TOWARDS AUTONOMIC MARKET MANAGEMENT IN CLOUD COMPUTING INFRASTRUCTURES

Ivan Breskovic, Michael Maurer, Vincent C. Emeakaroha, Ivona Brandic
Distributed Systems Group, Information Systems Institute, Vienna University of Technology, Austria
{breskovic, maurer, vincent, ivona}@infosys.tuwien.ac.at

Jörn Altmann
TEMEP, Department of Industrial Engineering, College of Engineering, Seoul National University, South Korea
jorn.altmann@acm.org

Keywords: Service Level Agreement, SLA management, electronic markets, autonomic computing

Abstract: Cloud computing markets face challenges, such as a large variety of different resource types in the market. A large variety of resource types results in a low number of matches of ask and bids. Consequently, the market has a low liquidity, which is economically inefficient and can lead to market failure. To counteract this problem, SLA templates (i.e., templates for electronic contracts) have been introduced. However, until now, SLA templates were static, not able to reflect changes in users' requirements. In this paper, we apply the adaptive SLA mapping approach for deriving public SLA templates from private SLA templates of users. To achieve the adaptation of SLA templates, we combine clustering algorithms with adaptation methods. The result is a set of new public SLA templates, which reflect the market situation more accurately. This way, we enable marketplaces to automatically adapt to observed changes of market conditions. For the simulation-based evaluation of our approach, we formalize a utility and cost model, calculate the overall net utility, and assess the scalability of the approach. Our results show that the use of clustering algorithms can significantly improve the performance of the adaptive SLA mapping approach.

1 INTRODUCTION

Cloud computing is the result of the convergence of several concepts, ranging from virtualization, distributed application design, Grid computing, and enterprise IT management. It enables a flexible approach for deploying and using applications (Buyya et al., 2009). In such a paradigm, hardware resources, software resources, and data are offered as on-demand services to the user in a pay-as-you go model. Functional and non-functional requirements of those services, termed quality of service (QoS) requirements, are expressed and negotiated by means of service level agreements (SLAs). Those are contracts in Cloud resource markets and are signed between a customer and a service provider.

For the successful implementation of the Cloud computing paradigm, well-developed hardware and software resource markets are fundamental (Risch et al., 2010). One essential feature of a well-developed market is liquidity of its goods. Illiquid goods have the risk of not being sellable or purchasable when needed, driving potential market participants away. Therefore, well-developed resource markets must enable immediate execution of standard orders, which is possible through a sufficiently large number of participants trading (Samuelson and Nordhaus, 2005).

Many electronic marketplaces suffer from challenging situations, such as a highly dynamic evolution of user requirements for goods. Due to the large resource variability in Cloud markets and still low number of market participants, a small difference in a single user requirement might prevent a match. This can result in very low percentage of matching of asks (i.e., consumers' requirements) and bids (i.e., providers' offers), making the market unattractive to both consumers and providers.

To counteract this problem, SLA templates have been introduced. They are documents comprising all required elements of an SLA but without QoS values assigned (Brandic et al., 2010). By creating publicly available SLA templates stored in searchable registries, finding an appropriate trade partner is simplified. However, since the generation of SLA templates has been static until now, they cannot efficiently react to market changes that require the adaptation of resource attributes within the SLA template.

To confront this challenge, we investigate autonomic creation and management of SLA templates with the goal of attracting participants to trade and, therefore, increasing the liquidity of goods. In detail, the approach presented in this paper is based on SLA mapping. SLA mappings are XSLT documents, which bridge the differences between two SLA templates by defining mappings between non-matching parts of the two SLAs. A service consumer can therefore define mappings between his private SLA template and the publicly available SLA template.

In this paper, in particular, we derive new public SLA templates by utilizing clustering methods and economic mechanisms, adapting public SLA templates to changes in demand in the market. This method leads to the automation of resource markets. By including economic mechanisms into Cloud computing infrastructures, they can automatically adapt to changes of market conditions. With the help of a simulation, we demonstrate the benefits of our approach. In particular, we show that this SLA management approach brings three additional benefits for market participants: (1) It allows market participants to keep their existing private SLA templates that may already be used in other business processes; (2) It enables participants to impact the evolution of Cloud resource markets; (3) It delivers better results for the Cloud market as a whole. We show this benefit by formalizing and applying a measure for assessing publicly available SLA templates within our simulation.

Concluding, the main contributions of this paper are: (1) the introduction of an approach for applying clustering algorithms to group similar SLA mappings and to enable autonomic market management; (2) the formalization of a measure for evaluating the SLA mapping approach by determining the utility and the cost to users; and (3) the evaluation of the proposed mapping approach using an experimental testbed.

The remainder of the paper is organized as follows. Section 2 describes related work. Section 3 defines the SLA mapping approach and introduces the idea of applying clustering algorithms to group similar SLA mappings. Section 4 introduces clustering algorithms and adaptation methods used for generating new public SLA templates. The formalization of the utility and cost model to assess the benefits of the approach is given in Section 5. A description of the simulation environment and evaluation results are presented and discussed in Section 6. Section 7 concludes the paper.

2 RELATED WORK

Over the past several years, many large IT companies have entered the utility computing market by offering different types of services. We divide them in three groups, based on types of resources offered: (1) group of service providers who offer their computing resources on a pay-per-use basis in infrastructure-as-a-service model, such as Amazon EC2[1] and S3[2], (2) providers who offer their computing resources in combination with their own software components, such as Google Apps[3] and Salesforce.com[4], and (3) group of providers offering not only a deployment platform, but an application development platform as well, such as Microsoft Azure[5] and Salesforce.com. However, although these providers offer a large variety of services, they usually sell a single type of resources and do not cover flexible SLAs.

