SaaS Cloud Provider Management Framework

Rima Grati[1], Khouloud Boukadi[1] and Hanêne Ben-Abdallah[2]
[1]*The Faculty of Economics and Management of Sfax, Route de l'Aéroport Km 4 Sfax 3018, Sfax, Tunisia*
[2]*Faculty of Computing and Information Technology King Abdulaziz University, Jeddah, K.S.A.*

Keywords: Cloud Computing, Service, SaaS Cloud Provider.

Abstract: The new mode of inter-enterprise cooperation based on the paradigm of composite SaaS keeps promising an ease and a fast implementation of an on demand cooperation. These collaborations called often, on demand cooperation, arising to meet a business opportunity over the Internet. However, this new paradigm poses a certain number of challenges for us. The main one is the lack of a management framework for the cloud provider while covering the different layers IaaS-PaaS-SaaS. To tackle this challenge, we propose a management framework for a cloud provider to define the typical concepts related to the intern management of a composite SaaS provider. The proposed concepts are related to the stakeholders' definition and the presentation of the different management features to consider while specifying the dependence between the cloud layers.

SCIENCE AND TECHNOLOGY PUBLICATIONS

1 INTRODUCTION

The concept of composite SaaS has been recently introduced by Yusho et al. (Yusoh and Maolin 2012) combining both the cloud paradigm and technology of the web service. Since it was introduced, the hype surrounding has followed by promising to revolutionize the way we undertake cooperation over the Internet. The new concept is in its infancy, it is essential to correctly identify the stakeholders (MangementEntities) and management features expected by a SaaS provider. However, up to now no standard definition of management entities or management features provided by a provider has been proposed in the literature. In addition, few industrial initiatives offering composite SaaS have emerged (such as Google Gmail and Facebook).

Besides, these initiatives are related to mailing features, video sharing and streaming video and they do not propose any business service with added value for enterprise.

These considerations complicate the provider' task for offering a composite SaaS to customers willing to establish an on demand cooperation based on service composition. The need is so strong for a management framework for a composite SaaS provider covering both entities and management features within the composite SaaS. To meet such a need, it is important to note that SaaS composite born from a combination of cloud and web service inherits the concepts proposed by these technologies and adds its own specificities. Composite SaaS stems from web service technology the concepts related to customer-provider relationships, abstract/concrete services, service composition, etc. From cloud paradigm, it results from the concept of layer, elasticity of resources, payment for use, etc. Thus, any framework must take into account these different concepts. In addition, it should include some specificities introduced by the composite SaaS concept. These specificities are mainly related to the relevant management functionalities that the SaaS provider is supposed to offer. The examples of the most relevant management features are the configuration required to provide composite SaaS based on web services, establishment of SLA contract and the monitoring of the performance of services according to the established contract. The management features must take into account the layers on which a composite SaaS is based, usually named as XaaS (everything as a Service).

In addition, XaaS layers are connected according to the customer-provider relationship where a layer benefits of services/resources from a lower layer and provides service/resource to a higher layer. This is a vertical dependence between the different XaaS layers. We mean by dependence, as defined by Oxford dictionary "The state of being determined, influenced, or controlled by something else." Dependence covers two aspects namely: the vertical

Grati R., Boukadi K. and Ben-Abdallah H..
SaaS Cloud Provider Management Framework.
DOI: 10.5220/0005550402210228
In *Proceedings of the 12th International Conference on e-Business* (ICE-B-2015), pages 221-228
ISBN: 978-989-758-113-7
Copyright © 2015 SCITEPRESS (Science and Technology Publications, Lda.)

dependence (also called interdependence) and the horizontal dependence (also called intra-dependence). The latter denotes the link between something belonging to the same level n. while the vertical dependence indicates a connection between something belonging to different levels (the notion of hierarchical levels).

Thus, this paper aims to define a management framework based on the concept of vertical dependencies between the different features considered by the cloud provider.

The remaining of this paper is organized as follows. Section 2 discusses the related work. Section 3 presents an overview of the proposed management framework. The concluding remarks are presented in Section 4.

