# SLA-based Service Selection for Multi-Cloud Environments

Ahmed Taha, Salman Manzoor and Neeraj Suri
*TU Darmstadt, Germany*
*Email:{ataha, salman, suri}@deeds.informatik.tu-darmstadt.de*

***Abstract*—The Multi-Cloud is a new paradigm that facilitates customers to host their applications and data onto multiple Clouds. It also allows the Cloud Service Providers (CSPs) to offer new functionalities in order to satisfy the diverse customer requirements. This is done by offering a combination of services from multiple CSPs that can individually only provide limited functionalities. The issue of "services selection" using Service Level Agreement's (SLA's) has been well-studied in Cloud computing. However, the Multi-Cloud environment introduces dependencies across the services thus entailing the need of new approaches.**
**In this paper, we propose a Multi-Cloud service selection approach. The novelty of our approach lies in extending and adapting a single Cloud decision-making technique for the Multi-Cloud environment. The proposed framework (i) assesses the service levels provided by varied CSPs and selects the optimal combination of services from multiple CSPs that best satisfy the customer requirements, and (ii) automatically detects conflicts resulting from dependencies between the selected services. Moreover, our framework provides an explanation for the detected conflicts allowing both customers and CSPs to resolve these conflicts. We validate our framework by applying it to real-world data that leverages the standardized Cloud service level agreements structure proposed in the ISO/IEC 19086 standard.**

***Keywords*-Multi-Cloud; SLA; service dependencies;**

## I. Introduction

Providing resources and services from multiple Clouds is the ongoing evolution in Cloud computing. Improving the services quality while optimizing the services cost, the ability to migrate among several Cloud Service Providers (CSPs), avoiding vendor lock-in, and the need of specific Cloud services which are not provided elsewhere are some of the reasons for using services from multiple Clouds. Currently, two basic delivery models exist for multiple Clouds: Multi-Cloud and Federated Cloud (cf., Section III-B). Each model defines its own degree of collaboration between the involved CSPs and the way it interacts with the customers. In the Multi-Cloud model, the services offered by multiple independent CSPs are aggregated as one or more composite services. In this paper we focus on the Multi-Cloud model as it gives customers the freedom to select the Cloud providers that best satisfy their requirements. This is achieved through a third-party (termed as Multi-Cloud middleware in this paper) that is responsible to deal with CSP's application programming interface (API) variations [1].
Despite the advocated economic and performance related advantages of utilizing multiple Clouds, two basic concerns are not yet fully addressed in the community. First, with the growth of public Cloud services, various CSPs offer similar services but with different prices. Second, most of these services include dependency[1] relations accross them. For example, Dropbox depends on Amazon's S3 service for storage and on EC2 for computation [3]. This example shows a SaaS provider that depends on other IaaS providers to support its software. These dependency relations can easily introduce conflicts[2]. Thus, the question remains as how can Cloud customers (i) assess and select services for each requirement, and (ii) select an optimal combination of services from multiple providers to satisfy their requirements, taking into consideration that these concerned services can be conflicting or have different degree of importance for each customer.
As an initial effort regarding the challenge (i), different stakeholders in the Cloud community have identified that specifying measurable Service Level Objectives (SLOs)[3] in Service Level Agreements (SLAs)[4] (explained in Section III-A) are a useful mean to express services using common semantics which allow to represent both the service level required by customers and the service levels offered by CSPs. Although the state of the art predominantly focuses on the methodologies to score and rank different services offered by various CSPs, most of these evaluation methodologies:

- Focus on a single Cloud infrastructure and on the best matching CSP selection, even if the selected CSP does not fulfill "all" the customer's needs.
- Do not consider dependencies across services.

Overall, it is important to provide customers with comprehensive support that can enable them to find the best services combination that "completely" fulfill their requirements and provide an automatic detection of services' conflicts. Stimulated by these challenges, it is becoming an important issue for customers to make decisions regarding how to (a) assess CSPs' qualitative and quantitative services, and (b) analyze

[1]Service dependencies or dependency relations are the direct relations between one or more services, where a service depends on other services for the provisioning of resources/data [2].

[2]Conflict arises when a service depends on data/resources which are not provided by the corresponding dependent service.

[3]SLOs are the measurable elements of an SLA that specify the service levels required by the customers and to be achieved by the CSP.

[4]The SLA specifies how provisioning takes place as well as the respective rights and duties of both the Cloud customer and the CSP.

the composite service dependencies to handle services' conflicts. This is done through the following contributions:

1) Proposing an SLA-based Multi-Cloud service allocation approach that includes two main building blocks: (a) the Multi-Cloud SLA (termed MCSLA) construction and (b) service selection and composition.
2) Proposing a dependency representation model which captures the dependencies across services. This model is used for validating the MCSLA by checking the existence of conflicts which occur due to dependency relations between services.
3) Validating the proposed framework by evaluating CSPs' SLAs found on the public STAR (Security, Trust and Assurance Registry) [4] repository, which are complaint with the relevant ISO/IEC 19086 standard [5].
4) Developing a prototype for a control panel that implements the proposed model[5].

The rest of the paper is organized as follows. Section II describes the related work. Section III develops the background and the basic terminologies related to Cloud SLAs and the Multi-Cloud environment. This is followed by a description of the proposed framework in Section IV. Section V presents the real-world use-cases that validate the Multi-Cloud service evaluation and composition as well as dependency management.

