Stakeholder Interactions to Support Service Creation in Cloud Computing

Lei Wang, Luís Ferreira Pires,
Andreas Wombacher, and Marten J. van Sinderen
Centre for Telematics and Information Technology
University of Twente
Enschede, the Netherlands
e-mail: {wangl, l.ferreirapires, a.wombacher,
m.j.vansinderen}@ewi.utwente.nl

Chihung Chi
School of Software
Tsinghua University
Beijing, China
e-mail: chichihung@mail.tsinghua.edu.cn

***Abstract*— Cloud computing is already a major trend in IT. Cloud services are being offered at application (software), platform and infrastructure levels. This paper presents our initial modeling efforts towards service creation at the infrastructure level. The purpose of these modeling efforts is to understand and reason about the service creation process. The paper presents a conceptual model represented as a UML class diagram, and identifies the interactions between service providers and infrastructure providers that are necessary in order to set up an execution environment and deploy services on top of infrastructure services. Although the models are far from complete, we are confident that they already give us some insight in the service creation process.**

Keywords-cloud computing; service creation; infrastructure services

I. INTRODUCTION

Cloud computing is already a major trend in IT [1] and is allowing computing to become the fifth utility, after water, electricity, gas and telephony [2, 3, 4]. According to the draft NIST definition [5], there are three service models in cloud computing, namely Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS). Cloud services are offered according to these models at different levels, namely at application (software), platform and infrastructure levels, respectively.

In this paper we claim that it is necessary to model the interactions between stakeholders in cloud computing in order to get insight in and reason about the service creation process. Our approach is to start with relatively simple models, and add aspects to these models in subsequent modeling steps. In this way we expect to be able to reason about issues like, for example, automated resource allocation, quality of service (QoS) monitoring and support, and resilience to failures. This paper reports on our initial modeling efforts.

The main distinctive features of cloud computing is that it allows rapid elasticity, making it straightforward for the service provider to dimension the resources necessary to support a service dynamically depending on the service demands. In this paper, we focus on the infrastructure level and we propose an initial model for the interactions between service provider and infrastructure provider, strongly based on the interactions necessary to deploy services on Amazon EC2 [6]. Examples of service providers are Alexa [7], which provides web information, and Salesforce, which provides services for developing business applications through their force.com platform [8].

This paper is further structured as follows. Section II presents a global architecture for services in the cloud. Section III identifies the concepts of the service creation process at the infrastructure level, and gives a conceptual model for services in the cloud based on these concepts. Section IV models the interactions between stakeholders. The resulting model helps understand and reason about service creation process. Section V briefly discusses some related work on modeling in cloud computing. Section VI gives our conclusions.

II. GLOBAL ARCHITECTURE

In order to define a global architecture, we generalize the currently available cloud computing services, especially those that offer hardware virtualization, like Amazon WS. Particularly Amazon S3 and EC2 are of interest for us.

Fig. 1 shows the global architecture we have assumed in this work. We structure this architecture in three layers. At the bottom layer we position the virtualized hardware elements provided by the infrastructure service provider. These elements offer computing capabilities (CPU and memory), and are instantiated with customized software (operating system, Database Management System and/or Application Server). Some infrastructure services, such as monitoring services, load balancing and security services, may also be offered at this layer.

The middle layer consists of the software packages that are developed by the service provider in order to implement services. The service provider creates, deploys and runs services, possibly by reusing services delivered by infrastructure provider. The services implemented by the service provider runs on the premises of the infrastructure provider. This layer delivers services to the end users.

Fig. 1 shows that three stakeholders have to interact in order to perform service provisioning, namely the end-user, service provider and infrastructure provider. In this paper we focus on the interactions between the service provider and infrastructure provider stakeholders, which are performed during the service creation process.

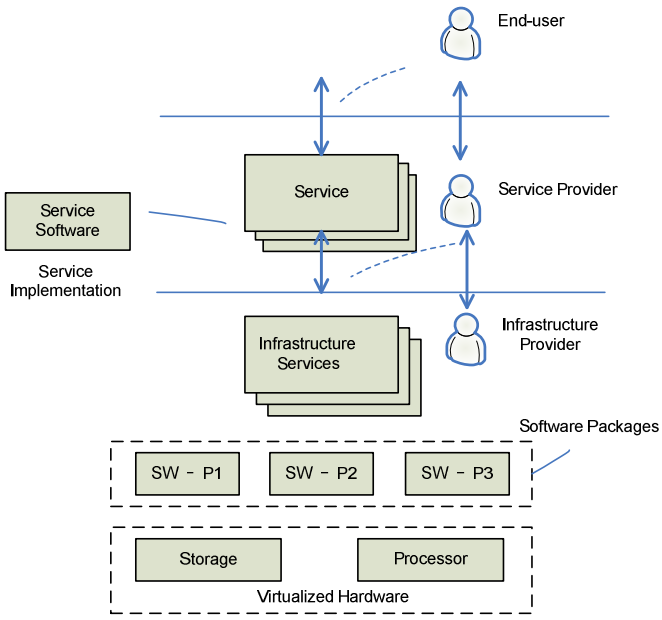

Figure 1. Global architecture for service creation in cloud computing.