2 RELATED WORK

(Zhao, 2012) presents a holistic security management framework, a model, processes, and controls based on appropriate standards to enable cloud service providers and consumers to be security certified. This framework helps evaluate initial cloud computing security risks and inform security decisions. The author assumes that the security responsibilities of both the provider and the consumer greatly differ between cloud service models. On an IaaS level, the responsibility for securing the underlying infrastructure and abstraction layers belongs to the provider and the PaaS and SaaS level fit to the consumer's responsibility. PaaS offers a balance somewhere in between, where securing the platform itself falls onto the provider, but securing the applications developed against the platform and developing them securely, both belong to the consumer. In his work, Zhao does not address a holistic management framework including all the features considered by the SaaS provider. He focuses only on the security level.

(Winkler and Schill, 2009) present the problem of dependencies between the composite service. They illustrate these dependencies through two examples from the logistics domain. The authors argue that dependencies between services lead to situations where the SLA violation of one service affects the provisioning of other services. Similarly, the renegotiation of the SLA of one service has effects on the SLAs of other services. To solve this problem, the authors present a conceptual architecture to manage the dependencies. The management dependencies is composed of two main steps: the first step is a design time step, it consists on the analysis of dependencies and the creation of a dependency model. The second step is a run time step composed of the dependency effects evaluation based on the dependency model. The proposed architecture is not yet implemented. The authors present a framework management for only the inter dependencies of the service composition. Unlike our approach, the services are not deployed in cloud environment so the authors do not consider the dependencies of the layers of cloud computing paradigm.

(Alcaraz Calero and Gutierrez Aguado, 2014) present a monitoring architecture addressed to the cloud provider and the cloud consumers. They provide a monitoring architecture namely MonPaaS (Monitoring Platform as a Service). It offers to the customers of cloud infrastructures the possibility to see automatically its cloud resources, define manually other resources to be monitored, configure, and customize what information is gathered over its resources. MonPaaS is addressed also to the cloud provider, it offers the ability to see a complete overview of the infrastructure by means of distributed architecture automatically deployed in the cloud infrastructure. MonPaaS in an open source monitoring solution combining Nagios and Openstack. In this work, the authors focus only on the monitoring functionality and specially on the IaaS level and neglect the other cloud stack and the dependency between them.

Table 1: Comparative table of related work.

Features / Works	Monitoring	PaaSConfiguration	Billing	Security	Dependencies
(Zhao, 2012)	No	No	No	Yes	No
(Winkler and Schill, 2009)	No	No	No	No	Yes, especially dependence between the composite service
(Alcaraz Calero and Gutierrez Aguado, 2014)	Yes	No	No	No	No
(Kouki, Jr et al., 2014) and (Kotsokalis, Rueda et al., 2011)	No	No	Yes	No	No
(Sharma, Sengupta et al., 2013)	No	Yes	No	No	No
Our approach	Yes	Yes	Yes	Yes	Yes

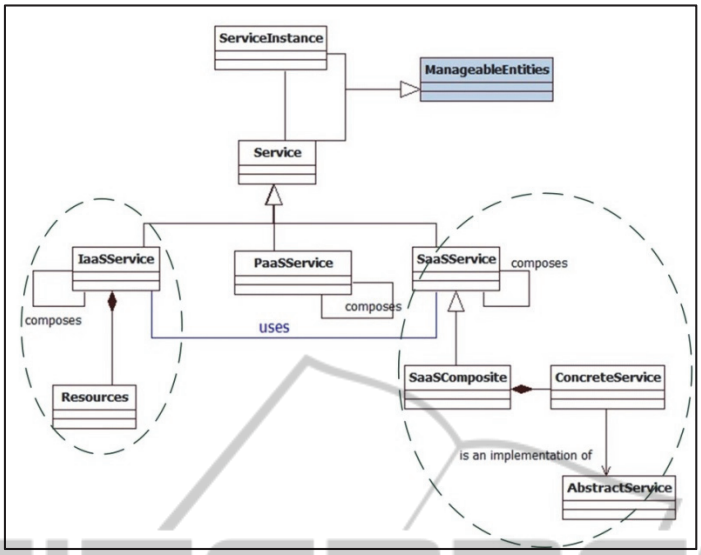

Figure 2: ManagementEntities Meta-model.

