SEOUL NATIONAL UNIVERSITY
College of Engineering
Technology Management, Economics, and Policy Program

Cost-Benefit Analysis of an SLA Mapping Approach for Defining Standardized Cloud Computing Goods

Michael Maurer, Vincent C. Emeakaroha, Ivona Brandic, Jörn Altmann

TEMEP Discussion Paper No. 2011:77

Jörn Altmann, Editor

Office: 37-305
Technology Management, Economics, and Policy Program
College of Engineering
Seoul National University
599 Gwanak-Ro, Gwanak-Gu
Seoul 151-742
South-Korea

Phone: +82-70-7678-6676
Fax: +1-501-641-5384
E-mail: jorn.altmann@acm.org

Cost-Benefit Analysis of an SLA Mapping Approach for Defining Standardized Cloud Computing Goods

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] TEMEP & Department of Industrial Engineering, College of Engineering
Seoul National University
Seoul, South-Korea
jorn.altmann@acm.org

July 2011

Abstract: Due to the large variety in computing resources and, consequently, the large number of different types of service level agreements (SLAs), computing resource markets face the problem of a low market liquidity. Restricting the number of different resource types to a small set of standardized computing resources seems to be the appropriate solution to counteract this problem. Standardized computing resources are defined through an SLA template. An SLA template defines the structure of an SLA, the service attributes, the names of the service attributes, and the service attribute values. However, since existing research results have only introduced static SLA templates so far, the SLA templates cannot reflect changes in user needs and market structures. To address this shortcoming, we present a novel approach of adaptive SLA matching. This approach adapts SLA templates based on SLA mappings of users. It allows Cloud users to define mappings between a public SLA template, which is 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 are adapted to the demand of Cloud users, we also analyze the costs and benefits 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, we investigate how the costs differ with respect to the public SLA template adaptation method. The simulation results show that the use of heuristics within adaptation methods allows balancing the costs and benefits of the SLA mapping approach.

Keywords: Service Level Agreements, Cloud Architecture, Market Liquidity, Cloud Markets, Cost-Utility Modeling, SLA Matching, Goods Standardization.

JEL Classification Numbers: C15, C61, C63, D40, D45, L22, L86.

1. Introduction

It Allocation of Cloud computing resources is based not only on functional requirements but also on different non-functional requirements. Non-functional requirements, e.g., application execution time, reliability, and availability, are termed as quality of service (QoS) requirements and are expressed by means of service level agreements (SLAs). In order to facilitate SLA creation and SLA management, SLA templates have been introduced. SLA templates represent popular SLA formats. They comprise elements such as names of trading parties, names of SLA attributes, measurement metrics, and attribute values [1].

Despite the existence of SLAs, buyers and sellers of computing resources face the problem of varying definitions of computing resources in Cloud computing markets. 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. 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 offers from sellers and bids from buyers becomes very unlikely, i.e., the market liquidity (the likelihood of matching offers and bids) becomes very low [1].

Approaches that tackle 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. These SLA templates are static meaning that they do not change nor adapt over time.

Consequently, the existing approaches for the specification of SLA templates cannot easily deal with demand changes. Demand changes of users are caused through different factors (e.g., changing market conditions). 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. The existing approaches for the specification of SLA templates involve heavy user-interactions to adapt existing SLA templates to demand changes.

In this paper, we apply adaptive SLA mapping, a new, semi-automatic 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. In our context private templates do not necessarily imply that they are inaccessible to others, but the word "private" is used to differentiate it from the "public" template of the (public) registry. So, all consumers' and providers' templates are called "private", whereas the registry's template is called "public". Since a user cannot easily change the private SLA template due to internal or legal organizational requirements, an SLA mapping is a convenient workaround.

Our adaptive SLA mapping approach can use different adaptation methods. The benefit of using an adaptation method is decreased by some cost for the user. Costs are only incurred, if a user has to define a new SLA mapping to a public SLA template due to its adaptation. Within this paper, we investigate these costs. In particular, we investigate how public SLA templates can be adapted to the demand of Cloud users and how the costs and benefits differ with respect to the public SLA template adaptation method used.

