Towards Exploiting the Full Adaptation Potential of Cloud Applications

Clarissa Cassales Marquezan, Florian Wessling, Andreas Metzger, Klaus Pohl
paluno, University of Duisburg-Essen, Germany
{clarissa.marquezan, florian.wessling, andreas.metzger, klaus.pohl}@paluno.uni-due.de

Chris Woods
Intel, Dublin, Ireland
chris.woods@intel.com

Karl Wallbom
Cloudmore, Stockholm, Sweden
karl.wallbom@cloudmore.com

ABSTRACT

Current technology for cloud application adaptation fails to capture two fundamental aspect of cloud environments: multiple adaptation options and interferences and dependencies among these multiple mechanisms. Addressing these aspects requires a significant extension of existing cloud tools and frameworks for engineering and executing cloud application adaptations. They should explicitly take into account: all entities of the cloud environment relevant for adaptation decisions; the concrete adaptation actions that these cloud entities may perform; and the mutual dependencies between those entities and actions. In this paper we provide the first insights towards such novel technology. As main contribution, we systematically elicit the key entities related to adaptations inside a cloud environment and explicitly document those in a conceptual model. To build this model we surveyed the literature, discussed with industrial partners with experience in cloud computing, and analyzed commercial solutions. We also provide a case study based on Amazon Web Services solutions, to show how our conceptual model can be instantiated and help developers to identify possible cloud application adaptation strategies.

Categories and Subject Descriptors

D.2.7 [Software Engineering]: Distribution, Maintenance, and Enhancement—Documentation; H.3.4 [Information Systems]: Systems and Software—Distributed systems; K.6.4 [Management of Computing and Information Systems]: System Management—Quality assurance

General Terms

Management, Documentation, Performance, Reliability

Keywords

Adaptation, cloud, automatization, model, layer, terminolo-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.

Copyright is held by the author/owner(s). Publication rights licensed to ACM.
PESOS’14, May 31, 2014, Hyderabad, India
ACM 978-1-4503-2841-8/14/05
http://dx.doi.org/10.1145/2593793.2593799

1. INTRODUCTION

Cloud computing is increasingly adopted by the industry to deliver, acquire and use IT infrastructures, platforms and software applications on demand. In the next two years, cloud computing is predicted to represent a major IT expenditure [29]. One of the main characteristics of cloud computing is elasticity [41, 24]. Elasticity refers to the ability of a cloud environment to scale up and down its resources and capabilities in order to respond to changing loads and demands. Elasticity is the “de facto” mechanism that is used for adaptation of cloud applications.

The cloud commercial solutions expose mechanisms for triggering scaling actions. For instance, Amazon Web Services [7] expose APIs to developers while Windows Azure [60] exposes a portal where owners of cloud services can manually configure the scaling rules. Research solutions have focused on delivering advanced adaptation mechanisms, which are variations of elasticity. Those mechanisms fall into two categories. First, mechanisms that allow to increase or decrease the amount of virtual resources provided to an application. They include: auto-scaling, i.e. policies that are defined to increase or decrease the resources [15, 47]; SLA-based scaling, i.e. the triggering or changing the amount of resources that is controlled by established SLAs between cloud consumer and provider [65, 56, 13]; cloud bursting, i.e. the up- and down-scaling of resources between private and public clouds [38]. Second, mechanisms that allow to reconfigure the amount of system capabilities (e.g., increase the amount of replicas from a Web servers to handle the number of requests) of applications inside the virtual machines of a single cloud or among multiple clouds [34, 6, 53]. However, the aforementioned adaptation mechanisms partially address two fundamental concerns in cloud environments:

