International Journal of Advances in Engineering and Management (IJAEM)
Volume 2, Issue 7, pp: 425-432 www.ijaem.net ISSN: 2395-5252

Sla Establishment and Monitoring in Cloud Based Software as a Service

Ajayi Oluwabukola, Awodele Oludele, Ogbonna A.C, Okolie S.O, Ajayi Adebowale

[1]*Department of Computer Science, Babcock University*
[2]*Department of Computer Science, Babcock University*
[3]*Department of Computer Science, Babcock University*
[4]*Department of Computer Science, Babcock University*
[5]*Department of Computer Science, Babcock University*

Date of Submission: 25-09-2020 Date of Acceptance: 08-10-2020

ABSTRACT: The goal of monitoring contractual Service Level Agreements (SLAs) is to measure the performance of a service, to evaluate whether its provider complies with the level of Quality of the Service (QoS) that the consumer expects. A service provisioning infrastructure should allow the establishment of SLA through coordinated negotiations among the potential stakeholders. However, SLA establishment can only partially serve the needs of SLA management if not linked to SLA monitoring. In a bid to ensure a precise definition of the SLA for easier automation of SLA monitoring, it is imperative to identify the relationships between establishing and monitoring of SLAs.

Therefore, this study focuses on the relationship between establishment and monitoring of such SLAs, showing how the two processes become tightly interleaved in order to provide meaningful mechanisms for SLA management.

KEYWORDS: Service Level Agreements, Quality of Service, SLA Establishment, SLA Monitoring, SLA Management, Service Provisioning**.**

I. INTRODUCTION

Before the advent of Cloud, the ICT administrative tasks were easy because the important objective of resource provisioning was performance [1]. Eventually, the complexity of applications grew, increasing the difficulties in their administration.

Consequently, enterprises realized that it is more efficient to outsource some of their applications to third-party Software as a Service (SaaS) providers enabled by Cloud computing due to some of the following reasons [2]:

- To reduce the maintenance cost, because as complexity increases the level of sophistication required to maintain the system also increases.
- Enterprises need not to invest in expensive software licenses and hardware before knowing the business value of the solution.

A service is a software system used to perform a specific task for its customers using request-response messages. Hence by moving to the SaaS model, customers tends to benefit from constantly maintained software and the complication of switching to new releases of software is entirely managed transparently by SaaS providers. The SaaS providers enlarges market share by accepting profitable requests and improving the Customer Satisfaction Level (CSL).

A service customer may choose a specific service from among similar ones that offer the same business. For this reason, it is a challenge for a service provider to maintain the running of the service at an adequate level in order to keep attracting potential customers [3, 4]. Customers' interest regarding the level of service offered may vary and this can be related to different dependability, performance and performability metrics such as response time, availability, throughput, reliability, exception handling, and security [5]. In this context, and in order to give customers the ability to choose which service is best suited to them, the term Quality of Service (QoS) has evolved to denote the quality of the non-functional properties of a service [3]. Service providers and customers choose QoS metrics and specify guarantees of their values over a certain period of time; these are called Service Level Objectives (SLOs) [6, 7]. Owing to their importance in attracting customers, SLOs have become a crucial part of a larger legal document called a Service Level Agreement (SLA) [3].

International Journal of Advances in Engineering and Management (IJAEM)
Volume 2, Issue 7, pp: 425-432 www.ijaem.net ISSN: 2395-5252

On the other hand, [8] highlighted that customer satisfaction is an important success factor to excel in the service industry and the best way to ensure the Quality of Service (QoS) is to define a legal contract which is a Service Level Agreement (SLA), between a service provider and a consumer [2] to measure the CSL.

II. SERVICE LEVEL AGREEMENT (SLA)

SLAs can be traced back to 1980s in telecommunication companies where telecommunication companies includes SLA within the terms of their contracts with customers to define the level(s) of service being sold them in plain language terms.