After introducing a reference adaption method for our analysis, we compare two additional adaptation methods which differ in the heuristics applied. The heuristics have been introduced in order to find a balance between the benefit of having a public SLA template that is identical to most of the private SLA templates and the cost of creating new SLA mappings and new public SLA templates. As the metrics for assessing the quality of the adaptation method, we define the overall system net utility of all users. The net utility considers the benefit of having the same attribute and attribute name in the public SLA template as in the private SLA template, as well as the cost of defining a new SLA attribute mapping.

The benefits of the adaptive SLA mapping approach 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 users; (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 cost–benefit 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 our work in context of the state-of-the-art, we briefly describe Cloud resource management, Cloud marketplaces, and the existing work on SLA matching

2.1. Cloud Resource Management

There is a large body of work about managing resource provisions, negotiations, and federation of Cloud and Grid resources. An example is [21]. They designed agent technology to address the federation problems in Grids, i.e., resource selection and policy reconciliation. [22] propose a new abstraction layer for managing the life cycle of services. It allows automatic service deployment and escalation depending on the service status. This abstraction layer can be positioned on top of different Cloud provider infrastructures, hence mitigating the potential lock-in problem and allowing the transparent federation of Clouds for the execution of services. [23] investigate three novel heuristics for scheduling parallel applications on utility Grids, optimizing the trade-off between time and cost constraints.

However, most of the related work on resource management considers resource provision from the provider's point of view and does not consider Cloud computing infrastructures in the context of a marketplace.

2.2. Cloud Market

Currently, a large number of commercial Cloud providers have entered the utility computing market, offering a number of different types of services. These services can be grouped into three types: 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] and [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, which is also called Infrastructure-as-a-Service (IaaS) 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 different types of virtual machines 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, which are called Software-as-a-Service (SaaS) are services offered by Google (Google Apps [6]) and Salesforce.com [14]. These companies provide access to software on pay-per-use basis. These SaaS solutions can hardly be integrated with other solutions, because of their complexity. Examples for the third kind of Cloud services, which are called Platform-as-a-Service (PaaS), 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 the PaaS service offerings is a seamless integration with the users' applications, standardization of interfaces is largely absent. Concluding, we can state that, apart from first attempts for the IaaS service type, standardization attempts do almost not exist.

2.3. Service Level Agreement Matching

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 ontology-based 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 5successful markets, as shown in our earlier work [1]. Additionally, several works on SLA management have been 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

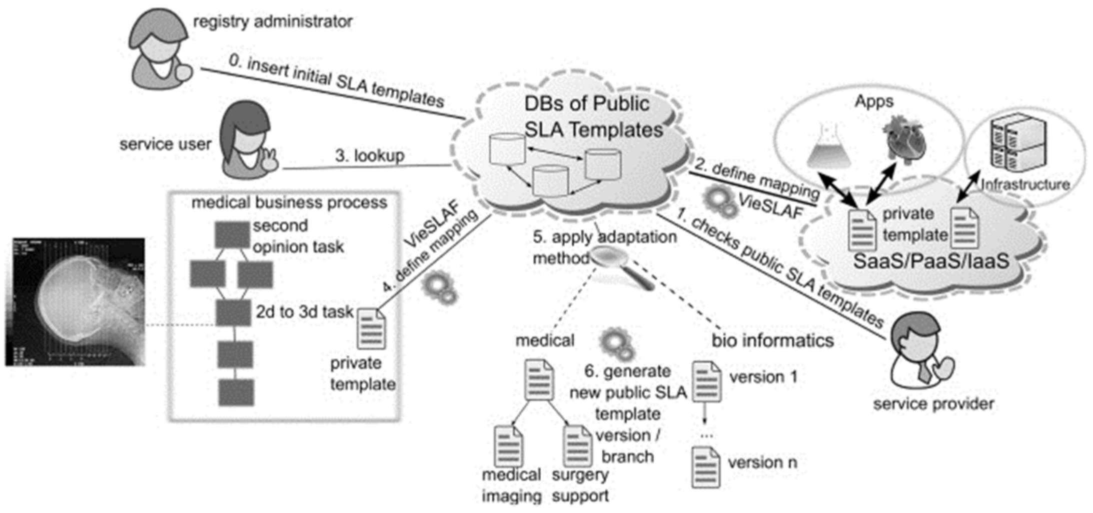

Figure 1: Use case of SLA mapping.