• Multiple adaptation mechanisms: A cloud environment is a complex environment made up of many different entities and layers [41]. Those entities include, to name only a few, physical resources, virtual resources deployed on top of physical resources, Web or application servers deployed inside virtual machines, as well as software code running inside a Web or application server. In principle, all these entities may be modified during runtime and thus may offer dedicated adaptation mechanisms. This means that the range of options that can be used for engineering cloud application adaptations is much broader than just elasticity.
Interferences among adaptation actions on different cloud layers: Currently, cloud applications and other cloud entities from different layers, such as physical resource managers [40, 48] are taking adaptation decisions and actions simultaneously and independently from each other. Studies have shown that there are many complex dependencies between the entities in a cloud environment. For instance, Zhang et al. [63] provide arguments that the decisions taken in the infrastructure level (i.e., the physical machines) impact the virtualization level (i.e., the virtual machines). Lloyd et al. [42] discuss the performance variations on the cloud application when choosing the model that this application will be deployed (in one or multiple VMs) and the relationship to auto-scaling. Sedaghat et al. [55] and Ferraris et al. [25] discuss the many trade-offs, dependencies, and problems on designing auto-scaling solutions. This means that various adaptation mechanisms applied simultaneously generate consequences that might negatively affect the cloud application quality and also the proper cloud infrastructure performance. If these consequences and dependencies were to be better understood and coordinated, these negative effects could be reduced.

Addressing the above listed concerns requires a significant extension of existing cloud tools and frameworks for engineering and executing cloud application adaptation. Specifically, those tools and frameworks need to explicitly take into account (i) all entities of the cloud environment that are relevant for adaptation decisions; (ii) the concrete adaptation mechanisms and adaptation actions that these cloud entities may perform; (iii) the mutual dependencies between those entities and actions such as to understand potential synergies and interferences between adaptations. We envision that such novel tools and frameworks will provide helpful support for cloud application developers, allowing them, for instance, to take informed decisions on which kind of adaptation mechanisms they exploit for their application. In addition, we foresee that those tools and frameworks will include powerful adaptation engines that are able to automatically negotiate among each other the most appropriate adaptation strategy based on the cloud environment status.

In this paper we pave the way towards such novel cloud application adaptation tools and frameworks for Web Applications. As main contribution, we systematically elicit the key entities related to automatic adaptations inside a cloud environment and explicitly document those in a conceptual model. The conceptual model presented in this paper is structured into four main layers: physical, virtualization, logical application architecture, and application business logic. These layers enclose different entities with capabilities to enact adaptations. For each of these entities we define those that can trigger changes and those that may be affected by changes. In addition, the conceptual model also captures the relationships among these entities that can provoke changes or that can be changed. To build this model we surveyed the literature, discussed with industrial partners with experience in cloud computing, and analyzed commercial solutions. As further contribution, we provide a case study based on the Amazon Web Service solutions exemplifying how cloud application developers can use our conceptual model to identify the available adaptation for their specific type of application.

The remainder of this paper is organized as follows. Section 2 describes the analysis of the current research and commercial solutions related to cloud adaptation technology. Section 3 introduces the conceptual model proposed in this paper, where the entities and their relationships are described. Section 4 depicts the case study. The related work is discussed in Section 5. Finally, in Section 6 we discuss the conclusions and future work.
new CHs (scaling up) or terminate such CHs (scaling down).

He et al. [34] also provide a framework for the deployment, migration and elasticity of applications based on Elastic Application Containers (EACs). The EAC is an abstraction inside a VM where applications are deployed. These application are then controlled in the EAC which can scale the application up and down (replicating the application in another EAC or shutting it down), load balancing of incoming requests for the applications (HTTP messages), and migrating the application among the EACs.

Ashraf et al. [6] also consider an abstraction level between the application and the VMs, called application server, which is the entity deployed inside the VM. Different scaling actions can be triggered in their proposal and the application server itself can be scaled. This means that applications can be migrated to other application servers in order to scale down the number of application servers (i.e., VMs) or in order to scale up the number of application servers to solve saturated situations. Applications themselves can also be scaled. In this case, new instances of saturated web applications can be deployed in other application servers or shutdown.

In addition to the technology proposed by the research community, well known commercial solutions also provide mechanisms for cloud application adaptation. For example, Google Compute Engine [31] allows developers to specify rules for load balancing of incoming traffic of the running applications. Windows Azure [60] offers a portal where owners of the applications deployed on their cloud can configure the auto-scaling policies for increasing and decreasing virtual resources associated with such application. Amazon Web Services offer multiple solutions to enable the adaptation of cloud applications. For instance, developers can make use of the EC2, Auto Scaling, and Load Balancing APIs from AWS to program and customize how the auto-scaling and load balancing mechanism for their application will work [8, 7]. In essence, the most known commercial solutions provide the basic functionalities of controlling the incoming traffic of the application and changing the amount of virtual resources associated with the application.