In addition to this, several research projects investigate SLA template generation (Oldham and Verma, 2006; Dobson and Sanchez-Macian, 2006; Green, 2006). As reported in (Karänke and Kirn, 2007), most projects use SLA specifications, based on WSLA and WS-Agreement, which lack support for economic attributes and negotiation protocols. To compensate these shortcomings, (Oldham and Verma, 2006) introduce utilization of semantic Web technologies based on WSDL-S and OWL for enhancement of WS-Agreement specifications to achieve autonomic SLA matching. Similar to that, (Green, 2006) introduces an ontology-based SLA formalization where OWL and SWRL are chosen to express the ontologies. (Dobson and Sanchez-Macian, 2006) suggest producing a unified QoS ontology applicable to the main scenarios such as QoS-based Web Services selection, QoS monitoring, and QoS adaptation.

Several research projects have also discussed the implementation of system resource markets (Buyya et al., 2001; Nimis et al., 2008; Neumann et al., 2008). GRACE (Buyya et al., 2001) developed a market architecture for Grid markets and outlined a market mechanism, while the good itself (i.e., computing resource) has not been defined. Moreover, the process of creating agreements between consumers and providers has not been addressed. The SORMA project (Nimis et al., 2008) has also considered open Grid markets. They identified several market requirements, such as allocative efficiency, budget-balance, truthfulness, and individual rational-

[1]http://aws.amazon.com/ec2/
[2]http://aws.amazon.com/s3/
[3]http://www.google.com/apps/
[4]http://www.salesforce.com/
[5]http://www.microsoft.com/windowsazure/

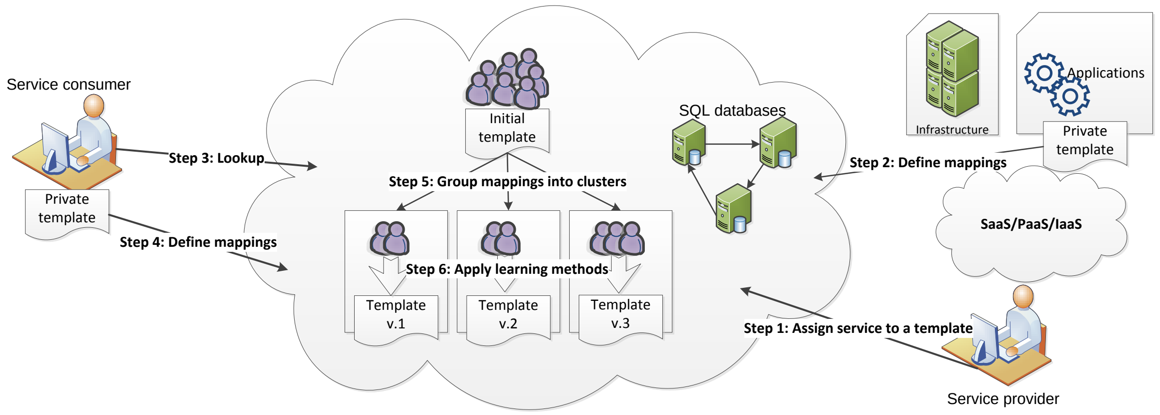

Figure 1: The SLA mapping process.

ity (Neumann et al., 2008). However, they have not considered that a market can only function efficiently if there is a sufficiently large liquidity of its goods.

Overall, most of the computing resource market research did not define the tradable good, or they worked only with simplified definitions of goods. Furthermore, most of the existing approaches in SLA management use static SLA templates and, therefore, do not consider market changes. This means that the issue of maintaining a sufficiently high liquidity in electronic market has not been addressed yet.

3 COMPUTING RESOURCE MARKETS

In this section, we present the SLA mapping approach and introduce the idea of applying clustering algorithms to group SLA mappings.

3.1 The SLA Mapping Approach

The SLA mapping process is presented in Figure 1. In the step 1, a service provider assigns his service (e.g., infrastructure resources) to a particular public SLA template. Since it is sometimes not possible to find a perfect match between a private SLA template and a public SLA template, service provider can define mappings to bridge the differences between two templates (step 2).

In the next step, a service consumer can look for appropriate Cloud services. Since it might happen that a consumer cannot change his private template due to existing business processes, legal issues, or other reasons, he can define SLA mappings to bridge the differences between his private template and the public template assigned to the service (step 4).

Both service provider and service consumer can specify two types of SLA mappings:

1. **Ad-hoc SLA mapping type** defines translation between a parameter existing in the user's private SLA template and the public SLA template. We distinguish simple ad-hoc mappings, i.e., mapping of different values for an SLA element (e.g., a mapping between the names "CPUCores" and "NumberOfCores" of an SLA parameter, or a mapping between two different values of a service level objective), and complex ad-hoc mappings. The later one maps between different functions used for calculating a value of an SLA parameter (e.g., defining a mapping for a metric unit of a value of an SLA parameter such as "Price" from "EUR" to "USD" can translate one function for calculating price to another one).
2. **Future SLA mapping type** defines a wish for adding (or deleting) a new SLA parameter that is supported by the private template to a public SLA template. Unlike ad-hoc mapping, future mapping cannot be applied immediately.