Besides, most of the works in litterature propose a price policy related to one layer: SaaS, PaaS or IaaS does not specify the strong dependence that connects these levels of billing (Naik, Beaty et al., 2014) and (Jain and Asadullah, 2012). In addition, these works do not present the detailed formulas for price policy calculation. They only mention the price per request, the price per resource, etc. Moreover few works (Kouki, Jr et al., 2014) and (Kotsokalis, Rueda et al., 2011) focus on the penalty in case of SLA violation in the cloud.

(Sharma, Sengupta et al., 2013) present an approach for assessing a SaaS application for migration to different PaaS cloud platforms. The migration assessment approach can be seen as a 5-stage process. The stages 1 through 3 are manual. In stage 1, the authors create a taxonomy and hierarchical categorization of different external technical utility services. Stage 2 consists in creating a corresponding hierarchical category of the technical utility services for a target PaaS platform, such as Heroku or CloudFoundry,in Stage 3 a set of advisories and recommendations are written to capture the changes required in the code. Stages 4 and 5 are automated, where the actual code analysis and migration assessment is performed in the MAT engine The proposed approach has been implemented in a prototype tool called Migration Assessment Tool or MAT using Java language and PaaS platforms like Heroku and CloudFoundry.

To the best of our knowledge, none of the discussed approaches deals with a holistic management framework that covers almost all of the relevant features for SaaS provider while emphasizing the strong dependence between these different features. Table 1 positions our approach with respect to the related work.

3 OVERVIEW OF THE FRAMEWORK MANAGEMENT

The framework management for a composite SaaS provider that we propose is based on the definitions of the concepts and the features related to the management dependencies across the various layers of cloud computing. This definition is achieved through CPM Framework meta-model shown in Figure 1.

- The *ManagementEntities* package covers the concepts of services and different mechanisms of service composition that are available to the software, platform and infrastructure layer.

In other words, the *ManagementEntities* represent the entities to manage in the framework management of the composite SaaS provider.

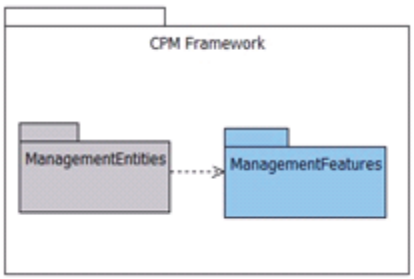

Figure 1: CPM Framework Meta-model.

- The *ManagementFeatures* package covers the concepts related to different features concerning different levels, namely SaaS, PaaS and IaaS that are considered by the cloud provider. All the composite SaaS is executed and delivered using the different elements of *ManagementEntities*.

In the following, we discuss the two core packages highlighting the existing dependencies.

3.1 Management Entities

As shown in Figure 2, the basic building block of CPM Framework is the concept of service. This concept encompasses different levels (SaaS, PaaS and IaaS). Although it is possible to consider other levels below IaaS (example Hardware as a Service), within the CPM framework, we consider that the IaaS level is the last visible and manageable level.

For each service (*Service*), it may be different service instances (*ServiceInstance*), each instance delivers functionalities to one or more customers. Each service instance (*ServiceInstance*) is an independent entity with respect to other instances; hence it can be managed independently. Some management features can be made to service and then applied to all instances of service. To enable the management at all levels, *service* and *serviceInstance* are two subclasses of *MangementEntities*. Management features (*ManagementFeatures*) will be described in Section 3.2.

Note that the *ManagementEntities* package defines two possible kinds of association between services, called «uses» and «composes». The association «uses» identifies a service given by a level uses and depends on another service offered by a lower level of the cloud layer, so it is a vertical dependence.

The association «composes» means that a value added services can be built by composing other services that exist in the same layer, therefore it represents horizontal dependence—it connects concepts belonging to the same layer. More precisely, the association «composes» indicates that the IaaS, PaaS our SaaS (*IaaSService*, *PaaSService* or *SaaSService*) service can be composed respectively by other services IaaS, PaaS or SaaS.

