Cost and Benefit of the SLA Mapping Approach for Defining Standardized Goods in Cloud Computing Markets

Michael Maurer[1], Vincent C. Emeakaroha[1], Ivona Brandic[1], Jörn Altmann[2]
[1]Vienna University of Technology, Vienna, Austria, {maurer,vincent,ivona}@infosys.tuwien.ac.at
[2]Technology Management, Economics, and Policy Program, Department of Industrial Engineering
College of Engineering, Seoul National University, South-Korea, jorn.altmann@acm.org

Abstract

Due to the large variety in computing resources and, consequently, the large number of different types of Service Level Agreements (SLAs), any market for computing resources faces the potential problem of a low market liquidity. To counteract this problem, offering a set of standardized computing resources is appropriate. Each of these standardized computing resources is defined through an SLA template. An SLA template defines the structure of an SLA, the attributes, the names of the attributes, and the attribute values. Since these SLA templates are currently static, they cannot reflect changes in users' needs. To address this shortcoming, we present the novel approach of adaptive SLA matching. This approach adapts SLA templates based on SLA mappings by allowing Cloud users to define mappings between public SLA templates, which are available in the Cloud market, and their private SLA templates, which are used for various in-house business processes of the Cloud user. Besides showing how public SLA templates adapt to the demand of users, we also analyze the benefits and costs of this approach. Costs are incurred every time a user has to define a new SLA mapping to a public SLA template due to its adaptation. In particular, within this paper, we investigate the cost depending on the use of different public SLA template adaptation methods. The simulation results show that the use of heuristics within adaptation methods helps balancing the cost and benefit of the SLA mapping approach.

1 Introduction

Computing resource allocations in Clouds are based not only on functional requirements but also on different non-functional requirements. These non-functional requirements (e.g., application execution time, reliability, and availability) are termed Quality of Service (QoS) requirements and are expressed and negotiated by means of Service Level Agreements (SLAs). In order to facilitate the creation and management of SLAs, SLA templates have been introduced. SLA templates, which represent popular SLA formats, comprise elements such as names of trading parties, names of SLA attributes, measurement metrics, and attribute values [1].

In Cloud computing markets, buyers and sellers of computing resources face the problem of varying definitions of computing resources. Computing resources are described through different non-standardized attributes (e.g., CPU cores, execution time, inbound bandwidth, outbound bandwidth, and processor type). [4]. Sellers use them to describe their supply of resources and buyers use them to describe their demand for resources. As a consequence, a large variety of different SLAs exists in the market. The success of matching asks (i.e, offers of sellers) and bids (i.e., offers of buyers) becomes very unlikely [1].

Approaches tackling this plethora of SLA attributes include the use of standardized SLA templates for a specific consumer base [5, 6], downloadable predefined provider-specific SLA templates [7], and the use of ontologies [8, 9]. These approaches clearly define SLA templates and require users to agree a priori on predefined requirements. The SLA templates are static.

However, the demand of users changes over time. For example, the emergence of multi-core architectures in computing resources required the inclusion of the new attribute "number of cores", which was not present in an SLA template a couple of years ago. However, the existing approaches for the specification of SLA templates cannot easily deal with demand changes. These approaches involve heavy user-interactions to adapt existing SLA templates to changing market conditions.

In this paper, we apply adaptive SLA mapping, a new approach that can react to changing market conditions [1]. This approach adapts public SLA templates, which are used in the Cloud market, based on SLA mappings. SLA mappings, which have been defined by users based on their

needs, bridge the differences between existing public SLA templates and the private SLA template (i.e., the SLA template of the user). Since a user cannot easily change the private SLA template due to internal or legal organizational requirements, an SLA mapping is a convenient workaround.

The benefits of SLA mappings for market participants are threefold. Firstly, traders can keep their private templates, which are required for other business processes. Secondly, based on their submitted mappings of private SLA templates to public SLA templates, they contribute to the evolution of the market's public SLA templates, reflecting all traders' needs. Thirdly, if a set of new products is introduced to the market, our approach can be applied to find a set of new public SLA templates. All these benefits result in satisfied users, who continue to use the market, therefore increasing liquidity in the Cloud market. However, these benefits come with some cost for the user. Whenever a public SLA template has been adapted, the users of this template have to re-define their SLA mappings.

The five contributions of this paper are: (1) the definition of an appropriate use case to exemplify the adaptive SLA mapping approach; (2) the definition of three adaptation methods for adapting public SLA templates to the needs of the user; (3) the investigation of conditions under which SLA templates should be adapted; (4) the formalization of measures (i.e., utility and cost) to assess SLA adaptations and SLA adaptation methods; and (5) the introduction of an emulation approach for the use cases.