[9] define an SLA as: "An explicit statement of expectations and obligations that exist in a business relationship between two organizations: the service provider and customer".

A Service Level Agreement (SLA) is a formal, negotiated document that defines (or attempts to define) in quantitative (and perhaps qualitative) terms the service being offered to a Customer. It is a contract that exists between the Service Provider (SP) and the Customer. It is designed to create a common understanding about Quality of Service (QoS), priorities, responsibilities, etc. SLAs can cover many aspects of the relationship between the Customer and the SP, such as performance of services, customer care, billing, service provisioning, etc.

SLAs are used to identify parties who engage in the electronic business, computation, and outsourcing processes and to specify the minimum expectations and obligations that exist between parties [10]. However, although a SLA can cover such aspects, agreement on the level of service is the primary purpose of a SLA".

The SLA is established and commenced automatically when a customer requests service with confirmed payment. If any clauses in the SLA are violated, the penalty should be enforced, such as the granting of more credit for future services to the customer.

III. SERVICE LEVEL AGREEMENT (SLA) IN CLOUD COMPUTING

In Clouds, infrastructure, platform and application or software services are available on-demand and companies are able to access their business services and applications anywhere in the world whenever they need.

As consumers delegate their tasks to cloud providers, Service Level Agreements (SLA) between consumers and providers emerge as a key aspect. Due to the dynamic nature of the cloud, continuous monitoring on Quality of Service (QoS) attributes is necessary to enforce SLAs. Also numerous other factors such as reliability, quality of the services become important aspects and trust (on the cloud provider) come into consideration, particularly for enterprise customers that may outsource its critical data. This complex nature of the cloud landscape warrants a sophisticated means of managing SLAs.

Since the demands of the service consumers vary significantly, it is not possible to fulfil all consumer expectations from the service provider perspective and hence a balance needs to be made via a negotiation process. At the end of the negotiation process, provider and consumer commit to an agreement.

This SLA serves as the foundation for the expected level of service between the consumer and the provider. The QoS attributes that are generally part of an SLA (such as response time and throughput) however change constantly and to enforce the agreement, these parameters need to be closely monitored [11].

Furthermore, the advent of Grid computing reinforces the necessity of using SLA, specifically, in service-oriented commercial Grid computing where resources are advertised and traded as services based on an SLA after users specify various levels of service required for processing their jobs [9]. However, SLAs have to be monitored and assured properly.

The most concise SLA includes both general and technical specifications, including business parties, pricing policy, and properties of the resources required to process the service [12].

The SLA may specify the levels of availability, serviceability, performance, operation, or other attributes of the service, such as billing. For each one of these functional and not functional QoS requirements a Service Level Objective (SLO) is defined, that is a threshold on the value assumed by the QoS metrics/attributes (e.g., the minimum availability averaged over a given period, the average response time or a stricter requirement such as the 90-percentile of the response time or the tail of the response time distribution). Non-compliance to the agreement may incur in penalties to the service providers. Typically, the SLA contains also an insurance clause offering monetary compensation to the user if the provider fails to provide the service objectives according to the minimum levels specified in the SLA (e.g., Amazon EC2 or Google App Engine SLA policies). The service contracts in the cloud

International Journal of Advances in Engineering and Management (IJAEM)
Volume 2, Issue 7, pp: 425-432 www.ijaem.net ISSN: 2395-5252

environment typically differ from traditional SLA contracts for providing a more flexible service accounting scheme, typically referred to as pay-per-use or pay-as-you-go.

