2012 ACM/IEEE 13th International Conference on Grid Computing

Automatic SLA Matching and Provider Selection in Grid and Cloud Computing Markets

Christoph Redl, Ivan Breskovic, Ivona Brandic, Schahram Dustdar
Distributed Systems Group, Institute of Information Systems
Vienna University of Technology, Austria
christoph.carl.redl@alumni.tuwien.ac.at, {breskovic, ivona, dustdar}@infosys.tuwien.ac.at

***Abstract*—Cloud computing is a novel computing paradigm that offers data, software, and hardware services in a manner similar to traditional utilities such as water, electricity, and telephony. Usually, in Cloud and Grid computing, contracts between traders are established using Service Level Agreements (SLAs), which include objectives of service usage. However, due to the rapidly growing number of service offerings and the lack of a standard for their specification, manual service selection is a costly task, preventing the successful implementation of ubiquitous computing on demand. In order to counteract these issues, automatic methods for matching SLAs are necessary. In this paper, we introduce a method for finding semantically equal SLA elements from differing SLAs by utilizing several machine learning algorithms. Moreover, we utilize this method to enable automatic selection of optimal service offerings for Cloud and Grid users. Finally, we introduce a framework for automatic SLA management, present a simulation-based evaluation, and demonstrate several significant benefits of our approach for Cloud and Grid users.**

***Index Terms*—Service Level Agreement; Cloud computing; SLA mapping; SLA matching; provider selection; machine learning; autonomic computing; electronic markets**

I. Introduction

Cloud computing is an emerging paradigm where highly scalable computing resources (e.g., infrastructure, platform, and software services) are offered as on-demand services in a pay-as-you-go model [1], [2]. Due to the large variety in services and vast number of service providers, the Cloud market is currently fragmented and inefficient [3], [4]. To counteract this issue, electronic marketplaces for trading Cloud services were introduced with the goal of facilitating trading in the heterogeneous environment.

In order to achieve the goal of ubiquitous computing as a commodity, Cloud marketplaces need to be flexible, dynamic, and have low entry barriers. However, today's Cloud landscape is static and cannot adequately adapt to the active user participation, i.e., dynamic user base and changes in user requirements for services [4], [5], [6]. In particular, due to the large variety of Cloud services and the vast number of market participants, Cloud marketplaces often suffer from low liquidity, i.e., the probability to sell or purchase a service, thus disadvantaging new providers and repelling potential consumers [4]. Moreover, although most of the existing marketplaces use Service Level Agreements (SLAs) to express and negotiate user requirements and offers for services, there exists no general standard for their specification. Therefore, although syntax specification of some requirements may vary among different SLAs, their semantic meaning may be the same from the perspective of market participants. This issue presents an obstacle for automatic SLA matching and partner selection in electronic markets. Instead, users have a high cost for searching through the large number of service offerings and manually comparing syntactically differing SLAs.

The most common solution for these issues in research is creation of SLA ontologies [7], [8], [9], [10], [11]. However, although ontologies may solve the problem for a static set of SLA elements by forcing market participants to specify the semantics of their requirements, they do not solve the problem of dynamic user base and ever-changing requirements for services. In order to address these issues, knowledge about user requirements, their characteristics and differences, as well as their evolution over time is crucial for taking regulating measures in the Cloud market. One approach that deals with this problem is the *SLA mapping technique*. SLA mappings are documents used to bridge the differences between two differing SLAs [12], [13], [3], [14]. Unlike ontologies, SLA mappings are dynamic and allow users to keep their SLAs that may already be used in other processes [3]. However, due to the lack of semantic description of SLA elements, mappings must be manually defined by market participants, thus keeping the high cost of the service selection process.

In this paper, we assess the cost of SLA matching and provider selection with the SLA mapping technique and discuss the methods for reducing this cost in Grid and Cloud marketplaces. In particular, we present an approach for automatic discovery of semantically equal SLA elements and creation of SLA mappings that map the differences in their syntax specification. Furthermore, using the automatic SLA matching algorithms, we allow the autonomic provider selection in Grid and Cloud computing marketplaces.

In order to achieve our goals, we propose storing and managing history data of user requirements as well as of mappings between corresponding requirements in a market knowledge repository. We suggest machine learning and automatic reasoning methods to analyze this data and automatic matching of SLA elements and generation of mappings based on the knowledge about common requirement specifications gained through this process. As the result, our approach facilitates automatic recommendations of trading partners and SLA mappings, thus reducing the cost of joining the market.

1550-5510/12 $26.00 © 2012 IEEE
DOI 10.1109/Grid.2012.18

The main scientific contributions of this paper are: (1) introduction of the algorithms for matching equal SLA elements from differing SLAs; (2) analysis of the methods for automatic provider selection; (3) development of a framework for automatic creation and management of SLA mappings; and (4) a simulation-based evaluation of the approach using an experimental testbed.