A SaaS composite is a specification of a *SaaSService*, which represents a composition of the concrete service (*ConcreteService*). The meta-model in Figure 2 illustrates the possibility of several concrete services belonging to different provider implementing the same abstract service with different QoS parameters. For instance, “Tunisair” and “AirFrance” are concrete services of the abstract service “flight reservation”.

In addition, *IaaSService* is a composition of various resources (Resources) belonging to the data centre of a cloud provider. Resources can be a

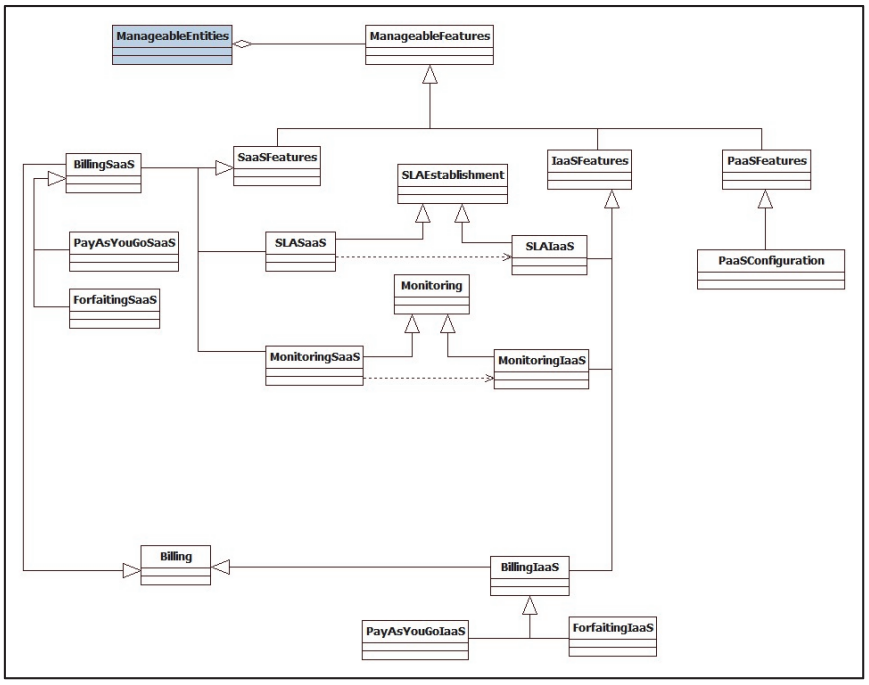

Figure 3: ManagementFeatures Meta-model.

physical machine, virtual machine, network device, etc.

In our work, we omit the dependence PaaS-IaaS, therefore, the vertical dependence related to the *«uses»* association can lead to only one possible case: a SaaS service uses an IaaS service (the case of a composite SaaS is deployed on a virtual machine configured correctly and belonging to the data centre of the cloud provider). In Figure 2, this dependence is represented by the association *« uses »* between different classes encapsulated in two dotted circles. This vertical dependence ensures the selection of a composite SaaS that meets the client's request with an optimal resource allocation (the adequate *IaaSService*).

3.2 Management Features

In addition to the relevant concepts of composite SaaS service, the management framework for the Cloud provider must offer a holistic management of cloud and covers all the layer SaaS, PaaS and IaaS. Indeed, the separate cloud levels are strongly linked and their definitions and their good operation can be valorized only by reference to all the layers. For example, a desfunctioning of a composite service (execution of the service with a too long response time) can be caused by a malfunction in one of the lower levels of the cloud (PaaS or IaaS).

For each problem the cloud provider must find the most effective solution by coordinating the different layers.

In the literature, several management works related to a single level of cloud and dealing with management in isolation of the other level have emerged (Spanoudakis and Mahbub, 2004), (Barbon, Traverso et al., 2006) and (Baresi and Guinea, 2011). Most of these approaches generally cover features on the IaaS layer and neglect the other levels as well as the vertical dependence between them. As illustrated in Figure 3, in our work, the management features considered by the provider (*ManagementFeatures*) encompasses the three levels of the cloud stack –

SaaSFeatures, *PaaSFeatures* and *IaaSFeatures*. More specially, our framework includes the following management features (see Figure 3): SLA establishment, Computing cost of service usage, Monitoring of execution service and Configuration PaaS. The CPM framework is extensible, other management features can be added to cover additional cloud requirements.

