CSLAM: A Framework for Cloud Service Level Agreement Management Based on WSLA

Milad Torkashvan
Faculty of Electrical and Computer Engineering
Shahid Beheshti University
Tehran, Iran
Milad.Torkashvan@gmail.com

Hassan Haghighi
Faculty of Electrical and Computer Engineering
Shahid Beheshti University
Tehran, Iran
h_haghighi@sbu.ac.ir

*Abstract*_**With impressive progress of cloud computing and its influence on many aspects of utility computing, many of companies and enterprises are tending to this new product of information technology; some of these companies are interested in consuming cloud services and others tend to provide cloud services. The most common thing for these two major classes of users is service level agreements (SLAs) which are set between them. Many frameworks have been introduced for SLA management in web service level, but there are few applicable frameworks that are specific for cloud computing services. In this paper, we propose a framework for SLA management in cloud computing and especially inter-cloud environments. This framework is based on WSLA which is introduced by IBM, but we have changed it to fit into cloud computing environments. This framework is in fact a component of the cloud computing framework which we have introduced in our previous work. In this paper we are not going to provide a list of cloud service SLA parameters; instead, we propose a framework to manage cloud service SLAs and metrics.**

Keywords: Service Level Agreement, Cloud SLA, SLA Management, Inter-Cloud SLA.

I. INTRODUCTION

In current days, cloud computing has been one of the most popular research topics in IT, and also it has been able to change the industrial trend to itself. Every technology which is intended to be industrial or commercial must have a solution for dealing with consumers; as a matter of fact, it must have a solution for making money through the service which is going to be provided. In this course, SLA (Service Level Agreement) is the way in which this deal happens. SLA is an agreement which stands between consumer and provider and specifies the level of services, QoS, prices and many other qualities.

Many approaches have been introduced for service level agreement for web based applications, such as WSLA [1] and WS-Agreement [2]; but these frameworks are not sufficient in cloud environments. Services in clouds are provided in three major categories as Software-as-a-service (SaaS), Platform-as-a-service (PaaS) and Infrastructure-as-a-service (IaaS) which can be broken down to eleven more detailed services [3] as are depicted in figure 1.

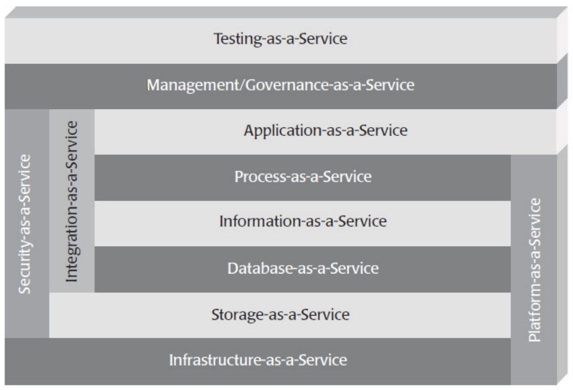

Figure 1. More detailed cloud services [3]

Since the nature of services in cloud environments is totally different from conventional Internet-based services, and consumers' expectations are different, cloud services need a different SLA management system. For example, suppose a conventional Internet-based service which is going to be used by a particular consumer; the SLA between the consumer and the provider is especially about this particular service, e. g., its availability, response time and reliability. But this type of SLA cannot be applied in cloud environments because the consumer pays for the major services (Infrastructure-as-a-service, Application-as-a-service, Security-as-a-service, etc.) [3]. Some cloud based SLA management frameworks, such as [4, 5, 6, 7, 8, 9], have been introduced which propose various approaches to resolve this need. Some of them do not support the whole SLA lifecycle; some others only talk about SLA parameters and do not say how to manage them.

In this paper we attempt to propose a framework, called CSLAM (Cloud Service Level Agreement Management), which cover SLA life cycle in cloud environments [1, 9]. CSLAM is based on WSLA which provides both SLA negotiation language and management framework. CSLAM also manages the SLA aware inter-cloud service provision. On the other hand, it is in fact a part of our major cloud computing framework which is introduced in [12].

The rest of this paper is organized as follows. Section 2 gives a brief description about WSLA. In section 3 some related work is reviewed. Section 4 describes CSLAM. CSLAM language structure is introduces in section 5, and Section 6 brings a discussion about CSLAM. Finally, section 7 gives the conclusion and future work.