Figure 2 represents a sample public SLA template and a service consumer's private SLA template. Consumer's private template differs from the public template in four ways: (1) the name of the SLA parameter "Price", (2) the unit of a value of the parameter "Performance" (i.e., the parameter metric), (3) the parameter "Availability", which does not exist in the consumer's private SLA template, and (4) the SLA parameter "ClockSpeed", which exists in the consumer's private template but not in the public SLA template. Therefore, the consumer will create four SLA mappings to bridge these difference, namely (i) a simple ad-hoc mapping to map the difference of the parameter name, (ii) a complex ad-hoc mapping for defining a mapping function for measuring performance, (iii) a deleting wish (future mapping) for wishing for the deletion of the "Availability" parameter from the public SLA template, and (iv) an adding

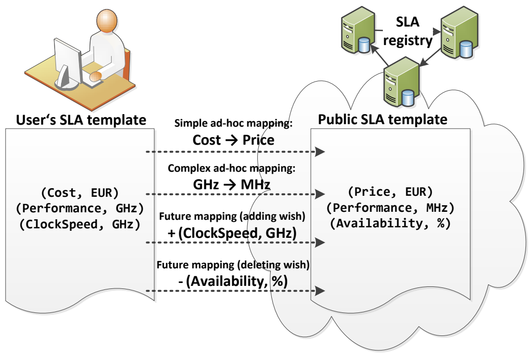

Figure 2: An example of the SLA mapping process.

wish (future mapping) for adding a new parameter "ClockSpeed" into the public SLA template.

3.2 Evolution of Public SLA Templates

After a certain period of time, SLA mappings are evaluated in order to adapt public SLA templates to the current market trends. By utilizing adaptation methods, we enable the creation of new branches of public templates that reflect consumers' needs more precisely. For example, a general template for internal medicine applications can be substituted by more specialized templates on cardiology and oncology. The adaptation process is executed in two steps. First, by applying clustering algorithms, similar SLA mappings are grouped (step 5 in Figure 1). For example, one group of SLA mappings could be for those templates that name their parameters in a certain language. In the second step (i.e., step 6 in Figure 1), adaptation methods are applied to each of the groups. Those methods analyze the SLA mappings and decide about the content of new public SLA templates. At the end of the process, new public templates are generated. With this approach, public SLA templates are adapted according to the need of market participants. This increases the likelihood that new requests of consumers will need less SLA mappings, reducing the cost of consumers and making the market more attractive.

After new public SLA templates have been generated, users can decide if they want to use the newly generated templates or if they prefer keeping the old ones. Note, although new templates reflect their requirements, users have to define new SLA mappings, which incur cost.

4 IMPLEMENTATION ISSUES

In this section, we discuss the algorithms and methods implemented for grouping similar SLA mappings and generating new public SLA templates. Besides, we formalize a measure of (dis)similarity of two SLA templates by utilizing an n-tuple SLA representation.

4.1 Clustering Algorithms for Grouping SLA Mappings

For the purpose of grouping similar sets of SLA mappings, we apply two clustering algorithms: DBSCAN and k-means. In our clustering algorithms, an item for clustering is a set of consumer's SLA mappings, and a cluster centroid is a newly generated public SLA template for a group of consumers (i.e., private SLA templates).

DBSCAN is a data clustering algorithm that is based on the density distribution of nodes and finds an appropriate number of clusters (Ester et al., 1996). The ε-neighborhood of a point p is defined as a set of points that are not farther away from p than a given distance ε. A point q is directly density-reachable from the point p, if q is in the ε-neighborhood of p, and the number of points in the ε-neighborhood of q is bigger than $MinPts$. A cluster satisfies two properties: (1) each two points are mutually density-reachable, and (2) if a point is mutually density-reachable to any point of the cluster, it is part of the cluster as well.

K-means is a clustering algorithm that, given an initial set of k means, assigns each observation to a cluster with the closest mean. It then calculates new means to be centroids of observations in the clusters and stops when the assignments no longer change (MacQueen, 1967). A frequent problem in k-means algorithm is the estimation of the number k. Within this paper, we explain two implemented approaches.

1. *Rule-of-Thumb* is a simple but very effective method in which k is set to $\sqrt{N/2}$, where N is the number of entities (Mardia and Kent, 1980).

2. *Hartigan's Index* is an internal index introduced in (Hartigan, 1975). Let $W(k)$ represents the sum of squared distances between cluster members and cluster centroid for k clusters. When grouping n items, the optimal number k is chosen so that the relative change of $W(k)$ multiplied with the correction index $\gamma(k) = n - k - 1$ does not significantly change for $k+1$, i.e.,

$$H(k) = \gamma(k)\frac{W(k) - W(k+1)}{W(k+1)} < 10$$

The threshold 10 shown in Hartigan's index is also used in our simulations. It is "a crude rule of thumb" suggested by Hartigan (Hartigan, 1975).

4.1.1 Computing Distance Between SLA Templates

In order to utilize clustering algorithms, the measure of distance between two clustering items must be defined, as well as the distance between a clustering item and a cluster centroid. As already mentioned in the previous section, an item for clustering is a set of consumer's SLA mappings, while a cluster centroid is an SLA template for the given group of consumers. Since having knowledge about the content of a public SLA template and about the SLA mappings a consumer created, we can exactly reconstruct consumer's private SLA template. Therefore, both the distance between two clustering items as well as between an item and a cluster centroid can be reduced to computing distances between two SLA templates. For this purpose, we introduce the n-tuple representation of SLAs.

