# SLAFM
## *A Service Level Agreement Formal Model for Cloud Computing*

Lucia De Marco[1,2], Filomena Ferrucci[2] and M-Tahar Kechadi[1]
[1]*School of Computer Science and Informatics, University College Dublin, Belfield, Dublin 4, Ireland*
[2]*Department of Management and Information Technology DISTRA-MIT, University of Salerno, Via Giovanni Paolo II, 132 - 84084 - Fisciano (SA), Italy*

Keywords: Service Level Agreement, SLA Management, Cloud Services Logs, SLA Formal Model.

Abstract: Cloud Computing services are regulated by a contract called Service Level Agreement (SLA). It is co-signed between the customers and the providers after a negotiation phase, and during their validity time several constraints have to be respected by the involved parties. Due to their popularity, cloud services are enormously used and unfortunately also abused, specially by cyber-criminals. Sometimes the crimes have the consequence of violating some contractual constraints without the parties are aware of. A manner for guaranteeing more control of the SLA respect is to consider a dedicated system interacting with the cloud services and detecting the SLA violations by analysing the log files. We introduce a formal model aimed to represent the contents of such SLAs necessary in the context of an automatic mechanism for detecting SLA violations.

## 1 INTRODUCTION

Cloud services are regulated by Service Level Agreement contracts (SLA) (Mell et. al, 2011), where all the constraints among a cloud service provider and its customer(s) are detailed; the contracts are co-signed by the parties and they have legal validity in case of a court litigation (Baset, 2012; **?**). Because of their importance and contents(CSA, 2013), SLAs have been being considered to be monitored to detect their violations. Many attempts to automate such monitoring there exist in literature, but to the best of our knowledge most of them do not consider cloud services logs; nevertheless the literature focuses on the monitoring of performances constraints.

In this paper a set of rules is presented in the shape of a formal model named SLAFM; we aim to formalize the necessary information needed by the SLA violation detection capability, including input, output, and intermediary computations. In addition, some diagrams will be presented in order to visualize the interactions among the components dedicated to implement our SLA violation detection capability based on our formal model.

The remainder of the paper is structured as follows: in Section 2 we synthesize the background literature about both formal modelling and automatic monitoring of SLAs; Section 3 provides an overview about contents and structure of cloud SLAs. In Section 4 we discuss the principal objectives of our proposal, while an example of the proposal is illustrated in Section 5. The details of the formal model are explained in Section 6, and a case study based on the previous example is provided in Section 7. Some pro and con are presented in Section 8, and conclusion and future work ideas close the paper in Section 9.

## 2 BACKGROUND

Automating the management of a textual document is a big challenge. Our focus is to target such topic to the SLAs stipulated for cloud services. The final aim is to detect contractual violations. The information involved in this process are the ones concerning cloud services logs. Our approach provides a formal model that represents both SLAs and cloud logs. The analysed literature covers different arguments, such as formal representation of SLAs and automatic SLA monitoring research projects using SLA formal representations, which provide the background of our proposal. Most work provide a customized manner to structure the SLAs contents, which are then formalized via mathematical tuple, e.g. in (Czajkowski et. al, 2002; Unger et. al, 2008; Ghosh et. al, 2012; Ishakian et. al, 2011; De Marco et. al, 2014); in other

De Marco L., Ferrucci F. and Kechadi M..
SLAFM - A Service Level Agreement Formal Model for Cloud Computing.
DOI: 10.5220/0005451805210528
In *Proceedings of the 5th International Conference on Cloud Computing and Services Science* (CLOSER-2015), pages 521-528
ISBN: 978-989-758-104-5

works the authors use concepts from set theory, e.g. in (Skene et. al, 2007) and (De Marco et. al, 2014); the other used formalism include derivation rules, reaction rules, integrity rules and deontic rules (Paschke et. al, 2008), or mathematical logic concepts (Unger et. al, 2008).

