A Model-driven Migration Approach among Cloud Providers

Julio Sandobalin¹, Emilio Insfran², Silvia Abrahao²

¹ Escuela Politécnica Nacional, Ladrón de Guevara, E11-253, P.O. Box 17-01-2759, Ecuador
  julio.sandobalin@epn.edu.ec

² Universitat Politècnica de València, Camino de Vera, s/n, 46022, España
  {jsandobalin, einsfran, sabrahao}@dsic.upv.es

Abstract. Cloud computing has become the primary model of pay-per-use to obtain cloud services in a short time. Companies are using the cloud services to get access to computing resources located in a virtualized environment. However, the traditional method of using a single cloud provider has numerous limitations in terms of privacy, security, performance, and geography reach. Furthermore, companies are focusing their efforts on avoiding dependence on a single vendor for products and services. As a result, companies start to use multiple clouds and look for methods to move or migrate their infrastructure from a cloud provider to another one. In previous work, we have presented ARGON, which is an infrastructure modeling tool for cloud provisioning. In this paper, we propose an extension of ARGON to provide a model-driven migration approach among cloud providers.

Keywords: Cloud Computing, Infrastructure as a Service, Infrastructure Migration, Model-driven Engineering, Infrastructure as Code.

1 Introduction

Cloud Computing [1] are hardware-based services offering computing, networking and storage capacity where hardware management is highly abstracted and infrastructure capacity is highly elastic. To leverage the cloud capacities in terms of Infrastructure as a Service (IaaS) both the industry and the academia are focusing their efforts on automating the cloud infrastructure provisioning. Infrastructure as Code [2] is an approach to infrastructure automation based on software development practices. In addition, DevOps [3] (Development & Operations) is a movement which is encouraging continuous collaboration between developers and operation staff through a set of principles, practices, and tools to optimize the software delivery time. DevOps community takes advantage the approach of the infrastructure as code to develop different tools to support the infrastructure provisioning for different cloud providers. DevOps provisioning tools use scripts to define the cloud infrastructure that will be deployed in a particular cloud provider. However, managing the cloud infrastructure provisioning process through scripts is a time-consuming and error-prone activity. To mitigate this issue, we
have developed ARGON [4], which is an infrastructure modeling tool for cloud provisioning. ARGON abstracts the complexity of work with different cloud providers and different provisioning tools through a Domain-Specific Language (DSL) to model the cloud infrastructure. In addition, ARGON generates scripts used by various DevOps provisioning tools to orchestrate the cloud infrastructure provisioning.

On the other hand, every team or department of a company has their requirements in terms of privacy, performance, or geography reach. Similarly, every cloud provider offers a different type of infrastructure and services to support the enterprise requirements. Moreover, lack of portability among cloud providers and vendor-lock-in are issues that should be avoided. For these reasons, companies are starting to use multiple clouds (i.e., multi-cloud) to satisfy their needs to reach flexibility and agility required by the market. However, migrating the infrastructure among cloud providers is still proven to be a challenge both to the industry and the academia. To mitigate this situation, we propose a novel model-driven approach to migrate the infrastructure among cloud providers. The main contribution of this work is proposing an extension of ARGON to support a flexible migration process of infrastructure among cloud providers.

The remainder of this paper is structured as follows. Section 2 discusses related works and identifies the needs of a flexible migration process among cloud providers. Section 3 presents a brief introduction to ARGON. Section 4 presents our approach to a flexible model-driven migration approach among cloud providers. Finally, Section 5 presents our conclusion and future work.

2 Relation Work

In recent years, there has been much interest in look for approaches or strategies to support the migration process among different cloud providers. Following, we describe the principal research works related to this topic.

Holmes [5] proposes overcome limitations of forward-engineering and consider existing infrastructure services a model-based round-trip engineering approach to offer a method which will be evolved accordingly. Holmes incorporates runtime reflection for applying differential changes to an existing service topology. Comparing two models, such as a target model with a current de facto runtime model. The research is applied to facilitate alignment and migration of infrastructure services between cloud deployments.

Nodehi et al. [6] propose a conceptual model for Intercloud Interoperability, to enable schedule dynamic operation for Infrastructure as a Service (IaaS) between different clouds. The research work provides a better understanding of elaborates on the cloud computing architecture, appropriate metrics for Service Level Agreements (SLA) and Quality of Service (QoS) models that are required for seamless integration and interoperability between cloud environments. Then, a conceptual model for the Intercloud Interoperability Framework for Workload Migration is proposed. The novel component of the framework that provides interoperability is the Transformation Engine that maps workload between heterogeneous cloud providers, while Model Driven Architecture
(MDA) is adopted as an applicable method for developing the Transformation Engine module.

