A Proposal for Migrating SOA Applications to Cloud Using Model-Driven Development

Miguel Botto-Tobar[1,2(✉)] and Emilio Insfran[3]

[1] Universidad de Guayaquil, Guayaquil, Ecuador
miguel.bottot@ug.edu.ec
[2] Eindhoven University of Technology, Eindhoven, The Netherlands
m.a.botto.tobar@tue.nl
[3] Universitat Politècnica de València, Valencia, Spain
einsfran@dsic.upv.es

Abstract. Software applications are currently considered an element essential and indispensable in all business activities. Nevertheless, for their construction and deployment to use all the resources that are available in remote and accessible locations on the network, which leads to inefficient operations in development and deployment, and enormous costs in the acquisition of IT equipment [6]. This paper aims to contribute proposing SOA2Cloud, a framework to migrate SOA applications to Cloud environments, following a Model-Driven Software Development approach. An example of an application that shows the feasibility of our approach was developed.

Keywords: SOA · Migration · Cloud Computing
Model-Driven Development

1 Introduction

The fast Internet expansion has created an ideal mean for the development of applications that carry out information exchange, such as electronic transactions, product sales, and on-line services, among other functionalities. It has driven organizations to increase their interactions with customers and other companies.

Service-Oriented Architecture (SOA) is based on services development and their reuse, with a clear business and a known functionality, independent and non-coupled, it is offered through a set of intelligent services and coupled interfaces that can provide the same way in other applications [3].

Cloud computing is linked to a novel model of network delivery of different types of IT resources, for instance: basic computing resources, storage, and networking, deploying and running applications, etc., that they are part of a set of resources that can be allocated, provisioned and delivered quickly with minimal interaction between customer and service provider [23].

Cloud Computing expands to SOA adding scalability and GRID Computing. Scalability is required when the software is used as a service, hence the hardware resources are used, and scalability becomes a transcendent requirement.

M. Botto-Tobar et al. (Eds.): CITT 2017, CCIS 798, pp. 171–184, 2018.
https://doi.org/10.1007/978-3-319-72727-1_13

This work was defined taking into account the need for many companies, both public and private, perform inefficient operations in construction and deployment of their applications, and enormous expenses in IT acquisition. With computing available as a service in Cloud Computing, many companies are considering it as a solution for service delivery. Therefore, SOA applications are being migrated to Cloud; and, for this reason, the need to build a framework that allows migrating applications developed in SOA to Cloud environment using Model-Driven Software Development (MDD) [4].

The paper is organized as follows: Sect. 2 discusses related work. Section 3 presents the research method. Section 4 describes an application example, and finally, Sect. 5 presents our conclusions and suggest areas for further investigation.

2 Related Work

Our goal is to develop a framework which allows migration of SOA applications to Cloud Computing environments, using Model-Driven Software Development. In order to fulfill this objective, we first have studied the state-of-art of cloud migration in [5,7], to know which methods and techniques have been implemented, especially those which have used Model-Driven Development.

Babar et al. [2] reported their experiences and observations gained from migrating an Open Source Software, Hackystat, to cloud computing.

Tran et al. [25] proposed a taxonomy of the migration tasks involved, and they showed the costs breakdown among categories of tasks, for a case-study which migrated a .NET n-tier application to Windows Azure.

Guillén et al. [11] proposed a framework MULTICLAPP. The framework follows a three-stage development process where applications can be modeled and coded without developers having to be familiar with the specification of any cloud platform. MDE [22] approaches rely on models as a means of abstracting the development process from the peculiarities of each cloud platform.

Mohagheghi et al. [18] presented a research project (REMICS) to define methodology and tools for model-driven migration of legacy applications to a service-oriented architecture with deployment in the cloud. The main project's objective is to develop a set of model-driven methods and tools that support organizations with legacy systems to modernize them according to the "Service Cloud paradigm".

Nevertheless, they presented either approaches or experiences to migrate applications to Cloud using MDD (applications were not developed using SOA) or legacy systems.

3 Research Method

This section presents a model-based approach defined to generate Cloud applications from SoaML [20] and SLA [21] models deployed in Windows Azure. We established a migration process to carry out this approach, which is based on the SPEM notation [19].