An SLA consists of SLA parameters with SLA metrics and service level objectives (SLOs) associated to the parameters. We distinguish between the *structure* of an SLA, i.e., the list of parameters with their names and SLA metrics, and the *values* of an SLA, i.e., a list of numerical, boolean and string values of SLOs and SLA attributes. We introduce the n-tuple representation of an SLA, where first $n-1$ elements contain values of the SLA, while the last element contains the SLA structure. By using such a representation, we can define the distance between two SLA templates as an n-tuple, where first $n-1$ elements contain the differences between the two values of each of the parameters, while the final element contains a value representing the difference between the structures of the SLAs.

To formalize, we observe two SLA templates T_1 and T_2 defined by their SLA values $\{\alpha, \beta, \gamma, ...\}$ and the SLA structure Δ.

$$T_1 = (\alpha_1, \beta_1, \gamma_1, ..., \Delta_1), T_2 = (\alpha_2, \beta_2, \gamma_2, ..., \Delta_2)$$

The n-tuple D_{T_1,T_2} representing the distance between two SLA templates is defined as:

$$D_{T_1,T_2} = (f(\alpha_1, \alpha_2), f(\beta_1, \beta_2), f(\gamma_1, \gamma_2), ..., F(\Delta_1, \Delta_2))$$

The result of the function f for calculating the difference between two SLA values α_1 and α_2 depends on the type of its arguments and is defined as follows.

$$f(\alpha_1, \alpha_2) = \begin{cases} |\alpha_2 - \alpha_1|, & \text{if } \alpha_1 \text{ and } \alpha_2 \text{ are numerical} \\ 0, & \text{if } \alpha_1 \text{ and } \alpha_2 \text{ are not numerical and } \alpha_1 = \alpha_2 \\ 1, & \text{if } \alpha_1 \text{ and } \alpha_2 \text{ are not numerical and } \alpha_1 \neq \alpha_2 \end{cases}$$

The distance between the structures of SLA templates is expressed as a number of differences between properties of SLA templates. This value is calculated by iterating through all SLA parameters contained by at least one of the SLA templates, calculating the distance between the SLA templates with respect to the parameters. We define the distance function F calculating the difference between structures Δ_1 and Δ_2 of two SLA templates T_1 and T_2as

$$F(\Delta_1, \Delta_2) = \sum_{p \in T_1 \cup T_2} d_p(T_1, T_2)$$

where the distance between two SLA parameters of two SLA templates with respect to its properties (i.e., its name and metric) is defined as:

$$d_p(T_1, T_2) = \begin{cases} 0, & \text{if name and metric of } p \text{ are the same in } T_1 \text{ and } T_2 \\ 1, & \text{if } T_1 \text{ or } T_2 \text{ does not contain } p \\ 1, & \text{if only name or metric of } p \text{ differs in } T_1 \text{ and } T_2 \\ 2, & \text{if both name and metric of } p \text{ differ in } T_1 \text{ and } T_2 \end{cases}$$

After the result tuple has been calculated, it can be used to generate a single numerical value representing the distance between the two SLA templates. In order to do so, the result tuple must be normalized beforehand so that the tuple elements can be mutually comparable. Normalization is executed on each of the n tuple elements separately, where a value of an element is divided by a range of possible values for the element (maximum value minus the minimum value). Then, the final value representing the distance between SLAs can be computed by a simple function (e.g., summation of element values).

Note, in this paper, we discuss only the final element of the distance tuple, i.e., the difference between structures of two SLA templates. We plan to consider the values of SLA templates in our future work.

4.2 Adaptation Methods for Evolving Public SLA Templates

For adaptation of public SLA templates to react to market changes, we utilize three adaptation methods. These methods analyze submitted SLA mappings and for each SLA parameter determine whether the current parameter name and metric should be changed, as well as if a new parameter should be added or an existing one deleted.

To demonstrate the methods, we use the following example. The example considers the evolution of an SLA parameter π occuring in an initial public SLA template T_{init} when adapting the template based on SLA mappings of 100 service consumers. The value of the parameter π in the initial template

Table 1: Distribution of mappings for the SLA parameter π.

a) Name of the SLA parameter π

Name:	*Cost*	*Charge*	*Rate*	(*Price*)
Mappings:	15%	15%	40%	(30%)

b) Unit of the SLA parameter π

Name:	*USD*	*GBP*	*YPI*	(*EUR*)
Mappings:	38%	2%	40%	(20%)

Table 2: Distribution of mappings for the SLA parameter μ.

a) Name of the SLA parameter μ

Name:	*MemoryConsumption*	*Consumption*
Mappings:	70%	30%

b) Unit of the SLA parameter μ

Name:	*Mbit*	*Gbit*	*Tbit*
Mappings:	5%	55%	40%

is $(Price, EUR)$, where the first member of the tuple represents the parameter name, and the second member represents the metric unit for expressing the parameter value (see Table 1). The example also considers an SLA parameter μ which has not been defined in the initial public SLA template. Let 20 consumers express a wish for deleting the parameter π from T_{init}, and the rest of them define mappings according to the distribution presented in Table 1. Note that the last column of the table represents the number of non-mapped values, i.e., the number of private SLA templates containing the same value as in the public SLA template. Let 75 consumers express a wish for adding the parameter μ to the public template, with a name and unit by the distribution depicted in Table 2.

The maximum method selects the option which has the highest number of SLA mappings (i.e., the maximum candidate). This candidate will then be used in the new SLA template. If there is more than one maximum candidate with the same number of SLA mappings, one of them is chosen randomly. In the given example, only 20% of consumers wished for deleting the parameter π, which is not sufficiently high. Therefore, the parameter stays in the template with "Rate" as the new parameter name because of the highest number of mappings (40% in comparison to 30% who did not create mappings, i.e., who want to keep the name "Price"). The price will be expressed in Japanese Yens due to the highest number of mappings. The parameter μ will also be in the new template, since there are more than 50 wishes for this parameter, and parameter name and unit will be $(MemoryConsumption, Gbit)$.