Beslic et al. [7] present an approach to deal with the issue of software migrations between Cloud Platforms. The goal is to provide assistance and tooling to re-engineer a software using the Model-Driven Architecture and Refactoring approach. It is divided into several stages. First, the discovery of a cloud software deployed on a platform using MoDisco [8]. Next, the transformation of the program structure using Model transformations on the discovered model is done. Then, fine-grained transformations are defined between an API and its counterpart on the targeted platform using a general-purpose transformation language such as TXL. Assistance on software transformation is provided by applying these rules. Finally, the migration on the targeted platform with offline and online tests suites validation to be sure that the software runs as usual.

Miranda et al. [9] present an approach based on software adaptation and Model-driven engineering (MDE) techniques, which describes a lightweight alternative for designing and building loosely coupled cloud applications composed of components that are grouped and distributed amongst different cloud providers. Due to the heterogeneity of the services and interfaces provided by such providers, elements have to be properly adapted to each provider taking into account the technical differences or mismatches that exist in each case. Miranda et al. analyze and measure the benefits of applying adaptation techniques to avoid the vendor lock-in effect. First, they analyze the impact of the vendor lock-in effect over the effort required to develop cloud-based applications. Next, the effort required for migrating the service dependent on parts of the application is measured. Finally, they describe an adaptation proposal and explain how it can be applied to significantly attenuate the negative aspects of the vendor lock-in effect in the construction.

The research works mentioned above focus their efforts on analyze models obtained from cloud services such as infrastructure and cloud-based applications. In contrast, our proposal takes advantages of model-driven techniques to automatically generate artifacts to support the infrastructure migration process among cloud providers.

3 ARGON

ARGON [4] (An infRstructure modelinG tool for clOud provisioniNg) is a tool which leverages the model-driven engineering (MDE) and supports the DevOps ideas. ARGON abstracts the cloud capacities (i.e., computing, storage, networking, and elasticity) to provide a Domain-Specific Language (DSL) for modeling a generic infrastructure model. In addition, ARGON allows automatically generate scripts used by DevOps provisioning tools for orchestrating the cloud infrastructure provisioning.

ARGON defines a generic Infrastructure Metamodel (see Fig. 1), which is the DSL’s abstract syntax and abstracts the cloud capacities. The metaclasses are explained according to the cloud capacities:

can be assigned to a Virtual Machine. A Load Balancer allows distributing the workload among Virtual Machines. A Listener checks connection requests to the Load Balancer. A Health Check validates that Virtual Machines attached to the Load Balancer are available.

- Storage capacity allows modeling Databases and File servers.
- Elasticity capacity allows modeling a Launch Configuration in which characteristics of a Virtual Machine are specified. An Auto Scaling Group determines the minimum and the maximum number of Virtual Machines to be created. Creation or elimination of Virtual Machines is done based on a Scaling Policy in which is executed an Alarm that monitors a metric at a time frame.
- Networking capacity allows modeling the associations among metaclasses.

The Infrastructure Metamodel only defines the DSL’s abstract syntax, but not a graphical notation for ARGON. To render the metamodel elements in a modeling editor we have defined a concrete syntax (see Fig. 2). We used EuGENia [10] tool to generate models needed to implement an editor for Graphical Modeling Framework in Eclipse tool. EuGENia tool uses Emfatic2 to represent the Infrastructure Metamodel in a textual manner and aggregate annotations for each infrastructure element. Fig. 2a represents a virtual machine with its security group. Fig. 2b shows the infrastructure element palette, which is used to model the cloud infrastructure elements.

---

1 http://www.eclipse.org/gmf-tooling/
2 https://www.eclipse.org/emfatic/
4 A Model-driven Migration Approach

ARGON allows modeling a generic infrastructure model (see Fig. 2). However, each cloud provider has a different type of infrastructure, which is causing many issues for supporting a migration process among cloud providers. For this reason, we have defined a metamodel for Amazon Web Services (Amazon for short) and Microsoft Azure (Azure for short). ARGON follows the Model-Driven Architecture (MDA) to abstract in layers the complexity to define both a cloud provider-independent infrastructure model (i.e., generic model) and several cloud provider-specific infrastructure models (i.e., Amazon model and Azure model). Fig. 3 shows the ARGON’s architecture which is according to MDA principles. Following, we explicate every layer in detail and present our proposal for a flexible model-drive migration approach among cloud providers.