3.1 Metamodels

SoaML. It extends the UML2 metamodel to support explicit service modeling in distributed environments. This extension aims to maintain the different service modeling scenarios such as the description of a single service, service-oriented architecture modeling, or service contract definition [20]. Our SoaML metamodel has been defined based on the specification proposed by OMG, as well as the contributions of Minerva project [9] and Irish Software Engineering Center [1].

Service Level Agreement. SLA stipulates conditions and parameters that commit the service provider (usually supplier) to comply with levels of quality of service against the contractor (usually customer) [21]. Its structure is based on elements presented in an exemplary structure that create in an SLA Fig. 1. It is clear that the description of services, or service level, objectives are the central aspect of each SLA [10].

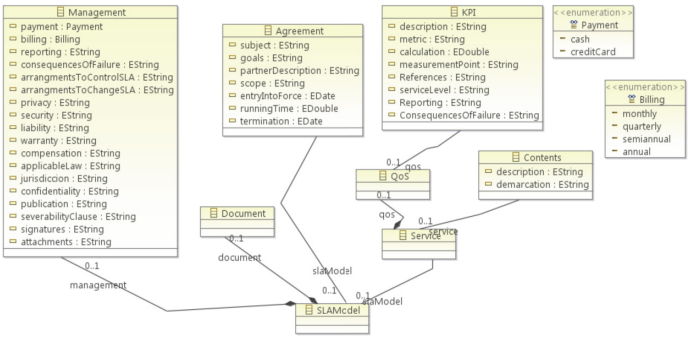

Fig. 1. SLA metamodel.

Cloud Application. SOA is an umbrella that describes any service. A Cloud application is a service. A cloud application metamodel is an SOA model conforming to SOA metamodel. It makes Cloud applications also SOA applications. However, SOA applications are not required in Cloud applications [12]. A Cloud application is an SOA application that runs under a specific environment, which is the cloud computing environment (platform). This context is characterized by horizontal scalability, fast provisioning, ease of access and flexible pricing [12].

Windows Azure. It is a technology set that provides a specific service set to application developers [8]. It also offers a platform where the ISV (independent software vendor) and companies can host their applications and data. It was selected for this research, and a generic metamodel was created. It has characteristics of the development environment and the platform. The Azure metamodel elements are detailed below (it has developed for this research):

- CloudServices. A cloudServices allows creating a solution with the two different types of instances provided by the Windows Azure platform for developers, Web Roles, and Work Roles.
- ServiceWebRole. A WebRole is a role that is customized for programming web applications with IIS 7 support, such as ASP.NET, PHP, Windows Communication Foundation and FastCGI. All information in a WebRole is stored in a file definition, and its default extension is .csdef [14].
- ServiceWorkRole. A WorkRole is a function that is useful for generalized development and can be performed background processing for a web role. All information in a WorkRole is stored in a definition file, and its default extension is .csdef [17].
- ServiceDefinition. It defines the service model for an application. It also contains the definitions of the functions available in a service, specifies the end services, and sets the configuration values for the service. The configuration values are found in the service configuration, as described by the Windows Azure service configuration schema (.file cscfg) [16].
- ServiceConfiguration. It specifies the number of instances to implement for each role in the service, the configuration parameter values and fingerprints of all certificates are associated with a role. If the service includes a Windows Azure virtual machine, the service configuration specifies the virtual hard disk for each virtual machine created. If the service is part of a virtual network, the configuration information for the virtual network must be provided in the service configuration as well as in the virtual network configuration [15] (Fig. 2).

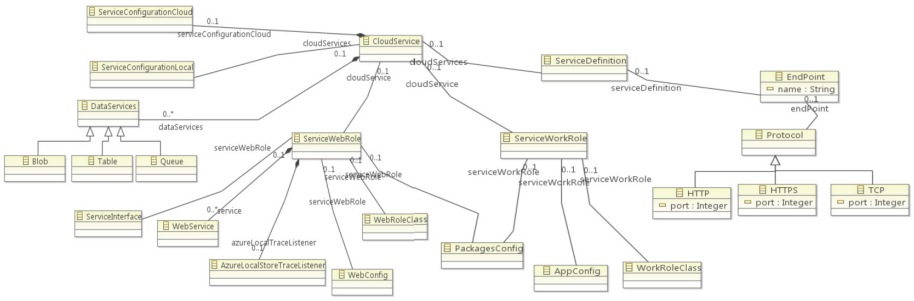

Fig. 2. Azure metamodel.

