A review of SLA specification languages in the Cloud Computing

Adil Maarouf*[1], Abderrahim Marzouk[1], Abdelkrim Haqiq[1,2]
[1]*Computer, Networks, Mobility and Modeling laboratory*,
FST, Hassan 1st University,
Settat, Morocco
[2]e-NGN research group, Africa and Middle East
E-mails: adil.maarouf@gmail.com, amarzouk2004@yahoo.fr, ahaqiq@gmail.com

***Abstract*—A Service Level Agreement (SLA) represents an agreement between a cloud provider and cloud customer in the context of a particular service provision. This raises the following question: How to describe the SLA terms between prospective signatories, such as service levels, Quality of Service properties, penalties in case of SLA violation, etc. SLA can be represented by specialized languages for easing SLA preparation, automating SLA negotiation, adapting services automatically according to SLA terms, and reasoning about their composition. Therefore, the aim of this work is to present a comprehensive review of how SLAs are created, managed and used in web services and cloud computing environment. This paper provides a review of SLA languages specification. Among them the WSLA, WS-Agreement, SLA*, CSLA and SLAC. Then, a comparison of these languages in terms of the requirements is presented, highlighting their strengths and weaknesses.**

Keywords-Cloud Computing; Service Level Agreement; SLA specification languages; WSLA; WS-Agreement; SLA*; CSLA; SLAC

I. Introduction

Actually, Service Level Agreements (SLAs) play a pivotal role in Cloud Computing, the current technological evolution of information and communications technology. The revolutionary technology of Cloud Computing offers a scalable and flexible paradigm where infrastructure, platform, and software are offered to users in the form of services [1] [2]. The provisioning of these computing services by Cloud providers are regulated by SLAs [3].

Service Level Agreements are one of the most common approaches for specifying some form of mutual understanding about business transactions between a Cloud provider (seller) and a Cloud consumer (buyer) in the software and telecommunications domain. An SLA is a representation of all features (including the functionality delivered by the service and the quality that the buyer experiences) a consumer should expect to receive by a service [3]. Thus, an SLA represents functional and non-functional properties of services and serves as a way for controlling and managing these properties. Typically, an SLA is a bilateral binding statement signed between a service provider and a service consumer, over the agreed terms and conditions of the given service [4]. An SLA also sets out the remedial action and any penalties that could take effect if performance falls below the promised standard. SLAs contain Quality of Service properties that must be maintained by a provider, generally defined as a set of Service Level Objectives (SLOs). These properties need to be measurable and must be monitored during the provision of the service that has been agreed in the SLA [5]. The SLA must also contain a set of penalty clauses specifying what happens when service providers fail to deliver the pre-agreed quality. This raises the following question: How to describe the SLA terms between prospective signatories, such as service levels, Quality of Service properties, penalties in case of SLA violation, etc.

SLA can be represented by specialized languages for easing SLA preparation, automating SLA negotiation, adapting services automatically according to SLA terms, and reasoning about their composition. Therefore, the aim of this work is to present a comprehensive review of how SLAs are created, managed and used in web services and cloud computing environment. This paper provides a review of SLA languages specification. Among them the WSLA, WS-Agreement, SLA*, CSLA and SLAC. Then, a comparison of these languages in terms of the requirements is presented, highlighting their strengths and weaknesses.

The rest of this paper is organized as follows: Section II provides an overview of SLA specification languages. Section III presents a comparative study of the SLA definition languages, characteristics and limitations. Section IV discusses the related work. Finally, section V provides concluding remarks on the paper.

II. An overview of SLA specification languages

Until a few years ago, SLA contracts were mostly written using natural expressions; examination of compliance to the agreement also had to be done manually [6]. One attempt to facilitate this process by using SLA templates was limited and unable to specify different service levels for different customers [6]. For this reason, it has become a necessity to automate the procedure through which different SLAs are flexibly described, provisioned and observed [6].

Several SLA specification languages have been developed by researchers within the service provision community to address the previous aspects [6]. Their aim is to simplify the contractual process for the parties involved and to minimize the time and cost included in this process. Many projects and solutions focus on the definition of SLA [7] [8] [9] [10] [11] [12]. We report here only work directly related to the Web services and Cloud services.