Other work are dedicated to manage the SLAs in terms of monitoring the respect of some constraints and to prevent their violation; for instance, a framework called DESVI(Emeakaroha et. al, 2010) is dedicated to monitor low level Cloud resources in order to detect if their measured value respects the constraints extracted from SLAs with the goal of detecting QoS violations. This work has been recovered in(Emeakaroha et. al 2012b) to demonstrate its efficiency in monitoring a single Cloud data centre, and also in (Brandic et. al, 2010), where some specific metrics are applied on the resources, whose values are required to match with a specified threshold to prevent possible contractual violations. In(Morshedlou et. al, 2014) a proactive resources allocation prototype is proposed for reducing the negative impact of SLAs' violations and for improving the level of users' satisfaction. In (Maurer et. al, 2012) a prototype for an autonomic SLA enhancement is discussed; it behaves as resource parameters reconfiguration tool at virtual machines level of cloud infrastructures, with the main advantage of reducing SLA violations and of optimizing resources utilization. Instead, in (Emeakaroha et. al, 2012a) the proposed SLA monitoring and violation detection architecture plays at the Cloud application provisioning level, where some metrics are exploited to monitor at runtime the resource consumption behaviour and performance of the application itself. In (Cedillo et. al, 2014) an approach to monitor some Cloud services non-functional SLA requirements is presented; a middle-ware interacting with services and applications at runtime is designed; it analyses the information and provides some reports as soon as an SLA violation is identified. Last but not least, SALMonADA (Muller et. al, 2014) is a platform that utilizes a structured language to represent the SLA, which is then automatically monitored to detect whether any violation occurs or not; such detection is performed by implementing a technique based on a constraint satisfaction problem.

## 3 SLA CONTENTS AND STRUCTURE

In a free trade context people have the freedom to choose for a specific need which services they prefer from a big offer pool. Once such choice is accomplished, a user contacts a specific service provider. This latter is responsible of delivering the service to the user after the instantiation of a particular Service Level Agreement (SLA) contract. An SLA concerning the provisioning of IT services is defied as a *formal, negotiated, document in qualitative and quantitative terms detailing the services offered to a customer* (ITIL). To the best of our understanding among cloud services, an SLA follows the life cycle depicted in the UML (Rumbaugh et. al, 2004) state chart digram in Figure 1.

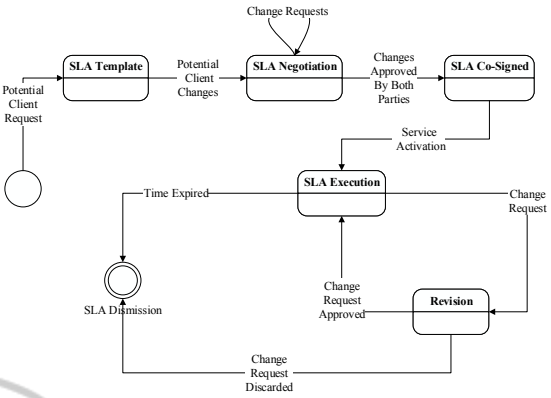


Figure 1: SLA Life Cycle.

After a potential customer request, a contractual template is customized by the cloud service provider depending on some change requests to the standard offer; subsequently a negotiation phase happens, where solutions to the change requests are included, together with information about expenses, penalties, and reports. The SLA co-signed state determines that both entities agree on the actual contractual contents, then the service provisioning can begin. The SLA has a validity time, that can be either explicitly expressed with start and end dates, or an initial date together with a time interval can be included in the document. Such validity time begins after the contract is hardly or digitally co-signed by both parties in the SLA Execution state. During its life cycle, an SLA can be subject to a revision to resolve some change requests instantiated by either party. The revision phase can provide solutions to such requests in order to continue the service provisioning and the contract validity. In case a solution is not met by the parties the service provisioning and the SLA validity are dismissed.

In an SLA regulating cloud services the duties and responsibilities of both parties are described, together with the possible presence and operations of a third party. The main contents of an SLA concern definitions and descriptions of the service, some rules and regulations about its delivering, some performance measures together with possible tolerance intervals about the levels of the services to guarantee,

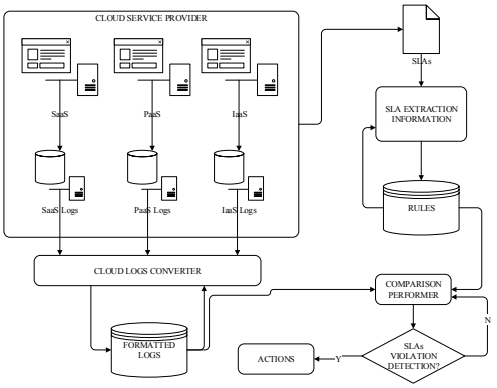


Figure 2: SLA Violation Detection Capability.

and the pricing and penalties measures in case such tolerances are not respected. For sake of monitoring and user satisfaction purposes, it is good practice to provide service levels that can be audited, managed, and measured (Larson, 1998).