3.2 Migration

One of the important aspects of this process is the use of two models as input artifacts, and execution of model transformations: an SOA application model based on SoaML specification, and a service level agreement model based on SLA. Unlike other transformation processes, which only use a source model as

input. The migration process is divided into three phases: analysis, transformation, and deployment.

Figure 3 shows migration process. It is entirely independent since it allows developers start migrating from scratch, that is, modeling SOA applications through SoaML specification, or from existing applications modeled in SoaML.

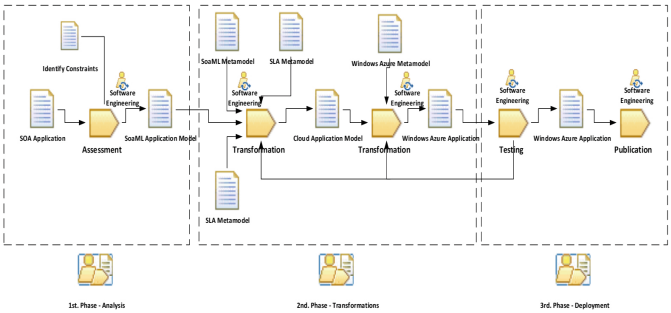

Fig. 3. Migration process.

In the analysis phase, the developer must assess SOA application to be migrated; it will be modeled in SoaML specification, identifying performance and quality restrictions that model may have. It is also important to take account in this first stage that an SOA application may also be source code written in a programming language; therefore, the developer has to use tools that allow obtaining the application model through reverse engineering. Once this phase is completed, a model is ready to be migrated.

The transformation phase is divided into two stages: (1) *transformation*, it uses two models as input artifacts, the first model is SOA application according to SoaML metamodel, and the second model is service level agreement based on SLA metamodel. Then, through model-to-model transformations, a cloud application model is obtained according to its metamodel. This model is transformed (using model-to-model transformations) into a model conforming to Azure metamodel. Finally, we proceed to perform a model-to-text transformation to obtain the source code, for this research, Windows Azure C #. (2) *model validation*, it verifies that all mappings specified in each transformation, from SoaML, SLA to Cloud, and from Cloud to Windows Azure, are matched between corresponding metamodels.

Deployment is the last phase, and the application will be deployed in a Cloud environment, Windows Azure.

3.3 Mappings

The correspondences between SoaML, SLA, and Cloud elements were defined. They specify which elements of source model conforming to source metamodel

will be transformed into the target model elements according to target metamodel when the transformations are executed.

Firstly, the correspondences between SoaML and Cloud metamodels are presented in Table 1. *SoaMLModel* element and *ServiceArchitecture* (if it exists in the model) correspond to *CloudApplication* element which includes the rest of model elements. *Participant* and *Agent* elements correspond to *CloudTask* which is where the actions of a cloud application are executed. *Consumer*, *Provider*, and *ServiceInterface* elements correspond to *TaskDefinition* which in the cloud provide the application structure, interface set, and their contracts. *MessageType* element corresponds to *Queue*, which in Cloud is responsible for message exchange coordination.

Table 1. Mappings between SoaML and Cloud.

SoaML (elements)	Cloud (elements)
SoaMLModel	CloudApplication
Participant	CloudTask
Agent	
Consumer	TaskDefinition
Provider	
ServiceInterface	
MessageType	Queue

Secondly, the correspondences between SLA and Cloud metamodels are presented in Table 2. *Agreement* element corresponds to *ConfigurationData* element which is where aspects related to service agreements of the Cloud application are determined. *KPI* element corresponds to *DataInjectionPort* element which is where QoS tasks are modified.

Table 2. Mappings between SLA and Cloud.

SLA (elements)	Cloud (elements)
Agreements	ConfigurationData
KPIs	DataInjectionPort

Finally, the correspondences between Cloud and Azure metamodels are presented in Table 3. *Agreement* element corresponds to *ConfigurationData* element which is where aspects related to service agreements of the Cloud application are determined. *KPI* element corresponds to *DataInjectionPort* element which is where QoS tasks are modified. *CloudApplication* element corresponds to *CloudServices* element which includes all other components of the model. *WebTask*

Table 3. Mappings between SLAs and Cloud.