The remainder of the paper is organized as follows: Section 2 describes related work. Section 3 introduces the adaptive SLA mapping approach and the utility and cost model. The simulation setup, the three adaptation methods, and the simulation infrastructure are described in Section 4. Section 5 presents the simulation results and a discussion. Section 6 concludes the paper.

2 Related Work

For putting this work in context of the state-of-the-art, we briefly describe Cloud marketplaces and the existing work on SLAs. Currently, a large number of commercial Cloud providers have entered the utility computing market, offering a number of different types of services. We distinguish between computing infrastructure services, which are pure computing resources on a pay-per-use basis [11, 12, 13], software services, which are computing resources in combination with a software solution [6, 14], and platform services, which allow customers to create their own services in combination with the help of supporting services of the platform provider. The first type of services consists of a virtual machine, as in the case of Amazon's EC2 service, or in the form of a computing cluster, as done by Tsunamic Technologies. The number of resources offered by a provider is low. For example, Amazon and EMC introduced only three derivations of their basic resource type [5]. Examples for the second type of services are services offered by Google (Google Apps [6]) and Salesforce.com [14]. These companies provide access to software on pay-per-use basis. These Software-as-a-Service (SaaS) solutions can hardly be integrated with other solutions, because of their large variety. Examples for the third kind of Cloud services are Sun N1 Grid [15], force.com [14], and Microsoft Azure [16]. In this category, the focus lies on provisioning essential basic services that are needed by a large number of applications. These basic services can be ordered on a pay-per-use basis. Although the goal of these offerings is a seamless integration with the users applications, standardization of interfaces is largely absent. Concluding, we can state that, apart from first attempts in the service type infrastructure as a service, standardization attempts do almost not exist.

The main SLA matching mechanisms are based on OWL, DAML-S, or similar semantic technologies. [8] describe a framework for semantic matching of SLAs based on WSDL-S and OWL. [9] present a unified QoS ontology applicable to specific scenarios such as QoS-based Web services selection, QoS monitoring, and QoS adaptation. [17] present an autonomic Grid architecture with mechanisms for dynamically reconfiguring service center infrastructures. It is exploited to fulfill varying QoS requirements. Besides those mechanisms, [10] discuss autonomous QoS management, using a proxy-like approach for defining QoS parameters that a service has to maintain during its interaction with a specific customer. The implementation is based on WS-Agreement, using predefined SLA templates. However, they cannot consider changes in user needs, which is essential for creating successful markets, as shown in our earlier work [1]. Several works on current SLA management are presented in [2]. Besides, regardless of the type of approach used, these approaches do not evaluate and explain the benefit and costs through the introduction of SLA matching mechanisms.

3 Adaptive SLA Mapping

In this section, we present a use case for adaptive SLA mapping. Besides, we discuss the SLA life cycle and introduce the utility and cost model for assessing SLA matching approaches.

3.1 Use Case

Since resources can be exposed as services using typical Cloud deployment technologies (i.e., SaaS/PaaS/IaaS), we assume that the service provider of Figure 1 registers its resources (e.g., infrastructure, software, platforms) to particular public templates (step 1, Figure 1). If some differences between its resources (private SLA template) and the public

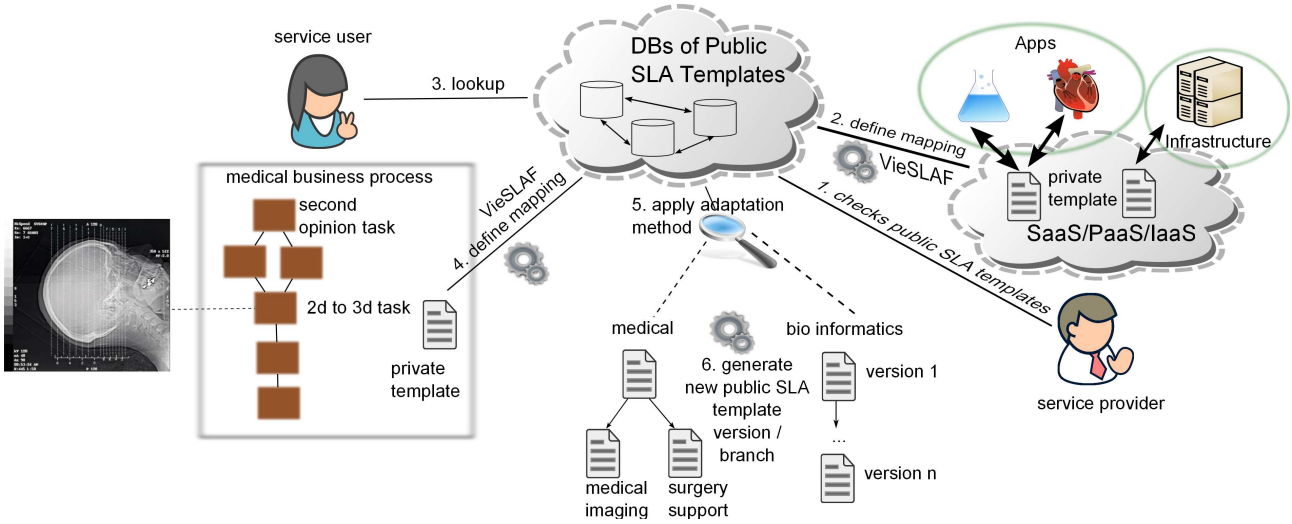

Figure 1: Use case of SLA mapping.