Fig. 3. ARGON’s Architecture.

- Fig. 3a depicts the most abstract layer of MDA, which represents the context, requirements, and purpose of the solution without any binding to cloud providers. Fig. 3a shows the layer in which the criteria to obtain a flexible migration approach among cloud providers is captured. As a result, we have defined the structure and behavior of the respective platform-independent model (which is unique and generic) and the platform-specific models (one for each cloud platform) by defining their metamodels.
- Fig. 3b represents the Platform-Independent Model (PIM). In this layer, we define the behavior and structure of a generic solution to model the cloud infrastructure by means of an Infrastructure Metamodel (see Fig. 1) regardless of the cloud providers. The Infrastructure Metamodel abstracts the cloud capacities (i.e., computing, stor-
age, networking, and elasticity) specifying the requirements used by ARGON's Domain-Specific Language (see Section 3). The Infrastructure Metamodel was defined using the metamodeling language Ecore [11].

- Fig. 3c represents the Platform-Specific Models (PSM). In this layer, we have two cases: (a) PIM to PSM, where a cloud provider-independent infrastructure model (i.e., a generic model) is transformed into a cloud provider-specific infrastructure model (i.e., Azure model); (b) PSM to PSM, where a cloud provider-specific infrastructure model (e.g., Amazon model) is transformed into another one (e.g., Azure model). The metamodels specified in the PSM layer are modeled using the metamodeling language Ecore. The transformation language for performing the model-to-model (M2M) transformations is ATL [12]. For example, to perform the transformation from the Amazon model to the Azure model, we define a transformation module (e.g., Amazon2Azure.atl), which contains the transformation mapping rules that represent the correspondences among concepts of the metamodels. The M2M transformation process (see Fig. 3c) starts when the ATL transformation engine takes the Amazon model (i.e., source model) and apply a set of transformation rules to obtain the Azure model (i.e., target model). It is worth noting that we can interchange the source and target models, meaning that the Azure model can be transformed into an Amazon model given that the correspondences among concepts are bidirectional. In addition, another cloud provider model can be used by defining the respective metamodel and transformation module. As a result, our proposal to support a flexible migration approach is the process to move/migrate from a cloud provider-specific infrastructure model to another one. In this context, we should accomplish two conditions (1) specify the cloud provider metamodel, and (2) define the transformation rules between the cloud providers’ models.

- Fig. 3d represents the Instance generated from a platform-specific model (PSM). In this layer, we generate the corresponding scripts for a particular DevOps provisioning tool (i.e., Ansible, Terraform, Chef, Puppet, etc.) by means of Model-to-Text (M2T) transformations. The source for this transformation process is a PSM (e.g., Azure model) and the output are scripts used by provisioning tools (see Fig. 3d). The scripts are used to manage the orchestration of the infrastructure provisioning in the cloud providers. Each DevOps provisioning tool uses a specific type of script, e.g., the Ansible tool uses a script called playbook. In addition, the DevOps provisioning tools use different script languages to define instructions for infrastructure provisioning, e.g., the Ansible tool uses the YAML language. We abstract the specific features of each script language to define transformation rules in templates, which allow us to generate the specific scripts. These templates are defined according to the Acceleo\(^3\) tool. The M2T transformation process takes as input a cloud provider-specific infrastructure model (i.e., Amazon model or Azure model) and applies a set of transformation rules to generate scripts, as output, for a DevOps provisioning tool selected.

\(^3\) https://www.eclipse.org/acceleo/
To demonstrate the feasibility of our approach we have modeled a virtual machine as a cloud provider-independent infrastructure model (see Fig. 4), which will be transformed into a cloud provider-specific infrastructure model (i.e., Amazon model). In the PSM layer (see Fig. 3c), we have obtained the Amazon model, which will be migrated toward an Azure model. The Amazon model is conforming to Amazon Web Services metamodel. Fig. 5a shows an excerpt of the Amazon Web Services metamodel. The Azure model is conforming to Microsoft Azure metamodel. Fig. 5b shows an excerpt of the Microsoft Azure metamodel.