In order to lower the requirements for selecting the maximum candidate, **the threshold method** introduces a threshold value. In this method, the value gets chosen if it is used more than the given threshold and has the highest count. If more than one value satisfies the conditions, one of them is chosen randomly. Throughout the evaluation in this paper, we fix the threshold to 60%. In the given example, neither the mappings for the name nor for the unit of the parameter π exceeds the threshold value. As for the parameter μ, it will be introduced in the new public template with the properties chosen according to the maximum method.

The significant-change method is a modification of the maximum-percentage-change method introduced in (Maurer et al., 2010). In this method, an SLA parameter property is only changed, if the percentage difference between the highest count value and the current public SLA template value exceeds a given threshold, which we assume to be significant at $\sigma_T > 15\%$. In the given example, 40 consumers have the name "Rate" for the "Price" parameter, while 30 consumers use the same name as in the public SLA template. Since the percentage difference of 33% is higher than the given threshold, "Rate" will be chosen as the new name for the parameter. As the new unit, "YPI" will be chosen. Since "MemoryConsumption" does not exist in the old public SLA template, the decision about this SLA parameter is made as described for the maximum method.

5 FORMALIZATION OF THE UTILITY AND COST MODEL

As already mentioned in Section 1, the approach presented in this paper brings many benefits to market participants. However, it also incurs cost. Namely, whenever a public SLA template has been adapted, the users of the template have to define new SLA mappings. Within this section, we investigate these costs and assess the benefits of our approach. For this purpose, we introduce the utility and cost model.

In the following, we observe a set of SLA parameters $P_{N,F}$, where possible names of a parameter $\alpha \in P_{N,F}$ are $N(\alpha) = \{A, A', A'', ...\}$, and the set of possible metrics is $F(\alpha) = \{f_\alpha, f'_\alpha, f''_\alpha, ...\}$. Let there be a set of consumers' private SLA templates $C = \{c_1, c_2, ..., c_n\}$ and two public SLA templates: the initial public SLA template T_{init}, and the newly generated public SLA template (for the given set of consumers) T_{new}.

For each SLA template $p \in C \cup \{T_{init}, T_{new}\}$, we define the relationship between a certain SLA parameter with its name and metric as

$$SLA_p : P_{N,F} \to \bigcup_{\alpha \in P_{N,F}} N(\alpha) \cup \emptyset \times \bigcup_{\alpha \in P_{N,F}} F(\alpha) \cup \emptyset$$

where a parameter of a template is represented as a tuple $(name, metric)$, i.e., for an SLA parameter α in an SLA template p it holds

$$SLA_p(\alpha) \mapsto (N(\alpha), F(\alpha))$$

If the value of α in a template p is $SLA_p(\alpha) = (A, f_\alpha)$, then we can define the projection function Π as

$$\Pi_N(SLA_p(\alpha)) = A, \quad \Pi_F(SLA_p(\alpha)) = f_\alpha$$

with the abbreviation

$$N_p(\alpha) = \Pi_N(SLA_p(\alpha)), \;\; F_p(\alpha) = \Pi_F(SLA_p(\alpha))$$

Additionally, we define the function χ to determine if an SLA template p contains an SLA parameter α:

$$\chi_p(\alpha) = \begin{cases} 0, & \text{if } p \text{ does not contain } \alpha \\ 1, & \text{if } p \text{ contains } \alpha \end{cases}$$

Finally, we can define the utility function for a consumer c and for an SLA parameter α as follows.

$$u_c^+(\alpha) = \begin{cases} 0, & \chi_c(\alpha) \neq \chi_{T_{new}}(\alpha) \vee \chi_c(\alpha) = \chi_{T_{new}}(\alpha) = 0 \\ A, & \chi_c(\alpha) = \chi_{T_{new}}(\alpha) = 1 \wedge \\ & N_c(\alpha) \neq N_{T_{new}}(\alpha) \wedge F_c(\alpha) \neq F_{T_{new}}(\alpha) \\ B, & \chi_c(\alpha) = \chi_{T_{new}}(\alpha) = 1 \wedge \\ & (N_c(\alpha) = N_{T_{new}}(\alpha) \wedge F_c(\alpha) \neq F_{T_{new}}(\alpha) \vee \\ & N_c(\alpha) \neq N_{T_{new}}(\alpha) \wedge F_c(\alpha) = F_{T_{new}}(\alpha)) \\ C, & \chi_c(\alpha) = \chi_{T_{new}}(\alpha) = 1 \wedge \\ & SLA_c(\alpha) = SLA_{T_{new}}(\alpha) \end{cases}$$

As stated in the definition of the utility function, the consumer gains no utility for an SLA parameter if it does not exist in one of the SLA templates (consumer's private template or the public template). If it does exist, the utility can be A (if both name and metric for an SLA parameter differ in user's private SLA template and the public SLA template), B (if only name or metric differ), or C (if templates do not differ with respect to the parameter).