templates exist, the provider defines SLA mappings, transforming the private template into the public template (step 2, Figure 1). The management of SLA mappings, which is performed with VieSLAF, is explained in detail in [3].

In step 3, Cloud users can look up Cloud services that they want to use in their workflow. In Figure 1, we exemplified a business process (i.e, workflow) for medical treatments [18]. It includes various interactions with human beings (e.g., the task of getting a second opinion on a diagnosis) as well as interaction with different infrastructure services. Some of these tasks (e.g., the reconstruction of 2-dimensional SPECT images to 3-dimensional SPECT images) can be outsourced to the Cloud [18]. Thereby, we assume that the private SLA template (representing the task) cannot be changed, since it is also part of some other local business processes and has to comply with different legal guidelines for electronic processing of medical data. Therefore, in case the user decides to outsource a task and discovers differences between the private SLA template and the public SLA template, the user defines an SLA mapping. The mapping describes the differences between the two SLA templates (step 4). A typical mapping is the mapping of an attribute name to another attribute name (e.g., *number of CPUs* to *cores*) or the inclusion of a new SLA attribute (e.g., *parallel programming models*) into the SLA template.

The public SLA templates are stored in searchable repositories using SQL and non-SQL-based databases (e.g., HadoopDB). The SLA mappings, which have been provided by users and providers to the entity managing the public SLA templates, are evaluated after certain time periods, in order to adapt the public SLA templates to the needs of the users. The adapted public SLA templates replace the existing public SLA templates in the repository, constituting our novel approach of adaptive SLA mapping. The adaptation method, which adapts the public SLA templates, performs it such that the new public SLA templates represent user needs better than the old SLA templates (step 5). Besides the adaptation of attribute names and attribute values, the adaptations can also include definitions of new branches of templates (e.g., a medical SLA template can be substituted by more specialized templates on medical imaging and surgery support). The definition of different versions of a particular template is also possible as shown for the templates in the bioinformatics domain (step 6).

3.2 Public SLA Template Lifecycle

To illustrate the lifecycle of public SLA templates, we give a short example as shown in Figure 2 first.

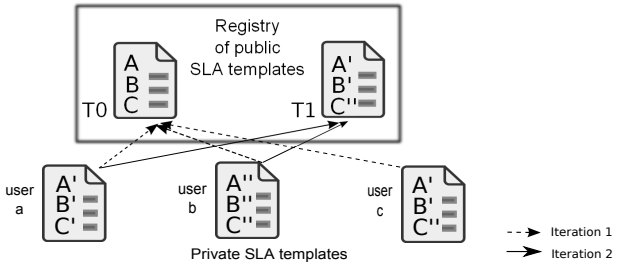

Figure 2: SLA mapping process.

Initially, the SLA template registry only holds the initial public SLA template T_0. In iteration 1, all users define mappings from their private templates to T_0. Since the attribute of the public SLA template *(A, B, C)* and the attribute names of each user differ, every user has to create 3 attribute

mappings. Based on these mappings, the new version T_1 of the public template is generated (according to the adaptation method used), containing the attribute names *A'*, *B'*, *C"*. Since the public template has changed, users need to change their mappings as well (iteration 2). Consequently, user *a* only needs one attribute mapping, user *b* can reduce the number of attribute mappings to 2, and user *c* does not need to issue any attribute mapping, since the public template is completely identical to her private template. This example shows how our adaptive SLA mapping approach adapts a public SLA template to the needs of users. In addition to this, since adapted public SLA templates represent the need of market participants, it is most likely that new requests of users need less attribute mappings, reducing the cost for users.

The formalized public SLA template lifecycle, which consists of five steps, is shown in Figure 3.

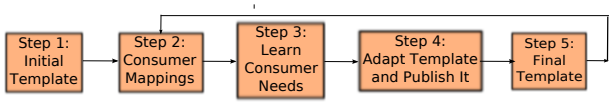

Figure 3: Formalized public SLA template lifecycle.

An initial template is created in the beginning of the lifecycle (step 1, Figure 3). Afterwards, consumers perform SLA mappings (step 2). Based on their needs, inferred from these mappings (step 3), and the predefined adaptation method, the public SLA template is adapted (step 4). Assuming that the demand of market participants does not change, a final template is generated (step 5). If the demand has changed during a fixed time period, the process continues with step 2. In practice, the time between two iterations could correspond to a time period of one week. During that time new SLA mappings are solicited from consumers and users.