The remainder of the paper is organized as follows: Section II describes the related work. Section III discusses the SLA mapping technique. The approach of autonomic SLA matching and provider selection is explained in detail in Section IV. Section V presents the simulation-based evaluation of the approach. Finally, Section VI concludes the paper.

II. Related Work

We classify the related work in four categories: (1) concepts for enhancing SLAs with semantics; (2) approaches for matching various types of entities; (3) approaches for automatic discovery of mappings between various types of entities; and (4) methods for learning from past experiences.

A general problem in the field of SLA matching is the lack of semantic description of SLA elements. To compensate these shortcomings, [15] introduce utilization of semantic Web technologies based on Web Service Semantics (WSDL-S) and Web Ontology Language (OWL) for enhancement of WS-Agreement specifications to achieve autonomic Quality of Service (QoS) and SLA matching. Similar to that, [16] introduce an ontology-based SLA formalization where OWL and Semantic Web Rule Language (SWRL) are chosen to express the ontologies. [17] suggest producing a unified QoS ontology applicable to the main scenarios such as QoS-based Web Services selection, QoS monitoring, and QoS adaptation. However, all of these approaches require market participants to specify the semantics of their requirements based on their knowledge about the target domain, which is a costly procedure. The approach discussed in this paper, on the other hand, tries to minimize the overall cost for market participants by autonomic analysis of semantic correlations between the managed SLAs and the users' feedback to automatic recommendations of SLAs and SLA mappings.

Several works deal with autonomic QoS matching in various distributed environments (e.g., [7], [8], [9], [10], [11]). Similarly to our approach, these works also focus on matching requirements of different SLAs. In contrast to our work, however, they try to facilitate the matching process by introducing ontologies as enhancement to plain SLA documents, thus forcing market participants to specify the semantics of their requirements in such ontologies. In the broader field of computer science, there are many other approaches for automatic matching of various types of entities. Recent works include matching records of databases (e.g., [18], [19], [20], [21], [22]) and ontologies for semantic web (e.g., [23]). These works, however, significantly differ from our context.

Approaches for automatic discovery of mappings between various types of entities have also been discussed in other fields of computer science. [24], for instance, provide an approach for automatic search for mapping rules for XML Schemas. [25] describe an approach for automatic derivation of XSLT transformations between XML documents. However, these approaches either try to match similar entities using traditional similarity algorithms that only operate on a syntactical level, build upon explicitly described semantics of the analyzed entities, or use synonym databases to find semantic relationships between individual words. On the contrary, the approach proposed in this paper focuses on the whole structure of SLAs rather than on individual entities. Moreover, our approach utilizes machine learning methods to learn from past experiences, thus continuously enhancing the knowledge about relationships between semantically equal SLA elements.

To facilitate automatic reasoning about the equality of SLA elements and creation of mappings between them, we propose strategies for learning from already established SLAs as well as from market participants' feedback in response to automatic recommendations. Learning methods for similar research problems have been evaluated in various fields of computer science. The ones most similar to our context deal with matching similar records in databases by utilizing classification methods (e.g., [26], [27], [20], [21], [28]). Most of these works use multiple classifiers allowing them to automatically adapt to the characteristics of record instances. Other learning methods deal with learning from past experiences. Many works use Case-Based Reasoning (CBR) for this purpose. For example, [29] used CBR to determine resource configurations of virtual machines in Cloud environments. However, this and other similar works cannot be directly applied to our research problem since the input features used for classification as well as the models of cases utilized by CBR may be highly different to our approach.

III. The SLA Mapping Approach

In our vision of Cloud markets, Cloud providers and Cloud consumers express their offers and requirements in special forms of SLAs named *SLA templates* before they negotiate and sign legally binding SLAs. SLA templates (as well as SLAs) comprise three principal elements: (1) *SLA metrics*, representing methods for calculation of Quality of Service (QoS) values; (2) *SLA parameters*, representing abstractions of SLA metrics, determined by their basic properties such as a name and a unit for expressing the parameter value; and (3) *Service Level Objectives (SLOs)*, determining contractually agreed objective values for specific SLA parameters. The SLA model can be expressed using any SLA specification scheme, such as WS-Agreement [31] and WSLA [30], with the latter used in our further discussions.

We distinguish two types of SLA templates: *private SLA templates*, which are used to specify requirements and offers of market participants, and *public SLA templates*, which are used to describe products offered for trade on the market. While the former are created by market participants when joining the market, the latter are autonomically generated and managed by the market [14]. The market allows only the trade of the products described by public SLA templates.