The cost function for a consumer c and an SLA parameter α is defined as follows.

$$u_c^-(\alpha) = \begin{cases} 0, & SLA_{T_{new}}(\alpha) = SLA_{T_{init}}(\alpha) \vee \\ & SLA_c(\alpha) = SLA_{T_{new}}(\alpha) \vee \\ & (\chi_c(\alpha) = \chi_{T_{new}}(\alpha) = \chi_{T_{init}}(\alpha) = 1 \wedge \\ & (N_c(\alpha) = N_{T_{new}}(\alpha) \wedge F_{T_{new}}(\alpha) = F_{T_{init}}(\alpha) \vee \\ & N_{T_{new}}(\alpha) = N_{T_{init}}(\alpha) \wedge F_c(\alpha) = F_{T_{new}}(\alpha))) \\ E, & \chi_{T_{init}}(\alpha) \neq \chi_{T_{new}}(\alpha) \wedge \chi_c(\alpha) \neq \chi_{T_{new}}(\alpha) \\ D, & \text{otherwise} \end{cases}$$

The cost function states that a user has no cost if he does not have to create new SLA mappings i.e., if the parameter properties are the same in his private template and in the newly generated public template, or the new template did not change with respect to the parameter. If a user has to create a future mapping, the cost is equal to E. Finally, if a user has to create ad-hoc mappings for name, metric, or both, the cost is equal to D.

At the end, we define the overall utility U^+ and the overall cost U^- for all service consumers C as:

$$U^+ = \sum_{c \in C} \sum_{\alpha \in P_{N,F}} u_c^+(\alpha), \quad U^- = \sum_{c \in C} \sum_{\alpha \in P_{N,F}} u_c^-(\alpha)$$

The overall net utility U^O is then calculated as

$$U^O = U^+ - U^- \tag{1}$$

Note that the values for the utility and the cost functions are not strictly defined, but considering the effort or benefit it should hold that $0 \leq A \leq B \leq C$ and $0 \leq D \leq E$.

6 EVALUATION

In this section, we measure and investigate the benefits and costs of defining SLA mappings and adapting public SLA templates to the current needs of market participants, using the utility and cost model presented in Section 5. Moreover, we assess the scalability of the approach and discuss the quality of newly generated public SLA templates.

6.1 Simulation Environment

For the simulation of our approach, we designed a testbed within the VieSLAF environment. VieSLAF is a framework for semi-automated management of SLA mappings and generation of SLA templates (Brandic et al., 2010). The major VieSLAF components are the knowledge base and the SLA mapping middleware.

1. **The knowledge base** is used for storing and managing public SLA templates. Browsing and creating SLA templates is enabled over searchable repositories implemented in Microsoft SQL 2008 DBMS with a Web service front-end to provide the interface for the SLA mapping management.
2. **The SLA mapping middleware** is based on several Windows Communication Foundation services for administration (e.g., creating SLA templates), accounting (e.g., creating consumer accounts), querying (e.g., finding an appropriate SLA template), management of SLA mappings (e.g., defining mappings for a user), and adaptation of public SLA templates.

Both knowledge base and the registry services can be accessed only with appropriate access rights depending on three access roles: *service consumer*, *service provider*, and *registry administrator*.

We describe the VieSLAF architecture and the testbed used for the simulation based on Figure 3. The simulation process is conducted in several steps. First, a service provider assigns his service (e.g., software, resources, infrastructure) to a public SLA template using VieSLAF's registry services. Service consumers than fetch the public template (step 2) and

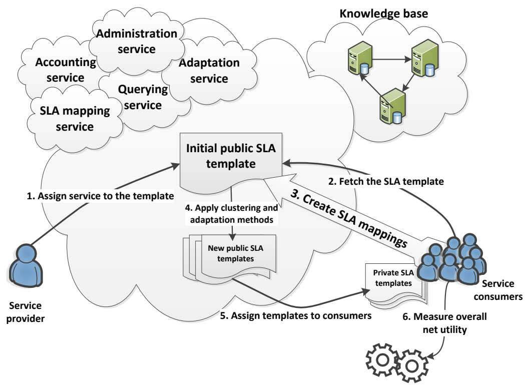

Figure 3: Simulation testbed.

generate SLA mappings to map the differences between their private templates and the public SLA template (step 3). After the end of the mapping process, the clustering algorithms are applied and consumers' SLA mappings are grouped (step 4). For each of the groups of SLA mappings, new public SLA templates are generated and assigned to the consumers (step 5).

The process is evaluated by investigating the overall net utility (step 6). For evaluation purposes, all described clustering algorithms and adaptation methods for creating new public SLA templates are used and compared. The number of service consumers creating SLA mappings is varying from 100 to 10000. The initial public SLA template is static and contains 8 SLA parameters, while the consumers' private SLA templates are generated randomly at the beginning of the evaluation process and can contain up to 11 SLA parameters, where, for each SLA parameter, consumers can choose between 5 names and 2 metrics.

6.2 Evaluation Results

6.2.1 Overall Net Utility

To assess the benefits and costs of the SLA mapping approach, we use the overall net utility defined by Equation (1), where the values for the constants of the utility and cost model are fixed as depicted in Table 3. For evaluation purposes, we simulate 500 service consumers creating SLA mappings to a public SLA template. In order to compare results achieved with different methods, all presented clustering algorithms and adaptation methods are used and compared. After the new SLA templates are generated, the overall net utility is calculated, representing the difference between gains and costs of the SLA mapping approach. The simulation results are depicted in Figure 4.

Using the results depicted in the graph, we can compare the approach described in this paper with our previous work on SLA mapping described in (Maurer et al., 2010), where we did not utilize clustering algorithms, but instead applied adaptation methods to create a single new public SLA template for all market participants. The first column of the graph presents the overall net utility achieved when creating a single new public template, while the rest of the columns represent results achieved by utilizing different clustering algorithms. As it can be seen from the graph, by generating only one public SLA template for all service consumers, the overall net utility is significantly lower than in the other cases. This is due to creating more new public SLA templates that reflect consumers' needs more precisely.