# 4 SLA VIOLATION DETECTION

An SLA violation detection capability is aimed to automatically guarantee in a real time manner the respect of the contract in relation to the Cloud behaviour. This capability has two different input. They are the SLA constraints and some Cloud logs. The capability intends to perform the comparison between them. The main challenge for implementing such capability is to convert the natural language expressions of the contract in rules that a machine can manipulate. About the management of the cloud logs, instead, the effort is less, because once the structure of the logs is known they can be manipulated. In most cloud service providers such structure is documented. In Figure 2 the interactions among the components dedicated to implement our SLA violation detection capability is designed. The beginning of such capability corresponds with the SLA to guarantee starting time. The necessary contractual sections are extracted and structured in a specific format. The information gathered from the cloud services logs are translated, aggregated and indexed according to another specific format, compatible with the one the SLA sections are converted to. The capability is responsible to perform the comparison between the SLAs constraints and the cloud service behaviour structured in logs. The comparison results are stored and utilized for subsequent comparison; they also feed a decision making module, responsible to manage the consequences of the detected SLA violations.

# 5 EXAMPLE: AMAZON S3

In this section we want to provide a real-world commercial well defined SLA contract documents as example for the implementation of the SLA violation detection capability depicted in Figure 2.

In order to obtain the necessary logs, we need to access to as many information as possible, among the quantity granted by a cloud service provider. It is well documented that the amount of information accessible by cloud services customers is bigger in IaaS level, medium in PaaS, and small in SaaS (Liu et. al, 2011). Thus, we chose a IaaS type of cloud service. Moreover, we considered Amazon as cloud provider because it is considered by both business and academia as a pioneer of a complete pool of cloud services delivering. From all these considerations, our choice fell on Amazon Simple Storage Service (S3) (Amazon), which is an Infrastructure as a Service (IaaS) provided by Amazon.com.

We accessed to both the S3 public SLA and the logs, available at a customer level. Among all the legal constraints, we consider the service level expressed as Monthly Uptime Percentage. The reader has to consider this parameter as a quality attribute of the Amazon S3 cloud service. *Amazon Web Services will use commercially reasonable efforts to make Amazon S3 available with a Monthly Uptime Percentage [...] of at least 99.9% during any monthly billing cycle*.

Scanning the SLA, we can find the definition of such attribute as: *Monthly Uptime Percentage is calculated by subtracting from 100% the average of the Error Rates from each five minute period in the monthly billing cycle.* Again, Amazon carefully defined the aforementioned Error Rates as *the total number of internal server errors returned by Amazon S3 as error status InternalError or ServiceUnavailable divided by the total number of requests during that five minute period*(Amazon).

In order to guarantee the respect of the monthly uptime percentage service level, an SLA violation detection capability has to consider the information about server responses to the HTTP requests made to S3 every five minutes. Such data is available in the Server Access Log files available in S3, which collects some information every request made to the service. An example of them is provided in Figure 3. The information in red circles are the ones necessary to calculate the Error Rates, i.e., the time and the Error Code.

In a specific five minute period time our capability has to collect the log files for all the requests made, i.e., from $\text{Time}_0$ to $\text{Time}_{+5min}$. Then the Error Rate is

calculated counting the number of Error Code equals to InternalError or ServiceUnavaiable, divided by the total number of requests during that five minute period.

Considering that every entry in the log has a related Error Code field, we can affirm that the total number of requests during that five minute period is obtained by the following formula:

$$5minRequests = \sum_{0}^{+5min} ErrorCode$$

The Error Rate of a five minute time period is obtained by the following formula:

$$5minErrorRate = (\sum_{0}^{+5min} ErrorCode = InternalError\ ORServiceUnavaiable)/5minRequests$$

In order to have a monthly value the capability has to compute this value along a billing month time period. Every solar day has $60/5 * 24 = 288$ five minutes time periods; this value has to be multiplied for the number of days of a billing month. Assuming that a billing month is composed of thirty days, we will have $288 * 30 = 8640$ five minutes time periods. Finally, the monthly uptime percentage will be obtained by the following formula:

$$AverageErrorRate = (\sum_{0}^{8640} 5minErrorRate)/8640$$

$$MUP = 100\% - AverageErrorRate$$

# 6 FORMAL MODEL

The SLA Formal Model (SLAFM) is aimed to provide a theoretical approach for a contractual violation detection capability for cloud computing services. A previous draft of the same formalism was discussed in (De Marco et. al, 2014), where only a specific scenario has been taken into account. We extend that formal model by adding the modelling of cloud services logs; according to that, some changes to the modelling of the SLA structure have been added. The model utilizes mathematical formalisms as tuple, set theory, functions (Ben-Ari, 1993).