[9]defined the components of a typical SLA of a cloud provider:

- Service guarantee: This specifies the SLO that a provider has to meet over a service guarantee time period. Examples are Availability ≥ 0.99%; Throughput ≥ 100requests/s; AverageResponseTime ≤ 0.5ms.
- Service guarantee time period: This describes the duration over which a service guarantee should be met (e.g., one month, one day, one hour).
- Service guarantee granularity: This describes the resource scale on which a provider specifies a service guarantee (e.g., per service, per datacenter, per instance, per transaction).
- Service guarantee exclusions: This specifies the instances that are excluded from service guarantee metric calculation (e.g., downtime period caused by scheduled maintenance).
- Service Credit: This determines the amount credited to the customers (i.e. the penalty that the provider has to pay) if the service guarantee is not met.
- Service violation measurement and reporting: This describes how and who measures and reports the violation of service guarantee.

There two typical types of SLA are provider predefined (static) and negotiated SLAs. The provider predefined SLA provides a standard SLA template for all customers. For example, Amazon EC2 has a predefined static SLA. On the other hand, customers may have special QoS requirements which may not be included in a predefined SLA, in such case the customer and the provider will go through negotiation processes to achieve a mutually agreed SLA (Negotiated SLA).

IV. NEGOTIATION AND RENEGOTIATION

During the negotiation phase, both parties are using their knowledge and assumptions for maximizing their profit and the value of the SLA at hand. The exact utility function to be optimized may be different for each party in the negotiation, for each business domain that SLA negotiation may be applied. The service provider and the customer exchange messages in order to agree on a well-defined set of guarantees governing service consumption by the specific customer. Guarantees may refer to interdependent obligations of both parties. This may include, for instance, the minimum performance of the service (provider side) as long as the invocation rate remains under a certain threshold (customer side).

Table 1.1 The Negotiation States and Description Summary

States	**Description**
Propose	The agent propose initial or counter offer to the opponent agent.
Reject	The agent does not accept the offer proposed by the opponent agent.
Accept	The agent accepts the offer proposed by the opponent agent.
Failure	System failure, trigger renegotiation.
Terminate	Negotiation is terminated due to timeout or no mutual agreement.

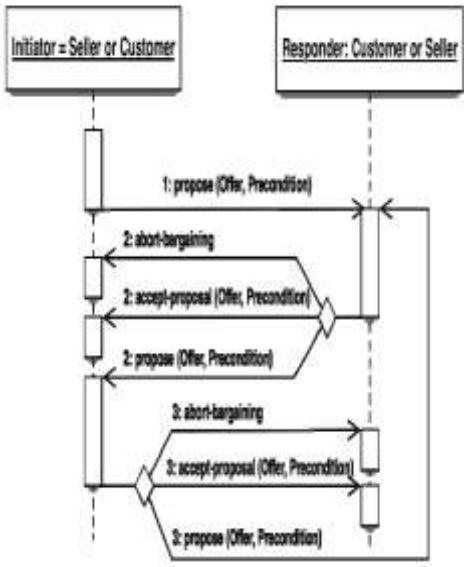

Figure 2: Negotiation process.

In figure 2 the sequential negotiation process is described as follows

Step 1: Initiator proposes a requests: Initiator who can either be the seller (provider) or Customer depending on who is proposing the offer first proposes an offer stating preconditions for the requests.

Step 2: The responder who can either be the seller (provider) or Customer evaluates the proposed request deciding what actions to take either to abort (reject) bargaining if the condition us not

International Journal of Advances in Engineering and Management (IJAEM)
Volume 2, Issue 7, pp: 425-432 www.ijaem.net ISSN: 2395-5252

favourable, accept proposal if condition is favourable or generate counter offer (i.e propose another offer for consideration).

Step 3: Depending on Step 2, if proposal is rejected or accepted then the process terminates else the negotiation process continues.

V. SLA MONITORING

SLA negotiation introduces two requirements for SLA monitoring:

1. The collections of SLA violations during the provisioning of a service under the terms of an SLA. On the Provider side, such violations should be made available as historical data to SLA negotiation, for optimization and planning while deciding whether to accept or not a SLA offer made by the customer;
2. The monitorability of the guarantee terms specified in a SLA offer madeby an agreement initiator to an agreement responder.This is necessary since auditing and enforcing an SLA that has non-monitorable guarantee terms would not be feasible.