## II. Related Work

***CSPs Assessment.*** Recent efforts have focused on specifying service levels in SLAs, e.g., Alhamad et al. [6] specified multiple metrics of IaaS in the SLA. However their technique is limited to a single CSP, and the authors did not assess or empirically validate the preseneted attributes. In [7]–[9], the authors utilized multi-criteria decision making (MCDM) methodologies to rank CSPs according to the customers' requirements. However, all these techniques focused on finding the best single CSP. Bernsmed et al. [10] presented a method for managing the SLA life cycle in the context of Federated Cloud services. However, they did not elaborate how to manage benchmarking. Jrad et al. [11] proposed SLA based brokering service for Federated Cloud. Their scheme is relevant to the initial phase of our scheme in which the definition of SLAs is collected and ranked according to customers' requirements. In [12], the authors presented a metrics framework for CSP's assessment and in [13] a methodology to quantitatively assess CSPs' SLAs. However in both of them, service dependencies were not considered. In our previous work [2], we presented a framework for the analysis of the SLA service dependencies. However, we only focused on a single-Cloud provider model. In this paper, we propose quantitative service-level assessment technique to find the service composition that satisfies the customer requirements. Furthermore, we extend our dependency model presented in [2] to check the existence of conflicts which occur due to dependency relations between different services offered by various CSPs.

***Multi-Cloud Model.*** A taxonomy of existing Inter-Cloud architectures and their brokering features is provided in [14]. The EU OPTIMIS project [15] developed a toolkit to optimize the Cloud life cycle. However, all the participating CSPs need to develop and maintain multiple vendor specific OPITMIS adapters to benefit from the entire toolkit (i.e., interoperability issues). In mOSAIC project [16], the development and deployment of Multi-Cloud applications are presented. However, the users need to rebuild their applications to integrate the mOSAIC API before using the framework.

## III. Basic concepts

### A. Service level agreements

A Cloud Service Level Agreement (SLA) specifies the provided services, and represents the binding commitment between a customer and a CSP. Basically each of these services contains a list of Service Level Objectives (SLOs), which are the measurable elements of an SLA that specify the service levels needed by the customers, and to be fulfilled by the CSP. The unfulfillment of any SLO would result in the violation of the SLA, and thus entail a possible penalty for the CSP.

To formalize the concept of SLA we extend the following definition presented in [2] to add services coming from different CSPs in order to support the Multi-Cloud environment.

**Definition 1.** *An SLA consists of a set of services $S = \{s_1, \ldots, s_n\}$. Each service consists of finite positive number n of SLOs $k_i$; where $i = 1 \ldots n$. Each SLO $k_i$ consists of m different values $v_i$; such that $k_i = v_{i,1}, v_{i,2}, \ldots, v_{i,m}$. Each value implies a different service level offered by the CSP and needed by the customer. The total order of service levels of each $k_i$ is defined using an order relation " $\prec_i$ ". Each $k_i$ value is mapped to a numerical value according to its order.*

### B. Multiple Clouds Delivery Models

The NIST report [17] has stated that the multiple Clouds can be used serially, when an application or service is moved from one Cloud to another, or simultaneously, when services from different Clouds are used. The reasons for selecting services and resources from multiple Clouds are various and have been reported in several research/practitioner publications (such as [18], [19]). Currently, there are two basic delivery models in place for multiple Clouds. The first is the Federated Cloud where the CSPs establish agreements with each other in order to enhance the service offer to their service consumers. The alternate is the Multi-Cloud where, unlike a federation of Clouds, the Multi-Cloud model does not imply volunteer interconnection and sharing of CSPs'

[5]The implementation is at https://github.com/amtaha/MCSLA

infrastructures. In the Multi-Cloud model, CSPs combine to provide a composite Cloud service, which is a set of services provisioned by different CSPs and aggregated as one service. In this model, the customer or a third-party (on behalf of the customer) contacts the CSPs, negotiates the terms of the services required by the customer and monitors the fulfillment of the SLAs. Where each service offered by a CSP is specified using the CSP's SLA.

It is important to note that variations between the current APIs often hinder the easy composition or configuration of services to be consumed from multiple Clouds. In order to use services from multiple Clouds in real world scenario, several additional technical barriers also need to be addressed such as interoperability, portability, data and services mobility, and middleware openness. Furthermore, to address the SLA interoperabiltiy issue and to control the interface between the customer and different providers in a Multi-Cloud scenario, a composite SLA is needed. This composite SLA (named Multi-Cloud SLA or MCSLA in this paper) contains (a) all contractual services of all involved SLAs and (b) the dependencies that exist between various services in different SLAs. In this paper we focus on the decision making problem of the selection and deployment of composite SLA while considering the customer requirements, and handling the dependencies between different services and/or SLOs.

## IV. Proposed framework

In this section we present a quantitative service-level assessment of CSPs to find the service composition that satisfies all the customer requirements. We assume that all transactions between the proposed framework and the CSPs are done through the Multi-Cloud middleware that communicates with the APIs of all the involved CSPs. In our framework, the selection of composite services and the dependency management for composing the services is performed in five main stages as shown in Figure 1. After the CSPs submit their SLAs and the customers specify their requirements in Stage (A), services assessment and selection algorithms are used to score services in Stage (B) according to the customer requirements. Based on this assessment, an optimal combination of services from multiple CSPs is chosen. This combination is used to create the MCSLA in Stage (C). Based on the MCSLA services allocation, a dependency model is created in Stage (D) to capture information about the composite services and the dependencies that occur across them. This model is specified using a machine readable format to allow automated validation for checking all the services conflict in Stage (E).

In order to allow customers to trust the result of the proposed evaluation and assessment framework, the CSPs' SLAs should be taken from a reliable/trusted source (e.g., the CSA STAR repository [4]) as shown in Figure 1.