3.3 Adaptation Methods

The adaptation methods determine for every attribute name separately, whether the current attribute name should be adapted or not. The first adaptation method is the maximum method (which has been applied to the example shown in Figure 2). The remaining two adaptation methods differ with respect to their use of heuristics to find a balance between benefit and cost.

3.3.1 Maximum Method

Applying this method, the SLA attribute name, which has the highest number of attribute name mappings, is selected (maximum candidate). The selected attribute name will become the next attribute name used by the next public SLA template.

Example: If we assume that all attribute names have the same count, this method would select any of the four possible attribute names randomly. If a public SLA template already exists, the method will choose the attribute name that is currently used in the public SLA template.

3.3.2 Threshold Method

In order to increase the requirements for selecting the maximum candidate, this method introduces a threshold value. If an attribute name is used more than this threshold (which can be adapted) and has the highest count, then this attribute name will be selected. If more than one is above the threshold and they have the same count, the method proceeds as described for the maximum method. If none is above the required threshold, then the method sticks to the currently used attribute name. Note, throughout the examples in this paper, we fix the threshold to 60%.

Example: Assuming an example in which none of the attribute names has a mapping percentage above 60% and all counts are equal, the threshold method sticks to the attribute name that is currently used in the public SLA template.

3.3.3 Maximum-Percentage-Change Method

This method is divided into two steps. In the first step, the attribute name is chosen according to the maximum method.

In the second step, which comprises τ iterations, attribute names will be changed, only if the percentage difference between the highest count attribute name and the currently selected attribute name exceeds a threshold. The threshold σ_T is set to 15%. A low threshold leads to more mappings, whereas a high threshold leads in average to fewer mappings. After τ iterations (e.g., $\tau = 10$), the method re-starts with executing the first step. It allows even slighter changes to take effect.

Example: Let's suppose the mapping count resulted in attribute name A' having the highest count. By applying the maximum method, A' is selected. In the next iteration, the number of mappings for each attribute name has changed. Attribute name A accounted for 10%, A' for 28%, A'' for 32%, and A''' for 30% of all mappings. Assuming a threshold of 15%, the chosen attribute does not change. The percentage difference between attribute name A' and the attribute name A'' with the highest count is only *13.3%*.

3.4 Utility and Cost Model

Since the aim of this paper is to assess the benefit and the cost of using the adaptive SLA mapping approach for finding the optimal standardized goods in a Cloud market, we define a utility model and a cost model. The utility function and the cost function, which take attributes of the customer's SLA template and the attributes of the public SLA template as input variables, helps to quantify the benefit and

cost. For our utility model, we assume an increase in benefit, if an attribute of both templates is identical. This is motivated by the fact that the Cloud resource traded is identical to the need of the buyer (or the provisioned resource of the provider) and, therefore, no inefficiency through resource over-provisioning occurs. The cost model captures the effort of changing an SLA mapping. A cost to the user is only incurred, if the user needs to change its SLA mapping because of a change in the public SLA template.

To formally introduce these models, we introduce some definitions. The set of SLA attributes is defined as T_{var}. As an example, we set $T_{var} = \{\alpha, \beta\}$, where α represents *Number of Cores in one CPU* and β represents *Amount of CPU Time* (Note, α and β could also represent attribute values). All possible attribute names that a user can map to a $\pi \in T_{var}$ are denoted as $Var(\pi)$. Within our example, we set $Var(\alpha) = \{A, A', A'', A'''\}$, representing *Var("Number of Cores in one CPU") = {CPU Cores, Cores of CPU, Number of CPUCores, Cores}*, and $Var(\beta) = \{B, B', B'', B'''\}$.

Assuming a set of consumers' private templates $C = \{c_1, c_2, \ldots, c_n\}$, we can now define the relationship of a specific SLA attribute to a specific name of this SLA attribute at the iteration $i \in N$ for every private and public template p, $p \in C \cup \{T\}$ as

$$SLA_{p,i} : T_{var} \rightarrow \bigcup_{\pi \in T_{var}} Var(\pi). \tag{1}$$

With respect to our example, we assume $SLA_{T,0}(\alpha) = A$ and $SLA_{T,0}(\beta) = B$ as our initial public template T at iteration 0.

