.

[25] A. Sulistio, C. Reich, and F. Doelitzscher, "Cloud infrastructure & applications–CloudIA," in *IEEE Intl. Conf. on Cloud Computing*. Springer, 2009, pp. 583–588.

[26] H. Fernandez, G. Pierre, and T. Kielmann, "Autoscaling web applications in heterogeneous cloud infrastructures," in *Intl. Conf. on Cloud Engineering (IC2E)*.IEEE, 2014, pp. 195–204.