In the following subsections, WSLA, WS-Agreement, SLA*, CSLA and SLAC are described in details. Then, a comparison of these languages in terms of the requirements of the proposed methodology is presented.

A. SLA @ Web services

There are two main specifications for describing a SLA for web services. In what follows we introduce WSLA and WS-Agreement which represent the most successful results.

1) Web Service Level Agreement (WSLA): WSLA [13] was introduced by IBM research as a framework for the definition and monitoring of SLAs for web services. The WSLA framework provides a formal language, an XML schema and a run-time architecture that interprets the language and handles SLA management tasks. According to WSLA [13], service level agreements are contracts between a service provider and a customer that specify the agreed QoS and define service level objectives (SLOs). An SLA may be monitored by either or both signatories. An SLA depicts anticipated service levels with respect to business and technical level objectives. The same service may be offered in diverse levels. WSLA describes all the aspects that are contracted between the signatory parties, the service supplier and the service consumer, regarding a specific service. The negotiation process for establishing such a document could be accomplished either on-line or off-line through the use of a WSLA template that comprises most of the defined and the agreed upon information needed to create the SLA contract [6].

WSLA represents, in its agreement, all the information that is normally contained within an SLA document. This information is situated in the following three main parts:

- **Parties:** contains information about the parties engaged in the SLA contract. These are the signatory parties, the service provider and the service consumer, in addition to the supporting parties that may be involved in measuring, monitoring or managing particular parts of the contract [13] [6].
- **Service Description:** contains an explanation of the Service Object on which the QoS metrics are defined, its SLAParameter(s), and what Metric(s) are used to compute their values. A metric will be measured from a source by identifying a measurement directive, if it is basic. However, in the case of a composite metric, its value will be computed using a function that takes other metrics or constants as its operands. The SLA parameters are one of the important parts of an SLA because they correlate the metrics to a particular consumer with specific accepted values [13] [6].
- **Obligations:** contains Service Level Objectives the service provider is obliged to maintain. These are the agreed values of SLA parameters during a specific duration and the actions to be taken in the case of a contract violation [13] [6].

Figure 1 illustrates the major components of WSLA in a UML class diagram. The latter offers the possibility to enrich the directory of performance criteria by creating new metrics. The latest version of the WSLA specification was published in 2003 [13]. Since then, WSLA is integrated into WS-Agreement. Despite specification discontinuation and latest version semi-stable in 2003, WSLA remains one of the most mature defining a SLA specifications. This specification covers the entire life cycle of SLA. However, the management of penalty is limited mainly to the notification in case of violation.

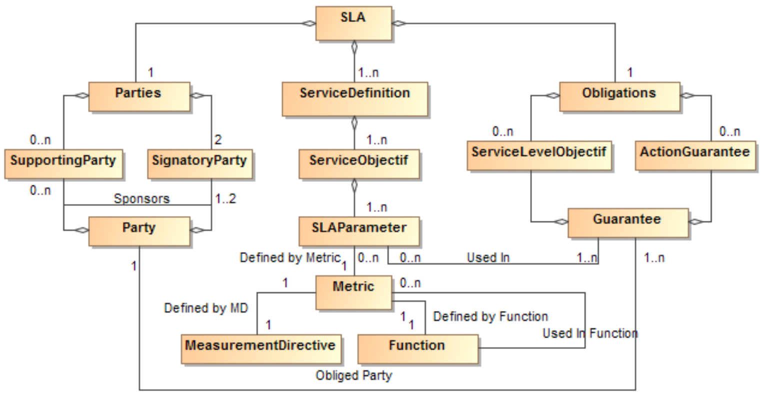

Fig. 1: WSLA Meta-model [14]