## 6.1 Service Level Agreement

SLAs are contractual documents written in natural language composed of a set of information structured as service levels. An sla $l$ is an element of the set of slas $L$. An sla is described by a mathematical tuple composed of a set of service levels $SL$, the validity starting time $t_{start}$ and the validity ending time $t_{end}$.

$$L = \{l_1, l_2, l_3, \ldots, l_j\}, j \in \mathbb{N}$$
$$l = \langle SL, t_{start}, t_{end} \rangle; t_{end} \geqslant t_{start}$$

A service level $sl$ is an element of the set of service levels $SL$. Each $sl$ has a validity time interval, determined by a starting and an ending time; during this time, some indicators related to a service level attribute for a specific service resource need to be verified, hence an $sl$ is described by the following tuple, composed of the set of indicators $I$, the attribute $a$ of the resource $r$, and the starting and finishing times, $t_s$ and $t_f$ respectively.

$$SL = \{sl_1, sl_2, sl_3, \ldots, sl_j\}, j \in \mathbb{N}$$
$$sl = \langle I, a^r, t_s, t_f \rangle; t_f \geqslant t_s$$

An indicator $i$ is an element of the set of the indicators $I$. An indicator is described by a mathematical tuple composed of a conditioned value $c\ k\ u$ of the attribute $a^{sl}$ for the resource $r^{sl}$. A condition $c$ belongs to the set of conditions $C$; in case any condition is not expressed in the contractual text, the value of $c$ will be set as $=$. A value $k$ is related to the attribute $a^{sl}$ of the resource $r^{sl}$; $u$ is the optional unit measure used to express the value $k$, belonging to the set of unit measures $U$. The conditioned value $c\ k\ u$ has to be verified through the application of the metric $m$ belonging to the set of metrics $M$. The metrics can be either atomic or composed, namely the value is obtained by combining more atomic metrics.

$$I = \{i_1, i_2, \ldots, i_j\}, j \in \mathbb{N}$$
$$i = \langle cku, m \rangle$$

$$c \in C; C = \{\leq; \geq; <; >; <>; =\}$$

$$u \in U; U = \{u_1, u_2, \ldots, u_j\}, j \in \mathbb{N}$$

$$m \in M; M = \{m_1, m_2, \ldots, m_j\}, j \in \mathbb{N}$$

## 6.2 Cloud Service Log

A cloud computing architecture is composed of a set of resources $R$, either physical or virtual.