In the following, we first detail each of these management capabilities. Next, we present the vertical dependence for each functionality across different levels of Cloud. Finally, we focus on the dependence SaaS-IaaS discussed at the *ManagementEntities* package in order to propose in section 4 a management method for vertical dependence between SaaS and IaaS services (Recall that our framework management assumes that the dependence SaaS-PaaS is provided by a configuration PaaS appropriate to web service deployment and execution).

3.2.1 SLA Establishment

On the client side, a request for a composite service is sent to the provider with QoS constraints such as cost, response time, availability, etc.

Once negotiated, the qualities of service form a clause in the SLA contract (Service Level Agreement) established between the provider and the customer. As part of our work, we are not interested in the negotiation phase, we refer the reader to the most relevant work in the theme (Li, 2011), (Linlin, Garg et al., 2013) and (Stantchev and Schrpfer, 2009). We believe that once the SLA is established, it will be agreed between the two parties (provider and customer).

An SLA can be expressed between different layers in the cloud stack, contractualising the vertical dependencies between layers. We are expected to focus on the vertical dependence between SaaS and IaaS. Solving the problem of vertical dependence means to express an SLA between a SaaS client and a provider offering SaaS service under an IaaS infrastructure. The solution is represented by the class SLAEstablishment in Figure 3.

The provider offers a SaaS service that runs on IaaS selected resources to meet the user's constraints. Thus, from the provider point of view, there are two levels of SLA: SLA for the SaaS level and SLA for the IaaS level (respectively SLASaaS and SLAIaaS in Figure 3). SLA for the SaaS level (SLASaaS) is the contract between the end user and the cloud provider. It specifies a set of objectives in SaaS level in terms of performance and availability. Regarding the IaaS level, it dictates to the cloud provider the adequate resources for the proper execution of services while respecting the SLASaaS. In our case, SLAIaaS represents an internal contract to the provider since it has its own resource infrastructure (IaaS). This type of contract indicates the level of IaaS service in terms of the resource infrastructures under which the composite service should be executed (virtual machine, memory, CPU, utilization rate, etc). The cloud provider can save the

SLAIaaS in a repository to be used later in the future when he is confronted to other service requests with similar QoS.

Otherwise, note that a violation of SLAIaaS consistently produces a violation of SLASaaS. In other words, the SLAs expressed in the SaaS and IaaS level represent a vertical dependence model. An example of vertical dependence of SLA is represented by how to express the availability of SaaS service depending on the resources (IaaS) availability.

3.2.2 Computing Cost of Service Usage

In addition to SLA establishment, our framework takes into account the management function related to the computing cost of service usage (Billing in Figure 3). This management function is related to the policy adopted by the cloud provider to bill the use of the requested service by the customer. Defining a strong vertical dependence between the cloud layers, the invoiced price of SaaS service includes the price of IaaS resources under which the service runs. In our work, we assume that the computing cost of service usage includes indirectly the price of PaaS since we consider that the cloud provider hold the adequate platform for the execution of composite SaaS.

To calculate cost, we propose a formula that aligns the specificities of composite SaaS. We include the price for the SaaS level and the IaaS level to cover the vertical dependence of the billing at both levels. Thus, we define Cost (n,t) the execution price of a composite service requested by the customer from n virtual machines and t types as follows:

$$\mathrm{Cost}_{n,t} = \mathrm{PC}_{n,t} + \mathrm{DTC}_t + \mathrm{PDC}_n \quad (1)$$

$\mathrm{PC}_{n,t}$ represents the cost for serving the request, it depends on the request's processing time ($\mathrm{procT}_{n,t}$), the price of the virtual machine of type t per hour (P_t) and the price of the service (PS) per unit of time.

$$\mathrm{PC}_{nt} = \mathrm{procT}_{n,t} \times (P_t + \mathrm{PS}) \quad \forall n \in N, t \in T \quad (2)$$

- DTC_t is the transfer cost which includes the cost of data in (inDS) and data out (outDS).

$$\mathrm{DTC}_t = \mathrm{inDS} \times \mathrm{inPri}_t + \mathrm{outDS} \times \mathrm{outPri}_t \quad (3)$$

where $\mathrm{inPri}_l$is the price of data transfer in and $\mathrm{outPri}_l$is the price of data transfer out.