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 databases (step 1, DBs of public SLA templates, Figure 1). If some differences between its resources (i.e., its private SLA templates) and the public templates exist, the provider defines SLA mappings, which can transform the private template into the public template and vice versa (step 2, Figure 1). The management of SLA mappings, which is performed with VieSLAF, is explained in detail in [3].

In step 3 of Figure 1, Cloud users can look up Cloud services that they want to use in their workflow. The figure exemplifies a business process 6(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 an 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. In general, the SLA 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. Then, 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). The adaptation of attributes, attribute names, and attribute values can not only replace SLA templates but also create new versions and branches of public SLA templates (step 6). A new branche of a public SLA template can be created, if specilization needs to be captured (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 public SLA template occurs, if different attribute combinations in the templates are used. Figure 1 shows n template versions in the bioinformatics domain.

3.2. Public SLA Template Life Cycle

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

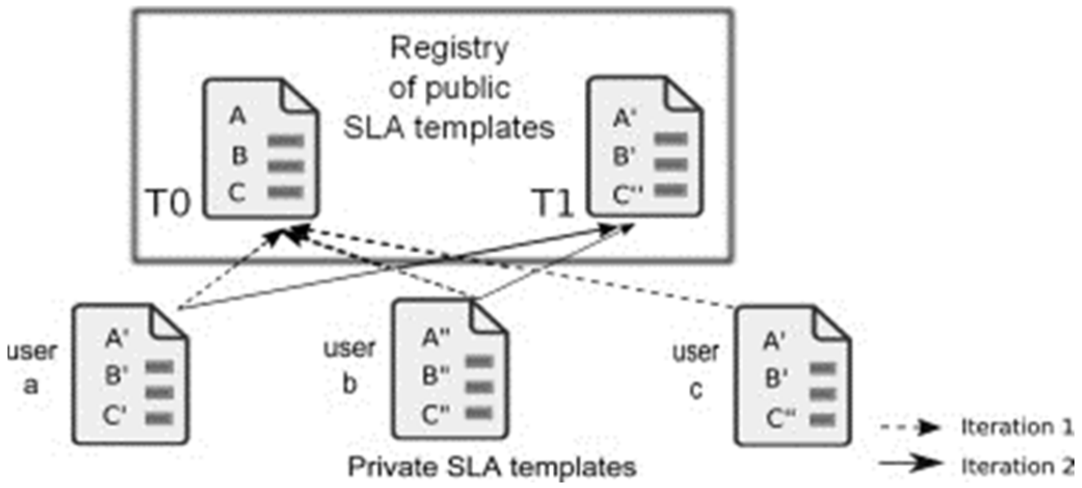

Figure 2: SLA mapping process.

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 names of the public SLA template *(A, B, C)* and the attribute names of each user differ, all users have 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 SLA template has changed, users need to change their mappings as well (iteration 2). Consequently, user *a* only needs one attribute mapping, user *b* needs two attribute mappings, 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 these users.

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

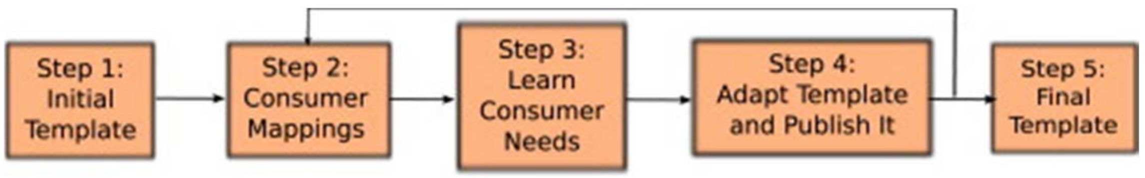
Step 1: Initial Template
Step 2: Consumer Mappings
Step 3: Learn Consumer Needs
Step 4: Adapt Template and Publish It
Step 5: Final Template

Figure 3: Formalized public SLA template life cycle.