II. WEB SERVICE LEVEL AGREEMENT (WSLA)

WSLA [1] consists of an XML-based definition language and an architecture which supports the SLA lifecycle. WSLA is a hierarchical language to define SLA parameters from resource through business objectives. Figure 2 shows a sample hierarchy in which SLA parameters are the top most elements. Also, a SLA parameter is made from at least one composite metric and each composite metric is the aggregation of some resource metrics [13].

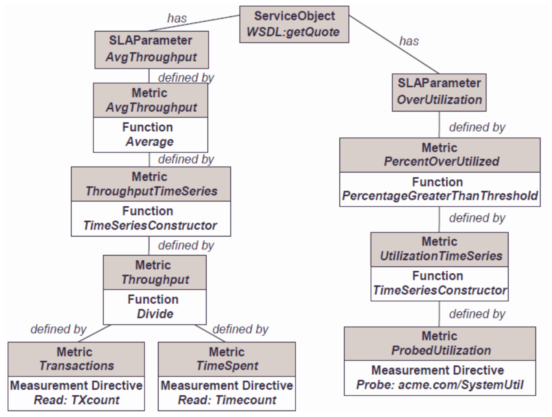
ServiceObject WSDL:getQuote
has
SLAParameter AvgThroughput
defined by
Metric AvgThroughput
Function Average
defined by
Metric ThroughputTimeSeries
Function TimeSeriesConstructor
defined by
Metric Throughput
Function Divide
defined by
Metric Transactions
Measurement Directive Read: TXcount
defined by
Metric TimeSpent
Measurement Directive Read: Timecount
has
SLAParameter OverUtilization
defined by
Metric PercentOverUtilized
Function PercentageGreaterThanThreshold
defined by
Metric UtilizationTimeSeries
Function TimeSeriesConstructor
defined by
Metric ProbedUtilization
Measurement Directive Probe: acme.com/SystemUtil

Figure 2. An example of hierarchical structure of WSLA [13]

Figure 3 depicts an overview of the SLA management architecture which supports the whole SLA lifecycle (Definition, Negotiation, Deployment, Monitoring and Enforcement). In this figure, the supposed service is assumed in the form of a web service which is provided through a servlet engine. Monitoring information is accessible by various services of the SLA monitoring framework for administrator. The interface of the web service which is going to be provided is defined by an XML document in the Web Service Description Language (WSDL). The SLA document will be referenced in the WSDL document. Typically, an SLA defines several SLA parameters, each referring to an operation of the web service. However, an SLA may also refer to the service as a whole, or even a composition of multiple web services [14].

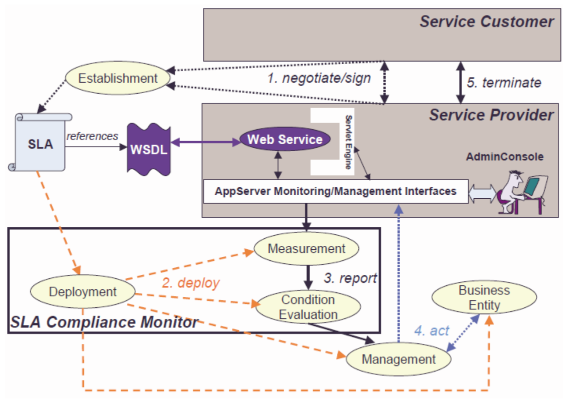

Figure 3.Overall WSLA architecture [13]

In fact, WSLA has both bottom-up and top-down architectures; SLA parameters are defined by consumers, resource metrics are defined by providers, and the framework maps them together to make monitoring, reporting, decision making, penalizing and obligating applicable, according to what is stated in the SLA document between a consumer and a provider. Due to the limited space, it is not possible to describe WSLA deeply, but interested readers can read about it in [1, 15].

III. RELATED WORK

Many works have been done to propose a suitable framework for SLA management in cloud environments. In fact, because of the nature of cloud environments and its services, its SLA management would be different from typical service provisioning. [4] proposes a framework named LoM2HiS for cloud SLA management; this is a subset of FoSII (Fundamental of Self-governing ICT Infrastructure) project [10] which is used to manage the whole lifecycle of self-adaptable cloud services [11]. The main purpose of LoM2HiS is to map the low-level resource metrics to high-level SLA parameters. Its first drawback is that it just supports monitoring and enforcement parts of the SLA lifecycle [1]. The other problem of this framework is that it does not have any plan for inter-cloud service provision.