2) Web Service Agreement Specification (WS-Agreement): WS-Agreement is an SLA specification presented by the Grid Resource Allocation and Agreement Protocol Working Group of the Compute Area of the Open Grid Forum [15]. It is an XML-based language and web service protocol for: firstly, promoting through templates a set of possible accepted agreement offers from agreement responders (which could be the service provider or consumer); secondly, generating a proposed agreement offer that the agreement initiator is keen to establish based on one of these templates; thirdly, negotiating this agreement according to specific constraints; fourthly, creating an agreement between the service provider and customer with all conditions and restrictions, and then finally observing its fulfillment [15]. The WS-Agreement is structured in three parts [15] as follows:

1. *Agreement Schema* (agreement creation offer schema): This is used by the agreement initiator to create an offer according to a specific template. The agreement creation offer and the agreement are structurally the same. The agreement offer, contains the agreement Name, the Context (this includes the involved parties and the agreement life span), and the Terms, which is the most important part of an agreement offer. Within the Terms, each term contains at least one Service Term and zero or more Guarantee Terms which could be combined using logical operators.

2. *Agreement Template Schema* : This is used by the agreement responder to promote acceptable agreement offers.

3. *Set of ports type and operations*: These are used for organizing and administrating the agreement life-cycle operations,

such as accepting or rejecting the offer, or identifying a type of a document for monitoring the agreement states.

Figure 2 illustrates the basic UML metamodel structure of an agreement according to WS-Agreement.

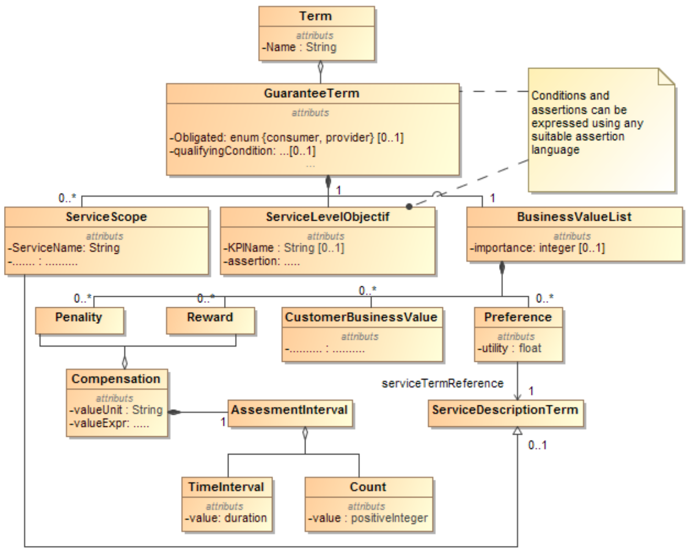

Fig. 2: WS-Agreement Meta-model [16]

One of the main strengths of WS-Agreement is that it is powerful in terms of extending new elements that are domain-specific [15]. In addition, it contains more information about the service functional's properties than WSLA does. Also, business values related to QoS metrics can be specified even if an accounting procedure is not supported [17]. WS-Agreement does not support a constructive ontology to define QoS metrics which is necessary to understand their exact definitions. Also its semantics are not defined precisely. However, WS-Agreement has recently been extended with SWAPS Extension [15] to overcome this semantic ambiguity. Oldham et al [18] present the Semantic WS-Agreement Partner Selection (SWAPS). This approach covers how to overcome limitations of WS-Agreement with respect to syntactical matching; it enables intelligent partner selection based on SLA technologies. These limitations exist due to the scantly defined XML domain vocabulary of WS-Agreement. The main focus of SWAPS is on closing these gaps, by adding more structure to the original specification, to enable automatic parsing and reasoning.

B. SLA @ Cloud

Regarding cloud services, NIST [19] proposed a definition of SLA. However, no attempt has been made to implement this definition until now. Similarly, Cloud Standards Customer Council [20] presented a practical guide to SLA which is a collaborative effort that brings together different experiences based on the client. Thus, Rackspace [21] defined a set of metrics through cloud monitoring to establish an SLA contract between two players Cloud. In addition the industrial solutions, several academic projects have been developed to meet the proposed limits by cloud services. In particular the project SLA@SOI offers the new language SLA* [22]. In what follows, we describe two major technologies SLA* and CSLA that consider SLA in Cloud environments.

