Adaptable Service Level Objective Agreement (A-SLO-A) for Cloud Services

Stefan Frey, Claudia Lüthje, Ralf Teckelmann and Christoph Reich
Furtwangen University, Faculty of Computer Science, Furtwangen, Germany

Keywords: SLA, Cloud Computing, Adaptive SLAs, Adaptable SLA Language Format.

Abstract: Reducing IT costs by using Cloud Computing is tempting for start up companies. To attract companies to outsource their services to Clouds, Cloud provider need to offer Service Level Objectives specified in SLAs individually for their customers. Cloud provider like Amazon can not afford to negotiate individual SLAs manually. Therefore, it becomes important to develop a format for machine-readable SLAs which can easily adapt to the individual Service Level Objectives requested by the customer any time. Because of its adaptability at run time by each individual customer on demand, this comply with the characteristics of Cloud Computing and to satisfies the customer’s requirements to be flexible. This paper describes an adaptable Service Level Objective Agreement (A-SLO-A) format being machine-processed to offer the possibility to integrate the SLA management into the highly automated processes of resource provisioning. Use cases show its applicability.

1 INTRODUCTION

Cloud providers do offer guarantees for QoS characteristics like, bandwidth, data backup, etc on a best-effort principal. But business requires QoS, monitoring and control of the Cloud services at any time, as stated in the “Architecture of Managing Clouds” (Distributed Management Task Force, 2010) and others e.g. Study Group Report of Cloud Computing (ISO/IEC SC 38 Study Group, 2011).

Service Level Agreements (SLAs) are required by the business customer to ensure risks and service qualities are prevented respectively provided in the way the customer wants. The the most a provider offers is a global SLA for all customers. This is insufficient because the individual customer requirements are not considered. The classical, manual negotiation process for SLAs is simply not feasible, either, considering the quick and easy way of resources allocation on demand. Cloud Computing needs more flexibility and adaptability, which can be accomplished by machine readable dynamic changeable SLAs. Thereby an integration of the SLA management into the automated process of resource allocation is required. Furthermore a transformation of the negotiations, agreement of guarantees, implementations, monitoring and reactions of violations of the SLAs must be achieved by technical measures.

Figure 1 depicts, how easy it could be for a customer to change requirements (Service Level Objective (SLO)) for the Cloud service needed.

![GUI for specifying customer SLOs.](image)

After discussing related work in Section 2, the new approach for SLA representation is described in Section 3 and justified by a reference implementation shown in Section 4. Section 5 illustrates applications by presenting some major use cases and a conclusion is drawn in Section 6.

2 RELATED WORK

As more and more companies are starting to use Cloud services the need for SLAs is growing stronger. NIST (Liu et al., 2011) has pointed out the necessity of SLAs, SLA management, definition of contracts, orientation of monitoring on Service Level Objects...
(SLOs) and how to enforce them. However a clear definition of a reference of a specific format is missing. This is also the case with the Internet Engineering Task Force (IETF) (Khasnabish et al., 2010). Besides these approaches of the companies and organizations there are further efforts to develop and to realize Cloud management architectures and systems. A basic discussion can be found in Service Level Agreements for Cloud Computing (Happe et al., 2011), but is mainly concerned about SLA definitions and negotiations.

In 2003 IBM developed the Web Service Agreement Language (WSAL) (Ludwig et al., 2003) with the focus on performance and availability metrics, but lacks expressiveness and therefore is simply not powerful enough. Also the required flexibility is missing, which is needed for dynamical changes at run time. WSAL has been mainly developed for Web services, the usage in other fields is questionable. It shows significant shortcomings regarding content as it was focused mainly on technical properties. The structural requirements, however, are met as discussed in Spillner (Spillner et al., 2009).

WS-Agreement (WS-A) was developed by the Open Grid Forum in 2007. The newest update, which is based on the work of the European SLA@SOI project, was done in 2011. The advantages are the expandability and the adaptability which is, on the other hand, also one of its greatest disadvantages, because it has not been specified in details by Kearney (Kearney et al., 2011). Based only on technical transformations, structural transformations have not been taken into account.

Although it has been enhanced within the SLA@SOI project (SLA@SOI, ) the development is unclear, because the SLA@SOI project developed its own format SLA(T), which is supported by the European IT industry. A comprehensive project result has been published on their web page but so far no independent analysis of the advantages or disadvantages of the SLA(T) format is available. SLA(T) provides all structural requirements of SLAs and it has the greatest intersection with regard to content. Therefore we choose SLA(T) as the basis of the proposed approach in this paper. Further it should be accentuated that a meta model SLA* is defined which simplifies the extension and adaptability for SLA(T).