Based on these definitions, we define the utility function $u^+_{c,i}$ and the cost function $u^-_{c,i}$ for consumer c, attribute $\pi \in T_{var}$, and iteration $i \geq 1$ as

$$u^+_{c,i}(\pi) = \begin{cases} 1, & SLA_{c,i}(\pi) = SLA_{T,i}(\pi) \\ 0, & SLA_{c,i}(\pi) \neq SLA_{T,i}(\pi) \end{cases} \tag{2}$$

$$u^-_{c,i}(\pi) = \begin{cases} 0, & SLA_{c,i}(\pi) = SLA_{T,i}(\pi) \\ 0, & SLA_{c,i}(\pi) \neq SLA_{T,i}(\pi) \wedge \\ & SLA_{T,i-1}(\pi) = SLA_{T,i}(\pi) \\ 1/2, & SLA_{c,i}(\pi) \neq SLA_{T,i}(\pi) \wedge \\ & SLA_{T,i-1}(\pi) \neq SLA_{T,i}(\pi) \end{cases} \tag{3}$$

We choose our utility function as exemplified in [20]. The utility function states that a consumer c receives a utility of 1, if the name of the attribute of the private SLA template matches the name of the public SLA template attribute, and a utility of 0 otherwise.

The cost function states that a consumer has a cost of $1/2$, if the attribute names do not match and the public template attribute of the last iteration changed to a new one. In this case, the consumer has to define a new attribute mapping, as he cannot use the old one anymore. In the other two cases, the consumer has no cost, since either the attribute names match or the public template attribute name did not change since the last iteration. That means he does not need any new mapping. Thus, for attribute π, the consumer c at iteration i gets the net utility

$$u^o_{c,i,\pi} = u^+_{c,i}(\pi) - u^-_{c,i}(\pi). \tag{4}$$

The net utility for all attributes at iteration i for consumer c is defined as the sum of the net utilities $u^o_{c,I,\pi}$:

$$u^o_{c,i} = \sum_{\pi \in T_{var}} u^o_{c,i,\pi}. \tag{5}$$

The overall utility and overall cost (i.e., the utility and cost of all users C and attributes π at iteration i) are defined as:

$$U^+_i = \sum_{c \in C} \sum_{\pi \in T_{var}} u^+_{c,i}(\pi) \tag{6}$$

$$U^-_i = \sum_{c \in C} \sum_{\pi \in T_{var}} u^-_{c,i}(\pi) \tag{7}$$

Consequently, the overall net utility at iteration i is defined as the difference between the overall utilities minus the overall cost:

$$U^o_i = U^+{}_i - U^-_i. \tag{8}$$

4 Simulation Environment

In order to analyze the performance of the three adaptation methods with respect to balancing between adapting the public SLA template to the current needs of all users and the cost of making new SLA mappings, we set up a simulation environment.

4.1 Testbed

For our simulation, we use a testbed that is composed of production-level software (i.e., *VieSLAF*) and software that simulates SLA mappings of users. Figure 4 illustrates our *emulation* testbed. The components that are drawn in white are production-level software. It comprises the knowledge base, components for managing SLA mappings provided by consumers and providers, and the adaptation method. The grey components indicate the simulated components.

The SLA mapping middleware, which follows a client/service design, facilitates the access to registries and provides a GUI used for browsing public SLA templates. The SLA mapping middleware is based on different Windows Communication Foundation (WCF) services, of which only a few are mentioned here. For example, the *SLAMappingService* is used for the management of SLA mappings (cf. (3), Figure 4) by users (i.e., consumers and providers). Consumers may search for appropriate services through *SLAQueryingService* in the registry and define appropriate SLA mappings by using the method *createAttributeMapping*. With each query, it is also checked

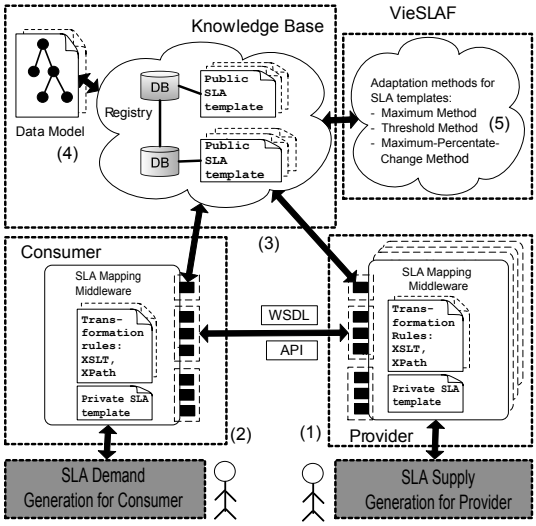

Figure 4: Adaptive SLA mapping architecture using VieSLAF.

whether a user has also specified SLA mappings. The rules necessary for the transformations of SLA attributes (or a set of SLA attributes) are stored in the database and can be applied by users to their private SLA templates.