1) SLA:* The SLA* [22] language is part of the SLA@SOI project, which aims at providing predictability, dependability and automation in all phases of the SLA life cycle. SLA@SOI propose SLA* [22] an abstract syntax for SLA. SLA*, is a domain-independent syntax for machine-readable Service Level Agreements and SLA templates [22]. Historically, SLA* was developed as a generalization and refinement of the web-service specific XML standards: WS-Agreement, WSLA, and WSDL. Instead of web services, however, SLA* deals with services in general, and instead of XML, it is language independent. SLA* provides a specification of SLA(T) content at a fine-grained level of detail, which is both richly expressive and inherently extensible: supporting controlled customization to arbitrary domain-specific requirements. The model was developed as part of the FP7 ICT Integrated Project SLA@SOI, and has been applied to a range of industrial use-cases, including; ERP hosting, Enterprise IT, live-media streaming and health-care provision. At the time of writing, the abstract syntax has been realised in concrete form as a Java API, XML-Schema, and BNF Grammar.

The solution is very expressive and extensible. SLA* promote the formalization of SLA in any language for any service by eliminating the restriction of XML. According to SLA@SOI, SLA is an SLA template with an extended attribute specifying the validity of the SLA contract. An SLA template contains five sections see figure 3: i) attributes template SLA, ii) the parties to the agreement, iii) service descriptions, iv) variable declarations, and v) the terms of the agreement, specifying the guarantees of Quality of Service. The parties in the abstract syntax of SLA(T) are identified by their role (supplier, customer). The description of services is defined by the statements of the interface. A declaration interface is used to assign an identifier to a local interface which may be a functional interface or a description of a resource. Variable declarations are provided to improve readability and avoid duplication of content. The terms of the agreement are formalized as guarantees of two types: Action warranty and status. In addition the simple expressions (for example, A constant, or a linear proportionality) SLA* propose a formal model to formalize penalties. The main motivation of this proposal is the opening and applicability to different domains without dependency on specific languages, taxonomies or technologies.

2) Cloud Service Level Agreement (CSLA): An SLA language to finely express SLA and address SLA violations in the context of Cloud services [23]. CSLA language has been influenced by related work, in particular WSLA and SLA@SOI [13], [22]. It allows to describe the SLA between a cloud service provider and a cloud customer. Thus, it describes QoS guarantees in the form of SLO clauses. Besides the standard formal definition of contracts comprising validity, parties, services definition and guarantees, CSLA [24] is enriched with features (QoS/functionality degradation and an

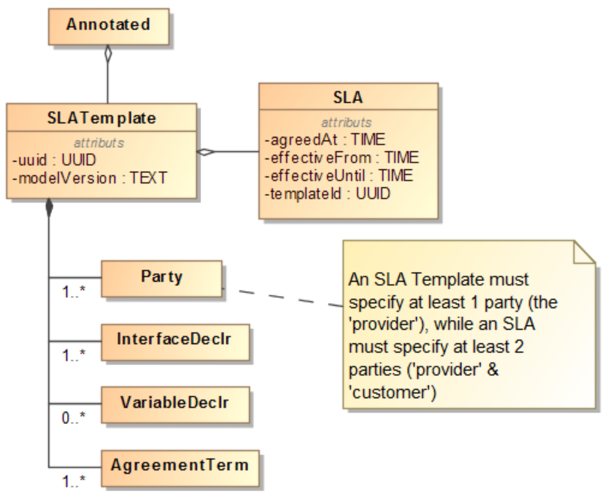

Fig. 3: SLA* (SLA@SOI) Meta-model [16]

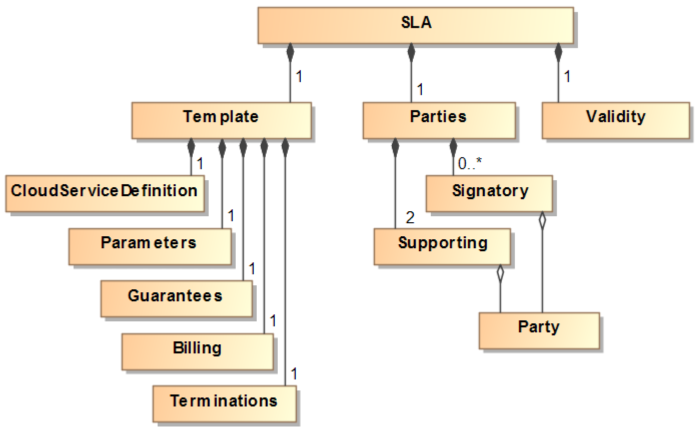

Fig. 4: CSLA Meta-model [16]

advanced penalty model) introducing a fine language support for Cloud elasticity management.