The Foundation of Self Governing ICT Infrastructures (FOSII)-Project (FOSII Team, 2012) is another research project which aims at the usage of autonomic principals for information and communication systems. Self determining infrastructures should be realized and made available through Cloud based services. Within the LoM2His autonomic SLA management is realized by translation of system parameters to abstract KPIs and SLOs (Brandic et al., 2009). The SLA specification is based on WSAL. Also the existing monitoring tools are only designed for monitoring of the system, but not for SLAs.

3 ADAPTABLE SERVICE LEVEL OBJECTIVE AGREEMENT (A-SLO-A) FORMAT

The reference implementation of the A-SLO-A format is based on the abstract SLA* model, developed within the SLA@SOI project. The SLA* model is an abstraction layer and follows the description of the Meta Object Facility (MOF), specified by the Object Management Group (OMG). The MOF describes a special meta data architecture and itself uses the highest abstraction layer M3. The SLA* model is on layer M2, this corresponds to the Platform Specific Model (PIM). The A-SLO-A Format is one abstraction layer below (M2). This describes the Platform Specific Model (PSM). So the A-SLO-A Format is on the same level like SLA(T) description of the SLA@SOI project, as shown in Figure 2. In the reference model A-SLO-A, the primitive data types of the SLA* model have not been used, instead the EMF data types of the Eclipse Module Ecore are used. This was necessary to get a functional prototype.

Figure 2: Model based development of A-SLO-A.

The main goal of the format is to develop adaptable, adequate machine readable agreements which are legally binding. To get customer-oriented, runtime-adaptable SLAs the structure is divided into a static and a dynamic part. The static part contains the contract partner IDs, addresses, etc. while the more interesting dynamic part focuses on SLOs like scaling limits, backup periods, etc. This dynamic SLA part must be monitor-able and is controlled by the customer.

Within the A-SLO-A format its possible to define
Top-Level SLAs and A-SLO-Agreements (A-SLO-As) which are single and dedicated for each SLO.

**Top-Level-SLA to a SLO** All Top-Level SLAs and A-SLO-Agreements are based on an appropriated template (see Figure 3). Also a Top-Level SLA always is displayed by a service which can have one or more interfaces, which are the base of the agreement. In contrast A-SLO-As are not based directly on a service. Here technically or contractually responsibilities can be modeled. Resulting characteristics of the model are:

- Technical services are characterized by a one to one assignment from a Top-Level-SLA to an SLO template, called service oriented SLO template. Also A-SLO-A templates can be referenced to another A-SLO-A template and build a hierarchy.
- A-SLO-As can be grouped by a higher A-SLO-A or a Top-Level SLA, building a hierarchical tree where each entity is referenced to the next higher A-SLO-A with a UUID and a naming convention. An entity directly references the low A-SLO-A and monitoring can be done over all.

**Top-Level SLAs and Top-Level SLA Templates.** Basically Top-Level SLA comprehend all the static contractual information and only a few dynamic information. Also references can be made to link outsourced documents like general terms and conditions. A Top-Level SLA necessary must contain information about accounting of services, IT continuity plans, service development plans, terminology, escalation plans, guidelines for priorities, responsibilities of both customer and company, data of both parties, service times, accomplishment penalties, signing, an ID, how reports have to be done and how they are displayed and in which frequency, the interfaces with IDs and descriptions, references of A-SLO-As with UUID and naming conventions, and the termination reason.

For a legally binding A-SLO-A it has to contain an A-SLO-A identifier, contact data and service times. Also the SLO itself with an ID, a value and a data type which has exactly one interface, the priority and boundary and marginal values has to be included. The A-SLO-A also should contain the A-SLO-A reference, the accounting, accomplishment penalties, monitoring, how and with with interface it is done, how the reporting is done and the termination clause.

**Workflow for a Top-Level SLA or A-SLO-A.** To generate a Top-Level SLA or a A-SLO-A, the customer first has to fill in the customer basic information, service times, how it will be payed for and how the reporting should be done with which interfaces. In case of a A-SLO-A there also has to be filled in how it should be monitored. After that a decision is made whether if its a standard template or a customer oriented template. For a standard template all information will be checked and if necessary it is asked for correction. Otherwise if its an customer oriented template, it is asked for a special template. If the value verification is accepted at the Top-Level SLA, all A-SLO-A are filled in and the inferior references are build. At the A-SLO-A template the reference to the higher entity is build directly. After that the templates gets signed and the contract is ready.

The incident an change management is mapped as a inferior grouped A-SLO-A, so they can be used as KPI. An extension of conditions and modality can involve more actions, which always have a condition, guidelines and postcondition, which describe how the action will be triggered.