[5] proposes a sort of SLA parameters for cloud computing environment, but it does not say anything about SLA management framework; in fact, it would be better to change the title of this work into “SLA Parameters for Cloud Computing” instead. [6] proposes a framework which is a SLA aware cloud service provider. It proposes a group-based monitoring; the groups send monitoring data to upper level to decide how to allocate the resources according to the predefined policies. The main problem is that it does not support the whole SLA lifecycle (it does not support SLA definition and negotiation).

[8] proposes a platform for SLA in cloud computing environment. This platform stands between cloud consumers and cloud providers, and services can be

found in a service list which is aggregated from cloud consumers. Cloud providers have also relationship with the monitoring center so as to receive data from various clouds. This platform also utilizes a reputation system which is based on the comments which are left from consumers. The main problem with this platform is that it stands beneath the cloud providers not over them. In this state, cloud providers must register into the platform to advertise their services and make them searchable; this issue makes this platform far from real world systems.

In this paper, our purpose is to introduce a novel cloud SLA framework to manage whole SLA lifecycle and SLA aware inter-cloud service provision. Details about the proposed framework are provided in the following section.

IV. CSLAM

Although CLSAM is intended as a part of a cloud computing framework introduced in [12], it is a generic framework which can be placed into any other cloud service provider. This section describes the components of CSLAM and shows how it supports SLA lifecycle. Figure 4 depicts an overall view of CSLAM; we describe its components in detail in the following sections and subsections.

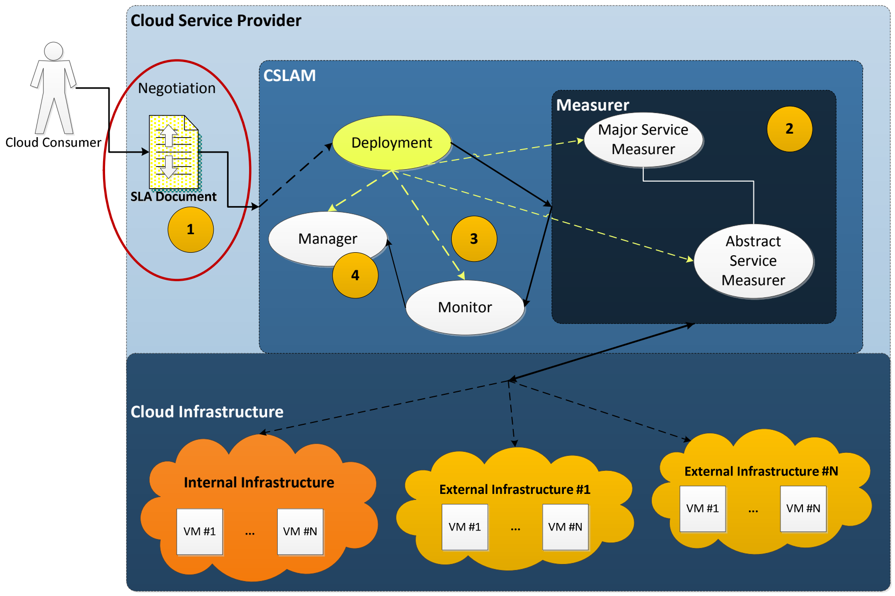

Figure 4.Overall view of CSLAM

In step (1) a negotiation happens between the consumer and provider which turn into the SLA document representing the SLA parameters of major cloud services. The language which defines these SLA parameters will be discussed in the next section. Then, after the negotiation, SLA documented will be deployed; it covers measurement, monitoring and management. Step (2) refers to SLA parameter measurement which is, in fact, hierarchical from major cloud service parameters to lower level metrics. One of the differences between WSLA and CSLAM is here: WSLA measures SLA parameters from metric functions, but in CSLAM, SLA parameter measurement values come from some services which are called “Abstract Service Measurer”; more descriptions will be given later. Step (3) considers the data provided by measurement services. It monitors the parameters to check whether they are over/under/equal to thresholds which are defined for the SLA parameters. Step (4) refers to the situation in which each one of cloud services are; if some violation happens about any parameter, the management component applies the appropriate action against the situation, such as punishment, invoking a special service or any other action which is agreed upon in the SLA document.

This framework supports the whole SLA lifecycle; according to WSLA [1], the lifecycle can be considered as figure 5.