### 2.2 Cloud Resource Adaptation

This section describes solutions that address adaptation related to the cloud resources. The studies described in Section 2.2.1 focus on adapting virtual resources of the cloud environment to accommodate application needs. Section 2.2.2, instead, is devoted to describing adaptation of cloud resources motivated by internal matters of the cloud environment and not by explicit application needs.

#### 2.2.1 Auto Scaling

Scaling cloud resources (e.g., VMs) based on SLAs has the primary goal of avoiding violation on the SLOs (Service Level Objectives) agreed between the cloud consumer and cloud provider. Several research projects have addressed adaptation driven by SLA. For instance, the research project SLA@SOI [1] addresses two aspects of adaptation driven by SLA negotiation: transparent SLA management, and automated SLA negotiation. The developed protocols and engines provide mechanisms for adapting the cloud environment in response to SLA violations. The SLA negotiation framework designed in that project considers multiple, diverse stakeholders (infrastructure, platform, software providers and service consumers). The work done in the RESERVOIR project [54] regarding adaptation provides dynamic deployment and relocation of virtual environments onto the underlying physical resources, based on quality of service requirements from an SLA [28]. The solutions provided by RESERVOIR focus on scaling the resources offered in a IaaS delivery model, which means the SLA is associated with the requirements of the virtual environment (e.g., virtual machines). In the context of adaptation, the OPTIMIS [52] project delivers SLA management between the IaaS and PaaS. This means that it is the responsibility of the IaaS to meet the SLA and provide the elasticity of the virtual infrastructure for a consumer of virtual machines and links [55]. Once again, the adaptation is performed in order to change the number and placement of virtual machines, in order to avoid SLA violations in the PaaS. In addition to the technology provided by large projects, the literature also shows multiple proposals for cloud resource adaptation driven by SLA management [65, 45, 39, 50]. The QoS requirements addressed by those solutions are typically related to application performance. The principle of how they operate is the same as they try to scale the resources used from the cloud provider in order to avoid an SLA violation.

Analyzing the literature, it is also possible to find other solutions that are only focussing on the mechanisms for enabling the automatic scaling of cloud resources. For instance, Yazdanov and Fetzer [62] developed a specific solution for vertical scaling of resources. The solution provided by the authors builds on top of the Xen Hypervisor [61] to provide a finer grain of control of the physical resources associated to the VM. Han et al. [33] also focus mainly on better decision algorithms for scaling resources of the VMs inside a physical machine (i.e., vertical scaling). Though, they are also able to handle the scaling process in the number of VMs for the same owner of these VMs (i.e., horizontal scaling). Mao et al. [43] designed a specific algorithm for autoscaling based on job deadline. In this case the application is composed of these jobs and their execution time is assumed to be quantifiable up front. The information about these jobs, like execution time, is part of the decision making of when and which resources need to be scaled. Recently, several solutions have been using prediction techniques to better support the scaling decision process. For instance, Bankole and Ajila [11] use machine learning for prediction of resources in multi-tier web applications. Jiang et al. [37] combined algorithms for prediction and modeled the dynamic provisioning of VMs as a time series problem, where provisioning and de-provisioning actions are equivalent to horizontal scaling.

#### 2.2.2 Internal Issues of Cloud Infrastructure

A cloud infrastructure typically receives requests to accept new virtual resources. This process is actually called virtual network embedding [26]. During the process of accepting a new request for virtual resources the infrastructure provider can trigger reorganizations of the current virtual resources in order to be able to accept the new request [18, 14]. Another group of works focuses on resource consolidation on the cloud infrastructure but observe the requirements of the SLAs from the consumers of the virtualized resources [35, 45, 36]. This means that during the consolidation process, e.g., shutdown some physical machines to reduce operational costs like energy consumption [32, 64, 48], the virtual resources are reconfigured with VM migration as the typical mechanism. There are also solutions targeting load balancing in the cloud.
infrastructure. They can also generate adaptation in the virtualized resources [51] and generate, e.g., the migration of VMs and reconfiguration of the virtual links [44, 30]. In summary, many of the management and adaptation actions performed in the cloud infrastructure affect the resources hosted by such infrastructure.