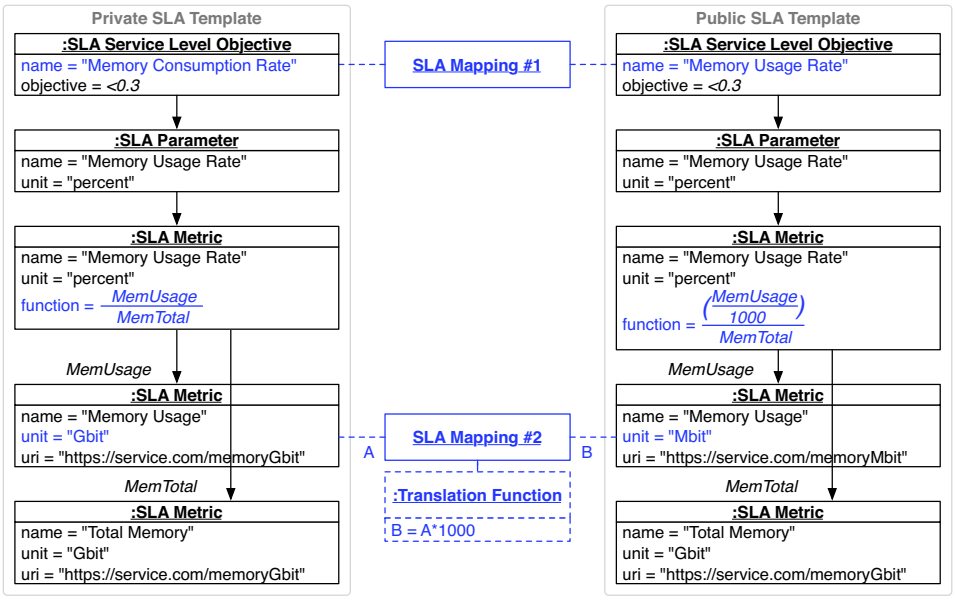


Fig. 1: Case study

When entering the market, users do not search for offerings of other users, but associate their private SLA template with public SLA templates that are closest to their requirements [5], [6]. Therefore, for the successful establishment of the contract between service buyers and sellers, their requirements must be matched with the ones specified in the associated public SLA template. Since there exists no general standard for specifying requirements in SLA templates, their definition may vary among templates. For that reason, *SLA mappings* have been introduced. They are used to bridge the differences between *syntax inequalities* of specific SLA elements that are *semantically equal* in two SLA templates [12], [13]. Semantically equal SLA elements have the same meaning from the perspective of the market participants participating in a trade. On the other hand, SLA elements are equal by syntax if their language specification is identical. Note that in our vision of Cloud markets, syntax equality of SLA elements implies their semantic equality.

Market participants are asked to create SLA mappings between their own private SLA template and the corresponding public SLA template when entering the market. Mapping translations can range from very simple, such as incompatible names of SLA elements (e.g., different names used for the same SLA parameter) to complex translations, such as methods for calculating parameter values with different metrics (e.g., different units used to express the same parameter value).

To explain the most important concepts involved in the SLA mapping approach, in Fig. 1 we present a case study comprising two semantically equal SLA templates containing an SLO to determine an objective value of the *memory usage rate* of a Cloud service. Although semantically equal, these SLOs differ in their syntax. Namely, they differ in the SLO name ("Memory Consumption Rate" in the public SLA template vs. "Memory Usage Rate" in the private template). The measurements for which the SLOs define objective values are specified by SLA parameters in each of the SLA templates. SLA metrics referenced in the two SLA parameters define a function for calculating memory usage rate out of two SLA metrics: *memory usage* and *total memory*. However, the values of the former are expressed in a different unit in the two SLA templates ("Gbit" from the private SLA template vs. "Mbit" from the public template). Correspondingly, the SLA metric function combining the parameter metrics differs in the SLA templates, which is a direct consequence of the differing measurement units.

To bridge the difference between two SLA templates depicted in Fig. 1, two SLA mappings should be created. First, a simple SLA mapping should map the difference between two SLO names. Secondly, a more complex SLA mapping should be defined between two differing SLA metrics to state a translation function for mapping the differing metric values. Note that an SLA mapping between syntactically different SLO functions is not necessary, since a mapping between differing metric units indirectly maps this inequality.

When specifying SLA mappings between elements of two SLA templates, market participants need to assess the equality of both elements from their own perspective. Contradictions caused by different opinions on the semantic equivalence of SLA elements play a subordinate role in autonomic market management since the market may be regulated according to their meaning for the majority of market participants. Only during the negation of final contracts such contradictions may be detected and solved by mutual agreements between two trading parties.

