2014 IEEE International Conference on Cloud Engineering

PSLA: a PaaS Level SLA Description Language

Ge Li, Frédéric Pourraz, Patrice Moreaux
Savoie University
Annecy le vieux, France
Email: {ge.li, frederic.pourraz, patrice.moreaux}@ univ-savoie.fr

***Abstract*—PaaSs emerge to help SaaS providers to conquer the challenges involved in guaranteeing the QoS of international SaaSs. A well structured machine readable SLA is critical for PaaS providers to automate InterCloud management to serve the worldwide elastic user demands. WS-Agreement is a widely accepted extendible language for describing agreements. In this paper, we propose PSLA, a well structured PaaS Level SLA description language based on WS-Agreement. We summarize and describe the semantic clauses needed to be considered in PaaS level SLA. In particular, some specific characteristics of PaaS are taken into account in PSLA, such as elasticity of workload, undefined metric properties and fuzzy value range.**

***Keywords*- Service Level Agreement (SLA); PaaS; QoS;**

I. INTRODUCTION

Most of the cloud services are provided in the way of "Pay as you go". Platform as a Service (PaaS) is one kind of these services. PaaS is a layer between Software as a Service (SaaS) and Infrastructure as a Service (IaaS). IaaS providers loan resources, like virtual machines, to their clients. SaaS providers loan software usage rights to their clients through internet. PaaS providers run their business by loaning a platform for whole SaaS application lifecycle, including development, testing, deployment, execution and software maintenance [1]. It is widely accepted that SaaS providers save investments on purchasing and maintaining hardware using IaaSs. But activities and difficulties involved in SaaS application lifecycle are quite different from past. So PaaS is designed to lower the sill of using cloud infrastructure and popularize the cloud. By using PaaS, SaaS providers can ignore the differences among IaaSs, accelerate SaaS application development and acquire elasticity of SaaS. Nowadays, there is no IaaS provider who can build data centers all around the world. International SaaS providers need to choose and run their applications into multiple IaaSs to serve worldwide elastic end user demands with expected quality of service (QoS) [2].

A Service Level Agreement (SLA) is a contract between the service consumer and the service provider. It is used to describe QoS target in a structured way. A SLA is needed for several reasons. The responsibilities of contracting parties should be clearly defined, for example the actions carried out by each part in various conditions. Usually, in a SLA, the services promised by the provider are quantitatively and comprehensively defined by a set of QoS metrics with value ranges. Not only the level of QoS should be guaranteed but also the corresponding rewards and penalties will also be specified for fulfilling or violating promises. Most of the management involved in cloud computing, including SLA management, is relied on automatic tools. A structured and machine readable SLA definition, e.g. in XML, is necessary so that the SLA can be used as input file of the SLA management system. Other management systems in the cloud may also get benefit from it. For example, billing systems can use penalty and reward attached to each QoS promise to charge to service consumer. Actually, most cloud service providers offer SLA in text documents. For example, public cloud service Google App Engine [3] and Amazon Elastic Compute Cloud (EC2) [4] publish their uniform and non custom SLA online.

If a SaaS provider authorizes PaaS providers to deploy and run applications, the SaaS level SLA should be guaranteed by PaaS providers. So SaaS level SLA will be expressed in corresponding PaaS level SLA with appropriate constraints. Since general application workload is unknown to PaaS providers, a set of constraints must be too expressed to define workload of applications and even elasticity properties of the load.

In this paper, we propose PSLA, a PaaS level SLA description language. PSLA is based on WS-Agreement [5] which is an extendable SLA skeleton language. To the best of our knowledge, we propose for the first time a commonly usable and machine readable PaaS level SLA description language. PSLA is being adopted by OpenCloudware [6] project which aims at building a PaaS platform to manage applications running over multiple IaaSs during their lifecycle.