3. CONCEPTUAL MODEL

This section presents the conceptual model for representing the entities and relationships in a cloud environment that can request or cause adaptations and the ones that can be changed in a cloud environment. This conceptual model assumes that the cloud applications deployed in this environment will be Web applications. The process of building this model included: (i) the analysis of the current technologies for cloud computing that are related to adaptation (as we describe in Section 2), (ii) discussions with industrial partners with experience in cloud computing in the context of the CloudWave Project [4], and (iii) analysis of current models for representing entities in a cloud environment (as detailed in Section 5). In Figure 1, we provide an overview on the layers composing the cloud environment: physical, virtualization, logical application architecture, and application business logic.

Figure 1: Overview of cloud environment layers

The definition of these layers helps to structure the cloud entities and their adaptation capabilities, regardless of the delivery model associated with the services being provided and consumed in a cloud environment, i.e., SaaS, PaaS, IaaS. Indeed, the key aspect that we consider in our work is that whenever a Web application is executing inside this environment there will be entities and relationships in different layers of the cloud environment that can change and affect each other, independently of who is selling or buying cloud services. Futhermore, in order to define the proposed conceptual model we assumed that: (i) the cloud environment is based on hypervisor virtualization, (ii) the virtualization of network devices is not part of the model, and (iii) we consider that the “management” term in the entities detailed in Figure 2 includes monitoring and configuration capabilities. In addition, we assume that ownership of resources in the cloud environment does not affect the visibility of information in our model. This assumption is based on the fact that there are recent initiatives providing Monitoring as a Service [46] (e.g., Amazon CloudWatch [5]) and Management as a Service [17] for cloud environments. Thus, for entities that are not totally visible we can make use of these services to achieve the visibility of their monitoring and adaptation capabilities.

As illustrated in Figure 1, the Physical Layer is in the base of this conceptual model. In a datacenter, the Physical Layer consists of several entities, such as physical machines, network devices and the necessary management tools. The Virtualization Layer includes all the entities related to the virtualized resources and the management entities controlling these resources. As previously mentioned, our model is able to represent a cloud environment that will host Web applications, which means that the application needs servers to host their components. In our model, the Logical Application Architecture Layer encloses the set of software components and management entities necessary to support the logical architecture of a Web Applications, i.e., the entities that will host the business logic of the application. The Application Business Logic Layer encloses the entities directly related to the business logic implementation.

The entities of these layers are related to each other. For example, the physical machines belonging to the Physical Layer are running multiple virtual machines (e.g., using OpenStack [58]), which are part of the Virtualization Layer. Each of these virtual machines can run different servers such as Web servers, database servers, application servers, etc. A server like Apache Tomcat [57] is classified as an entity of the Logical Application Architecture Layer. It is able to host and run different servlets which is an example of an entity of the Application Business Logic Layer. In addition, these four layers are interrelated with relations, i.e., the influences relationship in Figure 1, which indicates that adaptation decisions taken in one layer can propagate to entities of other layers or inside the same layer. This relation just models the possibility of an influence, it describes neither the request of the exact change nor states whether a change itself will really happen. The detailed conceptual model proposed in this paper with the cloud entities and their relationships is illustrated in Figure 2 and described in the following sections. It contains several placeholders (denoted by “...”) to underline that the elements are not restricted to the given examples and can be extended.