Table 3: Values of the utility and cost function variables used for the evaluation.

Variable	*A*	*B*	*C*	*D*	*E*
Value	1	2	3	1	2

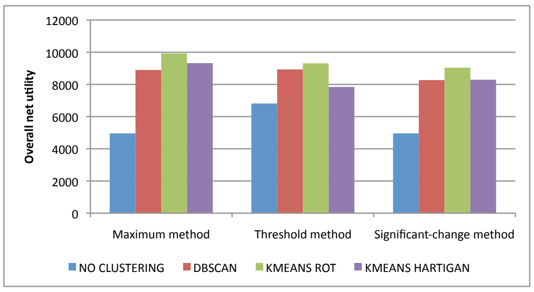

Figure 4: Evaluation results: overall net utility.

When comparing the overall net utility gained by utilizing different adaptation methods, we can conclude that the best results are achieved by using the maximum method. Although the threshold method causes significantly lower number of changes (due to the high threshold), and therefore very low cost, the maximum method adapts public template more dynamically and achieves higher utility rate. Highest overall net utility is achieved by the *k*-means clustering algorithm with the rule-of-thumb method, due to creating large number of very specific clusters.

The utility rate highly depends on the number of generated clusters. The more clusters are created, the higher utility will be achieved. This relation can be seen when combining the results depicted in Figure 4 with Table 4, representing the number of clusters per clustering algorithm and per number of service consumers creating SLA mappings. The maximum rate of overall net utility for n service consumers can be achieved by creating n clusters i.e., by generating one public SLA template per consumer. In this case, public SLA templates would be equal to consumers' private templates. Moreover, the utility rate would be maximum, and the cost would be equal to 0. However, by raising the number of new public SLA templates in the market, the liquidity of its goods is at the lowest point. Therefore, we must find a balance

Table 4: Number of generated clusters per algorithm and number of service consumers.

	DBSCAN	k-means (RoT)	k-means (H)
100	5	7	3
200	5	10	3
300	6	12	3
500	6	15	5
1000	7	22	6
2000	7	31	6
5000	8	50	6
10000	10	69	10

between keeping low cost of creating new SLA mappings and ensuring high liquidity of market goods. The problem of finding the optimal number of new public SLA templates to be introduced to the market will be addressed in our future work.

Since the number of generated clusters influences the overall net utility, we conclude that both the overall net utility and the dataset properties determine the quality of newly generated public SLA templates. Therefore, it is not sufficient to investigate only the utility gained, but also the properties of the simulation dataset that may have influenced the results of the utility measurements. For this purpose, we introduce the measure of compactness. In our discussion of dataset properties, C represents a set of clusters, where $C_j \in C$ is a cluster, and C_{jc} its centroid. R_i is a clustering item, and $D(R_i, R_k)$ a distance between two items. $|C|$ represents the total number of clusters, and $|C_i|$ represents a number of items in a cluster C_i.

Compactness is reciprocal of average distance between consumers' private SLA templates within clusters, and is formally defined as follows.

$$\left(\frac{1}{|C|}\sum_{C_j\in C}\left(\frac{1}{|C_j|(|C_j|-1)}\sum_{R_i\in C_j}\sum_{R_k\in C_j} D(R_i,R_k)\right)\right)^{-1}$$

Since a cluster represents a group of consumers with similar requirements, high compactness implies well formed market. Figure 5 depicts the compactness rates achieved by utilizing clustering algorithms. For this purpose, we have simulated SLA mappings specified by different number of service consumers, varying from 100 to 10000. The adaptation method used for the evaluation is the maximum method.

High utility is common for compact clusters, since all members of the cluster have similar properties, and the cluster centroid i.e., newly generated SLA template, reflects those properties more precisely. On the other side, items of a less compact cluster differ in a larger extent, and it is probable that the utility achieved by creating a new public SLA template will be lower. High compactness can be achieved by creating large number of smaller clusters and is maximized when there is one cluster per item.

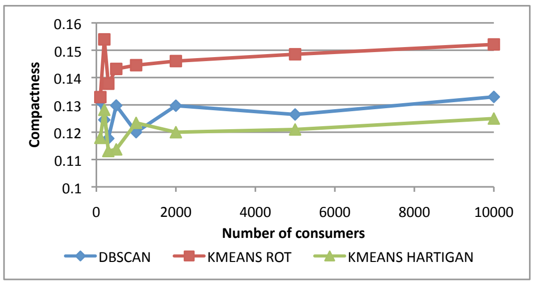

Figure 5: Evaluation results: compactness.

As depicted in Figure 5, the k-means algorithm with the Hartigan's index achieves lowest rate of compactness. This is due to creating small number of general clusters, where items mutually differ in a large extent. Therefore, as it can be seen on Figure 4, this algorithm results in low overall net utility. k-means algorithm with the-rule-of-thumb achieves larger rate of compactness, and consequently larger rate of overall net utility, due to large number of specific clusters.

6.2.2 Scalability

For Cloud environments, it is of great importance that the resource markets are highly scalable. In order to assess the scalability of our approach, we measured execution time of the clustering process for different number of users creating SLA mappings, varying from 100 to 10000. Results are depicted in Figure 6.