$$P = \{R^p | R^p \subseteq R\}$$
$$V = \{R^v | R^v \subseteq R\}$$
$$R = \{r_1, r_2, r_3, \ldots, r_j\}, j \in \mathbb{N}$$

```
Access to S3 file log

Bucket owner - 5927116389e7d406047097a41cba2ef5830ad74cdaf67351d74682eeaa07ea2b
Bucket - myownlogsbucket
Time - [18/Feb/2015:10:37:23 +0000]
Remote IP - 137.43.248.70
Requester - 5927116389e7d406047097a41cba2ef5830ad74cdaf67351d74682eeaa07ea2b
Request ID - E730ACFDEBFC0AD6
Operation - REST.GET.OBJECT
Key - myapp/AWSLogs/028578912368/elasticloadbalancing/us-east-1/2015/02/18/028578912368_elasticloadbalancing_us-
east-1_MyLoadbalancer_20150218T1000Z_54.85.108.72_dzdvvvy8.log
Request URI - "GET /myownlogsbucket/myapp/AWSLogs/028578912368/elasticloadbalancing/us-east-
1/2015/02/18/028578912368_elasticloadbalancing_us-east-1_MyLoadbalancer_20150218T1000Z_54.85.108.72_dzdvvvy8.log?
X-Amz-Date=20150218T103721Z&X-Amz-Expires=300&X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-
Signature=ca71da5a8639895ffae88c18ac998b4a457559041422c5e785d11dd460fc14d1&X-Amz-
Credential=ASIAJLWYBGZX77VM6P3Q/20150218/us-east-1/s3/aws4_request&X-Amz-SignedHeaders=Host&x-amz-security-
token=AQoDYXdzEEQagAKh3dXT/b%2BuxVLeigXA/jYaewrCDf6JQFBqDUpf88cJZylVFjh3yTwhxyAE%2BTM2hiUxQb79Y08Y3O1XgUiI026%2BCW
4Zjl623HTlFZOPz0rXMcZPkfjs2ibAcDP7TReTbr7WO9rtQQO2OH7MeVdu4pJHpWPOKKQuaPqB0gfCnfOtBtZshBJt8Ki/IgHgttxbau8EwNoXhP5S
gusf/hDZSLndjJ/P%2BQUyMqoXCVJYpiTovUu5lONJmO9ua6Vu7/thJH7JD0JBJhJ%2Bxv%2BFdOsb8HhPfbTA3Rz2xU%2BfAaRA8ijRqfdpfARRPf
88%2BMy6pvG9f6yWu4AQLGMU0wFuJ92g1leIILXWkacF HTTP/1.1"
HTTP Status - 200
Error code - -
Bytes Sent - 331
Object Size - 331
Total time - 43 [ms. Is measured from the time your request is received to the time that the last byte of the
response is sent.]
Turn-Around Time - 43 [ms. Is measured from the time the last byte of your request was received until the time the
first byte of the response was sent.]
Referrer - "https://s3-console-us-standard.console.aws.amazon.com/GetResource/Console.html?region=us-east-
1&pageLoadStartTime=1424255801228&locale=en"
User Agent - "Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/40.0.2214.111
Safari/537.36"
```

Figure 3: Amazon S3 Sever Access Log.

A resource $r$ is described by a set of attributes $A$, e.g., filename, date of creation, size. Let $a$ be an attribute, it belongs to the set of attributes $A$.

$$A = \{a_1, a_2, a_3, \ldots, a_j\}, j \in \mathbb{N}$$
$$r = \{A^r | A^r \subseteq A\}$$
$$|A^r| \neq 0 \forall r \in R$$

**Theorem 1.** The set of attributes $A$ cannot be empty.

*Proof.* Every resource $r \in R$ is described by a set of attributes $A^r$ which is subset of $A$.

$$A = \bigcup A^r_{\forall r \in R} \Rightarrow |A| = \bigcup |A^r|$$
$$\overset{by def.}{\Rightarrow} |A^r| \neq 0 \forall r \in R$$
$$\Rightarrow |A| \neq 0$$

□

During the execution of a Cloud service, the value of an attribute of a resource is subject to changes via an operation $o$. An operation $o$ is an element of the set of operations $O$. Each operation is described by a mathematical tuple composed of a sender $s$ that is the executor of such operation, a result $value(a^r)$ that describes the value assigned to attribute $a$ of resource $r$, an operation resource $r$, an attribute $a$, and an operation time $t_o$. The operation value is a mathematical function that assigns a value to an attribute of a resource. The assigned value can be of numeric type; moreover it can be associated to an optional unit measure $u$. A sender $s$ is an entity (process) performing operations in the Cloud, e.g., an IP address or a Dropbox process. Let $s$ be a sender, it belongs to the set of senders $S$.

$$S = \{s_1, s_2, s_3, \ldots, s_j\}, j \in \mathbb{N}$$
$$O = \{o_1, o_2, o_3, \ldots, o_j\} j \in \mathbb{N}$$
$$o = \langle s, a^r, value(a^r), u, t_o \rangle$$
$$u \in U$$
$$value : a^r \rightarrow value(a^r) = k \in \mathbb{Z}$$

The information about the sequence of operations are stored in a cloud log $cl$, which is an element of the set of cloud logs $CL$. Every log is composed of a set of operations $O^{cl}$ concerning an attribute $a$ of a resource $r$; the set $O^{cl}$ is included in the set of all the operations $O$.

$$CL = \{cl_1, cl_2, cl_3, \ldots, cl_j\}, j \in \mathbb{N}$$
$$cl = O^{cl}; O^{cl} \subseteq O$$

## 6.3 Violation Detection Capability

An SLA violation detection capability is responsible to perform a comparison between an indicator of an attribute for a resource expressed in a service level $i^{sl} \in I$ and a set of operations of a specific cloud logs $O^{cl}$ where the sequence of values of such attribute is contained. Such comparison is evaluated according to the metric $m^i$used to calculate the value. Formally, the aforementioned entities are correlated in the following tuple. A comparison $n$ belongs to the set of comparisons $N$. A comparison tuple is composed of a cloud log $cl$, an indicator that has to be verified $i^{sl} \in I$, and the comparison time $t_n$.