An initial template is created in the beginning of the life cycle (step 1, Figure 3). Afterward, consumers perform SLA mappings to their private SLA templates (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 (i.e., new tasks has to be executed or new users joined the marketplace), the process continues with step 2. In practice, the time between two iterations could correspond to a time period of one week, e.g., but can be set to any value depending on the volatility of the market. During that time new SLA mappings are solicited from users (i.e., consumers and providers).

3.3. Adaptation Methods

The adaptation methods determine for every attribute name of the public SLA template separately, whether the current attribute name should be adapted or not. In this paper, we investigate three adaptation methods. The first adaptation method is the maximum method (which has been applied in the example shown in Figure 2). The remaining two adaptation methods apply heuristics, in order 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 of 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 attribute name 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% within this paper. A low threshold leads to more mappings, whereas a high threshold leads on average to fewer mappings. After τ iterations (e.g., $\tau = 10$), the method re-starts with executing the first step. This allows slighter changes to take effect.

Example: Let us 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 32/28−1.0=14.3%.

3.2. Utility and Cost Model

To 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 and cost model. At its core, the model defines the utility function and the cost function. The utility function and the cost function, which take attributes of the private SLA template of the customer and the attributes of the public SLA template as input variables, help to quantify the benefit and the cost.

The model assumes an increase in benefit, if an attribute (or attribute name or attribute value) 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, in the other case, the provisioned resource of the provider) and, therefore, no inefficiency through resource over-provisioning occurs. The model also captures the effort (i.e., cost) of changing an SLA mapping. The cost 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 functions, 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 CPU Cores, Cores}*, and $Var(\beta) = \{B, B', B'', B'''\}$.

Assuming a set of private SLA templates $C = \{c_1, c_2, \ldots, c_n\}$ of customers, we can now define the relationship of a specific SLA attribute to a specific attribute name of this SLA attribute at a specific point in time (i.e., iteration) $i \in N$ for a SLA template p, $p \in C \cup \{T\}$ (i.e., private or public SLA template) as

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

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 time 0 (i.e., iteration 0).

Based on these definitions and the utility function exemplified in [20], 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

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

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

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 previous iteration has been adapted 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 previous 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 $u^o_{c,I,\pi}$:

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

In addition to this, 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 or as the sum of the net utility of all consumers *c* for all attributes at iteration *i*:

$$U_i^o = U_i^+ - U_i^- = \sum_{c \in C} u_{c,i}^o. \tag{8}$$

4. Simulation Environment

In order to analyze the performance of the three adaptation methods with respect to the balance 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 (*VieSLAF*) and software that simulates SLA mappings of users. Figure 4 illustrates our *emulation* testbed. The components that are drawn in white belong to *VieSLAF*. It comprises the knowledge base, the middleware for managing SLA mappings provided by consumers and providers, and the adaptation methods. The grey components indicate the components that simulate SLA mappings of users. A sample provider and a sample consumer are shown in the lower part of Figure 4.

The SLA mapping middleware, which follows a client-server design, facilitates the access by the provider and the consumer to registries. It provides to users a GUI 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 in the following paragraph.

The *RegistryAdministrationService* provides methods for the manipulation of the database. This service requires administrator rights. An example for these methods is the creation of template domains. Another service of the SLA mapping middleware is the *SLAMappingService*, which is used for the management of SLA mappings by service consumers and service providers (cf. (3) of Figure 4). Providers and consumers may also search for appropriate public SLA templates through *SLAQueryingService* and define appropriate SLA mappings by using the method *createAttributeMapping*. With each service request, it is also checked whether the user has also specified any new SLA mappings. The SLA mappings (i.e., transformation rules) are stored in the private database of the user and can be re-used by the user for her next SLA mapping.