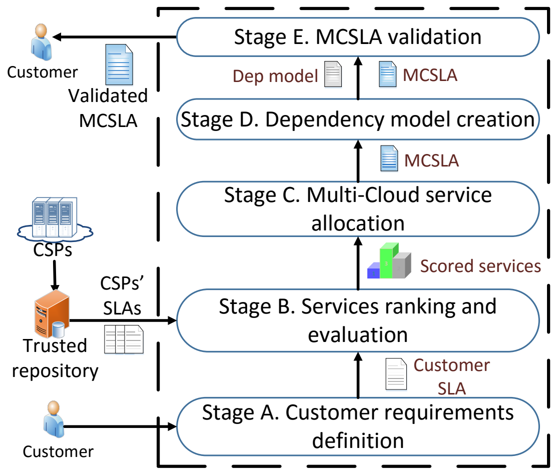
Stage E. MCSLA validation
Customer
Validated MCSLA
Dep model
MCSLA
Stage D. Dependency model creation
MCSLA
Stage C. Multi-Cloud service allocation
CSPs
Scored services
CSPs' SLAs
Stage B. Services ranking and evaluation
Trusted repository
Customer SLA
Stage A. Customer requirements definition
Customer

Figure 1: Proposed framework stages

### A. Stage A: Customer Requirements Definition

In this stage, the Cloud customers create their set of requirements and specify their preferences based on the same SLA template used by the CSPs to specify their offered services.

### B. Stage B: Services Evaluation

The CSPs quantitative service level assessment according to the customer requirements and preferences is developed in this stage. Using this assessment services are scored based on the customer requirement level for each service. The SLA services assessment is performed in the following four phases:

*1) Phase 1. Weights Assignment:* In order to find the best combination of services for each customer, the relative importance levels of the customer requirements for each service are assigned as weights. We utilize qualitative terms to specify the importance of each SLO, where customers can assign qualitative terms as weights to each SLO to indicate their priorities (*Not-Required (NR)*, *Low-Important (LI)*, *Highly-Important (HI)*, and *Extremely-Important (EI)*). These qualitative terms are further transformed to a quantitative values. The weights assignment is explained using a real-world case study in Section V.

*2) Phase 2. Services Quantification:* In order to evaluate CSPs' SLAs, a measurement model for different SLOs should be defined. We use the relative ranking model proposed in [13], which is based on a pairwise relation of services offered by the CSPs and required by the customers such that:

$$CSP_{1,k}/CSP_{2,k} = \frac{v_{1,k}}{v_{2,k}} \tag{1}$$

where $CSP_{1,k}/CSP_{2,k}$ indicates the relative rank of $CSP_1$ over $CSP_2$ for a particular SLO $k$. Similarly, $CSC_k/CSP_{1,k}$ indicates the relative rank of the Cloud Service Customer $CSC$ over $CSP_1$ regarding $k$, which specifies whether $CSP_1$

satisfies $CSC$ requirement or not. As a result, for each SLO we have a one to one evaluation matrix ($EM$) of size $(n+1)$ x $(n+1)$ if there is a total of $n$ CSPs and one CSC, so that:

$$EM_k = \begin{array}{c|cccc} & CSP_{1,k} & \dots & CSP_{n,k} & CSC_k \\ \hline CSP_{1,k} & V_{1,1}^k & \dots & V_{1,n}^k & V_{1,c}^k \\ CSP_{2,k} & V_{2,1}^k & \dots & V_{2,n}^k & V_{2,c}^k \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ CSC_k & V_{c,1}^k & \dots & V_{c,n}^k & V_{c,c}^k \end{array} \qquad (2)$$

where $V_{i,j}^k = CSP_{i,k}/CSP_{j,k} = \dfrac{v_{i,k}}{v_{j,k}}$ and $V_{i,c}^k = CSP_{i,k}/CSC$ Next, the respective scores for all the CSPs and the customer, for each SLO, are obtained by calculating the evaluation vector ($EV_k$) of the corresponding evaluation matrix $EM_k$ . The $EV$ is an approximation eigenvector of the $EM$ which indicates the numerical ranking of all the CSPs by specifying an order of preference among them, as indicated by the ratios of the numerical values.

$$EV_k = \begin{bmatrix} N_{1,k} & N_{2,k} & \dots & N_{n,k} & N_{c,k} \end{bmatrix}^T \qquad (3)$$

where $N_{1,k}$ is a numerical value representing the relative rank of $CSP_1$ to other CSPs as well as the CSC regarding an SLO $k$. $N_{c,k}$ is the relative rank of the CSC's required service level with respect to the service levels offered by the CSPs.

*3) Phase 4. Services Selection:* Based on the relative ranking of CSPs according to the CSC requirements (determined in the previous phase), finding the best combination of SLOs that can collectively satisfy the customer needs is performed in this phase. The selection of the set of feasible services with respect to the set of customer requirements is performed using Algorithm 1. Each SLO evaluation vector (Equation 3) serves as an input to this algorithm to determine the services from CSPs that are compatible with the customer requirements.