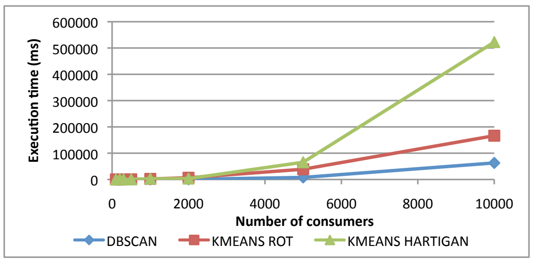

Figure 6: Evaluation results: execution time.

As showed in the graph, the k-means algorithm with the Hartigan's index takes most time to compute groups of SLA mappings. This is due to the nature of the Hartigan's index method, in which the clustering algorithm runs in several iterations and computes different number of clusters before it finds the optimal number k. The best performance is achieved by the DBSCAN algorithm. This result is also not surprising, since unlike the k-means algorithm, it creates smaller number of clusters and does not compute cluster centroids, which incurs cost.

The results show that the approach presented in this paper is highly scalable. It is important to notice that the clustering process in the worst scenario takes less than 0.08% of the total simulation execution time. This percentage decreases with the rising of the number of service consumers. Furthermore, the execution time of *k*-means algorithm with the Hartigan's index, which is far the highest, can be improved by introducing heuristics for determining the initial number *k*, which we will consider in our future work.

7 CONCLUSION AND FUTURE WORK

In this paper, we have extended the SLA mapping approach by introducing the idea of applying clustering algorithms to group users' SLA mappings. This enables the automatic creation of new public SLA templates, paving the way for autonomic market management. In particular, we introduced the *n*-tuple SLA representation to compute the difference between two SLA templates, utilized two clustering algorithms for grouping SLA mappings, and used three adaptation methods for generating new SLA templates. We have investigated the utility and cost to users when applying the adaptive SLA mapping approach. Our simulation results show that our approach facilitates continuous market evolution and adaptation of public SLA templates, responding to market trends and changes.

In the future, we will consider heuristic methods for improving the efficiency of clustering algorithms for grouping SLA mappings and investigate the full possibilities of the *n*-tuple representation of SLAs. Also, we will address the problem of finding the optimal number of new public SLA templates to be introduced to the market, in order to ensure liquidity on the market while keeping a high utility rate.

ACKNOWLEDGEMENT

The work described in this paper is supported by the Vienna Science and Technology Fund under the grant agreement ICT08-018 Foundations of Self-Governing ICT Infrastructures (FoSII), and by the National Research Foundation of the Ministry of Education, Science and Technology of Korea under the grant K21001001625-10B1300-03310.

REFERENCES

Brandic, I., Music, D., and Dustdar, S. (2010). Vieslaf framework: Facilitating negotiations in clouds by applying service mediation and negotiation bootstrapping.

Buyya, R., Abramson, D., and Giddy, J. (2001). A case for economy grid architecture for service oriented grid computing. *Parallel and Distributed Processing Symposium, International*, 2.

Buyya, R., Yeo, C. S., Venugopal, S., Broberg, J., and Brandic, I. (2009). Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Computer Systems*, 25(6):599 – 616.

Dobson, G. and Sanchez-Macian, A. (2006). Towards unified qos/sla ontologies. In *Services Computing Workshops, 2006. SCW '06. IEEE*, pages 169 –174.

Ester, M., Kriegel, H. P., Sander, J., and Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. pages 226–231. AAAI Press.

Green, L. (2006). Service level agreements: an ontological approach. In *Proceedings of the 8th international conference on Electronic commerce: The new e-commerce: innovations for conquering current barriers, obstacles and limitations to conducting successful business on the internet*, ICEC '06, pages 185–194, New York, NY, USA. ACM.

Hartigan, J. A. (1975). *Clustering Algorithms*. John Wiley & Sons Inc.

Karänke, P. and Kirn, S. (2007). Service level agreements: An evaluation from a business application perspective. In *In eChallenges e-2007 Conference*, pages 104–111. IEEE-CS Press.

MacQueen, J. B. (1967). Some methods for classification and analysis of multivariate observations. In Cam, L. M. L. and Neyman, J., editors, *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. University of California Press.

Mardia, K. V. and Kent, J. T. (1980). *Multivariate Analysis*. Academic Press.

Maurer, M., Emeakaroha, V. C., Risch, M., Brandic, I., and Altmann, J. (2010). Cost and benefit of the sla mapping approach for defining standardized goods in cloud computing markets. In *International Conference on Utility and Cloud Computing (UCC 2010) in conjunction with the International Conference on Advanced Computing (ICoAC 2010)*.

Neumann, D., Stösser, J., and Weinhardt, C. (2008). Bridging the adoption gap-developing a roadmap for trading in grids. *Electron. Market.*, 18:65–74.

Nimis, J., Anandasivam, A., Borissov, N., Smith, G., Neumann, D., Wirstrm, N., Rosenberg, E., and Villa, M. (2008). Sorma business cases for an open grid market: Concept and implementation. In *Grid Economics and Business Models*, volume 5206 of *Lecture Notes in Computer Science*, pages 173–184. Springer Berlin / Heidelberg.

Oldham, N. and Verma, K. (2006). Semantic ws-agreement partner selection. In *In 15th Int. WWW Conf*, pages 697–706. ACM Press.

Risch, M., Brandic, I., and Altmann, J. (2010). Using sla mapping to increase market liquidity. In *Service-Oriented Computing. ICSOC/ServiceWave 2009 Workshops*, volume 6275 of *Lecture Notes in Computer Science*, pages 238–247. Springer Berlin / Heidelberg.

Samuelson, P. and Nordhaus, W. (2005). *Economics*. McGraw-Hill/Irwin, 18. ed., internat. ed. edition.