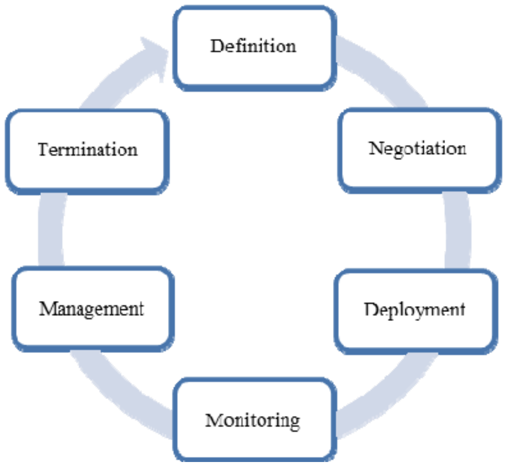

Figure 5. SLA lifecycle

This starts from the definition which is presented by CSLAM language; the other parts in the cycle are provided through the introduced framework, and finally the SLA will be terminated on its time expiration or the consumer request.

A. *SLA Definition and Negotiation*

Here is where the SLA is signed by both signatory parties and will be presented by a CLSAM document which is a XML-based language. According to the parameters and thresholds of the parameters which are going to be agreed, the price of services will be specified, and the SLA will be deployed as it is signed. Since CLSAM also supports inter-cloud communications, its language is hierarchical to cover inter-cloud metric measurement; this hierarchical view is shown in figure 6. It is worth mentioning that although some works have been done in the area of inter-cloud pricing, such as [16], in this paper, we point out how to support metric measurement when the services are spread over one or more than one cloud.

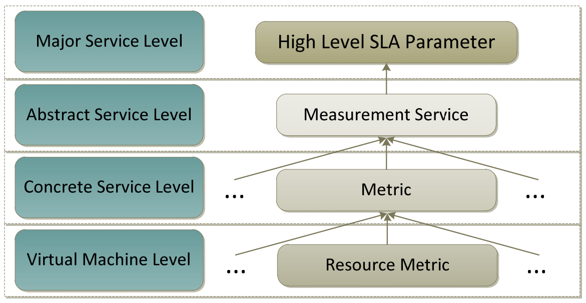

Figure 6. Hierarchy of CSLAM language

CSLAM language is divided into four levels, (1) major service level, which represents the services such as availability percentage of SaaS, IaaS, etc. (2) abstract service level, which defines the SLA parameters for underlying abstract services. Abstract services are groups of similar web services. In this work we assign concrete web services to abstract services and SLA parameters in this level will be defined for abstract services. (3) Concrete service level, which defines metrics for concrete web services which are assigned to abstract services, the matter is that, consumers are involved with abstract services and their functionalities, and total SLA metrics for High level major services are provided through abstract services. (4) Virtual machine level, which is defined for virtual machines in cloud providers. It's possible that concrete services are provided through different cloud providers, and cloud providers run services onto virtual machines, thus, the resource metrics will be defined for virtual machines.

It should be noticed that some SLA parameters are configuration parameters [5], such as PaaS-related ones, but some others are based on metrics which must be measured such as SaaS-related ones and IaaS-related ones. CSLAM language is designed for the latter type of parameters. Figure 7 provides an example of the usage of this language. It depicts a typical example when a consumer and a provider have agreed on a cloud major service (in this example, Application-as-a-service) availability to be more than x, and the figure shows how this parameter will be measured for a high-level major cloud service. Figure 7 shows, to measure the Application-as-a-service availability, it is needed to measure the SaaS availability and check to see if it is over the threshold or not. It also shows that to measure the SaaS availability, it is needed to measure all of services which are under SaaS. Each of these services can be multi-provisioned, and each version of these services could have been deployed either in the internal infrastructure or other external infrastructure which can belong to another cloud provider. Thus, it is needed to measure availability for each of the possible cloud providers. And finally, each virtual machine which deploys services must report its presence periodically. The flow of these function calls is bottom-up. What we have presented here is the structure of the CSLAM language which defines the agreement between signatory parts; now the SLA is ready to be deployed. To serialize the measurement, a *TimeSeriesConstrcutor* can be used. By calling this in a SLA document, one can serialize the lower level elements and put them in a timely period by minute, hour, day, etc. *TimeSeriesConstrcutor* can be used in any place of the SLA definition as needed, but one is mandatory at the first place (as shown in figure 7).