III. CONCEPTUAL MODEL

In order to define a conceptual model for services in cloud computing at the infrastructure level, we start by identifying the main concepts of the service creation process. The conceptual model is then defined as a UML class diagram with the OCL constraints that represent these main concepts and their relations. The model presented here is not yet complete, as in this paper we just want to illustrate our modeling approach and identify the fundamental concepts for modeling service creation in the cloud. Fig. 2 depicts our initial conceptual model.

Fig. 2 shows the stakeholders *EndUser*, *Service-Provider* and *InfrastructureProvider*, stereotyped as actors. The *InfrastrucureProvider* provides virtualized resources for running the service, including *ServerInstance* and *Storage*, shown as associations between *InfrastructureProvider* and *ServerInstance* and *Storage*, respectively.

The *Service* class represents services provided by service providers, and has as attributes an *URL*, through which the service can be reached, and one or more documents, written in WSDL, OWL-S, WSML, etc., which describe the service. A *Service* is built on top of the *Storage* and *ServerInstance* objects to which the *ServiceProvider* has subscribed with the *InfrastructureProvider*.

A *ServerInstance* object represents a virtual server instance, and has a software configuration (*SWConfig*) and a hardware configuration (*HWConfig*). A *Storage* object represents some storage space. The *ServiceProvider* subscribes to these two classes of virtual resources from the *InfrastructureProvider* before deploying the service. Subscriptions to virtual resources are shown as associations between *ServiceProvider* and *ServerInstance* and *Storage*.

OCL constraints can be defined to make the conceptual model more precise. In our model, we defined that the *URL* attribute of a *Service* is related to the *URL* of its *Server-Instance* in the following way:

context: *Service*
def: *string : String*
inv: *self.URL = concat(*
self.ServerInstance.external_ip_address,
string)

The invariant above defines that the *URL* of a *Service* object is a concatenation of the *ExternalIPAddress* of the *ServerInstance* and some String value that locally identifies the service.

Another constraint defined as an OCL invariant is that the *ServiceProvider* of a *Service* is the owner of the *ServerInstance*, which is defined as follows:

context: *Service*
inv: *self.ServiceProvider =*
self.ServerInstance.ServiceProvider

IV. INTERACTIONS

The conceptual model given in Section III defines the relations between concepts that can exist during service creation and provisioning. However, the service creation process can only be properly understood if we model the interactions between service providers and infrastructure provider, indicating how these interactions affect the configurations (valid objects and links) of the conceptual model.

The service providers develop service software package, subscribe storage spaces and virtual server instances from infrastructure provider, and deploy the service implementation on the premises of the infrastructure provider. We identify below step by step the interactions between the *ServiceProvider* and the *InfrastructureProvider* that are necessary in order to create an atomic service, i.e., a service that is neither composite nor replicated.

1) Subscribe Storage

With this interaction a *ServiceProvider* subscribes to some *Storage* space from the *InfrastructureProvider*. Some possible parameters for this interaction are:

- *ServiceProvider*: service provider requesting storage;
- *InfrastructureProvider*: infrastructure provider offering storage;
- *Config*: configuration of the storage, with parameters like URL, size, access key and region;
- *Storage*: identifier of the storage for further reference.

Once this interaction is successfully performed, the instances of our conceptual model of Fig. 2 are affected so that a *Storage* object instance is created and associated with a *ServiceProvider* and an *InfrastructureProvider* object.