**Algorithm 1** Services selection

```
1: procedure MCSLASELECTION
2:     if $N_{c,k} = N_{1,k} \vee \dots \vee N_{n,k}$ then          ▷ Condition 1
3:         $N_{m,k} = N_{1,k} \vee N_{2,k} \vee \dots \vee N_{n,k}$
4:     else if $N_{c,k} < N_{1,k} \vee \dots \vee N_{n,k}$ then     ▷ Condition 2
5:         $N_{m,k} = min(N_{1,k}, N_{2,k}, \dots N_{n,k})$
6:     else if $N_{c,k} > N_{1,k} \vee \dots \vee N_{n,k}$ then     ▷ Condition 3
7:         $N_{m,k} = max(N_{1,k}, N_{2,k}, \dots N_{n,k})$
8:     end if
9: end procedure
```

In the algorithm we describe different conditions for services selection in comparison with the customer requirements, which we explain using an example. Consider an SLO $k$ offered by three providers with three different service levels values $v_{1,k}$, $v_{2,k}$, $v_{3,k}$ and required by a customer with value $v_{c,k}$. After the $EV_k$ is calculated using Equation 3, the following conditions take place:

**First Condition:** If $N_{c,k}$ is equal to any of the three relative ranking values (e.g., $N_{c,k} = N_{3,k}$), the MCSLA numerical relative ranking value ($N_{m,k}$) will be assigned the same value as $N_{3,k}$ which means $CSP_3$ is selected to provide the required service.

**Second Condition:** If the first condition is not satisfied and $N_{c,k}$ is lower than the CSPs (e.g. $N_{c,k} < N_{1,k}$ and $N_{2,k}$, which means $CSP_{1,k}$ and $CSP_{2,k}$ are over-provisioning the customer's requirement). Then $N_{m,k}$ is equal to the minimum of the over-provisioning CSPs values ($N_m$ is equal to the closest value to the customer required value).

**Third Condition:** If neither the first nor the second conditions are satisfied, then the providers are under-provisioning the customer requirement. Thus, the provider with the highest offered level is chosen (the one closest to the customer's requirement).

*4) Phase 5. Services Aggregation:* In this phase, the evaluation vector of each SLO (Phase 3) is aggregated with the normalized weights assigned by the customer in Phase 1 in order to to give an overall assessment of the service levels.

$$EV_{aggregated} = \begin{bmatrix} EV_{k_1} & \dots & EV_{k_n} \end{bmatrix} . \begin{bmatrix} W \end{bmatrix}^T \qquad (4)$$

where $W$ is the set of normalized weights of different SLOs such that $W = w_{k_1}, w_{k_2}, \dots, w_{k_n}$.

## C. Stage C: Multi-Cloud Service Allocation

After finding the best combination of services that collectively satisfy "all" the customer needs in Stage B, mapping this combination to the Multi-Cloud SLA (named MCSLA) is the target of this stage. This is done by constructing the MCSLA template and then using the services scores provided in the previous stage to build the composition of the feasible services. The main purpose of constructing such MCSLA is addressing the SLA interoperability issue in a Multi-Cloud.

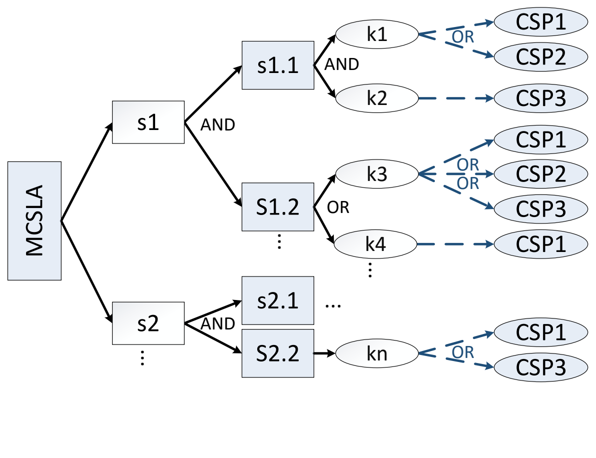

Figure 2: MCSLA "AND-OR" tree

In order to automatically and quantitatively assess composite services based on the MCSLA, we use the classical "AND-OR" trees to identify the set of services that satisfy the customer requirements. This is done by forming compositions of services represented by "AND" or "OR" relationships. As inferred from their name, "AND" relationships represent the "necessary" customer requirements and "OR" relationships are more adequate to model "optional" requirements. If more than one provider satisfy the customer requirement for a specific service, these providers are mapped using "OR" relationships and then based on additional criteria (e.g., cost, previous customers rating, and etc.) the best CSP is selected (i.e., the CSP with the lowest cost for example) as depicted in Figure 2.
After mapping the selected services to the MCSLA, a dependency model is created in Stage D to capture information about MCSLA services and the dependencies that occur across them.

### *D. Stage D: Dependency Model Creation*

In order to model services dependencies, it is important to specify on which SLOs the fulfillment of a specific service depends and how it depends on these SLOs.
We model a *MCSLA* by a tuple $MCSLA = (S, l, \rightarrow_S, K, \rightarrow_K, v)$, such that:

- $S$ is a set of services $s$ (i.e., composition of services offered by multiple CSPs as per their SLAs) such that $S = s_{i,1} \odot s_{i,2} \odot \ldots \odot s_{i,m}$; $i = 1 \ldots n$ where $n$ is the number of CSPs and $m$ is the number of services.
  - $s_{i,j}$ specifies service $s_j$ provided by $CSP_i$
  - $\odot$ is a generic composition operator. We are interested in which services are being composed but not how they are being composed, thus we model the composition process using a generic composition operator which represents any type of composition.
- $\rightarrow_S \subseteq S \times (S \cup K)$ models service dependencies.
- For two CSPs, $CSP_i$ and $CSP_j$, we write $s_{i,1} \rightarrow_S s_{j,2}$ if $s_{i,1}$ (dependent service) depends on $s_{j,2}$ (antecedent service) where $i \in \{1 \ldots n\}$ and $j \in \{1 \ldots n\}$. If $i = j$ then $s_1$ and $s_2$ are provided by the same CSP.
- Each service is decomposed into a Boolean combination of SLOs $k_1 \otimes k_2 \otimes \ldots \otimes k_o$; where $\otimes \in \vee, \wedge$ and $o$ is the number of SLOs.
- $K$ is a set of SLOs where:
  - $\rightarrow_K \subseteq K \times K$ models SLO dependencies. We have $k_{i,1} \rightarrow_K k_{j,2}$ if $k_{i,1}$ (dependent SLO) depends on $k_{j,2}$ (antecedent SLO).
  - $v : K \mapsto V$ is the values in $V$ assignment to SLOs, where $V$ defines the set of all values of each SLO in $K$.
  - $\forall k \in K \mid \nexists k \in K : v_{i,k} \prec v_{j,k}$ models the ordering over the possible CSPs for each SLO using a preference relation $\prec$; furthermore, $v_{i,k}$ and $v_{j,k}$ represent $CSP_i$ and $CSP_j$ offered values for an SLO ($k$). We define $v_{i,k} \prec v_{j,k}$ if only $CSP_j$ satisfies the customer requirement. If both CSPs satisfy the customer requirement regarding SLO $k$ ($v_{i,k} = v_{j,k}$) then the one with min cost is chosen.
  - A constraint on SLO dependency relation is specified using a constraint set $C_v^{\rightarrow_K} \subseteq K \times K \times \{=, \neq, <, \leq, >, \geq\}$. A constraint $(k_{i,1}, k_{j,2}, \equiv) \in C_v^{\rightarrow_K}$ is satisfied if the values of $k_{i,1}$ and $k_{j,2}$ are related by the given comparison, i.e., $v(k_{i,1}) \equiv v(k_{j,2})$. A dependency relation $k_{1,i} \rightarrow_K k_{2,j}$ is called valid, written $valid_{C_v^{\rightarrow_K}}(k_{i,1}, k_{j,2})$, if the relation satisfies all its constraints, i.e., $\forall (k'_{i,1}, k'_{j,2}, \equiv) \in C_v^{\rightarrow_K}.(k_{i,1} = k'_{i,1} \text{ and } k_{j,2} = k'_{j,2}) \Rightarrow v(k_{i,1}) \equiv v(k_{j,2})$.

Based on the defined dependency relations, a model that describes the SLOs and services along with the dependencies across them is developed. The model is specified using a machine readable format in order to have an automatic validation of the MCSLA. An example of the model format is an XML data structure using an XML Schema. In this Schema, service and SLOs dependencies are modeled, as well as the services/SLOs role as dependent or antecedent.
Following the MCSLA and dependency model construction, the MCSLA is validated to check service conflicts.

### *E. Stage E: MCSLA Validation*

The validation process is performed in the following steps:

Step 1. Extract the dependency model ID, SLO ID, SLA ID, and the dependency ID of each two dependent SLOs defined in the XML schema. Note that, each SLO has a unique ID and an SLA ID as depicted in the dependency model in Section IV-D. Moreover, each dependency relation in the same dependency model has a unique ID.

Step 2. Extract the SLO values of each two dependent SLOs as well extracting the constraint comparative (i.e., $=, \neq, <, \leq, >, \geq$) of the dependency relation.

Step 3. Checking if the two dependent SLOs' values satisfy the constraint.

If the constraint between the two SLOs is not satisfied, this indicates a conflict between these SLOs. The SLO ID, SLA ID, dependency ID and dependent SLO ID of the affected SLO are saved, and the evaluation continues to determine further conflicts. At the end of the process, a complete list of all conflicts between SLOs are reported along with the explanation of these conflicts. Algorithm 2 details the necessary steps for extracting the values of validating a dependency relation between SLOs in a pseudocode notation.

## V. Case Study: Evaluation of CSPs Based on their SLAs

This section shows an empirical validation of the proposed framework through a scenario that uses real world SLA

**Algorithm 2** Validation

```
Input: String mcslaID, String depmodelID, String depID,
String depSLO, String antSLO, String depConst
Output: List affectedSLOs
List depList = NIL
Value depValue = NIL
Value antValue = NIL
List depModels = getModels(depmodelID, mcslaID)
for model $\in$ depModels do
    depList = getDependencies(depID, depmodelID)
    for depend $\in$ depList do
        depValue=getValue(depend.depSLOvalue, depID)
        antValue=getValue(depend.antSLOvalue, depID)
        depConst=getString(depend.constraint, depID)
        if validate(depValue, antValue, depConst) $\neq true$ then
            affectedSLOs.add(depend.depSLO)
        end if
    end for
end for
```

information provided by the STAR repository. The rationale for this decision is that (i) to the best of our knowledge there are not other publicly available Cloud SLA repositories, (ii) most CSPs will not provide their SLA information to non-customers, and (iii) major CSPs are still in the process of restructuring their SLAs by leveraging the recently published ISO/IEC 19086. This scenario demonstrates how a Cloud customer can apply the framework presented in this paper to compare three CSPs based on their advertised SLAs (compliant with the ISO/IEC 19086 standard). Note that, the qualitative metrics are specified as service levels using Definition 1 such as $level_1, level_2$, and $level_3$ are modeled as $\frac{1}{3}, \frac{2}{3}, \frac{3}{3}$. Furthermore, $no, yes$ metrics are denoted as $level_0$,$level_1$ respectively. All CSPs' SLOs are normalized to the customer requirements to prevent the masquerading effect[6]. Weights assigned by the customers to indicate their priorities are specified as numerical value such that *EI* and *NR* indicate a relative value *1* and *0* respectively. *HI* and *LI* can be considered any intermediate values between *1* and *0*. In this analysis they indicate a relative rank value *0.7* and *0.3* respectively.