The first of these requirements is a typical functional requirement for any generic software system monitoring component.

The second requirement, regarding the assessment of the monitorability of SLA terms before SLA establishment.

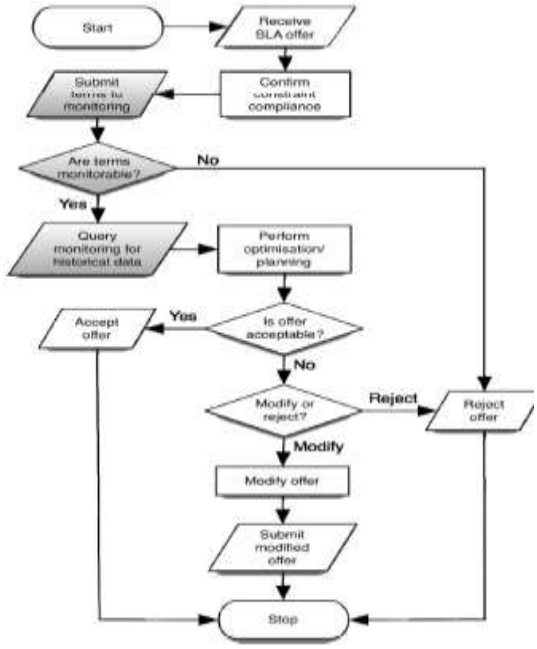

Figure 3: Negotiation from the provider's side

In figure 3, the sequential negotiation process is described as below: A provider

Step 1: receives SLA offer from a customer

Step 2: confirm constraint compliance

Step 3: Submit terms for monitoring

Step 4: evaluates if the terms are monitorable, if yes move to step 5 else terminate

Step 5: Query monitoring for historical data

Step 6: Performs optimization or planning on the offer received

Step 7: Evaluate whether the offer is acceptable if yes accept offer then terminate, else modify or reject offer

a. Components of a monitoring framework

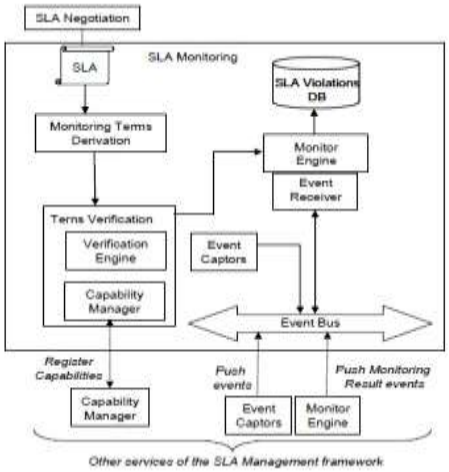

Figure 4: SLA monitoring architecture

The role and function of the components of a SLA monitoring framework are as follows:

Event Bus. The architecture of a SLA monitoring framework is event-based [13], i.e., it relies on capturing runtime information during SLA provisioning at the different services of the managed SBS by suitable event captors and making it available to different components of the monitoring framework as events. The exchange of events between the monitor and the event captors (internal to a node or from external nodes) is managed through an Event Bus that realizes a publish/subscribe architecture. In this architecture, event captors are event publishers and monitors are event subscribers and consumers. More specifically, event captors publish their events to the bus with appropriate tags enabling it to distribute them to monitors that have subscribed to them. Based on these events the monitors can detect violations of the terms of SLAs.

Monitoring Terms Derivation Module. The role of this module is to translate the agreed guarantee

International Journal of Advances in Engineering and Management (IJAEM)
Volume 2, Issue 7, pp: 425-432 www.ijaem.net ISSN: 2395-5252

terms of an SLA into specifications of patterns of events and computations over their features that can be checked at runtime. EC-Assertion was used to express the monitorable event patterns i.e., an XML language based on Event-Calculus. This is because the default monitor of the SLA monitoring framework is the EVEnt REaSoning Toolkit (EVEREST) that supports this language. However, the architecture of the monitoring framework allows the integration of other Monitoring Terms Derivation Modules to support different languages for expressing guarantee terms and monitors.