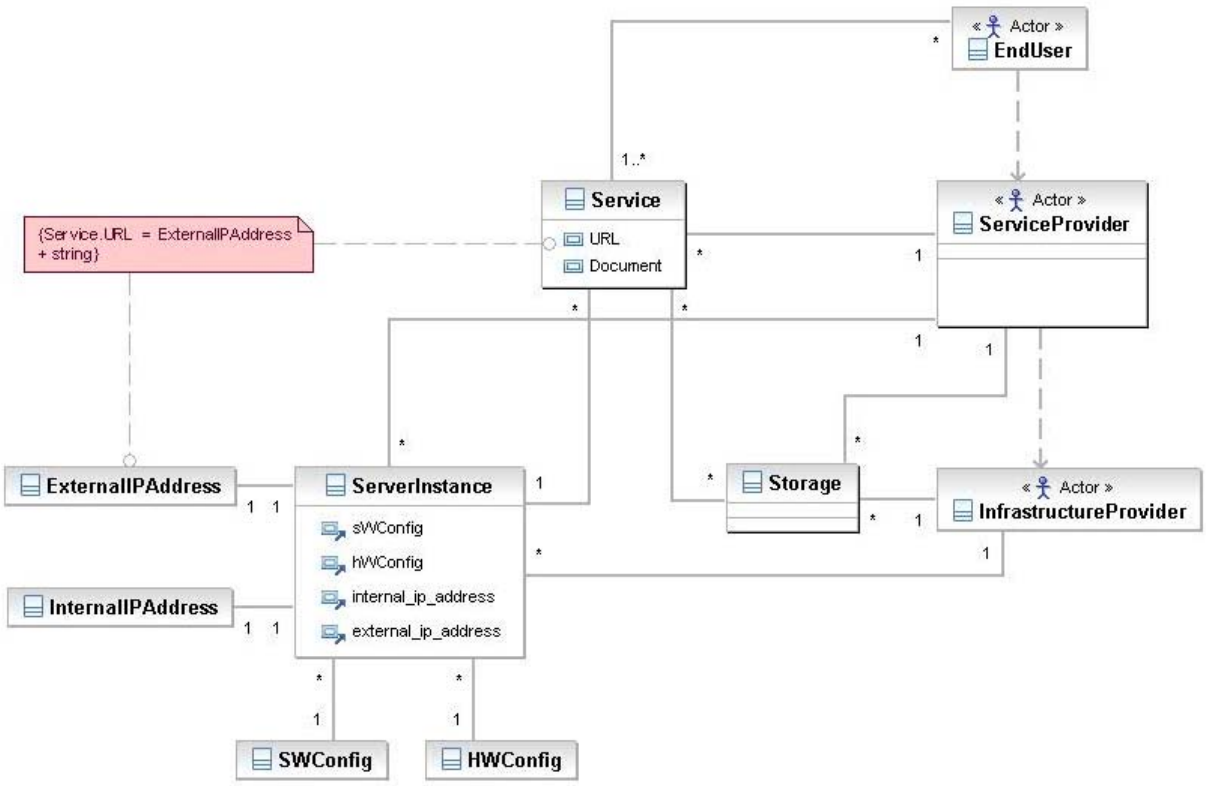

Figure 2. Conceptual model for service creation in the cloud (infrastructure level)

2) Upload Content

With this interaction a *ServiceProvider* uploads the service software package and corresponding application data to a storage to which it has subscribed before. Some possible parameters for this interaction are:

- *AccessKey*: the credentials used to access the storage space where the *Content* is expected to be uploaded;
- *Content*: the content to be uploaded, which represents the service software package and application data;
- *Storage*: the storage space subscribed from the *InfrastructureProvider* before.

Once this interaction is successfully performed, the instances of our conceptual model of Fig. 2 are affected so that the contents are associated with the storage (not explicitly represented in Fig. 2).

3) Subscribe Instance

With this interaction a *ServiceProvider* subscribes to *ServerInstance* from the *InfrastructureProvider*. After this interaction is successfully performed, all the building blocks needed for the creation of services are properly set up. Some possible parameters for this interaction are:

- *ServiceProvider*: service provider requesting a server instance;
- *InfrastructureProvider*: infrastructure provider offering server instances;
- *Config*: the software and hardware configuration for the *ServerInstance* being requested.
- *ServerInstance*: identifier of the server instance for further reference.

Once this interaction is successfully performed, the instances of our conceptual model of Fig. 2 are affected so that a *ServerInstance* object instance is created and associated with the *ServiceProvider* and the *InfrastructureProvider* objects. Furthermore, the values of the *SWConfig* and *HWConfig* attributes of the *ServerInstance* object are set according to the contents of the *Config* parameter. The *InfrastructureProvider* also assigns an *ExternalIPAddress* and an *InternalIPAddress* to the *ServerInstance. ExternalIPAddress* can be used to access the virtual instance from the Internet, while the *Internal-IPAddress* can be used for communication between *ServerInstances* in the same *InfrastructureProvider.*

4) Deploy

With this interaction a *ServiceProvider* transfers the *Content* (service software package and application data uploaded in interaction *UploadContent*) from a *Storage* to a *ServerInstance* and finishes the installation of a *Service.* Some possible parameters for this interaction are:

- *Storage*: the storage space subscribed from the *InfrastructureProvider* before;
- *ServerInstance*: the server instance subscribed from the *InfrastructureProvider* before;
- *Instructions*: instructions needed to transfer the *Content* from *Storage* to the *ServerInstance* and to install the *Service;*

- *Document*: documents that specify the *Service*, written in languages as, e.g., WSDL, OWL-S or WSML.