3.1 Physical Layer

The entities in the Physical Layer refers to physical equipment inside of one or multiple datacenters. In our model, the geographical boundaries are not taken into account, which means that the entities in this layer could be located in different countries and different datacenters. The central parts of this layer are the Physical Resource and Physical Resource Management entities. Physical Resource refers to Physical Machines, Storage and the Network Devices that connects them. Physical Machines are composed by Memory, Disk, CPU, and NIC (Network Interface Card). The Network Device entity can be represented by Routers or Switches. The Physical Resources can host several Virtual Resources, since this is one of the most basic concepts in cloud computing. The hosts relationship between the Physical Resources and Virtual Resources entities creates another entity called Physical vs. Virtualization Mapping entity. This entity rep-
Figure 2: Entities and relationships associated with runtime adaptations in the cloud environment
resresents the mapping between Physical Resources and Virtual Resources and can be used as input during the adapting decisions. The Physical Resource Management is responsible for controlling and adaptation actions inside the Physical Layer. As discussed in Section 2.2, there are adaptation actions executed in the cloud infrastructure, i.e., in the Physical Layer, that lead to changes in the rest of the cloud environment. In our detailed model, we reflect this in the Virtual Resource Embedding and Load Balancing entities, which are examples of types of Physical Resource Management found in the current literature. The results of the decision process from these types typically result in changes in the Physical Resource entities, as represented in Figure 2.

3.2 Virtualization Layer

There are two main entities in the Virtualization Layer: the Virtual Resource and Virtual Resource Management. As illustrated in the conceptual model, Virtual Machine, Virtual Storage and Virtual Link are types of Virtual Resource entities, with the first two connected by Virtual Link entities. The Virtual Machine entity is composed of Virtual Disk, Virtual Memory, Virtual CPU, and Virtual NIC. The Virtual Resource Management is the entity responsible for managing the Virtual Resources. This means that decisions made by the Virtual Resource Management changes the Virtual Resources. As discussed in Section 2.2, there are different mechanisms for dealing with cloud resources that create changes, i.e., adapting the Virtual Resources. In our model these mechanisms are: Auto-Scaling, Resource Consolidation, and Load Balancing. Changes in the Virtual Resources might generate other changes in the Physical Resources, as an indirect effect. This has been actually evidenced by Zhang et al. [63] as discussed in Section 1. In addition, adaptation actions decided and enforced by the Physical Resource Manager also change the Virtual Resources.

We identified in the literature the examples of how embedding new virtual resources can cause relocation of VMs, and this leads to changes in the status of the virtual resources [44, 30]. There is also another entity in the Virtualization Layer that can be changed at runtime: the Virtualization vs Logical Application Mapping. This entity reflects the host relationship between Virtual Machines and Guest OS, which will be further discussed as follows.

3.3 Logical Application Architecture Layer

The major entities in this layer are indeed reflecting the logical architecture of a Web application. The Server is the main entity in this layer. In a Web Application, many different types of servers can be used depending on the logical architecture of the application. In the conceptual model, examples of these servers are: Web Server, Application Server, Process Engines, Databases, etc. However, the Server entity does not exist alone and it is indeed part of the Guest OS which is hosted inside a Virtual Machine. In addition, OS Management entities can also be part of the Guest OS and they are responsible, for instance, for managing the processes inside the Guest OS. Another entity in this layer is the Server Management, which is responsible for configurations, monitoring and it can provoke changes in the Server entity. For example, Load Balancing and (Re-)Deployment are types of Server Management entities that we identified currently being used in cloud application adaptation (as discussed in Section 2.1). As represented in Figure 2, Server Management entities can also influence the Virtual Resource Management. It can request changes in the Virtual Resources in order to accommodate changes in the performance of the Server entity, as some proposals found in the literature indicate [53, 34]. The Logical Application Architecture layer has also an entity denoting a mapping which is the Logical Architecture vs Business Logic Mapping.

As it will be further detailed in the next section, the Server entity hosts entities from the Application Business Logic layer, in this case the Container entity. The mapping between the entities of both layers represents the Application Business Logic layer which can be changed whenever the Server Management redeploys such entities, as suggested, for instance by He et al. [34].

3.4 Application Business Logic Layer

In this layer, the entities represent the basic elements associated to Web application business logic used for adaptation at runtime. A Container is the entity in which implemented business logic is grouped together to be deployed in a Server. For example, it can be a “war” file. The Container can host several Core Logic Implementation entities, e.g., the “.class” files associated with the Java programming language. The Core Logic Implementation entities can be created based on a different paradigms such as Object-Oriented Programming (OOP), Aspect-Oriented Programming (AOP), Service-Oriented Architecture (SOA), etc. The relation between the Container and the Core Logic Implementation can be described by a Configuration entity, which will be necessary for the deployment and eventually during the execution of the Web application. Although current cloud application adaptation technology does focus so much on it, Core Logic Implementation can be endowed with self-adapting capabilities [49, 20, 59]. We express this capabilities in the Self-adaptation Management entity, which changes in the Core Logic Implementation. Examples of self-adapting capabilities are: Dynamic Weaving, Re-binding, Mismatch Resolution and Reflection. For example, a Web Application based on the SOA paradigm can use Core Logic Implementation, in this case a BPEL specification, and Re-binding in order to substitute Web services that fail to provide the expected quality of service (e.g., the response time is too high) [16].