Monitor Engine. Monitoring service based systems has been an area of focus lately and several systems have been proposed for monitoring composite or atomic services [14], and service infrastructures (Ganglia : http://ganglia.info). In this approach, SLA monitoring in each node may adopt a different Monitor Engine. The logic implemented by the Monitoring Terms Derivation module will then change according to the kind of properties/rules required by the adopted Monitor Engine. Detected SLA violations are stored in the SLA Violations DB, which is queried by SLA Negotiation when historical data are required for accepting/refusing a SLA offer.

Terms Verification Module. This module implements the main functionality required for assessing terms' monitorability. It receives as input the Monitoring terms, as obtained from the translation made by the Monitoring Terms Derivation, and assesses whether the terms can be monitored through a call to the Capability Manager.

The provision of runtime events to the SLA Monitoring framework is based on Event Captors. Event Captors are able to capture events generated by the SLA provisioning environment, and may be implemented differently depending on the entity that they need to provide information for. Event captors may, for example, be realized as instrumented BPEL processes in the case of composite software services implemented by BPEL service coordination workflows, which during execution can emit the required events [14] and state of the executing workflow. Service invocations and matching responses are typical examples of events that can be captured at the BPEL process execution level. Such events are required, forinstance, for monitoring the Completion Time agreement. In other cases they may be realized as service container/proxies that capture service calls and responses [14].

Regardlessof their implementation, event captors need to timestamp the events that they generate and, depending on the consumer of these events, even synchronize their clocks with the clock of a reference monitor [7]. Time stamping is critical for monitoring SLAs as most of the terms in them need to be expressed in relation to time.

b. Monitoring capabilities and monitorability assessment

The assessment of the monitorability of SLA terms relies on the definition of the monitoring capabilities of each service involved in the SLA Management Framework. The Monitoring Capabilities of a service are defined as the collection of

(i) the Events that can be produced by its local Event Captors and
(ii) the Monitoring Result Events that can be produced by its Monitor Engine, that is, the kind of agreement terms a service may locally monitor if requested to do so.

The exchange of monitoring capabilities between two services in the SLA management framework is implemented as the exchange of (XML-based) monitoring capabilities documents among the Capability Managers of the two services.

When a service receives an SLA offer, the generated Monitoring Terms are submitted to the Terms Verification module. The Terms Verification module will retrieve the (hierarchically defined) monitoring capabilities from its Capability Manager. Then, for each term, the Terms Verification module verify whether

(i) events required for monitoring the term are available or
(ii) the monitoring of the term can be delegated to another service in the hierarchy.

In case (i), the term will be monitored locally by the service, consuming the required events that will be published on the bus by Event Captors (local and from other peer services). In case (ii), the monitoring of the term can be delegated to another service down the hierarchy. If the monitoring of a term cannot be performed locally, i.e. required events are not available according to the exchanged monitoring capability documents, or delegated to other services, the SLA monitoring will notify the SLA negotiation that the term cannot be monitored. Therefore, the agreement offer will be rejected (or modified for further negotiation steps).

International Journal of Advances in Engineering and Management (IJAEM)
Volume 2, Issue 7, pp: 425-432 www.ijaem.net ISSN: 2395-5252