For storing the SLA templates in a predefined data model (cf. (4)), we implemented registries representing searchable repositories. Currently, we have implemented an MS-SQL 2008 database with a Web service frontend. To handle scalability issues, we intend to utilize non-SQL DBs (e.g., HadoopDB) with SQL-like frontends (e.g., Hive [22]). SLA templates are stored in a canonical form, enabling the comparison of these XML-based templates. The registry methods are also implemented as WCF services and can be accessed only with appropriate access rights. The access rights distinguish three access roles: *consumer*, *provider* and *registry administrator*.

Based on the submitted SLA mappings, public SLA templates are adapted, using an adaptation method (cf. (5)).

4.2 Simulation Parameter Settings

For our simulation, we define five scenarios on how often attribute names occur in average. That means each scenario defines an occurrence distribution of four different SLA attribute names. The five scenarios, which have been chosen such that they represent different situations, are defined as follows:

- Scenario a: All attribute name counts of an attribute are equal.
- Scenario b: The counts of three attribute names are equally large and larger than the remaining one.
- Scenario c: Two attribute name counts are equally large and are larger than the other two, which are equally large as well.
- Scenario d: One attribute name, which has been picked as the attribute name for the initial setting, has a larger count than the remaining three attribute names, which are equally large.
- Scenario e: One attribute name, which has not been picked as the attribute name for the initial setting, has a larger count than the remaining three attribute names.

The actual values of each of the five scenarios are shown in Table 1. The four attribute names chosen for this example are: A, A', A'', A'''.

Table 1: Average occurrence of attribute names in all scenarios.

	Scenarios [%]				
	a	b	c	d	e
A	25	10	10	30.0	23.3
A'	25	30	10	23.3	30.0
A"	25	30	40	23.3	23.3
A"'	25	30	40	23.3	23.3

For example, if the attribute α (*CPU Time*) is distributed according to scenario *c*, then the four attribute names occur in average as follows: 10% of the attribute names is A, 10% of the attribute names is A', 40% of the attribute names is A'', and 40% of the attribute names is A'''. However, as we intend to account for slight changes in the demand for attribute names by users, we draw randomly the attribute names according to the distribution given in Table 1 instead of generating the exact number of attribute names. Consequently, the actual counts of attribute names might vary compared to the average values shown in Table 1. As an example, the attribute names generated according to the distribution of scenario *c* might be 9%, 12%, 37%, and 42% instead of 10%, 10%, 40%, and 40%. This process of generation of attribute names is executed for each iteration.

Furthermore, another three simulation parameters are set. First, we limit the number of iterations to 20. At each iteration, 100 users perform SLA mappings to all SLA attributes. At the end of an iteration, a new public SLA template is generated, which is based on the adaptation method and the users' SLA mappings.

Table 2 summarizes these settings.

Table 2: Simulation parameter settings.

Simulation Parameter	**Value**
Number of scenarios	5
Number of users (consumers / providers)	100
Number of SLA attributes per SLA template	1
Number of SLA attributes names per attribute	4
Number of adaptation methods applied	3
Number of iterations	20

We used these parameter settings for each of the adaptation methods.

5 Experimental Results and Analysis

5.1 Net Utilities of Adaptation Methods

Using the SLA mapping approach, the user gets the benefit of having access to public SLA templates that reflect the overall market demand (i.e., the average user's demand). This gain of some user is expressed with equation 2. However, this comes with the cost for defining new SLA mappings whenever the public SLA template changed (equation 3). Within this section, we investigate the cost of all users (equation 7), the utility of all users (equation 6), and the net utility of all users (equation 8) for different adaptation methods. The net utility metric is used to decide which of the three adaptation methods is superior.

The first adaption method that we investigate is the maximum method. It is our reference method, since it does not use any heuristics. The simulation results, which are shown in this section, have been obtained from running the simulation with parameter settings as described in section 4.2. The simulation results shown are averages over all scenarios. The advantage of this method is that the public SLA template generated with this method minimizes the differences to all private SLA templates of all users. This method requires, however, many changes of SLA mappings.

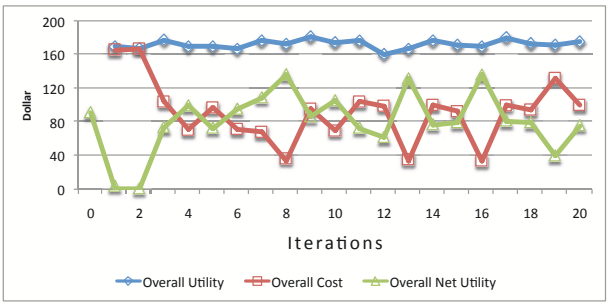

Figure 5: Utility, cost, and net utility for the maximum method.

Figure 5 shows, as expected, that the maximum method generates a high utility, since it achieves many matchings of attribute names of the public SLA template and the private SLA templates. Its net utility stays around its initial net utility value of about 170 for each iteration. However, as expected as well, it requires many new mappings and, thus, incurs high costs. Consequently, the net utility is far lower than the utility.