Even not being yet considered by most of the proposals on cloud application adaptation, self-adapting mechanisms can create several indirect changes in the cloud environment. Consider the example mentioned above, by changing the Web service to be invoked, the entities in the Application Business Logic layer could impact the load of the Server hosting the Core Logic Implementation. In turn, the Server could be forced to request more Virtual Resources, which can lead the Physical Resource Management to change the configuration of the Physical Resources. This last change, can impact the Virtual Resources hosted inside the changed Physical Resources. All these sequence of effects and changes among the cloud entities might lead to a recursive chain of adaptations caused by the relationships among these entities as we show in the conceptual model. Thus, the current developed technology fails to provide better mechanisms to visualize which kind of adaptations could happen to a cloud application and how is it possible to handle the existence of recursive chains of adaptations.
4. CASE STUDY

This section shows how the model proposed in Section 3 can be applied to describe the entities and relationships associated with runtime adaptations in the cloud environment. Our case study is based on a 3-Tier Amazon Web Application to exemplify how their services (i.e., Amazon Web Services – AWS) can be used for deploying and running this type of application [9] in their cloud environment. In Figure 3, we present a version of the Amazon example [9] depicting the services that are related to the Web Application architecture and the cloud resources used by the application. Extra services from the original example, such as Amazon SNS Notifications and Amazon CloudWatch Alarms, that serve as sources of information for the application but are not really part of the architecture of the application are not part of this case study.

Our goal with this case study is to show that using our conceptual model as reference, developers and cloud application operators can identify which are the cloud entities associated with their application that can provoke or suffer adaptation actions, and which kind of direct and indirect influences these entities might cause in their application. Having this information is vital to determine which are best adaptation options that they might want to use for their applications. The result of the mapping process between the application in Figure 3 and our conceptual model in Figure 2 is illustrated in the conceptual model of Figure 4.

The Amazon Web application example consists of an Elastic Load Balancer that distributes the incoming traffic on multiple Amazon EC2 Instances. Each instance contains Web servers. Application servers (App Server) and a database service which is connected to an Amazon S3 Bucket for data storage. Some of the entities from the Amazon example in Figure 3 have a direct counterpart in our conceptual model. For example, the Web Server in Figure 3 is directly mapped to Web Server in Figure 4. The same applies for the mapping between App Server and Application Server and between Amazon S3 Bucket and Virtual Storage. Amazon RDS is a scalable database with load balancing implemented by the Elastic Cache Tier and therefore these elements are mapped to the Database entity in our model and to Load Balancing as one form of Server Management.

In some cases the mapping is not explicit and requires a more detailed analysis. For instance, AWS offers the Auto Scaling Service enabling applications to request a change of their virtualized resources. Figure 3 only indicates the existence of an Auto Scaling Group. This indicates that this application is using AWS Auto Scaling Service. This mapping is made explicit in Figure 4 with the relationships between the Server Management and Virtual Resource Management entities and respective relationships. It is also possible with AWS to specify the geographical location where EC2 instances should be placed. The possibility of using this kind of attributes during the process of changing the relationship between the virtual and physical resources for this application is reflected by the relationship between the Physical Resource Management and the Virtual Resources. In this case, the developers and application operators can also give directives that influence the adaptation relationships among the Physical Resource Manager, Physical Resources, Virtual Resource Manager, and Virtual Resources. With our model we are able to help them to identify these relationships which will allow the developers and application operators to make better informed decisions about the adaptation strategies they will use in their applications.