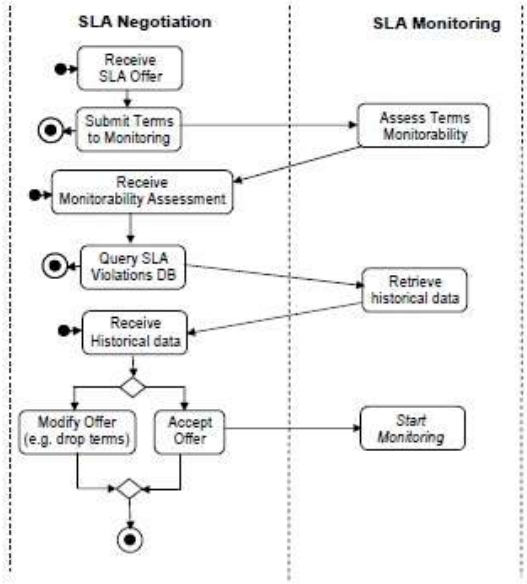

Figure 5: Interactions between SLA negotiation and monitoring

At runtime, when the SLA is provisioned, the Event Bus of the service will subscribe to the events required for monitoring or to the correspondent Monitoring Result event registered by other services, to which the monitoring of some terms has been delegated. A service's Monitor Engine, will then start receiving the events to which it has subscribed. Generic events are processed by the Monitor Engine to assess SLA violations, whereas Monitoring Result events are directly stored by the Event Receiver in the SLA Violations DB.

As a conclusion, Figure 5 explicates the negotiation time offer evaluation flow described in Figure 3, showing how SLA Negotiation acts as a client of SLA Monitoring, which exposes three atomicfunctionalities, i.e. VerifyMonitorability, Retrieve Historical Data, and Start

Monitoring. On the one hand, Verify Monitorability fulfills theneed for assessing the monitorability of agreement termsin an SLA offer, according to the exchange of monitoring capabilities. On the other hand, Retrieve Historical Data and Start Monitoring functionalities jointly fulfil requirement (1), i.e. makingmonitoring data available for the evaluation of SLAoffers. The former functionality, in particular, isimplemented by a set of queries that SLA negotiation mayrun on the SLA Violations DB.

VI. RELATED WORKS

Cloud computing is a paradigm of Service-Oriented utility computing.In Clouds, infrastructure, platform and application or software services are available on-demand and companies are able to access their business services and applications anywhere in the world whenever they need.

In Service-Oriented Architecture (SOA) terms, this agreement is referred to as a SLA. This SLA serves as the foundation for the expected level of service between the consumer and the provider. The QoS attributes that are generally part of an SLA (such as response time and throughput) however change constantly and to enforce the agreement, these parameters need to be closely monitored [4].

Furthermore, the advent of Grid computing reinforces the necessity of using SLA, specifically, in service-oriented commercial Grid computing where resources are advertised and traded as services based on an SLA after users specify various levels of service required for processing their jobs [15]. However, SLAs have to be monitored and assured properly.

SLAs are used to identify parties who engage in the electronic business, computation, and outsourcing processes and to specify the minimum expectations and obligations that exist between parties [16]. The most concise SLA includes both general and technical specifications, including business parties, pricing policy, and properties of the resources required to process the service [13].

Research related to SLA-based cost minimization and Customer Satisfaction Level (CSL) maximization for SaaS providers are still in their preliminary stages, and current research on Cloud computing [12,11,14] focus mostly on market oriented models for IaaS providers. Many authors do not consider customer driven resource management, where resources have to be dynamically reallocated according to the customer's on-demand requirements.

SLA negotiation and SLA monitoring have been heavily researched in the past, for what concerns runtime monitoring of service based systems (SBS), intrusive monitoring relies on alternating the execution of the service and monitoring activities at runtime. This can be done directly in the BPEL engine, interleaving monitoring code with the process executable code [10].

System properties' monitorability cannot be achieved with intrusive monitoring, since the properties to be monitored and the actions required for monitoring must be interleaved with service

execution code and, therefore, known a priori by the system designer. Non-intrusive monitoring [17, 10, 9] requires the establishment of mechanisms for capturing runtime information on service execution, e.g. service operation calls and responses. In this way, the business logic of the SBS process and the monitoring logic remain separate. The cited approaches to non-intrusive monitoring take for granted the availability of events required for monitoring and do not consider the issue of monitorability of rules/properties submitted to a generic monitor engine. The concept of local monitors attached to services has been introduced in [18].