### 4 REFERENCE IMPLEMENTATION

The SLA Format implementation was done at HFU in the cloud research center (Cloud Research Center, ). There have been made many extensions to the SLA Templates proposed by the SLA@SOI project. One important part of a SLA Template is the extension of the attribute Type. It differs from the Type of the SLA Template by the possibility to have 3 values: top-level, service and customer. This causes an ‘IF’ clause to load a concrete structure into the SLA template or SLA. This means, when Type is either service or customer it is an A-SLO-A Template or A-SLO-A. For the unique identification the UUID attribute is used. Also the model version can be found in the attribute modelVersion. The following subsections discuss the main features of the A-SLO-A Model in more detail.

**SLAs and SLA Templates.** There are two segments, that contains the documentation part of the
SLA/A-SLO-A Templates. The first segment descriptions keeps the general terms of the agreement. The second segment fuDocs (short for further Documents) contains the interface description. Both structures will be described later. Normally it is possible to use SLA and SLA Templates without an agreement term segment, but for compatibility reasons to SLA* respectively SLA(T) this segment is included.

Description of Agreements. The class Agreement Description as the segment description of SLA templates and SLA has its own structure, which differs from the SLA* model. In the A-SLO-A Format the segment descriptions contains exactly one element of the class Property with the segment entries. These entries use the class Entry, which has two attributes key and value. The attribute key contains the type of the following document in the attribute value. The type can be either a CoverageDescr, a AgreementDescr or a Disclaimer. The information itself is stored in the attribute value.

Signatures in SLAs and SLA Templates. New attributes are in the class SLATemplate the segment ProviderSignature and in the class SLATemplate the segment customerSignature. These contain the signature of the corresponding party, for signing a SLA online. If the SLA is accepted by the customer/provider, the signature will be added to the SLA (segment customerSig) and/or SLATemplate (segment providerSig).

Interface Declaration. The interface declaration is part of the service description and is used to connect the interfaces of a service and the SLOs. Also this is needed for the reference of further documents. There are two ways to implement that useing the abstract needed for the reference of further documents. There the interfaces of a service and the SLOs. Also this is part of the service description and is used to connect the service with the same name, which contains through the class ResourceType the type of the document. This is in uppercase the type of the document. For example for an PDF document the shortcut "PDF" is used. Second the class SpecVersion, that can be defined by

WSDL (Web Service Description Language). This is also part of the SAL* interface specification.

Agreement Terms. The AgreementTerm segment is the core part of the SLA* Agreement because it contains the states GuaranteedStates and the resulting actions GuaranteedAction and is a segment of SLATemplate. A state describes the relation between a SLO and a measure point based on an interface and in general contain an attribute priority. Both (stats and actions) have conditions. Thereby actions must have a precondition and a postcondition, while a state can have many preconditions. The representation of conditions happens by the ground expressions of the SLA* model. The actions have an attribute actorRef. This field contains the unit, which is responsible to work on it. The class AgreementTerm is contained in the segment term of the top level SLAs or A-SLO-As. This is a difference between the SLA* model and the A-SLO-A Format. Important is the fact, that the class AgreementTerm can have only the segment GuaranteedStates, when the type is not top-level. Therefore three state rules exist: (1) Top-Level SLA (Template) hold no states, (2) A-SLO-A (Template) hold one or more states, and (3) A-SLO-A (Template) without a state is a group KPI. For ActionRules it is essential that a Top-Level SLA (Template) or a A-SLO-A (Template) can hold any number of actions.

Dependencies. The Dependencies of the documents itself are solved simply with the classes Dependency and Property. Also there are no existing solutions in the SLA* model or SLA(T). The idea behind this is, to set the documents itself in relation. All Agreement Terms can have many superordinate (segments depending) and subordinate (segments anteceding) Terms over the class Dependency as segment dependency. To store the information below the class Dependency the class Property is used. The attribute key contains the name and the attribute value references the document either as URI or as UUID. Also
rules can be applied. Important is the CustomerSLO-ARule, which allows for non customer A-SLO-A (see attribute type) only exactly one entry for anteceding. This structure is quite simpler than the mechanism in the SLA* model.

**Service Level Objectives (SLOs).** SLOs contain two components: The first component is the declaration of objectives and its values and the thresholds. A threshold will be defined by the variable declaration. This is a mechanism to define a name, a value and a data type. Also the threshold values will be added, so that all required information are available. The state definition creates a connection between the values defined in the SLO and at the runtime estimated value (measure point). The second component of the SLAS is the state definition. This is done by the class GuaranteedState, which has a Priority, and is an optional set. This class connects the values defined in the SLO and the measured real time values. These information are called measure point. Important is that the name of a defined interface and a UUID of a top level template are required. This relation is known by the SAL* model but was extended. An association to an interface of a A-SLO-A is possible by storing an UUID of the top level SLA.