Furthermore, we consider dependencies between services and SLOs which are going to be validated using the Multi-Cloud validation model presented in Section IV-D so that:

- $IVS2.1$ depends on $IVS1.1$ (this dependency relation is denoted as $Dep_1$ in Table I). This is modeled as $IVS2.1 \rightarrow_K IVS1.1$ with constraint $(IVS2.1, IVS1.1, =) \in C_v^{\overrightarrow{K}}$.

[6]The masquerading effect happens when the overall aggregated service level values depend on the controls with high number of SLOs, thus negatively affecting controls with fewer number of SLOs, even though possibly more critical

- In the same way, $Dep_2$ and $Dep_3$ are specified as $IVS2.1 \rightarrow_K IVS3.1$ with $(IVS2.1, IVS3.1, =) \in C_v^{\overrightarrow{K}}$ and $IVS2.3 \rightarrow_K IVS3.2$ with $(IVS2.3, IVS3.2, =) \in C_v^{\overrightarrow{K}}$ respectively.
- Finally, $IS1.1$ and $IS1.2$ are symmetrically dependent where $IS1.1 \rightarrow_K IS1.2 \wedge IS1.2 \rightarrow_K IS1.1$ with $(IS1.1, IS1.2, =) \in C_v^{\overrightarrow{K}}$ (i.e., $Dep_4$).

For this evaluation technique, the customer specifies her/his requirements and considers different relative importance (i.e., weights) for all of the SLOs. For the *Infrastructure & Virtualization Security* control of Cloud SLA, there are three sub-services ($IVS1, IVS2$ and $IVS3$) which are further divided into SLOs ($IVS1.1$, $IVS1.2$, $IVS2.1$, ...). Using the data shown in Table I, Equation 1 is used to define the $IVS1.2$ pairwise relation such that:

$$CSP_{1,IVS1.2}/CSP_{2,IVS1.2} = \frac{2}{3}/\frac{3}{3},$$
$$CSC_{IVS1.2}/CSP_{3,IVS1.2} = \frac{3}{3}/\frac{2}{3}$$

The evaluation matrix of $IVS1.2$ is then calculated using Equation 2 so that:

$$EM_{IVS1.2} = \begin{array}{c c} & \begin{array}{cccc} CSP_1 & CSP_2 & CSP_3 & CSC \end{array} \\ \begin{array}{c} CSP_1 \\ CSP_2 \\ CSP_3 \\ CSC \end{array} & \begin{pmatrix} 1 & 2/3 & 1 & 2/3 \\ 3/2 & 1 & 3/2 & 1 \\ 1 & 2/3 & 1 & 2/3 \\ 3/2 & 1 & 3/2 & 1 \end{pmatrix} \end{array}$$

Then the relative ranking of the CSPs for $IVS1.2$ is calculated using Equation 3.

$$EV_{IVS1.2} = \begin{array}{c} \begin{array}{cccc} N_1 & N_2 & N_3 & N_c \end{array} \\ \begin{pmatrix} 0.2 & 0.3 & 0.2 & 0.3 \end{pmatrix} \end{array}$$

This implies that only $CSP_2$ satisfies the customer requirement for $IVS1.2$ as $N_{2,IVS1.2} = N_{c,CO1.2}$. Using Algorithm 1, since $N_{c,IVS1.2}$ is equal to $N_{2,IVS1.2}$ then $N_{m,IVS1.2}$ is equal to $N_{2,IVS1.2}$, this means $CSP_2$ is selected for providing *IVS1.2* SLO according to the customer requirement.

Similarly, we calculate $EM_{IVS1.1}$, $EV_{IVS1.1}$ and $N_{m,IVS1.1}$ such that:

$$EV_{IVS1.1} = \begin{array}{c} \begin{array}{cccc} N_1 & N_2 & N_3 & N_c \end{array} \\ \begin{pmatrix} 0.333 & 0 & 0.333 & 0.3333 \end{pmatrix} \end{array}$$

This implies that both $CSP_1$ and $CSP_3$ satisfy the customer requirement for $IVS1.1$. Using Algorithm 1, $N_{m,IVS1.1}$ is equal to $N_{1,IVS1.1}$ and $N_{3,IVS1.1}$, this means either $CSP_1$ or $CSP_3$ is selected for providing *IVS1.1* SLO according to the customer requirement. Thus the customer can choose any of two providers according to other factors such as cost or previous customers feedback.

The $IVS1$ evaluation vector is then premeditated by aggregating $EV_{IVS1.1}$ and $EV_{IVS1.2}$ with the "normalized" weights where the customer specified *EI* and *HI* for *IVS1.1* and

Table I: Excerpt of SLA's from CSPs and customer requirements.