Cloud (elements)	Azure (elements)
CloudApplication	CloudServices
WebTask + ServiceTask	ServiceWebRole
CloudRotorTask	ServiceWorkerRole
Table	SQL Azure Table
Queue	SQL Azure Queue
Blob	SQL Azure Blob
EndPoint	EndPoint
Protocol	Protocol
ConfigurationData	ServiceConfigurationLocal + ServiceConfigurationCloud

and *ServiceTask* elements correspond to ServiceWebRole which are the application services to be specified. *CloudRotorTask* element corresponds to *ServiceWorkRole* which is in charge of processing web roles in an application on the Azure platform. *Table* element corresponds to *SQL Azure Table* which in Azure is responsible for storing structured information. *Queue* element corresponds to *SQL Azure Queue* which in Azure is responsible for the storage of scalable messages. *Blob* element corresponds to *SQL Azure Blob* which in Azure is responsible for storing unstructured data. *EndPoint* element corresponds to *EndPoint* which in Azure is responsible for the exchange of information between web roles and work roles. *Protocol* element corresponds to *Protocol* which in Azure is the port used by the web roles. *ConfigurationData* element corresponds to *ServiceConfiguration* which in Azure stores configuration data of the web roles and work roles.

3.4 Transformations

Model-to-model transformations were defined through ATL [13], defining the object templates of source domains (SoaML, and SLA) to target domain (Cloud). Since SoaML, SLA, and Cloud metamodels refer to elements of the UML metamodel, UML elements are also included which are accessible from the rules. The defined rules allow ATL transformations to automatically generate the elements of the target Cloud model as defined in the mappings as well as the relationships between the generated elements in the resulting model.

Model-to-model transformations were made between SoaML, SLA to Cloud, and from Cloud to Azure, where all identified key elements of SoaML and key SLA elements corresponded to each key element identified in the Cloud metamodel. Model transformation extract is presented below:

```
-- @nsURI Cloud=http://Cloud.ecore
-- @nsURI SoaML=http://SoaML.ecore
-- @nsURI SLA=http://sla.ecore
```

```
module soaml2cloud;
create OUT: Cloud from IN: SoaML, IN1: SLA;
-- SoaMLModel To CloudApplication
-- In this part was mandatory to include SLA Elements.
rule SoaMLModelToCloudApplication {
from
sm: SoaML!SoaMLModel,
sl: SLA!SLAModel
to
ca: Cloud!CloudApplication (
clientDependency <- sm.clientDependency,
name <- sm.name.toString(),
templateParameter <- sm.templateParameter,
URI <- sm.URI,
visibility <- sm.visibility,
packagedElement <- sm.packagedElement,
packagedElement <- sl.packagedElement
)
}
```

This transformation rule allows a SoaMLModel element to be converted to a CloudApplication. It is also very important to note that the SLAModel element must be included to execute the transformation between SoaML, SLA, and Cloud. A complete list of transformation rules, can be found on: http://www.win.tue.nl/~mbottoto/resources/citt/transformations-rule.pdf.

Finally, model-to-text transformations were made from Azure metamodel to source code. It corresponds to C # programming language used in the Visual Studio tool.

4 Application Example

Our example was a business scenario (see Fig. 4), it is a community of independent distributors, manufacturers, and carriers who want to be able to work together consistently and not re-design business processes or systems when working with other parties of the community. They want to be able to have their business processes, rules, and information. The community has decided to define an architecture oriented to the service of the community to allow this open and agile business environment. This example has been defined to represent the SoaML specification made by the OMG [20].

4.1 Definition

The dealer network is defined as a "collaboration" community that involves three main roles for the participants in this community: dealer, manufacturer, and shipper. In the same way, participants participate in the three "B2B" services, a shopping service, a shipping service and a status service.

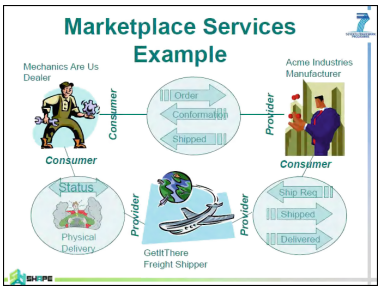

Fig. 4. Dealer network community [20].