In order to address this issue of high cost, we use heuristics in the following two adaptation methods. The heuristics help to find a balance between the utility of having a public SLA template, whose attribute names are identical to most of the attribute names of the private SLA templates, and the cost of creating new SLA attribute mappings. The first heuristics-based adaptation method, which we investigate, is the threshold method. The simulation results are shown in Figure 6.

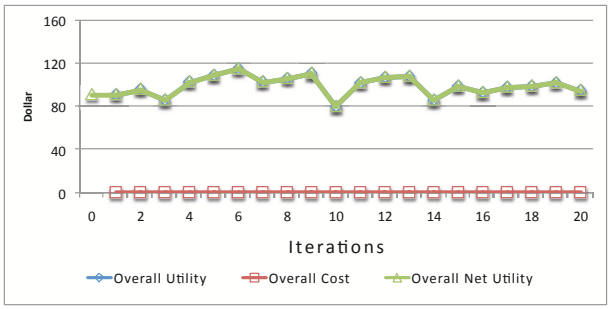

Figure 6: Utility, cost, and net utility, for the threshold method.

Figure 6 illustrates that the threshold method does not incur any cost to users at all. This is due to the high threshold (i.e., 60%), resulting in no changes of the SLA template attribute names. Nevertheless, the utility (and net utility) is not higher than the maximum method, just more stable across the 20 iterations. Therefore, the threshold method with a threshold of 60% could be considered the other extrem strategy, in which the initial public SLA template does not get adapted at all. By lowering the threshold parameter such that the threshold parameter in a few iterations is lower than the highest count of an attribute name, it is expected that the net utility improves. If the threshold parameter is lower than the minimum count of an attribute name in all iterations, then this method is identical to the maximum method.

The maximum-percentage-change method is the second heuristics-based adaptation method, which we investigate and the results are shown in Figure 7.

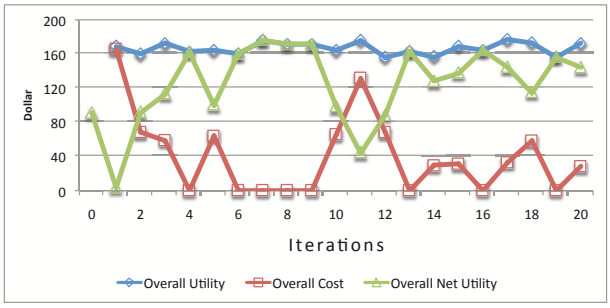

Figure 7: Utility, cost, and net utility for the maximum-percentage-change method with $\tau = 10$.

The simulation results show that in the first iteration and every tenth iteration ($\tau = 10$) the overall net utility decreases significantly due to the high amount of changes of SLA mappings (Figure 7). The cost is very high. At these iterations, this method chooses the attribute names with the maximum number of counts (not considering the threshold of 15%). In the subsequent iterations, however, the cost is low and the overall net utility increases significantly. It achieves even higher values than the other two methods.

5.2 Average Cost and Average Net Utility

Table 3 shows the average overall utility, average overall cost, and the average overall net utility for all three adaptation methods. The averages are calculated over all iterations. The maximum method has achieved the highest average overall utility. It satisfies the largest number of users. However, since it also incurs the highest costs, it becomes the method with the lowest average overall net utility.

Table 3: Overall utility, overall costs, and overall net utilities averaged across all iterations (The best values are highlighted in bold).

	Maximum	**Threshold**	**Max.-Perc.-Change**
avg. overall utility	**171.9**	99.5	166.6
avg. overall cost	91.3	**0.0**	39.95
avg. overall net utilities	80.6	99.5	**126.65**

The threshold method does slightly better with respect to the average net utility than the maximum method. This is due to the zero cost. The threshold method (with a high threshold) stays with a fixed set of SLA attribute names for the public SLA template.

The best adaptation method with respect to the average overall net utility is the maximum-percentage-change method. We observe that the average overall net utility is better than the other two adaptation methods, although the average overall utility is not the highest among the three adaptation methods. The reason is that the cost is low. The low cost is a result of the fact that the SLA attribute names of the public SLA template are not changed frequently. They are only changed in iterations $k\tau + 1, k \in N_0$ (i.e., when the method behaves like the maximum method) and whenever the threshold of 15% is exceeded.

Based on the result shown in this section, we can state the adaptive SLA mapping approach is a good way of generating standardized goods, which address the needs of the market. To reduce the cost for creating SLA mappings frequently, the introduction of heuristics into the adaptation methods is helpful. Results show that a significant reduction of costs can be achieved, balancing the benefit and the cost of SLA mapping.