However, the proposed approach considers the static allocation of properties monitoring based on a predefined service network topology.

A multitude of research papers discuss the topic of SLA negotiation with some reference to monitoring, but without exploring it explicitly in the context of a complete, multi-layer service economy. [19] is using a "Situation Assessment Module" to evaluate the feasibility of a SLA based on monitoring info, but only looking at isolated SLAs. Conversely, [20, 21] looked into SLA hierarchies and negotiation in this context, without any reference to consultation with monitoring though. [15] refers to using events for evaluating the validity of offers, but without further discussion on using monitoring for provider-side optimization of the negotiation process. [22] presented a negotiation framework and decision strategies are mentioned, but without any explicit links to monitoring information. Several projects have also focused on SLA definition, establishment, and provisioning both in the context of Web and Grid services.

The TrustCOM project looked deeply into the subject of SLA negotiation and monitoring, and also produced a reference implementation. However, SLA hierarchies and dependencies are not taken into account, and the problem is solved for isolated agreements only [23]. The same holds for AssessGrid, which concentrated on SLAs and risk management [24]. Also, AssessGrid has a focus on Grid computing, therefore assuming certain system organization and architecture, while our approach has a wider view on autonomic service providers and the respective service economies.

A. GENERAL COMMENT

As consumers delegate their tasks to cloud providers, Service Level Agreements (SLA) between consumers and providers emerge as a key aspect. Due to the dynamic nature of the cloud, continuous monitoring on Quality of Service (QoS) attributes is necessary to enforce SLAs. Also numerous other factors such as reliability, quality of the services become important aspects and trust (on the cloud provider) come into consideration, particularly for enterprise customers that may outsource its critical data. This complex nature of the cloud landscape warrants a sophisticated means of managing SLAs.

Since the demands of the service consumers vary significantly, it is not possible to fulfil all consumer expectations from the service provider perspective and hence a balance needs to be made via a negotiation process.

SLA negotiation and SLA monitoring have been heavily researched in the past, but the two research streams have usually been kept separated. In some cases, they have been brought together in more unified architectures, but never viewed in such a way where negotiation relies on monitoring and vice versa.

VII. CONCLUSION

After illustrating and analysing the explicit link between SLA negotiation and SLA monitoring, we showed why this relationship cannot be disregarded, especiallysince cloud consumers do not have control over the underlying computing resources, therefore cloud providers need to ensure the quality, availability, reliability, and performance of these resources when consumers have migrated their core business functions onto their entrusted cloud.

Hence it is vital for consumers to obtain guarantees from providers on service delivery and these are provided through Service Level Agreements (SLAs) negotiated between the providers and consumers.

REFERENCES

[1]. Yeo, C. S., and Buyya, R. (2006). A Taxonomy of Market-based Resource Management Systems for Utility-driven Cluster Computing. Software: Practice and Experience (SPE), 36 (13), (pp.1381-1419).

[2]. Jennings, N., Norman, T., Faratin, P., O'Brien, P., Odgers, B. (2000). Autonomous Agents For Business Process Management. Applied Artificial Intelligence, 14, 145-189.

[3]. Buco, J., Chang, N., Luan, Z., Ward, C., Wolf, L., and Yu, S. (2004). Utility Computing SLA Management based upon Business Objectives. IBM Systems Journal, 43(1), (pp. 159-178).