4.2 Services

The service architecture does not define the service, and it uses the service specification. The service is defined independently and then released into the service architecture so that it can be used and reused.

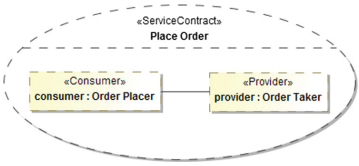

Fig. 5. Place order service [20].

The place order service is a simple service. The diagram in Fig. 5 diagram identifies the service contract, the terms, and conditions of the service, as well as defining the two roles: order placer and order taker.

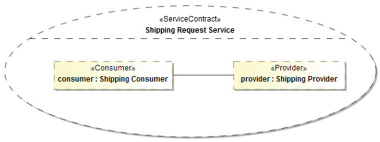

Fig. 6. Shipping request service [20].

The service request service a simple service. The diagram in Fig. 6 identifies the service contract, the terms, and conditions of the service, as well as defining the two roles: shipping consumer and shipping provider.

4.3 Procedure

In IDE Eclipse [24] - SOA2Cloud, SoaML model was created and saved with a format ".soaml". Besides, the model to specify the service agreements between the consumer and SLA service provider was also created in ".sla" format. This environment uses as input for transformations two models ".soaml" (Fig. 7) and ".sla" (Fig. 8), respectively.

Figure 7 depicts the DealersNetwork community modeled through SoaML specification. This model consists of a service architecture, three participants, three service contracts, three consumers, three providers, and three messages types. They interact with each other for forming a service community.

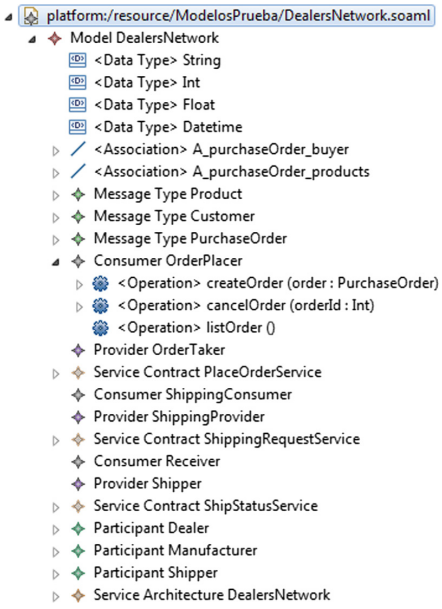

Fig. 7. Dealer network in SoaML.

The model in Fig. 8 represents service level agreements for DealersNetwork community. This model consists of an agreement, a document, a management, a service, a QoS, and a KPI.

Once the models have been created/modeled, model-to-model transformation rules were executed through ATL, and the result is a new model conforming to Cloud metamodel (Fig. 9).

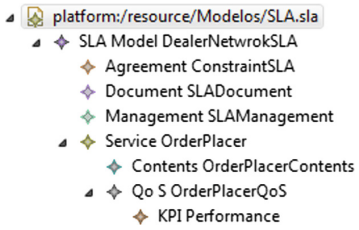

Fig. 8. Dealer network in SLA.

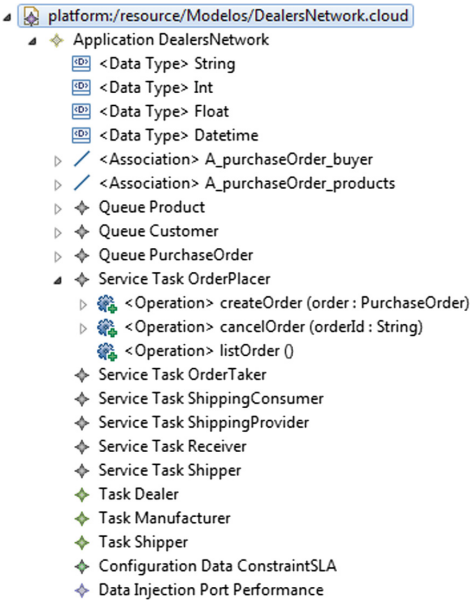

Fig. 9. Dealer network in Cloud.