CSLA allows to define SLA in any language for any Cloud service (XaaS). CSLA addresses intrinsically (i) QoS uncertainty in unpredictable and dynamic environment; (ii) the cost model of Cloud computing. Its structure is similar to WS-Agreement. and

Conceptual Model : A SLA in the CSLA language contains three sections see figure 4: a section describing the validity of the agreement, a section defining the parties involved in the agreement and a section referencing the template used to create the agreement. The validity defines how long an agreement is valid. CSLA distinguishes two types of parties: Signatory parties, namely service provider and service customer, and Supporting parties (e.g., trusted third party). A CSLA template is like a pattern for SLA. It contains five elements: Cloud services definition, Parameters, Guarantees, Billing and Terminations. *A Cloud service definition* refers to any XaaS service (SaaS, PaaS or IaaS). In order to deal with unpredictable and dynamic environment, we propose the concept of functionality degradation (i.e., normal vs degraded mode) for each SaaS and PaaS services (e.g., 3D vs 2D display). *Parameters* provide a way to define variables in the context of the agreement. *Guarantees* contain a set of guarantees. Each guarantee consists of four elements: Scope, Requirements, Terms and Penalties. CSLA supports two types of *billing*: Pay as You Go and All-in package. Finally, the Agreement starts on effectiveFrom and ends on the earlier of the effectiveUntil or in accordance with *Terminations* section.

3) Service-Level-Agreement for Clouds (SLAC): SLAC [25] is a language to define SLAs specifically devised to the cloud computing domain, which is inspired by WS-Agreement and shares many features with the definitions and structure of this language. SLAC was defined as a domain specific language for clouds. The metrics available in the language are pre-defined in the light of the requirements of the cloud domain. The set of characteristics of the SLAC language, such as multi-party, group definition and specification of the involved parties on each term, enables and leverages the definition of SLAs including a broker [25].

This core-language provides the basic ingredients that enable the description of a simple SLA for the cloud computing domain. Differently from most of the existing SLA definition languages, SLAC does not differentiate service description terms and quality requirements. The formal definition of the syntax of the core language is defined in the Extended Backus Naur Form (EBNF) [26].

The semantics of the evaluation of a SLA is formulated as a Constraint Satisfaction Problem (CSP) that verifies: (i) at negotiation-time, whether the terms composing the agreement are consistent; and (ii) at enforcement time, whether the characteristics of the service are within the specified values. More specifically, the formal semantics of SLAC is given in a denotational style [26]. Denotational semantics is based on the recognition that the elements of the syntax are symbolic realizations of abstract mathematical objects or functions. Although originally developed to describe the behaviour of imperative programming languages, the denotational approach is also appropriate for declarative languages (as required for SLAs) due to its capacity to translate the static and dynamic elements of the domain. This use of the semantics provides the formalism that is needed to express the meaning of the elements of a SLA definition language.

Finally, it is worth noticing that SLAC focusses on Infrastructure-as-a-Service (IaaS) [25]. Since most requirements of the other service models are also taken into account, we do not envisage any major issue in extending SLAC to cover them, especially the case of Platform-as-a-Service (PaaS). We plan to specifically address the other models in future works.