The rest of this paper is arranged as follows. Section II introduces briefly related works. PSLA specificities are analysed in section III. In section IV, the semantic expression of PSLA is summarized. We provide a scenario and a plain text SLA in section V. In section VI, we use PSLA to express the SLA of the example of section V. Finally, we conclude this paper in section VII.

II. RELATED WORKS

A. SLA

Researches about SLA emerged many years ago. Among these, Web Services Agreement Language (WSLA), Web

978-1-4799-3766-0/14 $31.00 © 2014 IEEE
DOI 10.1109/IC2E.2014.29

Services Agreement (WS-Agreement), SLA model proposed in SLA@SOI, SLAng and CC-Pi are markable.

WSLA [7] can be used to define QoS statements for contracts but it is not actively maintained. Moreover, works in WSLA have been transferred into WS-Agreement. SLA model proposed in SLA@SOI [8] is language independent, by using abstract syntax. But it is not actively maintained too. SLAng [9] uses Meta Object Facility (MOF). Thus, it is independent from language to some extent. However, it focuses on services and is weak in expressing QoS constraints. CC-Pi [10] is also a generic solution but it is also weak in expressing basic concepts such as agreement parties.

WS-Agreement [5] is widely used and has many existing implementations. WS-Agreement for Java framework (WSAG4J) [11] is a tool to create and automatically manage SLAs. SLA managements, like offer validation, service level monitoring, persistence and accounting are supported. This makes the design and implementation of SLAs based on WS-Agreement much easier. Using WS-Agreement to create contract requires extensions for domain specific quality expressions.

B. SLA for Cloud Computing

Nowadays, there is only few researches focusing on description language in the context of the Cloud. Among them, [12] defines CSLA, specially designed for cloud computing. CSLA consults well known SLA languages like [5], [7], [13]. However CSLA presently still needs a lot of completion of basic clauses before being usable. "Confidence" and "Fuzziness" were first introduced in CSLA to express the ambiguous of the boundary value of target range of QoS metrics. They are adopted in PSLA.

So far as we know, there is no generic PaaS level SLA description language, motivating our PSLA proposal, a XML language based on WS-Agreement skeleton.

III. PSLA Specificities

A. Elasticity

The elasticity at a given time can be expressed by a four-dimensional vector (s, c, t, l):

- s, Data size, means the total amount of data during a given unit of time.
- c, Workload composition, means the proportion of each kind of load which form the workload. Workload composition itself can be described by a multi-dimensional vector and each dimension can be defined by the resources required by a unit amount of the load. The details will be described in the following content.
- t, Time, means the time when the workload happened.
- l, Location, means this workload acts on which IaaS.

1) Load: Different services need different amount and various kinds of resources. From the PaaS providers' point of view, resource needs is the fundamental gist to ensure the promised QoS. So we classify the loads according to the amount of each kind of resources demanded on each component (We consider the component based software system). We use matrix $L^i_{M\times N} = \left\{l^i_{m\times n} | l^i_{m\times n} \geq 0, 0 \leq m < M, 0 \leq n < N\right\}$ Where $L^i_{M\times N}$ represents the load i. $l^i_{m\times n}$ represents the amount of $resources_n$ required on $component_m$ by $load_i$. M represents the number of components in software system. N represents the number of kinds of resources. Since there are limited kinds of resources, like memory usage rate, we can summarise commonly used resource metrics and use a fix single dimension vector here. In this paper, we describe it as $(CPU, RAM, Network)$.

2) Workload: Workload is composed by several loads. Different ratios of loads can lead to different distribution of total resources demanding.

B. Metric

For PaaS level SLA, as far as we know, there is no existing metric model. The studies in [14] show some of the commonly used performance metrics, including throughput, reliability, Load balancing, Durability, Elasticity, Linearity, Agility, Automation and customer service response times, are listed.