After obtaining the Cloud application model, it was necessary to carry out a new model-to-model transformation, however, for this time, the Cloud model was needed as input to obtain the model for Azure (Fig. 10).

Finally, we performed a model-to-text transformation to generate the Windows Azure application, and it was implemented on Visual Studio for its compilation, testing and deployment/publication on Microsoft Windows Azure platform. To see the correct service functioning, we entered to the link provided by the service, which is: http://orderplacer.cloudapp.net/, and it showed the service published (Fig. 11).

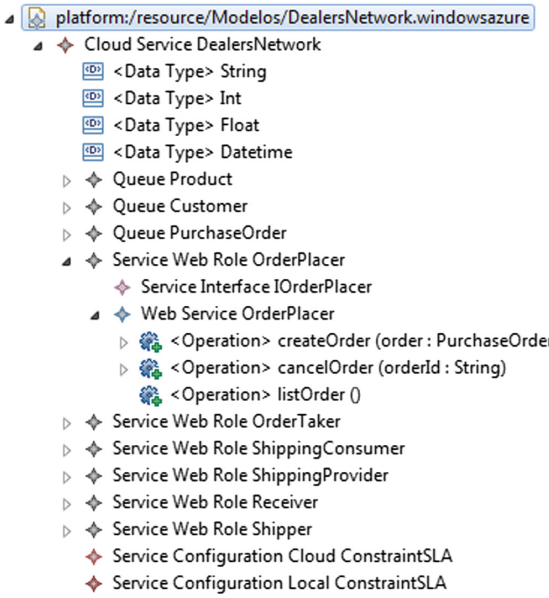

Fig. 10. Dealer network in Azure.

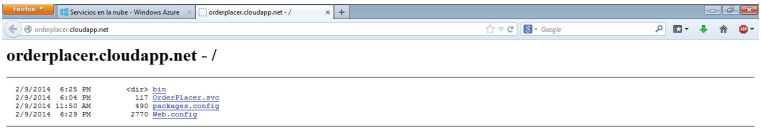

Fig. 11. Order place service.

5 Conclusions

The main motivation of this research is to have a framework to migrate SOA applications to Cloud, following a model-driven approach, since there are proposals that perform this process but do not cover aspects of development model-driven software.

To carry out this work we defined a generic process of model transformations (model-to-model and model-to-text), from SoaML and SLA models, to obtain a target model (Cloud) and its respective source code in Azure.

The final product has been taken into account, which it has been addressed through an application example, and its functionality and performance/deployment in a Cloud service provider have been viewed globally.

6 Future Work

According to our experiences, we can propose for future studies the following:

There is a need for a contribution for migrating SOA applications to other Cloud providers, such as Google App Engine and Amazon Web Services.

Another aspect needed to do is the refinement of the Cloud model by evaluating a wider set of artifacts based on all existing platforms.

Subsequently, an empirical validation of proposed framework will be carried out through controlled experiments, where the framework is evaluated objectively according to its effectiveness and efficiency; and subjectively to its ease of use and satisfaction.

Acknowledgments. This research was supported by the SENESCYT - Ecuador (scholarship program 2011).