III. Comparative study of the SLA definition languages

In this section we present a comparative study of SLA languages highlighting their strengths and weaknesses. Table I shows the comparison between the SLA definition lan-

		WSOL	SLAng	WSLA	WS-Agreement	SLA*	CSLA	SLAC
General	Cloud Domain	-	-	-	-	□	■	■
	Cloud Service Models	-	-	-	-	□	■	□
	Multi-Party	-	-	-	-	-	■	■
	Broker Support	-	-	-	-	-	□	■
	Ease of Use	□	□	-	□	□	■	□
Business	Business Metrics	□	□	□	□	□	■	■
	Price schemes	□	-	□	□	□	■	■
Formal	Syntax	■	■	■	■	■	■	■
	Semantics	-	□	-	-	-	■	■
	Verification	-	□	-	-	-	■	■
Tools	Evaluation	■	■	■	■	■	■	■
	Open-Source	■	■	-	■	■	■	■

TABLE I: Comparison of the SLA languages according to the domain requirements [25].

guages according to the domain requirements. For the sake of simplicity and standardization, we classify the languages in three levels: the ■symbol represents a feature covered in the language, □stands for a partially covered feature and - represents a not covered feature [25].

- *Cloud Domain* - This criterion defines when a language was devised for the cloud domain. Languages for a broader domain (e.g. IT Services) are not considered to cover the criterion [26];
- *Cloud Delivery Models* - Fully supported when a language covers IaaS, PaaS and SaaS, and partially supported when only one of these models is covered;
- *Multi-party* - Fully supported when the language enables the use of: (i) all roles defined; (ii) multiple-parties with the same role; and (iii) multiple roles for the same party [26]. Partially supported when at least one of the characteristics is present;
- *Broker support* - To support the broker, a SLA definition language should enable the specification of: (i) the parties involved in each term, which define the parties in charge of providing and consuming the service; (ii) offers and requests [26], which enable the creation of specification of services needed by consumers and of services available by providers; (iii) the role of the broker in the partys definition; and (iv) the actions and metrics which enable the parties to state who is the responsible for the service, who consumes it, who pays for it and the parties involved in each action. When at least two of these characteristics are supported we determine that the language partially fulfils this criterion;
- *Ease of Use* - This criterion is considered fully supported when a language is ready to be used in the domain [25](without or with minimal extensions) and, at the same time, it is easy to understand, also for human actors. If a language has only one of these characteristics it is considered as partially supported;
- *Business Metrics and Actions* - Fully supported when the language provides metrics related to the business aspects, such as performance indicators (KPIs) and enables the use of actions which express common business behaviours, for example the payment of penalties. It is partially supported if at least one of them is available in the language [26];
- *Price schemes* - The language should consider flat and variable pricing schemes and the main models of the domain, which are fixed pricing, bilateral agreement, exchange, auction, posted price and tender. A language offers partial support when it includes at least one type of variable pricing;
- *Formal Syntax* - Supported only when a formal definition of the syntax, e.g using BNF [26], is available;
- *Formal Semantics* - Fully supported if a formal definition of the SLA evaluation process is available. By formal we intend the use of any recognized formal tool (e.g. operational or denotational semantics) to avoid ambiguity on its interpretation. It is partially supported if possibly ambiguous tools are used in its definition;
- *Formal Verification* - Fully supported when the formal verification of agreement exists before execution time. We consider as partially supported if only the syntax is verified;
- *Evaluation Tools* - A language fully supports it when an implementation of a tool which evaluates the SLAs against the services monitoring information is available. If only design time verification is available this characteristic is only partially supported;

Although we specify the criteria for the comparison of the languages, their classification is subjective since they are not quantitative [26]. However, this comparison provides the base to understand to which extent the existing languages support the domain. The results of the comparison among the language show that the existing languages do not support some of the main requirements of the domain.

IV. Related works and discussion

The Web services community has performed significant level of research in SLAs languages. Several languages, such as SLAng [27], a domain specific language for IT services; and WSOL, WSLA [13] and WS-Agreement [15] have been proposed for SLA specification using a XML-based language. WS-Agreement is the most flexible language in the level definition of SLA. This means that users are free to define their terms as they want, but instead, a problem of automation is put seen that there is no common definition. All these works have contributed significantly to the emergence of SLA. However, none meets the needs for Cloud computing environment and particularly they do not address the SLA violations in this kind of unpredictable and dynamic environment.