| Cloud SLA based on STAR [4] | | | | | CSPs | | | Customer (CSC) | |
|---|---|---|---|---|---|---|---|---|---|
| Services | | | SLOs | | $CSP_1$ | $CSP_2$ | $CSP_3$ | req | weight |
| | | | name | dep. | | | | | |
| *Root* | Infrastructure & Virtualization Security $IVS$ | Intrusion Detection $IVS1$ | $IVS1.1$ | $Dep_1$ | *yes* | *no* | *yes* | *yes* | EI |
| | | | $IVS1.2$ | | $level_2$ | $level_3$ | $level_2$ | $level_3$ | HI |
| | | Audit Logging $IVS2$ | $IVS2.1$ | $Dep_1$ $Dep_2$ | *yes* | *yes* | *no* | *yes* | HI |
| | | | $IVS2.2$ | | *no* | *yes* | *yes* | *no* | NR |
| | | | $IVS2.3$ | $Dep_3$ | *no* | *yes* | *yes* | *yes* | EI |
| | | | $IVS2.4$ | | *yes* | *no* | *yes* | *yes* | EI |
| | | | $IVS3.1$ | | *no* | *yes* | *no* | *yes* | EI |
| | | | $IVS3.2$ | | *no* | *yes* | *yes* | *yes* | EI |
| | | Third Party vulnerability assessment $IVS3$ | $IVS3.1$ | $Dep_2$ | *no* | *yes* | *no* | *yes* | EI |
| | | | $IVS3.2$ | $Dep_3$ | *no* | *yes* | *yes* | *yes* | EI |
| | | | $IVS3.3$ | | $level_2$ | $level_1$ | $level_3$ | $level_3$ | LI |
| | Threat & Vulnerability Assessment $TVM$ | Handling Security Incidents$TVM1$ | $TVM1.1$ | | *yes* | *yes* | *yes* | *no* | NR |
| | | | $TVM1.2$ | | *no* | *yes* | *no* | *no* | NR |
| | | Reporting $TVM2$ | $TVM2.1$ | | *yes* | *yes* | *yes* | *yes* | LI |
| | | | $TVM2.2$ | | $level_3$ | $level_2$ | $level_3$ | $level_3$ | LI |
| | Interface Security $IS$ | Application Security $IS1$ | $IS1.1$ | $Dep_4$ | $level_2$ | $level_2$ | $level_2$ | $level_2$ | EI |
| | | | $IS1.2$ | $Dep_4$ | $level_1$ | $level_2$ | $level_1$ | $level_2$ | EI |

*IVS1.2* respectively. $EV_{IVS1}$ is then calculated using Equation 4 such that:

$$EV_{IVS1} = \begin{array}{c} \\ CSP_1 \\ CSP_2 \\ CSP_3 \\ CSC \end{array} \begin{array}{c} \begin{array}{cc} EV_{IVS1.1} & EV_{IVS1.2} \end{array} \\ \begin{pmatrix} 0.333 & 0.2 \\ 0 & 0.3 \\ 0.333 & 0.2 \\ 0.333 & 0.3 \end{pmatrix} \end{array} \begin{array}{c} w_{IVS1} \\ \begin{pmatrix} 0.588 \\ 0.412 \end{pmatrix} \end{array}$$

Therefore,

$$EV_{IVS1} = \begin{array}{c} \begin{array}{cccc} CSP_1 & CSP_2 & CSP_3 & CSC \end{array} \\ \begin{pmatrix} 0.278 & 0.124 & 0.278 & 0.32 \end{pmatrix} \end{array}$$

This means no single provider is offering *IVS1* service regarding the customer requirements. Both $CSP_1$ and $CSP_3$ are satisfying customer requirement *IVS1.1* and only $CSP_2$ satisfies customer requirement *IVS1.2*. Therefore, a composition of services from different providers satisfies the *IVS1* customer requirements. The service composition for $IVS1$, specified as $k_{IVS1.1} \wedge k_{IVS1.2}$, is (($CSP_1$[7] OR $CSP_3$) AND ($CSP_2$)).

In a similar way as in *IVS1*, $EV_{IVS2}$ is calculated. Where *IVS2* is composed of six SLOs $(k_{IVS2.1} \wedge k_{IVS2.2} \wedge k_{IVS2.3} \wedge k_{IVS2.4}) \wedge (k_{IVS3.1} \vee k_{IVS3.2})$ such that:

$EV_{IVS2} =$

$$\begin{array}{ccccc} EV_{IVS2.1} & EV_{IVS2.3} & EV_{IVS2.4} & EV_{IVS3.1} & EV_{IVS3.2} \end{array}$$
$$\begin{pmatrix} 0.333 & 0 & 0.3333 & 0 & 0 \\ 0.333 & 0.3333 & 0 & 0.5 & 0.3333 \\ 0 & 0.3333 & 0.3333 & 0 & 0.3333 \\ 0.333 & 0.3333 & 0.3333 & 0.5 & 0.3333 \end{pmatrix}$$

Thus, the best service combination for satisfying customer requirement *IVS2* is offered by (($CSP_1$ OR $CSP_2$) AND ($CSP_2$ OR $CSP_3$) AND ($CSP_1$ OR $CSP_3$) AND $CSP_2$). Note that, *IVS2.2* is not required by the customer so it is not offered in the Multi-Cloud composite service.

[7]$CSP_1$ here means the service offered by $CSP_1$ which is *IVS1.2*

Similarly, *Third Party Audits* evaluation vectors ($EV_{IVS3.1}$, $EV_{IVS3.2}$ and $EV_{IVS3.3}$) are calculated and then $N_{m,IVS3.1}$, $N_{m,IVS3.2}$ and $N_{m,IVS3.3}$ are determined. In a similar way the *Handling Security Incidents*, *Reporting* and *Application Security* EVs are considered.

The set of SLOs that satisfy the customer requirements are selected as shown in the MCSLA "AND-OR" tree which shows the CSPs fulfilling the customer requirements for each SLO using AND/OR relations. Each SLO is specified in order to have the overall MC composition. After the set of SLOs selection the MCSLA is validated to detect SLOs conflicts.

### *MCSLA Validation*

Conflicts are detected in an automated manner, based on the defined dependency model, such that:

1) $Dep_1$ **Validation**: $IVS2.1$ service level ($level_1$) is *equal* to the $IVS1.1$ service level ($level_1$),$v(IVS2.1) = v(IVS1.1)$. **Result:** Valid for $CSP_1$ only. $CSP_2$ shows a conflict in $Dep_1$. Thus, only $CSP_1$ is selected for offering *IVS2.1*.
2) $Dep_2$ **Validation**: $IVS2.1$ service level ($level_1$) is *equal* to $IVS3.1$ service level ($level_1$). **Result:** Valid for $CSP_2$ however $CSP_1$ shows a conflict in $Dep_2$. Thus, only $CSP_2$ is selected for offering $IVS3.1$.
3) $Dep_3$ **Validation**: $IVS2.3$ service level ($level_1$) is *equal* to $IVS3.2$ service level ($level_1$). **Result:** Valid for both $CSP_2$ and $CSP_3$.
4) $Dep_4$ **Validation**: $IS1.1$ service level is *equal* to the $IS1.2$ service level. **Result:** Valid as $CSP_2$ the one chosen to offer $IS1.1$ and $IS1.2$ satisfy the constraint.