References

1. Ali, N., Nellipaiappan, R., Chandran, R., Babar, M.A.: Model driven support for the Service Oriented Architecture modeling language. In: Proceedings of the 2nd International Workshop on Principles of Engineering Service-Oriented Systems - PESOS 2010, September 2009, p. 8. ACM Press, New York (2010). http://portal.acm.org/citation.cfm?doid=1808885.1808888
2. Babar, M.A., Chauhan, M.A.: A tale of migration to cloud computing for sharing experiences and observations. In: Proceedings of the 2nd International Workshop on Software Engineering for Cloud Computing, pp. 50–56. ACM (2011)
3. Barry, D.K.: Service-Oriented Architecture (SOA) Definition. http://www.service-architecture.com/articles/web-services/service-oriented_architecture_soa_definition.html
4. Beydeda, S., Book, M., Gruhn, V.: Model-Driven Software Development, vol. 1, 464 pages. Springer. Heidelberg (2005). https://doi.org/10.1007/3-540-28554-7_9
5. Botto, M., González-Huerta, J., Insfran, E.: Are model-driven techniques used as a means to migrate SOA applications to cloud computing? In: WEBIST (1), pp. 208–213 (2014). https://doi.org/10.5220/0004963802080213
6. Botto-Tobar, M., Insfrán, E.: Soa2cloud: Un marco de trabajo para la migración de aplicaciones soa a cloud siguiendo una aproximación dirigida por modelos (2014)
7. Botto-Tobar, M., Ramirez-Anormaliza, R., Cevallos-Torres, L.J., Cevallos-Ayon, E.: Migrating SOA applications to cloud: a systematic mapping study. In: Valencia-García, R., Lagos-Ortiz, K., Alcaraz-Mármol, G., Del Cioppo, J., Vera-Lucio, N., Bucaram-Leverone, M. (eds.) CITI 2017. CCIS, vol. 749, pp. 3–16. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-67283-0_1
8. Chappel, D.: Introducing the Windows Azure Platform (2010). http://www.davidchappell.com/writing/white_papers/Introducing_the_Windows_Azure_Platform,_v1.4--Chappell.pdf
9. Delgado, A., Ruiz, F., García-Rodríguez de Guzmán, I., Piattini, M.: MINERVA: Model drIveN and sErvice oRiented framework for the continuous business process improVement and relAted tools. In: Dan, A., Gittler, F., Toumani, F. (eds.) ICSOC/ServiceWave -2009. LNCS, vol. 6275, pp. 456–466. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-16132-2_43

10. Frey, S., Reich, C., Lüthje, C.: Key performance indicators for cloud computing SLAs. In: EMERGING 2013, The Fifth International Conference on Emerging Network Intelligence (2013). http://www.thinkmind.org/index.php?view=article&articleid=emerging_2013_3_30_40082
11. Guillén, J., Miranda, J., Murillo, J.M., Canal, C.: Developing migratable multi-cloud applications based on mde and adaptation techniques. In: Proceedings of the Second Nordic Symposium on Cloud Computing & Internet Technologies, pp. 30–37. ACM (2013)
12. Hamdaqa, M., Livogiannis, T., Tahvildari, L.: A reference model for developing cloud applications. In: Proceedings of the 1st International Conference on Cloud Computing and Services Science, pp. 98–103 (2011). http://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0003393800980103
13. Jouault, F., Allilaire, F., Bézivin, J., Kurtev, I.: ATL: a model transformation tool. Sci. Comput. Program. **72**(1–2), 31–39 (2008). http://linkinghub.elsevier.com/retrieve/pii/S0167642308000439
14. Microsoft: WebRole Schema. http://msdn.microsoft.com/en-us/library/windowsazure/gg557553.aspx
15. Microsoft: Windows Azure Service Configuration Schema (.cscfg File). http://msdn.microsoft.com/en-us/library/windowsazure/ee758710.aspx
16. Microsoft: Windows Azure Service Definition Schema (.csdef File). http://msdn.microsoft.com/en-us/library/windowsazure/ee758711.aspx
17. Microsoft: WorkerRole Schema. http://msdn.microsoft.com/en-us/library/windowsazure/gg557552.aspx
18. Mohagheghi, P., Sæther, T.: Software engineering challenges for migration to the service cloud paradigm: ongoing work in the remics project. In: 2011 IEEE World Congress on Services (SERVICES), pp. 507–514. IEEE (2011)
19. OMG: Software & Systems Process Engineering Meta-Model Specification, April 2008
20. OMG: Service oriented architecture Modeling Language (SoaML) Specification. Object Management Group (OMG), May 2012
21. Patel, P., Ranabahu, A.H., Sheth, A.P.: Service level agreement in cloud computing (2009)
22. Schmidt, D.C.: Guest editor's introduction: Model-driven engineering. Computer **39**(2), 25–31 (2006). https://doi.org/10.1109/MC.2006.58
23. SPIRENT: The Ins and Outs of Cloud Computing (2010). http://www.spirent.com/White-Papers/Broadband/PAB/Cloud_Computing_WhitePaper.aspx
24. The Eclipse Foundation: Eclipse Open Source Community. http://www.eclipse.org/
25. Tran, V., Keung, J., Liu, A., Fekete, A.: Application migration to cloud: a taxonomy of critical factors. In: Proceedings of the 2nd International Workshop on Software Engineering for Cloud Computing, pp. 22–28. ACM (2011)