More recently, initiatives such as SLA@SOI [28] or Optimis [8] have addressed SLA specification for Clouds. The SLA@SOI [28] language (SLA*) is based on the WS-Agreement while the Optimis language (WSAG4J) is a full Java-based implementation of WSAgreement and WS-Agreement Negotiation. Their solutions cover SLA lifecycle. However, the violations management does not reflect Cloud characteristics, the model does not support multi-party agreements, does not provide the formal semantics of the language and requires the definition of the vocabulary for the cloud domain. In addition, in SLA*, the description of SLA is just limited to guarantee terms while external files are needed (e.g., Open Virtualization Format) is needed to describe IaaS services. The CSLA [23] language shares motivations with the SLA@SOI project and goes further by taking into account the cross-layer nature of Cloud and proposes the definition of SLAs using a mechanism to support fuzziness in the numeric specification of metrics. However, this approach does not consider multi-party agreements, the deployment models (e.g. community cloud) or the prominent role of broker in the domain. Moreover, the support of CSLA for the delivery models is restricted to a few metrics and terms, which also requires extensions for real-world deployment.

V. Conclusion

It's true that research on the SLA languages for cloud computing is not a new topic but it is still in infancy state. Research work is booming in this field. SLA specification languages are crucial in the definition of the SLA contract. In this paper we discussed significant works in the field of the SLA definition languages (WSLA, WS-Agreement, SLA*, CSLA and SLAC) for web services and cloud computing. Moreover, we presented a detailed review of the characteristics for each language. Also, we presented a comparative study of these languages in terms of the requirements, highlighting their strengths and weaknesses. The results of the comparison among these languages show that the more appropriate language is CSLA which proposes new features related to services functionality/QoS degradation and an advanced penalty model that allow service providers to maintain its consumers satisfaction while minimizing the service costs due to resources fees.

References

[1] A. Maarouf, A. Marzouk, A. Haqiq, and M. El Hamlaoui, “Towards a mde approach for the establishment of a contract service level monitoring by third party in the cloud computing,” in *Proceedings of the 2014 Tenth International Conference on Signal-Image Technology and Internet-Based Systems*, ser. SITIS '14. Washington, DC, USA: IEEE Computer Society, 2014, pp. 715–720. [Online]. Available: http://dx.doi.org/10.1109/SITIS.2014.30

[2] A. Maarouf, A. Marzouk, and A. Haqiq, “Automatic control of the quality of service contract by a third party in the cloud computing,” in *Complex Systems (WCCS), 2014 Second World Conference on*, Nov 2014, pp. 599–603.

[3] A. Leff, J. T. Rayfield, and D. M. Dias, “Service-level agreements and commercial grids,” *IEEE Internet Computing*, vol. 7, no. 4, pp. 44–50, 2003.

[4] K. Stanoevska-Slabeva, T. Wozniak, and S. Ristol, *Grid and cloud computing: a business perspective on technology and applications*. Springer Science & Business Media, 2009.

[5] A. Maarouf, A. Marzouk, and A. Haqiq, “Towards a trusted third party based on multi-agent systems for automatic control of the quality of service contract in the cloud computing,” in *Electrical and Information Technologies (ICEIT), 2015 International Conference on*. IEEE, 2015, pp. 311–315.

[6] A. Keller and H. Ludwig, “The wsla framework: Specifying and monitoring service level agreements for web services,” *Journal of Network and Systems Management*, vol. 11, no. 1, pp. 57–81, 2003.

[7] R. G. Cascella, L. Blasi, Y. Jegou, M. Coppola, and C. Morin, *Contrail: Distributed application deployment under SLA in federated heterogeneous clouds*. Springer, 2013.

[8] W. Ziegler, M. Jiang, and K. Konstanteli, “Optimis sla framework and term languages for slas in cloud environment,” *OPTIMIS Project Deliverable D*, vol. 2, 2011.

[9] D. Petcu, B. Di Martino, S. Venticinque, M. Rak, T. Máhr, G. E. Lopez, F. Brito, R. Cossu, M. Stopar, S. Šperka *et al.*, “Experiences in building a mosaic of clouds,” *Journal of Cloud Computing*, vol. 2, no. 1, pp. 1–22, 2013.