6 Conclusion and Outlook

In this paper, we have investigated cost, utility, and net utility of the adaptive SLA mapping approach, in which market participants may define SLA mappings for translating their private SLA templates to public SLA templates. Contrary to all other available SLA matching approaches, the adaptive SLA mapping approach facilitates continuous adaptation of public SLA templates based on market trends. However, the adaptation of SLA mappings comes with a cost for users in the form of effort for generating new SLA mappings to the adapted public SLA template. To calculate the cost and benefits of the SLA mapping approach, we utilized the SLA management framework VieSLAF and simulated different market situations. Our findings show that the cost for SLA mappings can be reduced by introducing heuristics into the adaptation methods for generating adapted public SLA templates. The methods show cost reduction and increase in average overall net utility.

Acknowledgment

The authors would like to thank Marcel Risch for his valuable discussions. The research was partially supported by the National Research Foundation of Korea (grant number K21001001625-10B1300-03310) and the Vienna Science and Technology Fund (grant agreement ICT08-018).

References

[1] M. Risch, I. Brandic, J. Altmann. *Using SLA Mapping to Increase Market Liquidity*. NFPSLAM-SOC 2009. In conjunction with The 7th International Joint Conference on Service Oriented Computing, Stockholm, Sweden, November 2009.

[2] R. Buyya, K. Bubendorfer. *Market Oriented Grid and Utility Computing*. John Wiley & Sons, Inc., New Jersey, USA, 2008

[3] I. Brandic, D. Music, P. Leitner, S. Dustdar. *VieSLAF Framework: Enabling Adaptive and Versatile SLA-Management*. GECON2009. In conjunction with Euro-Par 2009, 25- 28 August 2009, Delft, The Netherlands.

[4] M. Risch, J. Altmann. *Enabling Open Cloud Markets Through WS-Agreement Extensions*. Service Level Agreements in Grids Workshop, in conjunction with GRID 2009, CoreGRID Springer Series, Banff, Canada, October 2009.

[5] Amazon Elastic Compute Cloud (Amazon EC2), http://aws.amazon.com/ec2/, 2010.

[6] Google Apps, http://www.google.com/apps/, March 2010.

[7] Business Objective Driven Reliable and Intelligent Grids for Real Business (BREIN), http://www.eu-brein.com/, February 2010.

[8] N. Oldham, K. Verma, A. P. Sheth, and F. Hakimpour. *Semantic WS-agreement partner selection*. 15th International Conference on World Wide Web, WWW 2006, Edinburgh, Scotland, UK, May 2006.

[9] G. Dobson, A. Sanchez-Macian. *Towards Unified QoS/SLA Ontologies*. IEEE Services Computing Workshops (SCW), Chicago, Illinois, USA, pp.18-22, September 2006.

[10] B. Koller, L. Schubert. *Towards Autonomous SLA Management Using a Proxy-Like Approach*. Multiagent Grid Systems. vol.3, no.3, IOS Press, Amsterdam, The Netherlands, 2007.

[11] M. Risch, J. Altmann, L. Guo, A. Fleming, C. Courcoubetis. *The GridEcon Platform: A Business Scenario Testbed for Commercial Cloud Services.* 6th international Workshop on Grid Economics and Business Models, Delft, The Netherlands, August 2009.

[12] Tsunamic Tech. Inc., http://www.clusterondemand.com/, 2010.

[13] EMC Atmos Online, https://mgmt.atmosonline.com/, 2010.

[14] Salesforce.com, http://www.salesforce.com, March 2010.

[15] Sun Grid, http://www.sun.com/service/sungrid/index.jsp, 2010.

[16] Microsoft Azure, http://www.microsoft.com/windowsazure/, 2010.

[17] D. Ardagna, G. Giunta, N. Ingraa, R. Mirandola, and B. Pernici. *QoS-Driven Web Services Selection in Autonomic Grid Environments*. International Conference on Grid Computing, High Performance and Distributed Applications (GADA), Montpellier, France, November 2006.

[18] I. Brandic, S. Benkner, G. Engelbrecht, R. Schmidt. *QoS Support for Time-Critical Grid Workflow Applications*. 1st IEEE International Conference on e-Science and Grid Computing, Melbourne, Australia, December 2005.

[19] E. Oberortner, U. Zdun and S. Dustdar: *Tailoring a Model-Driven Quality-of-Service DSL for Various Stakeholders*. MiSE 2009.

[20] J. Chen, B. Lu. *An Universal Flexible Utility Function in Grid Economy*. 2008 IEEE Pacific-Asia Workshop on Computational Intelligence and Industrial Application.

[21] R. A. Fisher. *Statistical Methods for Research Workers*. ed. 12, Edinburgh, Oliver and Boyd, 1954.

[22] Ashish Thusoo, Joydeep Sen Sarma, Namit Jain et. Al. *Hive - A Warehousing Solution Over a Map-Reduce Framework*. VLDB 2009.