**Fig. 4.** Infrastructure model of a virtual machine modeled by ARGON.

Fig. 4 shows a cloud provider-independent infrastructure model of a virtual machine modeled by ARGON. In this case, by using M2M transformations, the infrastructure model is specified as an Amazon model (PIM to PSM, see Fig. 3). The infrastructure model has a webserver connected to a security-group, which performs like a firewall. The security-group has an outbound rule (all), which enables all outgoing connections of the webserver. Additionally, the security-group has two inbound rules, which enable the ingoing connections to HTTP protocol and Remote Desktop Connection (RDC) for a Windows Server. The webserver has connected an external-disk and an elastic-ip.

The webserver (see Fig. 4) has settings corresponding to a virtual machine from Amazon Web Services, however, to add features for a virtual machine from Azure platform (such as publisher, offer, and SKU) is necessary to use an azure-profile element.

**Fig. 5.** Model-driven migration of a virtual machine between Amazon model and Azure model.
Fig. 5 shows the model-driven migration approach based on M2M transformations to migrate a virtual machine from Amazon Web Services to Microsoft Azure. Following are explained changes:

- **Service2Network** and **Service2ResourceGroup** specify that for each *Service* in the Amazon model should be created a *Resource Group* and a *Virtual Network* with its *Subnet* in the Azure model.
- **Tag2Tag** specifies that for each *Tag* found in an element in the Amazon model should create the same *Tag* in the corresponding element in the Azure model.
- **SecurityGroup2SecurityGroup** specifies that for each *Security Group* with its *Inbound rules* and *Outbound rules* in the Amazon model should be created the corresponding *Security Group* with its *Inbound rules* and *Outbound rules* in the Azure model.
- **Volume2Disk** specifies that for each *Volume* attached to a virtual machine in the Amazon model should be created the corresponding *Disk* connected to a virtual machine in the Azure model.
- **VirtualMachine2VirtualMachine** specifies that for each *Virtual Machine* in the Amazon model should be created the corresponding *Virtual Machine* with its *Network Interface* in the Azure model.
- **ElasticIP2PublicIP** specifies that for each *Elastic IP* assigned to a virtual machine in the Amazon model should create the corresponding *Public IP* assigned to the virtual machine in the Azure model.

![Fig. 6. Matching of elements between Amazon model and Azure model.](image)

Fig. 6 presents the matching among infrastructure elements of the Amazon model (see Fig. 5a) and the Azure model (see Fig. 5b). The hierarchical tree view in Eclipse tool shows the infrastructure Service called *virtual-machine* that group the infrastructure elements in the Amazon model (group A), which is matched to a Resource Group that group the infrastructure elements, and a Virtual Network with its Subnet in the Azure Model (group A’). The Virtual Machine called *webserver* and its Azure Profile in the Amazon model (group B) are matching to Virtual Machine and its Network Interface in the Azure model (group B’). The rest of the infrastructure elements correspond one by one in the matching between the Amazon model and the Azure model.
Finally, according to ARGON’s architecture (i.e., Model-driven architecture, see Fig. 3), the Azure model is the input of the Acceleo tool, which generates the corresponding scripts for a particular DevOps provisioning tool. In this case, the provisioning tool selected (e.g., Terraform tool) uses the script to orchestrate the infrastructure provisioning in Microsoft Azure. Fig. 7 shows the infrastructure elements deployed in Microsoft Azure.

![Fig. 7. Infrastructure model of a virtual machine deployed in Microsoft Azure.](image)

### 5 Conclusion and Future Work

In this paper, we have presented a model-driven migration approach among cloud providers. The purpose is to provide a novel approach for a flexible migration process based on model-driven techniques and DevOps philosophy. We have demonstrated the feasibility of our proposal using the migration of an infrastructure model from the Amazon model to the Azure model. Despite the fact that we have shown the migration from Amazon Web Services to Microsoft Azure, we can interchange the cloud providers, or even add another cloud provider. The migration process in ARGON should accomplish two conditions (1) specify the cloud provider metamodel, and (2) define the transformation rules between the cloud providers’ models.

As a future work, we plan to run experiments with students and practitioners with experience in cloud computing development and in particular with knowledge of provisioning resources in the cloud. This will help us to validate the effectiveness of the proposed solution concerning the migration process among different cloud providers, such as Google Computing Engine, Rackspace, etc.

#### Acknowledgment

This research is supported by the MINECO within the Value@Cloud (TIN2013-46300-R) and Adapt@Cloud (TIN2017-84550-R) projects.
References