[10] E. Kamateri, N. Loutas, D. Zeginis, J. Ahtes, F. DAndria, S. Bocconi, P. Gouvas, G. Ledakis, F. Ravagli, O. Lobunets *et al.*, “Cloud4soa: A semantic-interoperability paas solution for multi-cloud platform management and portability,” in *Service-Oriented and Cloud Computing*. Springer, 2013, pp. 64–78.

[11] T. Ledoux and Y. Kouki, “Sla-driven capacity planning for cloud applications,” in *Proceedings of the 2012 IEEE 4th International Conference on Cloud Computing Technology and Science (CloudCom)*. IEEE Computer Society, 2012, pp. 135–140.

[12] J. Carpentier, J. Gelas, L. Lefevre, M. Morel, O. Mornard, and J. Laisne, “Compatibleone: Designing an energy efficient open source cloud broker,” in *Cloud and Green Computing (CGC), 2012 Second International Conference on*. IEEE, 2012, pp. 199–205.

[13] H. Ludwig, A. Keller, A. Dan, R. P. King, and R. Franck, “Web service level agreement (wsla) language specification,” *IBM Corporation*, pp. 815–824, 2003.

[14] P. Patel, A. H. Ranabahu, and A. P. Sheth, “Service level agreement in cloud computing,” 2009.

[15] A. Andrieux, K. Czajkowski, A. Dan, K. Keahey, H. Ludwig, T. Nakata, J. Pruyne, J. Rofrano, S. Tuecke, and M. Xu, “Web services agreement specification (ws-agreement),” in *Open Grid Forum*, vol. 128, 2007.

[16] Y. Kouki, “Approche dirigée par les contrats de niveaux de service pour la gestion de l'élasticité du'nuage',” Ph.D. dissertation, Ecole des Mines de Nantes, 2013.

[17] P. Karaenke and S. Kirn, “Service level agreements: An evaluation from a business application perspective,” in *Proceedings eChallenges e-2007 Conference*, vol. 44, 2007.

[18] N. Oldham, K. Verma, A. Sheth, and F. Hakimpour, “Semantic ws-agreement partner selection,” in *Proceedings of the 15th international conference on World Wide Web*. ACM, 2006, pp. 697–706.

[19] P. Mell and T. Grance, “The nist definition of cloud computing (draft),” *NIST special publication*, vol. 800, no. 145, p. 7, 2011.

[20] C. S. Council, “Practical guide to cloud service level agreements version 1.0,” 2012.

[21] “Rackspace,” http://www.rackspace.com/cloud/, 2013.

[22] K. T. Kearney, F. Torelli, and C. Kotsokalis, “Sla: An abstract syntax for service level agreements,” in *Grid Computing (GRID), 2010 11th IEEE/ACM International Conference on*. IEEE, 2010, pp. 217–224.

[23] Y. Kouki, F. A. d. Oliveira, S. Dupont, and T. Ledoux, “A language support for cloud elasticity management,” in *Cluster, Cloud and Grid Computing (CCGrid), 2014 14th IEEE/ACM International Symposium on*. IEEE, 2014, pp. 206–215.

[24] D. Serrano, S. Bouchenak, Y. Kouki, T. Ledoux, J. Lejeune, J. Sopena, L. Arantes, and P. Sens, “Towards qos-oriented sla guarantees for online cloud services,” in *Cluster, Cloud and Grid Computing (CCGrid), 2013 13th IEEE/ACM International Symposium on*. IEEE, 2013, pp. 50–57.

[25] R. B. Uriarte, F. Tiezzi, and R. D. Nicola, “Slac: A formal service-level-agreement language for cloud computing,” in *Proceedings of the 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing*. IEEE Computer Society, 2014, pp. 419–426.

[26] R. B. Uriarte, “Supporting autonomic management of clouds: Service-level-agreement, cloud monitoring and similarity learning,” 2015.

[27] D. D. Lamanna, J. Skene, and W. Emmerich, “Slang: a language for service level agreements,” 2003.

[28] “Sla@soi,” http://www.sla-at-soi.eu/, 2013.