$$N = \{n_1, n_2, n_3, \ldots, n_j\}, j \in \mathbb{N}$$
$$n = \langle cl, i^{sl}, t_n \rangle$$
$$t_e \geq t_{o^{cl}}$$
$$\forall o \in O^{cl}, \forall i \in I^{sl} (a^r)^{O^{cl}} = (a^r)^{I^{sl}}$$

**Definition 1.** SLA Violation
Given a comparison $n \in N$ and a service level $sl \in SL$, the comparison is considered an SLA violation if the values of the set of operations $O^{cl}$ composing the cloud log $cl$ on the attribute $a$ of the resource $r$ at the time $t$ are different from the conditioned value $cku$ of the related indicator $i^{sl}$ about the service level $sl$, on the same attribute $a$ of the same resource $r$. The validity has to be determined during the correct time interval, namely the times of the operations have to be included in the time interval $t_f(i) - t_s(i)$; the value $value(a^r)_t$ is obtained by the application of the metric $m^i$.

$$m^i(a^r) = value(a^r) \neq (cku)^i \quad (1)$$
$$t_s(i) \leq t_{value(a^r)} \leq t_f(i) \quad (2)$$

# 7 CASE STUDY: AMAZON S3

The aim of this section is to map the formal model of the previous section onto the Amazon S3 information illustrated in Section 5.

## 7.1 Service Level Agreement

The Amazon S3 SLA has a validity time that begins when a customer subscribes begins to have access to the S3 service until it decides to terminate. We consider the solar year 2015 as the S3 SLA validity time.

$$L = \{S3\}$$
$$S3 = \langle S3SL, 01/01/2015, 31/12/2015 \rangle$$

Among all the legal constraints, we consider the service level expressed as Monthly Uptime Percentage, MUP for short. *Amazon Web Services will use commercially reasonable efforts to make Amazon S3 available with a Monthly Uptime Percentage [...] of at least 99.9% during any monthly billing cycle*. We consider February 2015 as the billing month range.

$$a = MonthlyUptimePercentage$$
$$r = S3server$$
$$S3SL = \{MUP\}$$
$$MUP = \langle I, MonthlyUptimePercentage^{S3server}, 01/02/2015, 28/02/2015 \rangle$$
$$I = \{i_1\}$$
$$c = atleast = \geq$$
$$i = \langle \geq 99, , MUP_m \rangle$$

The unit measure is not expressed, so we can discard it due it is defined optional.

Recalling the definition of such attribute: *Monthly Uptime Percentage is calculated by subtracting from 100% the average of the Error Rates from each five minute period in the monthly billing cycle.* Error Rate is *the total number of internal server errors returned by Amazon S3 as error status InternalError or ServiceUnavailable divided by the total number of requests during that five minute period*(Amazon).

The metric for calculating MUP is a composed metric, because it needs the computation of the *AverageErrorRate* that depends on *5minErrorRate*, depending again on *5minRequests* .

$$M = \{MUP_m, AverageErrorRate_m, 5minErrorRate_m, 5minRequests_m\}$$
$$MUP_m = 100\% - AverageErrorRate_m$$
$$AverageErrorRate_m = (\sum_{0}^{8640} 5minErrorRate_m)/8640$$
$$5minRequests_m = \sum_{0}^{+5min} ErrorCode$$
$$5minErrorRate_m = (\sum_{0}^{+5min} ErrorCode = InternalError\ ORServiceUnavaiable)/5minRequests$$

## 7.2 S3 Server Access Log

The SLA violation detection capability collects every five minute period time the S3 Server Access Log files available in S3 where every HTTP request made to the service and its response is stored (see Figure 3). The information from those logs are mapped in our formal model in the following way.

$$R = \{S3server\}$$
$$A = \{MonthlyUptimePercentage\}$$
$$S3server = \{\{MonthlyUptimePercentage\}^{S3server}\}$$