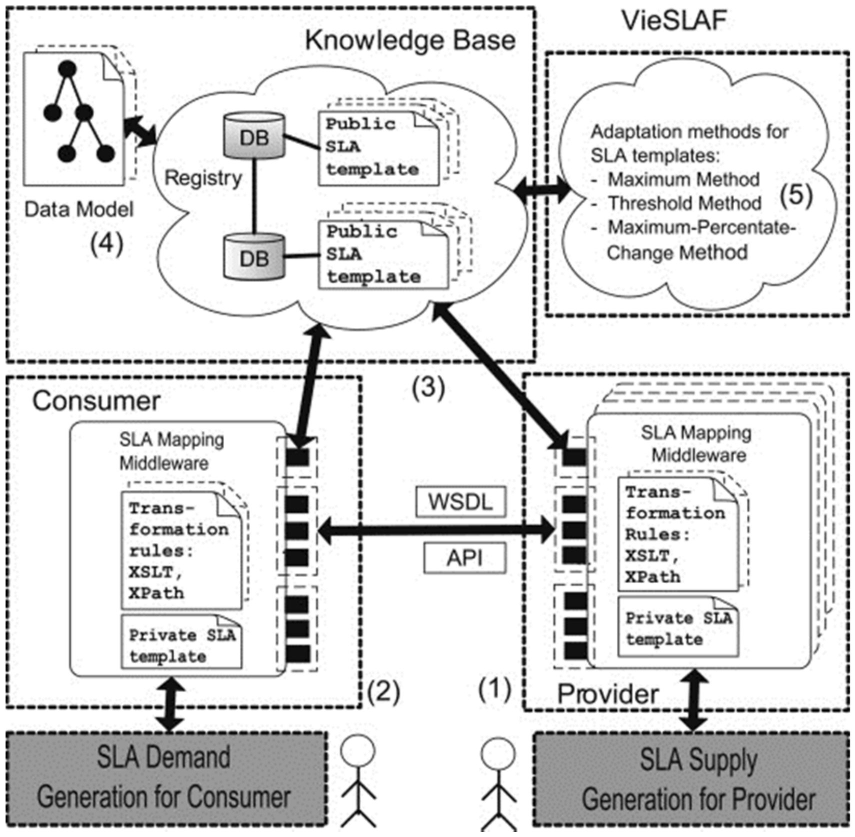

Figure 4: Adaptive SLA mapping architecture using VieSLAF.

The knowledge base for storing the SLA templates in a predefined data model ((4) of Figure 4) is implemented as registries representing searchable repositories. Currently, we have implemented anMS-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 [25]). SLA templates are stored in a canonical form, enabling the comparison of the XML-based SLA 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*. The registry administrator may create new SLA templates. A service consumer and a service provider may search for SLA templates and can submit their SLA mappings.

Based on the submitted SLA mappings, public SLA templates are adapted by the registry administrator, using one of the adaptation methods ((5) of Figure 4), introduced in section 3.3.

4.2. Simulation Parameter Settings

For our simulation, we define five scenarios on how often attribute names occur in private SLA templaes on average. In particular, 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 counts of 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 counts of remaining three attribute names, which are equally large.

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'''. The initial setting of attribute α is the attribute name 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

As an example for the use of the scenarios, we take scenario c. If the attribute α (*Number of Cores in one CPU*) 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 SLA mappings of the user. We used these parameter settings for each of the three adaptation methods. 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

5. Experimental Results and Analysis

5.1. Net Utilities of Adaptation Methods

Using our SLA mapping approach, the user gets benefit of having access to public SLA templates that reflect the overall market demand (i.e., the demand of all users). This benefit of a user is expressed with equation 2. However, this benefit comes with the cost for defining new SLA mappings whenever the public SLA template changed (equation 3)

Within this section, we investigate the cost for all users (equation 7), the utility of all users (equation 6), and the net utility of all users (equation 8) with respect to three adaptation methods. The net utility metric is used to decide which of the three adaptation methods investigated 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 five scenarios. The advantage of the maximum method is that the public SLA template generated with this method minimizes the differences to all private SLA templates of all users. This method, however, requires many SLA mappings.