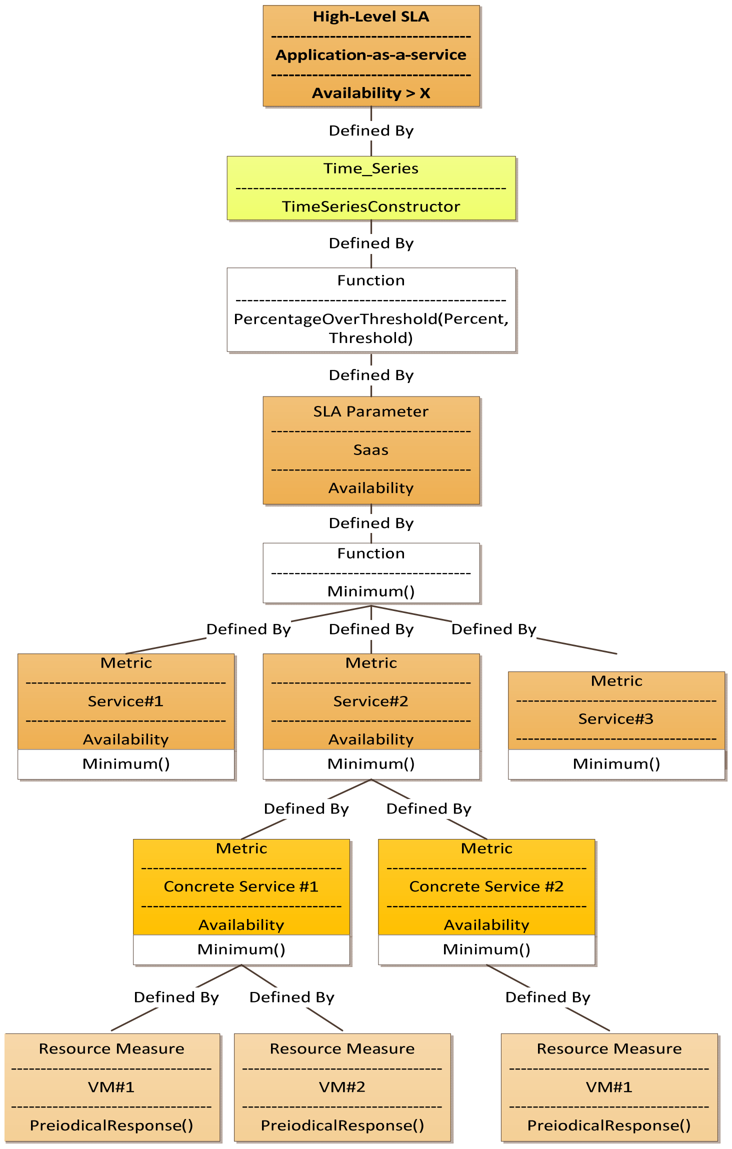

Figure 7. A CSLAM example

B. *SLA Deployment*

A main component in the deployment phase is the measurer component. This component is in charge of acquiring metric values for the parameters. The SLA deployment also consists of monitoring and management of SLA parameters. Figure 8 shows that SLA deployment is realized in a two-dimensioned vector (value, time).

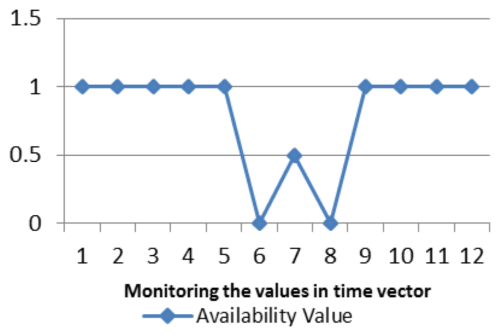

Figure 8.Two-dimensioned deployment behavior

It means that deployment is done in a time vector, i. e., check values of SLA parameters, such as availability in figure 8, will be fetched from measurers periodically (once in any time border), and the monitor inspects the fetched values in the line of time to see whether there is any deviation or not.

As the SLA is agreed between signatory parts, the SLA document is stored in a SLA repository; this repository stores a SLA document for each consumer. For each SLA document (agreement), an instance of measurer is initialized. As shown in figure 6, the measurement structure is hierarchical; hence, it needs a hierarchical instantiation of measurers from top to down to calculate metric values and pass them to the higher level. Finally, resulting values will be saved in a SLA log base so as to be ready for the monitor component.

C. *SLA Monitoring*

The monitoring component is in charge of checking the measurement values to see whether they violate the threshold or not. Checking type of this violation differs from one SLA parameter to another SLA parameter. For instance, availability (see figure 8) checking is influenced totally by time factor: if a service is not available for more than 3% of time during the last 10 time units, it is a violation (we suppose it has been agreed upon), and the service provider must be treated as the SLA document says.

D. *SLA Management*

In the SLA document, it must be defined explicitly what should be done when a parameter value violates the agreed value. This action can be a service call, a monetary punishment to service provider and so forth.