During the execution of a Cloud service, the value of an attribute of a resource is subject to change via an operation $o$. An operation $o$ is an element of the set of operations $O$. Each operation is described by a mathematical tuple composed of a sender $s$ that is the executor of such operation, a result $value(a^r)$ that describes the value assigned to attribute $a$ of resource $r$,

an operation resource $r$, an attribute $a$, and an operation time $t_o$. The operation value is a mathematical function that assigns a value to an attribute of a resource. The assigned value can be either a numeric or textual; moreover it can be associated to an optional unit measure. A sender $s$ is an entity (process) performing operations in the Cloud. Let $s$ be a sender, it belongs to the set of senders $S$.

$$S = \{137.43.248.70\}$$

The sender is the Remote IP address field of the log in Figure 3.

$$O = \{REST.GET.OBJECT\}$$

The operation is the Operation field of the same S3 Server Access Log file. The components of the operation are described in the following tuple.

$$REST.GET.OBJECT = \langle 137.43.248.70,$$
$$MonthlyUptimePercentage^{S3server},,,$$
$$18/Feb/2015:10:37:23+0000\rangle$$

The value component of the tuple is empty, as well as the unit measure. More precisely, the value component is not either InternalError or ServiceUnavailable.

### 7.3 Violation Detection Capability

In order to build a log $cl \in CL$ the SLA violation detection capability translates many S3 Server Access Log file, covering an amount of time to be decomposed in five minutes periods From this mapping, the S3 Server Access Log file operations mapped to cloud log $O^{cl}$ feed the metric $MUP_m^i$ in order to determine the elements of the set of comparisons $N$.

## 8 DISCUSSION

SLAFM is devoted to formalize a capability to manage SLAs violation detections for cloud services. The proposed approach concerns the representation of information from both the SLAs and the cloud logs in a specific format. One of the strengths of such approach is its extensibility; because the formal model is a high level abstraction, it can be enriched with additional information.

The entity sender is included in this formal model because considered necessary to easy the eventual forensic investigation triggered once a violation happens. The senders are important to reconstruct an events time-line more efficiently and to relate the flow of the operations stored by cloud logs per authors.

An extension of the model can be the necessity of managing some legal principles, thus a specific set of formal rules can be added to the formal model. The comparison among a service level and the operations of the cloud logs have to be memorized because they can be necessary for computing comparisons of other service levels. This depends on how the SLAs relate the service levels and the indicators.

A dedicated system module reacting to the detected SLA violations is out of the SLAFM formal model duties, but it can be considered as a possible extension or integration of both the formal model and the system implementing our capability designed in Figure 2.

## 9 CONCLUSION AND FUTURE WORK

The management of Service Level Agreement contracts for cloud services provisioning is an extremely challenging and active research trend. Several proposals have been made in literature to approach the QoS levels guaranteeing issues described in the contracts with the purpose of monitoring a platform behaviour.

The main objective of such research works is to detect whether the agreed resources performances are respected, without considering the cloud services logs. In some papers, the issue is formally modelled for being subsequently implemented; in other ones, a specific framework is proposed demonstrating the manner how such monitoring is performed.

In the future we intend to prototype a system designed in Figure 2 based on the SLAFM formal model proposed in Section 6. The prototype will simulate cyber attacks to a cloud service regulated by an SLA, and it will perform comparisons among the logs and the SLA constraints. Moreover, we want to test the number of SLA formal rules violations that can be detected in a specific amount of time.

We strongly believe that such capability can enhance the security strategies of a Cloud platform, so that it can be considered as a must requirement for it, and very likely becoming a standard in the next years.

## REFERENCES

Amazon Simple Storage Service (S3), [online] http://aws.amazon.com/s3/sla/, accessed on 22/02/2015.

Baset, S.A., 2012. Cloud SLAs: present and future. *ACM*

*SIGOPS Operating Systems Review*, vol. 46, no. 2, pp. 57-66.

Ben-Ari, M., 1993. *Mathematical Logic for Computer Science*, SPRINGER, first edition.

Brandic, I., Emeakaroha, V.C., Maurer, M., Dustdar, S., Acs, S., Kertesz, A., Kecskemeti, G., 2010. LAYSI: A Layered Approach for SLA-Violation Propagation in Self-Manageable Cloud Infrastructures, *COMPSACW Workshop*, 2010, pp.365 - 370, IEEE.

Cedillo, P., Gonzalez-Huerta, J., Abrahao, S., Insfran, E., 2014. Towards Monitoring Cloud Services Using Models@run.time, *International Workshop on Models at run.time*, to appear.