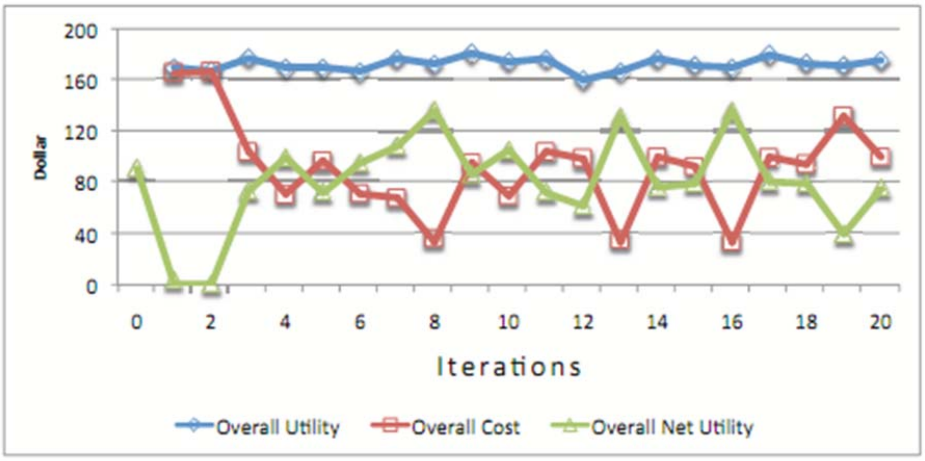

Figure 5: Utility, cost, and net utility for the maximum method.

Figure 5 shows, as expected, that the maximum method generates a high utility, since it achieves many matches 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 to the users. Consequently, the net utility is far lower than the utility.

In order to address this issue of high cost of the maximum method, 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.

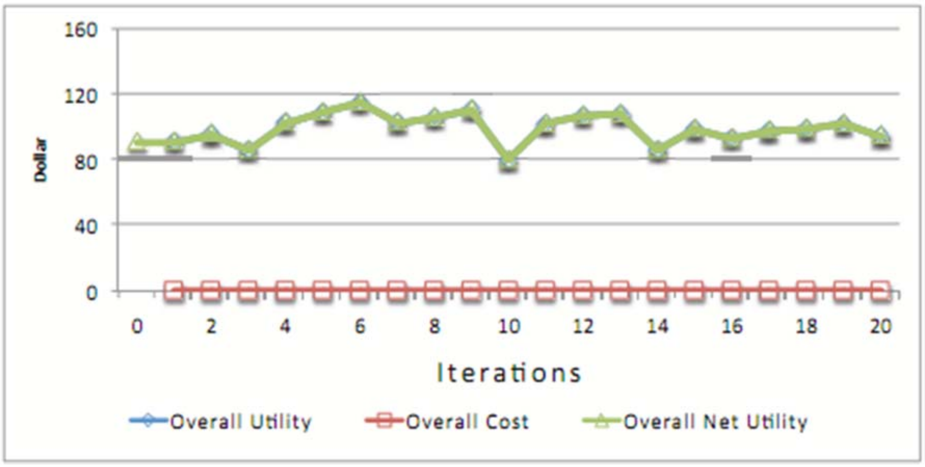

Figure 6: Utility, cost, and net utility, for the threshold method.

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., a threshold of 60%), resulting in no changes of the public SLA template attribute names. Nevertheless, the utility (and net utility) is not higher than ones of the maximum method, just more stable across the 20 iterations. Therefore, the threshold method with a threshold of 60% could be considered the opposite strategy to the maximum method. That means, the initial public SLA template does not get adapted at all. By lowering the threshold parameter such that the threshold parameter for 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 that we investigate. The results are shown in Figure 7.