V. CSLAM LANGUAGE STRUCTURE

In this section, the structure of CSLAM language is presented. Figure 9 depicts a class view of this structure and the relationships between the objects. This language is XML based, and each tag in its structure is shown as an UML object).

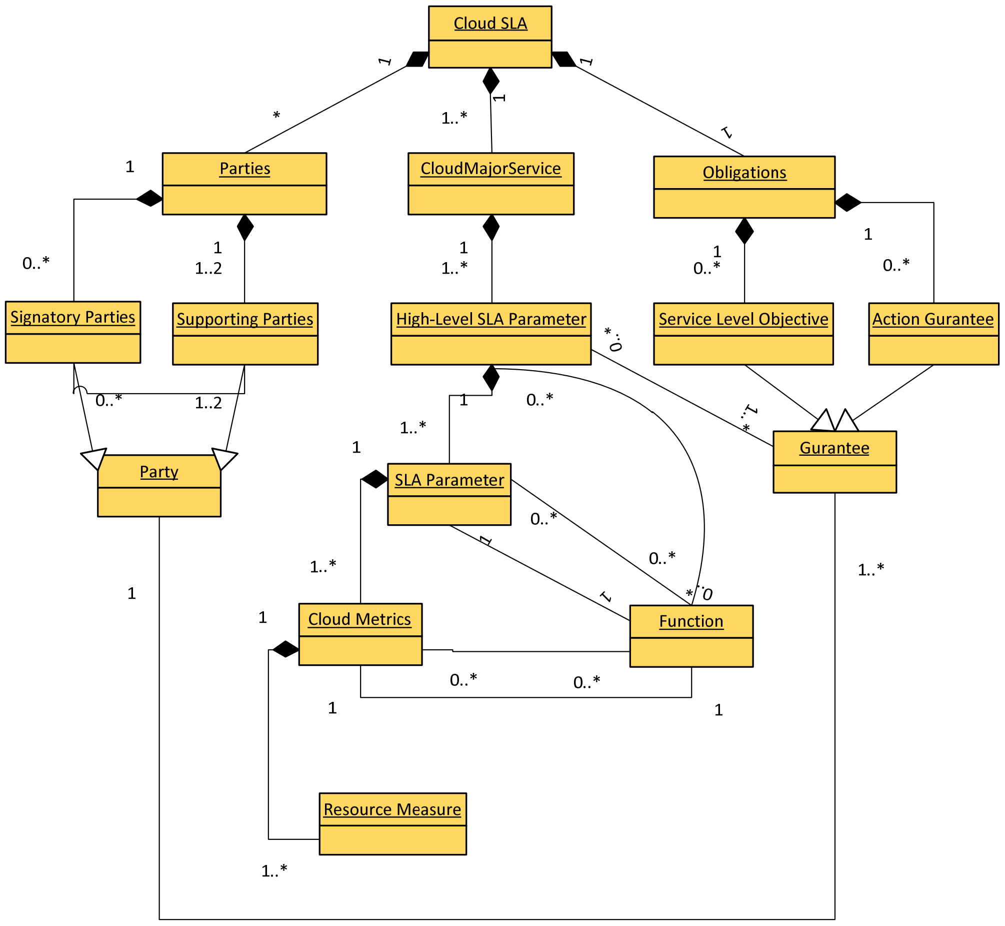

Figure 9. CSLAM language structure

As shown in figure 9, CSLAM language has three basic substructures, named parties, cloud major service and obligations:

- Parties. This is the part that defines the consumer and provider as signatory parties that are going to be agreed on some level of services. There is also a supporting party that can be defined by one or both signatory parties to measure the parameters and evaluate the condition of services.

- Obligations. This part contains two types of obligations: A service level objective which is a guarantee of a particular state of Cloud SLA parameters. The other type of guarantee is an action guarantee which is the promise to do something in a defined situation (notification for example). Each guaranty should be related to one of the parties (often the provider).