Using MCSLA "AND-OR" tree, the MCSLA can simply remove the SLO that is causing conflict (offered by a specific

CSP) and chooses another CSP to offer this SLO and validate the MCSLA again to detect any further conflicts.
**Note:** We have implemented a simulation environment for the proposed framework using the Java language. The full implementation is publicly available at https://github.com/amtaha/MCSLA.

## VI. Conclusion

Although the popularity of Multi-Cloud is on the rise due to the improved quality and availability of services. It also opens up new functionalities for the customers e.g, selecting services from multiple independent providers that best satisfy their requirements. Despite the advocated performance and economic advantages of the Multi-Cloud model, selecting a set of suitable services from different providers is a challenging task. Given the increasing number of CSPs which are offering a variety of services with different capabilities and prices, customers would need to assess these offered services, to be able to choose the services matching their requirements. Another problem for the customers is the lack of an efficient service selection and SLA management solution for the Multi-Cloud model. In this paper, we introduced an SLA-based service selection for Multi-Cloud environment methodology that involves: i) MCSLA construction and management and ii) service selection. In the former, we investigated the MCSLA hierarchy issue and considered the dependencies between different SLOs. Moreover, our approach automatically validates the MCSLA by exploiting dependencies between SLOs. In the latter, we used a selection technique to score the CSPs' offered services based on their SLAs and the customer requirements. The case study showed that the presented approach effectively selects the set of services that satisfies the customers' requirements best.

## Acknowledgment

Research supported in part by EC H2020 ESCUDO-CLOUD GA #644579 and MSCA-ITN NECS GA #675320

## References

[1] S. Farokhi, "Towards an sla-based service allocation in multi-cloud environments," *Proc. of CCGrid*, pp. 591–594, 2014.

[2] A. Taha, P. Metzler, R. Trapero, J. Luna, and N. Suri, "Identifying and utilizing dependencies across cloud security services," *Proc. of AsiaCCS*, pp. 329–340, 2016.

[3] I. Drago, M. Mellia, M. Munafò, A. Sperotto, R. Sadre, and A. Pras, "Inside dropbox: Understanding personal cloud storage services," *Proc. of IMC*, pp. 481–494, 2012.

[4] Cloud Security Alliance, "The Security, Trust & Assurance Registry (STAR)," *https://cloudsecurityalliance.org/star/*, 2011.

[5] "(Draft) Information Technology - Cloud Computing - Service Level Agreement (SLA) Framework and Terminology," International Organization for Standardization, Tech. Rep. ISO/IEC 19086, 2014.

[6] M. Alhamad, T. Dillon, and E. Chang, "Sla-based trust model for cloud computing," *Proc. of NBiS*, pp. 321–324, 2010.

[7] Z. Rehman, F. Hussain, and O. Hussain, "Iaas cloud selection using mcdm methods," *Proc. of ICEBE*, pp. 246–251, 2012.

[8] S. Garg, S. Versteeg, and R. Buyya, "Smicloud: A framework for comparing and ranking cloud services," *Proc. of UCC*, pp. 210–218, 2011.

[9] Z. Rehman, F. Hussain, and O. Hussain, "Towards multi-criteria cloud service selection," *Proc. of IMIS*, pp. 44–48, 2011.

[10] K. Bernsmed, M. Jaatun, P. Meland, and A. Undheim, "Security SLAs for Federated Cloud Services," *Proc. of ARES*, pp. 202–209, 2011.

[11] F. Jrad, J. Tao, and A. Streit, "Sla based service brokering in intercloud environments." *Proc. of CLOSER*, pp. 76–81, 2012.

[12] J. Luna, H. Ghani, D. Germanus, and N. Suri, "A security metrics framework for the cloud," *Proc. of Security and Cryptography*, pp. 245–250, 2011.

[13] A. Taha, R. Trapero, J. Luna, and N. Suri, "AHP-Based Quantitative Approach for Assessing and Comparing Cloud Security," *Proc. of TrustCom*, pp. 284–291, 2014.

[14] N. Grozev and R. Buyya, "Inter-cloud architectures and application brokering: taxonomy and survey," *In Software: Practice and Experience*, vol. 44, no. 3, pp. 369–390, 2014.

[15] A. Ferrer, F. Hernández *et al.*, "OPTIMIS: A holistic approach to cloud service provisioning," *In Future Generation Computer Systems*, vol. 28, no. 1, pp. 66–77, 2012.

[16] "MOSAIC EU Project," *http://www.mosaic-fp7.eu/*.

[17] NIST, "Special 9 Publication 500291," *Cloud computing standards roadmap-version 1.0*, 2011.

[18] R. Prodan, M. Wieczorek, and H. Fard, "Double auction-based scheduling of scientific applications in distributed grid and cloud environments," *In Grid Computing*, vol. 9, no. 4, pp. 531–548, 2011.

[19] S. Sotiriadis, N. Bessis, P. Kuonen, and N. Antonopoulos, "The inter-cloud meta-scheduling (icms) framework," *Proc. of AINA*, pp. 64–73, 2013.