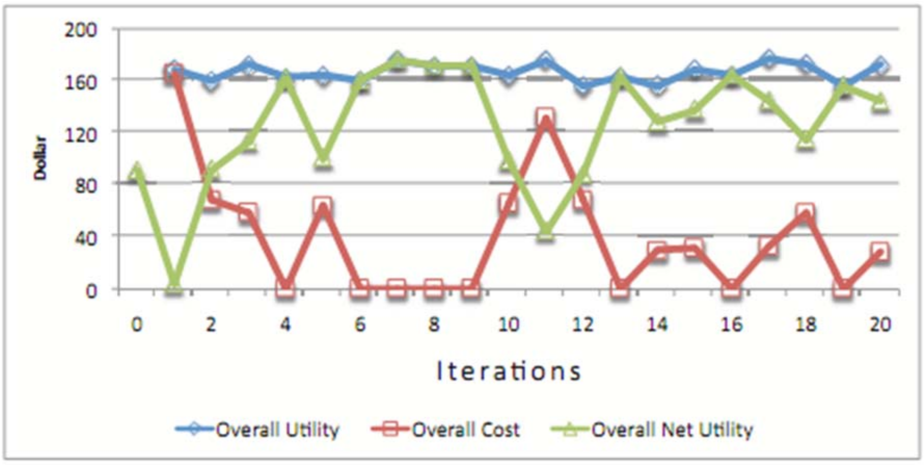

Figure 7: Utility, cost, and net utility for the maximum-percentage-change method with $\tau = 10$

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 new SLA mappings needed (Figure 7). At these iterations, the cost of the SLA mappings is very high, since 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, therefore, 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 the initial SLA attribute name 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 ones of 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 that 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 while preserving the benefit of adapted public SLA templates.

5. 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 an increase in average overall net utility. The best-performing adaptation method is the maximum-percentage-change method.

Acknowledgements

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), Foundations of Self-governing ICT Infrastructures (FoSII).

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 internationalWorkshop 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] Wai-Khuen Cheng, Boon-Yaik Ooi, and Huah-Yong Chan. Resource federation in grid using automated intelligent agent negotiation. Future Generation Computer Systems, Volume 26, Issue 8, October 2010, Pages 1116-1126.

[22] Luis Rodero-Merino, Luis M. Vaquero, Victor Gil, Fermin Galan, Javier Fontan, Ruben S. Montero, Ignacio M. Llorente, From infrastructure delivery to service management in clouds, Future Generation Computer Systems, Volume 26, Issue 8, October 2010, Pages 1226-1240.

[23] Saurabh Kumar Garg, Rajkumar Buyya, Howard Jay Siegel, Time and cost trade-off management for scheduling parallel applications on Utility Grids, Future Generation Computer Systems, Volume 26, Issue 8, October 2010, Pages 1344-1355.

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

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

2010-48: Jörn Altmann, Alireza Abbasi and Junseok Hwang, "Evaluating the Productivity of Researchers and their Communities: The RP-Index and the CP-Index"

2010-49: Jörn Altmann and Zelalem Berhanu Bedane, "A P2P File Sharing Network Topology Formation Algorithm Based on Social Network Information"

2010-50: Tai-Yoo Kim, Seunghyun Kim and Jongsu Lee, "The Gene of an Accelerating Industrial Society: Expansive Reproduction"

2010-51: Almas Heshmati and Sangchoon Lee, "The Relationship between Globalization, Economic Growth and Income Inequality"

2010-52: Flávio Lenz-Cesar and Almas Heshmati, "Agent-based Simulation of Cooperative Innovation"

2010-53: Erkhemchimeg Byambasuren and Almas Heshmati, "Economic Development in Mongolia"

2010-54: Almas Heshmati and Subal C. Kumbhakar, "Technical Change and Total Factor Productivity Growth: The Case of Chinese Provinces"

2010-55: Bory Seng and Almas Heshmati, "Digital Divide and Its Variations Amongst OECD, NIE and ASEAN Countries"