- Cloud Major Service. This part defines the hierarchical structure of cloud services and the parameters which are going to be met. Each CSLAM has one or more cloud major service, and each cloud major service contains one or more high level SLA parameters. Each high level SLA parameter can directly have one or more SLA parameters or use functions to define SLA parameters. In addition, each SLA parameter can directly define cloud metrics or use a function (or a composition of functions) to define them. Finally, each cloud metric will be mapped to one or more than one resource measure to generate the physical measurement values. In fact, two different types of SLA parameters exist in a cloud SLA: (1) some parameters which pertain to Platform-based agreements, such as VM scale up/down, Auto Scaling, Max VMs Number, Costs, Backup Enabled and Monitoring [5]. These parameters are counted as settings and appear in Testing-as-a-service, Management-as-a-service, Information-as-a-service, Integration-as-a-service, Security-as-a-service, Platform-as-a-service and sometimes Infrastructure-as-a-service services; see figure 1. Some other parameters need to be measured, such as Availability, Response Time, Boot Time and Reliability [5] which are more in Infrastructure-as-a-service, Application-as-a-service, Process-as-a-service, Application-as-a-service and so forth.

VI. DISCUSSION

To show the applicability of CSLAM, in this section, we discuss about a real world example with two instances of SLA parameters, i.e., availability and response time. Suppose that a consumer has signed up in a cloud platform which uses the proposed SLA framework. Also, suppose that the cloud provider has been committed to provide SaaS with 98.5% of availability and 99.9% of response time. After the negotiation happens, service provisioning gets along with the SLA lifecycle as follows:

Step 1. For the first step, after the negotiation in the SLA lifecycle, is that the CSLAM document will be generated. As aforementioned, it is a XML document in which the SLA parameters and parties will be specified.

Step 2. After the step 1 is completed, the SLA should be deployed. Now we are to describe what will happen during the deployment process. For simplicity, we assume that whole of cloud infrastructure belongs to the cloud provider, and for the time being, there is no need to use other cloud provider services. At first, a major service measurer will be instantiated which is responsible to record the cloud SLA parameters values. It starts from a zero hour with a shared timestamp and considering a particular time period (e.g., measuring the parameter once per 10 minutes). Since every major cloud service consists of some fine grained services, for each of these fine grained services, one abstract service measurer will be instantiated (as figure 10).

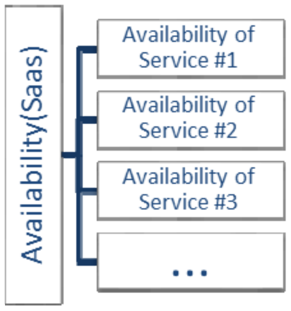
Availability(Saas)
Availability of Service #1
Availability of Service #2
Availability of Service #3
...

Figure 10. Breaking a Cloud service to its subsidiary services

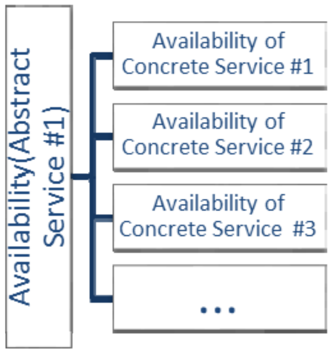
Availability(Abstract Service #1)
Availability of Concrete Service #1
Availability of Concrete Service #2
Availability of Concrete Service #3
...

Figure 11. Breaking an abstract service to its underlying cloud providers

Each of these abstract services is the delegates of services which are provided as subsidiaries of the major services. These are called abstract services because there is the possibility that more than one underlying concrete services provide. Thus, each abstract service measurer acquires the metric values from each of existing cloud providers and finally normalizes the values to one value (through a function which is defined by user). For each concrete service, one concrete measure will be instantiated (figure 11). These concrete measures are in charge of gathering the metric values from virtual machines which are executing the particular service.

Step 3. After required data is acquired, the monitor component will check data deviation according to the type of SLA parameter. There are two possible types:

- The monitor component investigates the aggregated data for a particular period of time, such as once per hour, and checks this data. Table 1 shows availability data for a particular cloud service which have been acquired once per 5 minutes by the measurer component (1 and 0, are the responses to availability check).

0	5	10	15	20	25	30	35	40	45	50	55	60
1	1	1	1	0	0	0	1	1	1	1	1	1

Table 1. Stored responses to rolling up a cloud service per 5 minutes

In this table, the monitor component checks the data to see how much percent of time the service had responded. In this case the service has been available for 77% of time.

- The other possible type is almost like the type mentioned before; the only difference is that the latter type just checks the values immediately. Considering table 1, the monitor component will do its job each 5 minutes, and when it reaches minute 20, a deviation will be reported (unavailability is a deviation) .

Step 4. This step refers to the SLA management. In this step, according to what has been agreed in the SLA, some service or operation will be invoked to manage the reported deviation.