We consider the following properties initially:

- Unit: The measurement unit of this metric. Common used units, like second, are built-in in PSLA, but there is also mechanisms for user defined unit.
- Value type: Which type the value of the metric should be, for example string, integer or decimal.
- Value source: Where the value of this metric is acquired from. PSLA allows to use a URI to describe the location of metric value.
- Value update mode: The way new metric value is gotten from the value source.

C. Fuzzy Value Range

We use elementary logical expression to describe value range. We define logical operators, comparison operators and boundary values by the book. However, it is difficult to give a fine line as boundary in the real world. So we adopt "fuzziness" and "confidence" as in [12] to make the boundary more practical. "confidence" is renamed as "trust" to distinguish statistical term.

- Logical operators: Including $\cap$, $\cup$ and $\neg$.
- Comparison operators: Including$<$, $>$, $\leq$, $\geq$ and $=$.
- Metric: Value range acts on which metric.
- Boundary values: the boundary of range.
- Trust: At least how many percentage of the metric value should be within the value range described by logical operators, comparison operators and boundary values.

- Fuzziness: The acceptable degree of deviation of metric values which are out of bounds.

IV. Semantic Description

In this part, we list and explain semantic clause involved in PaaS level SLA. PaaS level SLA includes two kinds of informations. They are general informations about the SLA named as "SLA Metadata" and detailed QoS target descriptions of the SLA named as "QoS Criterion".

A. SLA Metadata

SLA Metadata defines the basic informations of the SLA:

1) Agreement Name: The identification of the agreement. It is a unique ID or name.

2) Agreement Initiator: Which party starts the agreement creation request.

3) Agreement Responder: The party who responds to the agreement creation request.

4) Service provider: The party who will provision services. The contrary party is service consumer.

5) Agreement Expiration Time: At which time, this agreement is no longer valid. All the "Valid Period"s defined in "QoS Criterion" should be subsets of it.

B. QoS Criterion

One important purpose of creating a SLA is giving an insurance according to the QoS delivered by the service provider. In this part, the detailed clauses are described. The rights and obligations of each party are defined clearly.

1) Valid Period: Valid Period specifies the period of time that this QoS Criterion will be taken into effect. Valid Period can either be continuous or regularly discrete (see section III-A). This is achieved by a "Start Time", a "End Time" and a optional "Time Interval".

2) Valid Workload: Valid Workload describes that the QoS Criterion will be taken into effect for what kinds of workload acting on SaaS application. The workload can be described from three aspects: "Data Size", "Data Composition" and "Location" (see section III-A).

3) Service Level Objective: Service Level Objective (SLO) is the core part of QoS Criterion. It expresses what exactly the QoS target is by quantifying the acceptable QoS metric "Value Range". Sometimes, it is even difficult for service consumer themselves to specify a value as exactly threshold between satisfactory and unsatisfactory, that's why we use "Trust" and "Fuzziness" to allow the fuzzy description of threshold (see section III-C).

4) Metric Description: In QoS Criterion, each SLO is quantified by QoS metric value scopes. The metrics are described to make sure that there is no ambiguity about the metric among parties (see section III-B). This is achieved by a "Unit", a "Value Type", "Value Source " and "Value Update Mode".

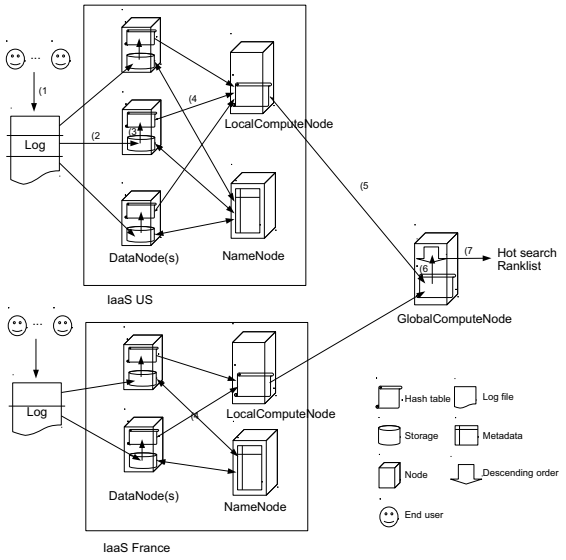

Figure 1: Hot Search Ranklist in InterCloud

5) Business Values: Different SLOs are usually reflected in different business values, like "Penalty" and "Reward". These values will be used in billing system to calculate the cost of service.