**Action Definitions.** The next part are the actions of the A-SLO-As. In general there are two different types: Business Level Guaranteed Actions and Pricing Business Terms. The first one is from the point of view of system functionality and is used for the service provision. In that terms and templates are stored as well as information that describes the execution of the system functionality. When an event happens, this information describes all further steps, which must be executed. For example this could be an agreement between provider and customer about the start time of the monitoring or describe the automatic sending of messages after an incident.

**Business Level Guaranteed Actions.** The Business Level Guaranteed Actions has currently monitoring and reporting actions implemented. The monitoring classes are extended for the A-SLO-As. Also the attribute ParameterSource was added, which allows to recognize properties through their names, data types and its source (UUID). Also frequencies for the definition of report, aggregation and acquisition intervals have been added for triggering actions. For example: onEvent: an event trigger, onTime: a constant time stamp, perJob: runtime of a job, or perVolume: volume of a resource (e.g. 30GB disk space is full).

A major change is the relation to the class Parameter. This was changed to composition with the 1:n relation, where n must be at least 1. This allows the declaration of multiple parameters and together with the source of the parameter (SourceParameter) it is possible to assign values of the depended A-SLO-As.

## 5 USE CASES

To illustrate possible applications this section presents use cases and gives the corresponding sample implementations. Cloud Computing hugely benefits of the underlying self-serve principle. Therefore it is particular important to give customers the possibility to fill in the SLO templates with only a few steps and in an self-explanatory way. The following use cases will show the modeling possibilities of A-SLA-A:

**Availability Use Case.** For the first use case we assume that a customer runs a Cloud application on an specific regular basis. Therefore he wants to ensure that the resources are available for this specific times and can be used. We assume that the customer already has completed the initial process of creating a valid SLA with an provider for the service. The contract thereby covers the specific service times, in this case every friday from 14:00 to 18:00. For repeating ServiceTimes the customer additionally specifies an interval and the associated day or calendar date. The resulting XML file can be seen in Figure 5.

Lets assume due to the changing business environment the customer would like to change the created SLA, because he needs the same service to be available on every wednesday from 09:00 to 14:00. Therefore the customer loads the existing SLA and adapts the ServiceTime so that his new requirements are met. This however requires that the newly chosen parameters are inside the limitations given by the provider and are valid. The validation is done automatic within the adjustment tool. Did the validation succeed, the SLA gets signed online by both parties and is then legally binding. In the case of mismatching customer demands and provider offerings the SLA can not be signed and a manual negotiation process has to be performed.

**Additional KPI Use Case.** As exemplary second use case a customer cloud want to extend an existing SLA in order to match the advanced or varied requirements of his services, so the service quality can be better monitored and ensured. In such a case the customer would want to add an additional KPI to the existing SLA. That means another it AgreementTerm
has to be added to the SLA. Inside this it Agreement-Term the it GuaranteedState, itGuaranteedAction, etc. have to be specified, and a it ServiceLevelObjective has to be created. For example if a customer wants to add a guarantee for the minimum bandwidth his service can use, the KPI and the corresponding it interfaceDeclr can be easily added by using the graphical interface like shown in Figure 1. Therefore the customer chooses a representing KPI from a list of available KPIs, fills in the appropriate values and so adds the new contract clause to the existing SLA. Again it has to be checked if newly added KPI values are within the providers range of offerings or not, and based on that be signed online.

In terms of a pricing model providers could give certain offers in regard of predefined KPI ranges. In this way a provider would be able to achieve a better resource allocation and usage prediction, while customers could easily choose upon these predefined sets.

6 CONCLUSIONS

In this paper we demonstrated how adaptable SLA management is essential for companies that want to use Cloud services and complies with the Cloud Computing self-service, on-demand characteristics to change SLAs during runtime. High utilization of the infrastructure for the provider and an ideal pay per use basis for the companies can therefore be achieved.

With A-SLO-A it is possible to get customer specific SLAs automated in acceptable conditions for both parties. But to get an economical efficient adaptation, more automation is essential. It has been shown, that the new adaptable SLA Agreement (A-SLA-A) language can model static SLA information and dynamic SLA objectives to be the basis of an adaptable SLA management. Use cases have been presented to visualize the power of the A-SLA-A.

REFERENCES


SLA@SOI. SLA@SOI projekt website. http://sla-at-soi.eu/