The main purposes are met as following:

- Supporting SLA Life Cycle: the proposed framework supports whole SLA life cycle. The framework proposes a SLA language which supports SLA definition and negotiation; it also proposes an approach to monitor services from major cloud services to virtual machines. And finally, it shows how to manage SLAs.

- Inter-Cloud SLA management: the proposed framework separates similar concrete web services into abstract services (concrete services are provided through different cloud providers). Cloud consumer is involved with major services such as SaaS, and the service level of these major services is gathered through abstract services. It helps consumers to have an overview on services without being involved with every provided service though different clouds.

- Being based on WSLA: this framework is based on WSLA, because it used the hierarchical view which is used in WSLA (gathering metric values from resource to high level services), but the different between CSLAM and WSLA is that CSLAM supports major services such as SaaS which are comprised of many other web services, but WSLA is restricted to one web service.

-

VII. CONCLUSION AND FUTURE WORK

In this paper we have proposed a cloud SLA framework, called CSLAM, which has a language to present the agreements between signatory parties. In addition, it has a management mechanism to deploy and monitor SLA parameters. CSLAM is based on WSLA proposed for web service level agreement with some extensions and changes to fit into cloud environment. This is an applicable framework because its hierarchical nature helps it to support major services which are provided through cloud service providers. As our future work, we aim at implementing CSLAM as a part of our cloud computing framework [12] in order to provide a whole package of cloud computing.

ACKNOWLEDGMENT

This work is supported by Research Institute of Information and Communication Technology of Iran.

REFERENCES

[1]. WSLA, IBM, http://www.research.ibm.com/wsla/.

[2]. A. Andrieux, K. Czajkowski, A. Dan, K. Keahey, et al., Web Services Agreement Specification (WS-Agreement), Version 1.0, Grid Forum Document, FD.107, The Open Grid Forum,Joliet, Illinois, United States, 2007.

[3]. D.S. Linthicum, Cloud Computing and SOA Convergence in Your Enterprise, Addison-Wesley, 2010.

[4]. V.C. Emeakaroha, I. Brandic, M. Maurer, et al., "Low Level Metrics to High Level SLAs - LoM2HiS Framework: Bridging the GapBetween Monitored Metrics and SLA Parameters in Cloud Environments," IEEE Internation on High Performance Computing and Simulation (HPCS), pp. 48-52, 2010

[5]. M. Alhamad, T. Dillon, E. Chang, "Conceptual SLA Framework for CloudComputing," 4th IEEE International Conference on Digital Ecosystems and Technologies, 2010.

[6]. N. Bonvin, T.G. Papaioannou and K. Aberer, "Autonomic SLA-driven Provisioning for Cloud Applications," 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, 2011.

[7]. Z. Wang, X. Tang and X. Luo, "Policy-Based SLA-Aware Cloud Service Provision Framework," Seventh International Conference on Semantics, Knowledge and Grids, 2011

[8]. M. C. Wang, X. Wu, W. Zhang, et al., "A Conceptual Platform of SLA in Cloud Computing," Ninth IEEE International Conference on Dependable, Autonomic and Secure Computing, 2011.

[9]. R. Buyya, S. K. Garg and R.N. Calheiros, “SLA-Oriented Resource Provisioning for CloudComputing: Challenges, Architecture, and Solutions,” International Conference on Cloud and Service Computing, 2011.

[10]. Foundation of Self-governing ICT Infrastructures (FoSII),http://www.infosys.tuwien.ac.at/linksites/FOSII/index.html.

[11]. I. Brandic, “Towards Self-manageable Cloud Services,” 2nd IEEE International Workshop on Real-Time Service-Oriented Architecture and Applications, 2009.

[12]. M.Torkashvan, H.Haghighi, “A Service Oriented Framework for Cloud Computing,” The 3rd International Conference on Information and Communication Systems, Jordan, 2012.

[13]. K White,“Definition of Managed Objects for Service Level Agreements Performance Monitoring”. RFC 2758,IETF, February 2000.

[14]. V.Tosic, B.Pagurek, B.Esfandiari, et al., “Management of Compositions of E- and M-Business Web Services with multiple Classes of Service,” Proceedings of the 8th IEEE/IFIP Network Operations and Management Symposium (NOMS 2002), pp 935–937, Apr 2002.

[15]. Web Service Level Agreement (WSLA) Language Specification, IBM Publication, 2003.

[16]. i. Goiri, J. Guitart and J. Torres, “Economic model of a Cloud provider operatingin a federated Cloud,” Springer-Information System Front Journal, 2011.