6) Check Plan: Billing system need to check the QoS and service promised in the SLA to do the calculation. The time when this action should be take is specified in the contract.

7) Priority: Sometimes service provider don't have enough resources to meet all SLOs at the same time, so it is necessary to specify priority for each ones, so that the service provider can make a trade-off.

V. An example

In this section, we describe a SLA scenario and the corresponding plain text SLA.

A. SLA Scenario

The scenario is as follows. Ebay is a international C2C electronic business web service provider. Ebay wants to provide a service as "Hot Search Ranklist". It computes the most popular search term during the past one hour. The architecture of this example is demonstrated in figure 1. Ebay's PaaS provider deployed system into two IaaSs, one in US and another in France, according to a certain algorithm. So there are two duplications running to serve the worldwide workloads. In each IaaS, local calculation is processed independently. Global "Hot Search Ranklist" will be summarized when local calculations are finished. The detailed description is as the following:

1) The end user use the search bar of webpage of ebay. Search logs are generated.
2) If the search bar is largely used by the end users, the search log file can be too big to be stored in a local

file system. In this case, distributed file system (DFS) will be used. For example, in the IaaS in US, the big log file will be cut into three parts, and each part will be stored on one DataNode separately.

3) On each DataNode, the frequency of occurrence of each kind of log record is also calculated. This calculation can, for example, be done by hash table (Key: log record - Value: frequency of occurrence).
4) So there are three small hash tables generated on DataNodes. These hash tables are sent to the LocalComputeNode instead of all parts of log file. In this way, the network resources spent on transforming huge amount of data are saved.
5) In LocalComputeNode, these three hash tables are merged into one big aggregated local hash table. In this later, the value is the frequency of occurrence of each kind of log record in the whole log file.
6) Aggregated local hash table generated in each LocalComputeNode will be sent to one GlobalComputeNode.
7) Finally, on the GlobalComputeNode, the values will be sorted from large to small, and the corresponding key will be displayed to the end user as "Hot Search Ranklist".

B. Plain Text SLA

The plain text SLA is created according to the semantic description in section IV. It is as follows.

1) SLA Metadata:

- *Agreement Name*: "TopHotSearchTerm".
- *Agreement Initiator*: "Ebay".
- *Agreement Responder*: OpenCloudware.
- *Service provider*: OpenCloudware.
- *Agreement Expiration Time*: 10/01/2020 at 00:00:00.

2) First QoS Criterion:

- *Valid Period*

  Start Time: from 12/01/2013 00:00:00

  End Time: to 12/31/2013 23:59:59

  Time Interval: everyday from 20:00 to 23:00
- *Valid Workload*

  Data Size: the amount of log generated in one hour is lower than 1000000

  Data Composition: ratio of long log is lower than 30%

  Location: the workload act on IaaS located in US.
- *Service Level Objective*

  Value Range: the time spend on compute "Hot Search Ranklist" should be lower than 3 seconds.

  Trust: at least 90% of these value should be met.

  Fuzziness: for the rest 10%, the value should be lower than 3.15s.
- *Metric Description*

  Unit: The unit of this value is "second"

  Value Type: float

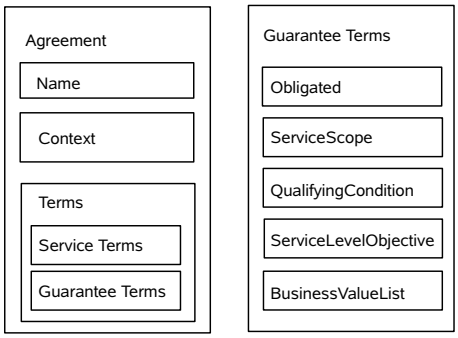

Figure 2: WS-Agreement Architecture

Value Update Mode: This time is periodically collected

Value Source: from interface01 provided by an external monitoring system/service.