2010-56: Kibae Kim, Jörn Altmann and Junseok Hwang, "The Impact of the Subgroup Structure on the Evolution of Networks: An Economic Model of Network Evolution"

2010-57: Kibae Kim, Jörn Altmann and Junseok Hwang, "Measuring and Analyzing the Openness of the Web2.0 Service Network for Improving the Innovation Capacity of the Web2.0 System through Collective Intelligence"

2010-58: Alireza Abbasi and Jörn Altmann, "A Social Network System for Analyzing Publication Activities of Researchers"

2010-59: Jörn Altmann, Costas Courcoubetis and Marcel Risch, "A Marketplace and its Market Mechanism for Trading Commoditized Computing Resources"

2010-60: Fatmawati Zifa and Jörn Altmann, "A empirically validated framework for limiting free-riding in P2P network through the use of social network"

2010-61: Ashraf Bany Mohammed, Jörn Altmann and Junseok Hwang, "Cloud Computing Value Chains-Understanding Business and Value Creation in the Cloud"

2010-62: Ashraf Bany Mohammed and Jörn Altmann, "A Funding and Governing Model for Achieving Sustainable Growth of Computing e-Infrastructures"

2010-63: Junseok Hwang, Jihyoun Park and Jörn Altmann, "Two Risk-Aware Resource Brokering Strategies in Grid Computing: Broker-driven vs. User-driven Methods"

2010-64: Juthasit Rohitratana and Jörn Altmann, "Agent-Based Simulations of the Software Market under Different Pricing Schemes for Software-as-a-Service and Perpetual Software"

2010-65: Radoslaw R. Okulski and Almas Heshmati, "Time Series Analysis of Global Airline Passengers Transportation Industry"

2010-66: Alireza Abbasi and Jörn Altmann, "On the Correlation between Research Performance and Social Network Analysis Measures Applied to Research Collaboration Networks"

2010-67: Kibae Kim, Jörn Altmann and Junseok Hwang, "An Analysis of the Openness of the Web2.0 Service Network Based on Two Sets of Indices for Measuring the Impact of Service Ownership"

2010-68: Bory Seng, "The Driving Forces Underlying the Growth of Total Factor Productivity in Cambodia, Quantitative and Qualitative Studies"

2010-69: Michael Maurer, Vincent C. Emeakaroha, Ivona Brandic, and Jörn Altmann, "Cost and Benefit of the SLA Mapping Approach for Defining Standardized Goods in Cloud Computing Markets"

2010-70: Khin Swe Latt and Jörn Altmann, "A Cost-Benefit-Based Analytical Model for Finding the Optimal Offering of Software Services"

2010-71: Jung Eun Lee, Younghoon Kim, Yeonbae Kim and Donghyuk Choi, "The Impact of Technology Licensing Payment Mechanisms on Firms' Innovative Performance"

2010-72: Dongook Choi and Yeonbae Kim, "Effects of Piracy and Digital Rights Management on the Online Music Market in Korea"

2011-73: Tai-Yoo Kim, Jihyoun Park, Eungdo Kim and Junseok Hwang, "The Faster-Accelerating Digital Economy"

2011-74: Ivan Breskovic, Michael Maurer, Vincent C. Emeakaroha, Ivona Brandic and Jörn Altmann, "Towards Autonomic Market Management in Cloud Computing Infrastructures"

2011-75: Kiran Rupakhetee and Almas Heshmati, "Rhetorics vs. Realities in Implementation of e-Government Master Plan in Nepal"

2011-76: Alireza Abbasi, Jörn Altmann and Liaquat Hossain, "Identifying the Effects of Co-Authorship Networks on the Performance of Scholars: A Correlation and Regression Analysis of Performance Measures and Social Network Analysis Measures"

2011-77: Ivona Brandic, Michael Maurer, Vincent C. Emeakaroha, Jörn Altmann, "Cost-Benefit Analysis of the SLA Mapping Approach for Defining Standardized Cloud Computing Goods"