Figure 3: Example of 3-Tier Web Application using Amazon Web Services [9]

Figure 4: Mapping the entities and relationships related to adaptation between the 3-Tier Web Application using Amazon Web Services and the proposed conceptual model
5. RELATED WORK

The NIST (National Institute of Standards and Technology) standard [41] defines three hierarchical layers: Operating System Layer (positioned at the bottom of the hierarchy), Middleware Layer (at the middle of the hierarchy), and Application Layers (at top of the hierarchy). The goal on defining these layers is to express the scope of control between provider and consumers of cloud services. For example, if a cloud provider offers a SaaS service this means that it has the control over the three layers, while the cloud consumer would only have the control over the application layer. Refinements of these scopes of control have been also provided in another NIST reference document [10]. The software stack involved in each of the scopes of control is described in more detail. However, it is not yet clear which entities exactly are associated with this software stack and no special focus is placed on identifying the changes that can happen in the entities of these scopes of control.

The DMTF (Distributed Management Task Force) defined a series of standards that specify the profiles, interfaces and models for managing cloud infrastructures [22, 23]. These documents provide a very detailed description of many entities inside the physical and virtual layers. For instance, they define UML classes for describing the elements of a virtual system configuration [22], or a detailed model of the basic resources associated with IaaS providers, such as machines, network and storage [23]. Nevertheless, it is not the major goal of these specifications and models to focus on how to identify adaptations in a cloud environment.

Recently OASIS (Organization for the Advancement of Structured Information Standards) defined a standard for specifying cloud applications topology and deployment dependencies. The major advancement this specification brings is the possibility of explicitly expressing how components of an application (e.g., in the sense of Web and database servers) are dependent and how they should be deployed in a cloud infrastructure. Despite of the major contribution introduced by this standard it does not capture the changes and dependencies when cloud applications are running, it defines these relationships only up to the deployment phase.

In addition to the initiatives from standardization bodies, the research community has also proposed ways of representing the entities in a cloud environment. For example, Basmadjian et al. [12] provided a detailed model with the goal to identify entities in the cloud environment that influence the energy consumption. Franceschelli et al. [27] describe the models used in MODAClouds project. The major goal of their model is to evaluate the performance and costs of cloud applications but it is not able to explicitly represent the relationship among the entities associated with runtime adaptations. Chapman et al. [19] define a model for representing elasticity between virtual resources and physical resources. They describe a model for specifying the KPIs from the cloud application point of view. This model is tied together with models representing another model representing the elasticity rules. In general the models provided by the authors are very thoroughly defined but are very strict only to represent the entities and relationship related to scaling up and down the virtual resources associated with the cloud application. There are also other models representing the entities associated with the deployment of an application in a cloud environment, like PCIM4Cloud [2] and MODACloudML [3]. Although this is not their explicit goal, these models can be used to identify adaptation entities not only related with auto-scaling. In summary, there is a lot of work done both in describing in very details the entities inside the cloud infrastructure and on the other hand the deployment needs of cloud applications.

Different from the work listed above, our model focuses on representing the relationship of these two perspectives when it comes to determining which entities are triggering changes and which are affected by these changes. As a future work, we intend to combine our work with the already detailed models adding the relationships that are missing for representing the options of adaptations emerging from a cloud environment.

6. CONCLUSION

In this paper we presented a conceptual model to represent the entities and relationships inside the cloud environment that are related to adaptation. The major contributions of this model are the identification of direct and indirect relationships among the cloud entities, and the possibility to identify dependencies among adaptation actions. As discussed, inside a cloud environment there are many adaptation decisions that can create interferences and conflicting actions. We plan to develop adaptation engines able to coordinate among the different adaptation options. For this purpose the proposed conceptual model has to be converted into a formal representation and extended with attributes to represent the specific adaptation capabilities of each cloud entity and the adaptation interferences among these different capabilities. The ultimate goal is to build an engine that enables developers to choose the desired adaptation options inside a cloud environment and that automatically handles the decision of which is the best option to be used given the conditions of the cloud environment.

7. ACKNOWLEDGMENTS

The research leading to these results has received funding from the European Union’s Seventh Framework Programme (FP7/2007-2013) under grant agreement 610802 (CloudWave).

8. REFERENCES