- *Business Values*

  Reward: If it is abide by OpenCloudware, USD 3 will be gained.

  Penalty: If it is violated, USD 2 will be charged.
- *Check Plan*: This term should be checked once per hour.
- *Priority*: The priority of this term is 2.

3) Second QoS Criterion: Due to lack of space, we only detail two entries of the second criterion which are significantly different from the first one.

- *Service Level Objective*

  Value Range: pay expenses per hour should be strictly no more than 10 USD.
- *Business Values*

  Reward: If it is abide by OpenCloudware, USD 0 will be gained.

  Penalty: If it is violated, USD 10 will be charged.

VI. Expressing SLA using PSLA

A. Basic WS-Agreement

In this section, the whole structure of WS-Agreement is introduced. How does PSLA uses and extends WS-Agreement skeleton is mentioned. The extension is described in next subsection. Like shown in figure 2, an agreement is composed by a $Name$, a $Context$ and $Terms$.

$Context$ describes "SLA Metadata" (see section IV-A), including "Agreement Name", "Agreement Initiator", "Agreement Responder", "Service provider" and "Agreement Expiration Time", of the $AgreementOffer$. It is extendable.

$Terms$ is the core part of an agreement. Services and QoS promised by service provider are described in $Terms$ and most of our works are inside its. There are two kinds of terms in WS-Agreement: $ServiceTerm$ and $GuaranteeTerm$. These terms are organised together by All, $ExactlyOne$

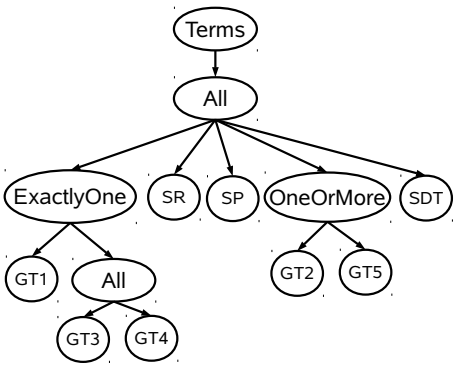

Figure 3: An example of term tree in WS-Agreement

and $OneOrMore$ as a $TermTree$. Figure 3 is an example of $TermTree$.

1) Service Terms: $ServiceTerm$ mainly contains functional descriptions. There are three kinds of $ServiceTerm$ in WS-Ageement. They are $ServiceDescriptionTerm(SDT)$, $ServiceReference(SR)$ and $ServiceProperties(SP)$. SDT and SR have got the same syntax expression and must be extended. SDT is designed for domain specific description, and we extended it to express "Metric Description" (see section IV-B4). In PSLA, SR is extended to describe a service provided by PaaS provider. One or more URIs are used to refer to the entry point of the service. In PSLA, SP is used to packet all the QoS metrics as $Variable$ which will be used in $GuaranteeTerm$.

2) Guarantee Terms: $GuaranteeTerm$ is designed for non-functional part of the agreement, including preconditions which are related to workload elasticity (see sections IV-B1 IV-B2), QoS target (see section IV-B3) and related business clause (see section IV-B5). One $GuaranteeTerm$ is composed of $Obligated$, $ServiceScope$, $QualifyingCondition$, $ServiceLevelObjectives$ and $BusinessValueList$.