- PDCn is the delay cost. It represents how much the service provider has to give discount to users in case of SLA violation. It depends on the penalty rate β (fixed in the SLA contract signed by the cloud provider and the customer) and the penalty delay time (PDT) period.

$$\mathrm{PDC}_n = \beta \times \mathrm{PDT} \quad (4)$$

3.2.3 Monitoring Services Execution

The cloud provider must be able to monitor the parameters of its resources (IaaS Monitoring) to meet the objectives agreed in the SLA. In turn, the monitoring must meet the dependence of monitoring the low level metric (resource IaaS) and the monitoring of SLA parameters (MonitoringSaaS). This dependence is modeled in Figure 3 by a dotted association between MonitoringSaaS and MonitoringIaaS. In Cloud environment, the requested composite runs on physical and virtual resources. These services are characterized by their performance and their availability, which are described as high level SLA parameters such as response time, price and availability, etc. However, the physical or virtual resources, under which services run, are characterized by low level metrics such as packetsize, bandwidth, CPU, memory, etc. Thus, there is a discrepancy between low level metrics and SLA parameters. To bridge the gap between low level metrics and SLA parameters and to solve the problem of vertical dependence between monitoring SaaS and monitoring IaaS, we proposed in (Grati. R., Boukadi. K. et al., 2014c) a framework for monitoring the QoS of composite SaaS.

3.2.4 Configuration PaaS

Configuration in the PaaS level (PaaSConfiguration in Figure 3) is necessary for the execution of composite SaaS. It ensures the request of deployment elementary web service belonging to different service providers.

These features deal with the PaaS level and more particularly the management and the provisioning of software. When a SaaS provider wants to generate a PaaS from heterogeneous PaaS, he meets several management difficulties such as the lack of compatibility of APIs and the lack of PaaS devoted to the provider of composite service, etc. Indeed, each available PaaS requires the use of specific APIs. For example, to interact with the PaaS Force.com, Apex API is provided; Cloud Foundry also provides its own API, etc. Each PaaS exposes a different interface and no standard or generic API is available. So a provider of composite SaaS can not achieve an easy interaction with heterogeneous PaaS.

To overcome the difficulty of managing heterogeneous PaaS, we assume that the provider has an independent PaaS for provisioning and managing software assigned for running composite SaaS. Figure 4 represents the various entities that compose the platform. By platform, we mean all the software needed for the execution of web service deployed on an IaaS layer.

As modeled in Figure 4, each platform is characterized by its unique name and it is instantiated from *PlatformTemplate* which is, in turn, characterized by a name. *PlatformTemplate* is constructed from a set of entities:

- *PlatformComponent* is the list of software associated with *PlatformTemplate*. The possible values of this list are: a database, a container, a router and an engine.
- *PlatformInterface* defines the interfaces between the components of the PlatformComponent
- *PlatformConfiguration* consists of several configuration actions (*ConfigurationAction*).

Certainly, there are functions associated with the configuration in the IaaS level. However we do not deal with this configuration on this level since there are several mature tools to perform IaaS configurations tasks (Georgiou, Tsakalozos et al., 2013) and (Papagianni, Leivadeas et al., 2013).

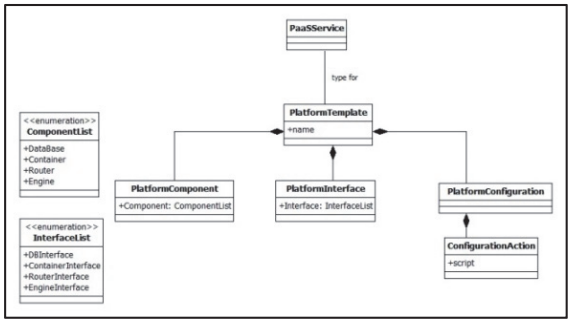

Figure 4: PaaS configuration meta-model.

So far, we have presented the building blocks of a framework management of the cloud provider that means the different stakeholders and different features to manage the cloud provider while emphasizing vertical dependence between the cloud layers. In the following, we present a method to manage the vertical dependence between SaaS and IaaS services.