Once this interaction is successfully performed, a *Service* is created and can be referenced by its *URI*. The end-user can access the service specifications represented as *Document* in order to learn how and where to access the service.

V. Related Work

J. Varia [9] presents models of architecture for a service in the cloud environment provided by Amazon Web Services. This service allows a developer to do pattern-matching across millions of web documents. Based on that, some best practices for using each Amazon Web Service Amazon S3, Amazon SQS, Amazon SimpleDB and Amazon EC2 to build an industrial-strength scalable application are discussed. Their approach differs from ours, because their model of the application is strongly dependent of the Amazon Infrastructure Provider, while we built our model by generalizing the infrastructure level, i.e., making the model independent of any specific infrastructure provider. Furthermore, an already existing service is modeled in [6], while we define a model that describes how services are created in the cloud.

T. Harmer et al. [10] give an outline of a usage model common to all cloud providers. Based on this model they define an abstract layer that is a generalization of the various infrastructure providers, and give examples of service development in a way that is neutral with respect to the provider. In our model, we define typical interactions and concepts for the whole service creation process. Their approach differs from ours in that their models are defined as snippets of programming code, while we provide an abstract model in terms of a UML class diagram and a set of interactions. Furthermore, they mainly focus on the task of subscribing the virtual machine instances and displaying their IP address.

R. Dodda et al. [11] present models of a cloud computing architecture to implement the cross-cloud system management. They introduce the concept of compute instance and its supported operations at the level of the infrastructure provider. After that, they present an implementation and empirical results of this implementation. Their approach differs from ours because their models concentrate on the virtual compute resource subscription phase while we aim at modeling the whole service creation process.

VI. Conclusion

This paper identifies the main concepts for the service creation process in cloud computing environment at infrastructure level, and identifies and models the interactions between stakeholders to support service creation. We are aware that these models are far from complete, but we are confident that we captured the most basic interactions and represented their effect on the instances of the concepts of the conceptual model. In this way the conceptual model gives semantics to the interactions, in that it explains the effects of the interactions.

We are currently planning to extend our models to represent load balancing functionality, security (authentication and authorization) and QoS support. With this work we will be able to explain, for instance, how infrastructure services should be published and how services can be composed. Another extension of this work is to model services offered at the other levels (application and platform levels), and use the resulting models to reason about service compositions across the different cloud service models. We expect that our models will facilitate the service creation process, for example, by automating these interactions, so that service creation and composition can be automated. Once the models are extended to incorporate QoS issues, we should be able to describe, monitor, reason about and control the QoS of the services being offered in a cloud environment.

References

[1] I. Sriram and A. Khajeh-Hosseini, "Research agenda in cloud technologies," unpublished, 2010, arXiv:1001.3259v1

[2] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility," Future Gener. Comput. Syst., vol. 25, pp. 599-616, Jun. 2009, doi:10.1016/j.future.2008.12.001.

[3] R. Buyya, C. S. Yeo, and S. Venugopal, "Market oriented cloud computing: vision, hype, and reality for delivering IT services as computing utilities," Proc. 10th IEEE Int. Conf. on High Performance Computing and Communications (HPCC 2008), IEEE CS Press, 2008, pp. 5-13, doi:10.1109/HPCC.2008.172.

[4] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia. "Above the clouds: a Berkeley view of cloud computing," unpublished, Feb. 2009.

[5] P. Mell, and T. Grance, "Draft NIST working definition of cloud computing," 2009.

[6] Amazon Web Services home page: http://aws.amazon.com/.

[7] Alexa home page: http://www.alexa.com/

[8] Force.com home page: http://www.salesforce.com/platform/

[9] J. Varia. "Cloud architectures," unpublished, Amazon Webservices, 2008.

[10] T. Harmer, P. Wright, C. Cunningham, and R. Perrott. "Provider-independent use of the cloud," Proc. 15th Int. Euro-Par Conference on Parallel Processing, LNCS, vol. 5704, pp. 454-465, doi:10.1007/978-3-642-03869-3_44.

[11] R. Dodda, C. Smith, and A. Moorsel, "An architecture for cross-cloud system management," Proc. Second Int. Conf Contemporary Computing (IC3 2009), Springer, pp. 556-567, doi: 10.1007/978-3-642-03547-0_53