- $ServiceScope$ indicates this $GuaranteeTerm$ is about which services.
- $QualifyingCondition$ declares this $GuaranteeTerm$ becomes effective under which premises. $QualifyingCondition$ usually constraints for the one who take advantages from the $GuaranteeTerm$, e.g. ebay in the example. $QualifyingCondition$ is also open to be extended.
- $ServiceLevelObjectives$ has been extended by us to express the quantify expression of the requirement on QoS metrics.
- $BusinessValueList$ describes the importance of this $GuaranteeTerm$ from many aspects. $BusinessValueList$ describes "Check Plan" (see section IV-B6), "Priority" (see section IV-B7) and "Business Values" (see section IV-B5), including "Penalty" and "Reward". It is extendable. These informations can be helpful for PaaS provider to make a choice when the resources are not enough to meet all $GuaranteeTerm$s. In our example, second $GuaranteeTerm$ (see section V-B3) will be considered if it is not possible to meet both of them at the same time. "Check Plan" (see section IV-B6) is expressed in $Penalty$ and $Reward$ in $BusinessValueList$.

We can see that WS-Agreement skeleton provides the commonly used clauses of web service based agreement. We extend it to meet domain specific needs. In PSLA, we extend WS-Agreement by considering the characteristic of PaaS level service and QoS mentioned in section **??**. Metric (see section III-B) will be expressed in SDT extension. Elasticity (see section III-A) will be expressed in $QualifyingCondition$ extension and $ServiceLevelObjectives$ will use fuzzy value range (see section III-C) to express the uncertain boundary of QoS target.

B. PSLA Extension of WS-Agreement

In this subsection, we will partly demonstrate PSLA's expression power by describing the plain text SLA listed in section V-B. The ideas behind the design of corresponding XML schema is based on section **??** and section IV.

1) Metrics: We take, for example, the "Metric Description" of "First QoS Criterion" (see section V-B2). We name this metric as "RtOfComputeComponent". This is a typical Customer service response times type metric. In this example, "Unit", "Value Type", "Value Source" and "Value Update Mode" are expressed. The unit of metric is "second". The value of this metric is "positive float"(e.g.3.0). Value source is "collected from interface01 of monitoring system". Value update mode is "periodically collected". The metric is expressed as follows XML fragment 1.

Listing 1: Metric example

```
<Metric Unit="second" MetricName="RtOfComputeComponent">
  <ValueType>float</ValueType>
        <ValueGenerate>
          <ValueSource>interface01</ValueSource>
          <ValueUpdateSchedule>PT1H</ValueUpdateSchedule>
  </ValueGenerate>
</Metric>
```

2) Qualifying Condition: $QualifyingCondition$ is mainly used to express the condition that this $GuaranteeTerm$ will become effective under which condition. If we put all the $QualifyingCondition$s in this $AgreementOffer$ together, it will be a panorama of the elasticity of workload III-A. We extend $QualifyingCondition$ to describe "Valid Period" (see section IV-B1) and "Valid Workload" (see section IV-B2). "Valid Period" includes "Start Time", "End Time" and "Time Interval". "Valid Workload"

includes “Data Size”, “Data Composition” and “Location”. $QualifyingCondition$ part of the first QoS criterion (see section V-B2) is as follows (see section 2).

Listing 2: QualifyingCondition example

```
<ValidPeriods> <ValidPeriod>
    <Start>2013−12−01T20:00:00</Start>
    <End>2013−12−31T23:59:59</End>
    <TimeInterval>PT3H</TimeInterval>
    <Step>PT21H</Step>
</ValidPeriod> </ValidPeriods>
<ValidWorkloads> <ValidWorkload>
    <DataSize>
        <Expression> <Predicate> <Less>
            <VariableName>LogNumber</VariableName>
            <Threshold>
                <Value>1000000</Value>
                <Trust>1</Trust>
                <Fuzziness>0</Fuzziness>
            </Threshold>
        </Less> </Predicate> </Expression>
    </DataSize>
    <Composition>
        <CompositionPart Name=”long”>
            <TypeName>longlog</TypeName>
            <LowerBound>0.0</LowerBound>
            <UpperBound>0.3</UpperBound>
        </CompositionPart>
        <CompositionPart Name=”short”>
            <TypeName>shortlog</TypeName>
            <LowerBound>0.7</LowerBound>
            <UpperBound>1</UpperBound>
        </CompositionPart>
    </Composition>
    <Location>US</Location>
</ValidWorkload> </ValidWorkloads>
```