4 CONCLUSIONS

Our proposed approach addresses the lack of management framework for the SaaS cloud provider. This framework is described through a set of key concepts. In addition, it includes some specificities introduced by the composite SaaS concept which is born from a combination of cloud and web service. These specificities are mainly related to the relevant management functionalities that the SaaS provider is supposed to offer. The examples of the most relevant management features are the configuration required to provide composite SaaS based on web services, establishment of SLA contract and the monitoring of the performance of services according to the established contract. The management features takes into account the layers on which a composite SaaS is based, usually named as XaaS (everything as a Service).

REFERENCES

Alcaraz Calero, J. and J. Gutierrez Aguado (2014). "MonPaaS: An Adaptive Monitoring Platform as a Service for Cloud Computing Infrastructures and Services." Services Computing, *IEEE Transactions on PP(99): 1-1.*

Barbon, F., et al. (2006). Run-Time Monitoring of Instances and Classes of Web Service Compositions. *International Conference on Web Services ICWS '06, pp. 63-71.*

Baresi, L. and S. Guinea (2011). "Self-Supervising BPEL Processes." Software Engineering, *IEEE Transactions on 37(2): 247-263.*

Georgiou, S., et al. (2013). Exploiting Network-Topology Awareness for VM Placement in IaaS Clouds. Third *International Conference on Cloud and Green Computing (CGC).*

Grati. R., et al. (2014c). A framework for IaaS to SaaS monitoring of BPEL processes in the Cloud: design and evaluation. 11th ACS/IEEE International Conference on Computer Systems and Applications (AICCSA' 2014), *IEEE Computer Societ, Doha, Qatar.*

Jain, S. and A. M. Asadullah (2012). Aggregating Bills and Invoices on Cloud for Anytime Anywhere Access: A Sustainable System. Third International Conference on Services in Emerging Markets (ICSEM), 2012.

Kotsokalis, C., et al. (2011). Penalty Management in the SLA@SOI Project. Book: Service Level Agreements for Cloud Computing, *Springer New York: 105-121.*

Kouki, Y., et al. (2014). A Language Support for Cloud Elasticity Management. IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, CCGrid, Chicago, United States.

Li, P. (2011). Towards a framework for automated service negotiation in cloud computing. *IEEE International Conference on Cloud Computing and Intelligence Systems (CCIS).*

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS

Linlin, W., et al. (2013). Automated SLA Negotiation Framework for Cloud Computing. 13th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid).

Naik, V. K., et al. (2014). Service Usage Metering in Hybrid Cloud Environments. *IEEE International Conference on Cloud Engineering (IC2E), 2014.*

Papagianni, C., et al. (2013). "On the optimal allocation of virtual resources in cloud computing networks." Computers, *IEEE Transactions on 62(6): 1060-1071.*

Sharma, V. S., et al. (2013). MAT: A Migration Assessment Toolkit for PaaS Clouds. *IEEE Sixth International Conference on Cloud Computing (CLOUD), 2013.*

Spanoudakis, G. and K. Mahbub (2004). Requirements monitoring for service-based systems: towards a framework based on event calculus. *19th International Conference on Automated Software Engineering, 2004. Proceedings.*

Stantchev, V. and C. Schrpfer (2009). Negotiating and Enforcing QoS and SLAs in Grid and Cloud Computing. Proceedings of the 4th International Conference on Advances in Grid and Pervasive Computing. Geneva, Switzerland, *Springer-Verlag: 25-35.*

Winkler, M. and A. Schill (2009). Towards Dependency Management in Service Compositions. Int. Conf. on E-Business and Telecommunication Networks: 79-84.

Yusoh, Z. I. M. and T. Maolin (2012). Clustering composite SaaS components in Cloud computing using a Grouping Genetic Algorithm. *IEEE Congress on Evolutionary Computation (CEC).*

Zhao, G. (2012). Holistic framework of security management for cloud service providers. *10th IEEE International Conference on Industrial Informatics* (INDIN), 2012.

SCITEPRESS
SCIENCE AND TECHNOLOGY PUBLICATIONS