[4]. Mensce, D., and Almeida, V. (2002). Capacity Planning for Web Performance: Metrics, Models and Methods. Prentice-Hall, Upper Sadale River, NJ.
[5]. Ayatollahzadeh, M., Barfouroush, A. (2008). A Conceptual Framework for Modeling Automated Negotiations in Multiagent Systems. Negotiation Journal, 24(1), pp. 45-70.
[6]. Snelling, D., Anjomshoaa, A., Wray, F., Basermann, A., Fisher, M., Surridge, M., Wieder. P. (2007). NextGRID Architectural Concepts. Towards Next Generation Grids: Proc. CoreGRID Symposium.
[7]. Van der Aalst, W., Dumas, M., Ouyang, C., Rozinat, A., and Verbeek, E. (2008). Conformance checking of Service Behavior, ACM TOIT, 8 (3).
[8]. Padgett, J., Gourlay, I., Djemame, K. (eds). AssessGrid D1.3: System Architecture Specification and Developed Scenarios (v0.30). December 2006.
[9]. Yang F., Zhang, Y., Wu L., Liu L., and Liu J. (2011). A Hybrid Approach to Placement of Tenants for Service-Based Multi-tenant SaaS Application. Proceedings of the 6th IEEE Asia-Pacific Services Computing Conference, Korea.
[10]. Brzostowski, J., Chhetri, M., Kowalczyk, R. (2007). Three Decision-making Mechanisms to facilitate Negotiation of Service Level Agreements for Web Service Compositions. Proc. Joint Conference of the INFORMS Section on Group Decision and Negotiation, pp. 37-44.
[11]. Moser, O., Rosenberg, F., and Dustdar, S. (2008). Non-intrusive monitoring and service adaptation for WS-BPEL, WWW.
[12]. Somefun, D., Gerding, E., Bohte, S., La Poutré, J. (2004). Automated Negotiation and Bundling of Information Goods. Agent-Mediated Electronic Commerce V, pp.1-17.
[13]. Spanoudakis G., Mahbub K. (2006). Non-Intrusive Monitoring of Service Based Systems, International Journal of Cooperative Information Systems, 15 (3), pp. 325-358.
[14]. Barbon, F., Traverso, P., Pistore, M., Trainotti, M. (2006). Run-Time Monitoring of Instances and Classes of Web Service Compositions, Proc. IEEE ICWS.
[15]. Baresi, L., and Guinea, S. (2005). Towards Dynamic Monitoring of WS-BPEL Processes, Proc. ICSOC.
[16]. Baset, S. (2012). "Cloud slas: Present and future," SIGOPS Oper. Syst. Rev., vol. 46, no. 2, pp. 57–66. [Online]. Available: http: //doi.acm.org/10.1145/2331576.2331586
[17]. Jank, K. (2005). Reference Architecture. Adaptive Services Grid Deliverable D6.V-1.
[18]. Keller, A., Ludwig, H. (2003). The wsla framework: Specifying and monitoring service level agreements for web services. J. Netw. Syst. Manage. 11(1) (2003) 57{81
[19]. Kloukinas, C., Spanoudakis, G., and Mahbub, K. (2008). Estimating Event Lifetimes for Distributed Runtime Verification, Proc. SEKE.
[20]. Machiraju, V., Sahai, A., and van Moorsel, A. (2003). Web Services Management Network: An Overlay Network for Federated Service Management, Proc. IFIP/IEEE 8th Int. Symposium on Integrated Management.
[21]. Nitto, E., Di Penta, M., Gambi, A., Ripa, G., Villani, M. (2007). Negotiation of Service Level Agreements: An Architecture and a Search-Based Approach. Proc. ICSOC, pp. 295-306.
[22]. Spanoudakis G, Kloukinas C. Mahbub K. (nd). The SERENITY Runtime Monitoring Framework, In Security and Dependability for Ambient Intelligence, Information Security Series, Springer, pp. 213-238 (to appear)
[23]. The TrustCOM project. (2007). Deliverable 64: Final TrustCoM Reference implementation and associated tools and user manual, (v3.0).
[24]. Tserpes, K., Kyriazis, D., Menychtas, A., Varvarigou, T., Silvestri, F., Laforenza, D. (2007). An Open Architecture for QoS Information in Business Grids. Towards Next Generation Grids: Proc. CoreGRID Symposium.