3) Service Level Objectives: Service Level Objectives is the core part of GuaranteeTerm. It is extended to describe “Value Range”, “Trust” and “ Fuzziness” (see section IV-B3). For example, the “Service Level Objective” of the first QoS criterion (see section V-B2) is expressed as the following XML fragment (listing 3).

Listing 3: Service Level Objective example

```
<Expression> <Predicate> <Less>
    <VariableName>RtOfComputeComponent</VariableName>
    <Threshold>
        <Value>3.0</Value>
        <Trust>0.9</Trust>
        <Fuzziness>0.05</Fuzziness>
    </Threshold>
</Less> </Predicate> </Expression>
```

VII. Conclusion

SLA, as a contract which specify the service and QoS criterion, is mandatory for service oriented business. In PaaS, the QoS of SaaS and elasticity of workload are cooperated by PaaS provider. So it is necessary to consider the elasticity and SaaS QoS target characters in PaaS level SLA. In this paper, we introduced PSLA, the first SLA description language for PaaS level SLA. PSLA is adopted by the OpenCloudware project. PSLA is based on XML as it extend WS-Agreement. The XML schema grammar of PSLA is available and the XML example described in this paper can also be found by contacting the authors. Next steps of our work are modelling most of the common used metrics and workload of systems to improve the expressing ability of PSLA.

Acknowledgment

The research described in this paper is supported by the OpenCloudware project [6].

References

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

[2] R. Buyya, R. Ranjan, and R. N. Calheiros, “Intercloud: Utility-oriented federation of cloud computing environments for scaling of application services,” in *Algorithms and architectures for parallel processing*. Springer, 2010, pp. 13–31.

[3] “Google App Engine Service Level Agreement,” 2013. [Online]. Available: https://developers.google.com/appengine/sla

[4] “Amazon Elastic Compute Cloud (EC2) Service Level Agreement,” 2013. [Online]. Available: http://aws.amazon.com/ec2-sla/

[5] 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 *Global Grid Forum*, vol. 2, 2004.

[6] “OpenCloudware,” 2013. [Online]. Available: http://opencloudware.org/

[7] H. Ludwig, A. Keller, A. Dan, R. P. King, and R. Franck, “Web service level agreement (WSLA) language specification, version 1.0,” IBM Corporation, Tech. Rep., 2003. [Online]. Available: http://www.research.ibm.com/people/a/akeller/Data/WSLASpecV1-20030128.pdf (2013)

[8] P. Wieder, *Service level agreements for cloud computing*. Springer, 2011.

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

[10] M. G. Buscemi and U. Montanari, “CC-pi: A constraint-based language for specifying service level agreements,” in *Programming Languages and Systems*. Springer, 2007, pp. 18–32.

[11] O. Wäldrich, “WS-Agreement for JAVA (WSAG4J).” [Online]. Available: http://packcse0. scai. fraunhofer. de/wsag4j

[12] Y. Kouki, T. Ledoux *et al.*, “CSLA: a language for improving cloud SLA management,” in *Proceedings of the International Conference on Cloud Computing and Services Science*, Porto, Portugal, Apr. 18-21 2012.

[13] T. K. Keven and T. Francesco, *The SLA Model*. springer, jun 2011, vol. 2, ch. 4, pp. 43–68, service Level Agreements for Cloud Computing.

[14] M. Ahronovitz, D. Amrhein, P. Anderson, A. de Andrade, J. Armstrong, B. Arasan, J. Bartlett, R. Bruklis, K. Cameron, M. Carlson *et al.*, “Cloud computing use cases white paper,” 2010.