CSA, 2013. Mapping the Forensic Standard ISO IEC 27037 to Cloud Computing, *Cloud Security Alliance*, [online] https://cloudsecurityalliance.org/download/mapping-the-forensic-standard-isoiec-27037 -to-cloud-computing/

Czajkowski, K., Foster, I., Kesselman, C., Sander, V., Tuecke, S., 2002. SNAP: A Protocol for Negotiating Service Level Agreements and Coordinating Resource Management in Distributed Systems, *Job scheduling strategies for parallel processing*, Springer Berlin Heidelberg, pp. 153-183.

De Marco, L., Abdalla, S., Ferrucci, F., Kechadi, M-T., 2014. Formalization of SLAs for Cloud Forensic Readiness, *Proc. ICCSM Conference*, Academic Conferences and Publishing International Limited, Reading, UK, Dr. Barbara Endicott-Popovsky University of Washington, Seattle, USA Edition, pp. 42 - 50.

Emeakaroha, V.C., Calheiros, R.N., Netto, M.A., Brandic, I., De Rose, C.A., 2010. DeSVi: An Architecture for Detecting SLA Violations in Cloud Computing Infrastructures, *ICST Cloud Comp Conference.*

Emeakaroha, V.C., Ferreto, T.C., Netto, M.A.S., Brandic, I., De Rose, C.A.F., 2012. CASViD: Application Level Monitoring for SLA Violation Detection in Clouds, *IEEE COMPSAC Conference*, pp. 499 - 508.

Emeakaroha, V.C., Netto, M.A., Calheiros, R.N., Brandic, I., Buyya, R., De Rose, C.A., 2012. Towards Autonomic Detection of SLA Violations in Cloud Infrastructures, *Future Generation Computer Systems*, vol. 28, issue 7, pp. 1017-1029.

Ghosh, N., Ghosh, S.K., 2012. An Approach to Identify and Monitor SLA Parameters for Storage-as-a-Service Cloud Delivery Model, *GC Wkshps*, pp. 724-729.

Information Technology Infrastructure Library (ITIL), http://www.itil-officialsite.com

Ishakian, V., Lapets, A., Bestavros, A., Kfoury, A., 2011. Formal Verification of SLA Transformations, *IEEE World Congress on Services*, pp. 540-547.

Larson, K. D., The role of service level agreements in IT service delivery, *Information Management & Computer Security*, vol. 6, issue 3, pp. 128-132, 1998.

Liu, F., Tong, J., Mao, J., Bohn, R., Messina, J., Badger, L., Leaf, D., 2011. *NIST cloud computing reference architecture*, NIST special publication, 500, 292.

Maurer, M., Brandic, I., Sakellariou, R., 2012. Self-Adaptive and Resource-Efficient SLA Enactment for Cloud Computing Infrastructures, *IEEE CLOUD Conference*, pp. 368 - 375.

Mell, P., Grance, T., 2011. *Final Version of NIST Cloud Computing Definition*, [online], http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

Morshedlou, H., Meybodi, M.R., 2014. Decreasing Impact of SLA Violations: A Proactive Resource Allocation Approach for Cloud Computing Environments, *IEEE Transactions on Cloud Computing*, vol.2, no.2, pp. 156-167.

Muller, C., Oriol, M., Franch, X., Marco, J., Resinas, M., Ruiz-Corts, A., Rodriguez, M., 2014. Comprehensive explanation of SLA violations at runtime. *IEEE Transactions on Services Computing*, vol. 7, issue 2, pp. 168-183.

Paschke, A., Bichler, M., 2008. Knowledge Representation Concepts for Automated SLA Management, *Decision Support Systems*, vol. 46, issue 1, pp. 187-205.

Patel, P., Ranabahu, A.H., Sheth, A.P., 2009. Service Level Agreement in Cloud Computing, [online], http://corescholar.libraries.wright.edu/knoesis/78

Rumbaugh, J., Jacobson, I., Booch, G. 2004. *Unified Modeling Language Reference Manual, The*. Pearson Higher Education.

Skene, J., Skene, A., Crampton, J., Emmerich, W., 2007. The Monitorability of Service-Level Agreements for Application-Service Provision, *Proc. International Workshop on Software and Performance*, pp. 3-14.

Unger, T., Leymann, F., Mauchart, S., Scheibler, T., 2008. Aggregation of Service Level Agreements in the Context of Business Processes, *Proc. ICEDOC Conference*, pp. 43-52.