IV. Towards Autonomic Provider Selection

In order to enable autonomic SLA management in Cloud markets, we must extract the knowledge from specifications of user requirement for services, which is only possible if SLAs and SLA mappings are managed by the Cloud market platform. Specifications of requirements and their variations over different SLA templates can then be analyzed and generalized knowledge can be learned and reused for generation of SLA mappings between unknown SLA templates. To realize this goal, a Cloud market should implement additional knowledge mechanisms dealing with several important tasks such as storing and managing data about the history of SLA templates and mappings, learning the semantics necessary for finding equivalent SLA elements and generation of adequate SLA mappings, and automatic SLA element matching and generation of SLA mappings between SLA elements.

To facilitate continuous learning, we propose autonomic knowledge management in Cloud markets based on a control loop similar to the traditional MAPE cycle containing the following phases: (1) continuous **M**onitoring of the learning progress by testing whether available knowledge yields correct recommendation results; (2) **A**nalyzing available knowledge and deciding whether new knowledge should be added or outdated knowledge forgotten; (3) **P**lanning and scheduling training and revision phases; and (4) **E**xecuting the actual training. Implementation of such a control loop facilitates continuous fitting of the learned knowledge to any potential changes in formalization of requirements due to the evolution of market participants' requirements over time.

The market knowledge is finally used for two main purposes: **automatic SLA matching**, i.e., recognition of equal elements of two SLAs and autonomic creation of SLA mappings between them, and **automatic provider selection**, i.e., recommendation of those public SLA templates to the market participants that are the closest to the requirements specified by their private SLA templates. In the following, we explain the processes of achieving these goals.

A. Automatic Matching of SLA Elements

The process of matching semantically equal SLA elements from differing SLA templates is important for enabling automatic provider selection in Cloud markets. This process is executed in three steps. First, for each pair of two SLA elements, the probability for their equivalence (i.e., their similarity) is computed. Then, based on the computed probabilities, the equivalence of the elements is determined. Finally, if two SLA elements are semantically equal, the SLA mappings are automatically created to bridge the possible differences. When measuring the similarity of a pair of SLA elements, we consider their two properties: element definition, stating basic properties such as name and measurement unit, and element metric, describing a method for measuring the element value.

1) Similarity of element definitions: Properties of element definitions are represented by string values. For calculating the similarity of these properties, possible characteristics of their specifications have to be considered. The most common characteristics we have identified are variations in the order of single characters or character blocks (e.g., "Memory Consumption" vs. "Consumption Memory"), additional characters or words (e.g., "Consumption Memory" vs. "Consumption of Memory"), and usage of abbreviations or synonyms (e.g., "Memory Consumption" vs. "Memory Usage").

Calculation of the similarity between definition properties can be realized by utilizing string similarity metrics. A large number of these metrics has been discussed in existing research [32], [20], [33]. They may be roughly separated into two groups: character-based and vector-space based methods [21]. While the former rely on character edit operations (e.g. insertions, deletions, and substitutions), the latter "transform strings into vector representations on which similarity computations are conducted" [21]. In our evaluation, we used the character-based string similarity metric *Levenshtein distance* for this purpose due to its good performance with short strings that only contain a small number of variations in their syntax. Levenshtein distance is defined as "the minimum number of insertions, deletions or substitutions necessary to transform one string into another" [21]. In our model, the similarity of two strings is computed as the difference between the length of the longer of the two strings and the number of character modifications needed to convert one string to another (i.e., the Levenshtein distance) divided by the same maximum distance:

$$p_{equal}(S_1, S_2) = \frac{max(|S_1|, |S_2|) - d_L(S_1, S_2)}{max(|S_1|, |S_2|)} \quad (1)$$

where S_1 and S_2 are the two strings, $|S_1|$ and $|S_2|$ their lengths, and $d_L(S_1, S_2)$ their Levenshtein distance.

Although string similarity metrics promise good results for identification of small variations in definition properties, they are not able to recognize semantically equal properties that differ significantly, as it is the case of synonyms or abbreviations. These differences may only be detected by having concrete knowledge of possible synonyms or abbreviations. Hence, besides string similarity, our approach utilizes Case-Based Reasoning (CBR) methods for learning semantically equal definition properties from established SLA mappings. In particular, semantic equivalence of individual definition properties is determined in four phases: (1) retrieval of the most similar case to the new case by calculating the similarity between SLA element definition properties used in both cases and selecting the case with the highest similarity; (2) reuse of the knowledge of the similar case to solve the new problem; (3) revision of the proposed solution by analyzing users' feedback to the recommendation of the matching SLA elements; and (4) retainment of the parts of user experience likely to be useful for future problem solving (i.e., storing users' feedback as part of the utility of a case). If reoccurring patterns are detected by CBR, the similarity of two element definition properties is calculated based on the similarity between the old case and the new case, which is again computed with the Levenshtein similarity metric.

Fig. 2 illustrates how an existing CBR case could be reused for solving a similar new case (the symbol $\cong^S$ is used to de-

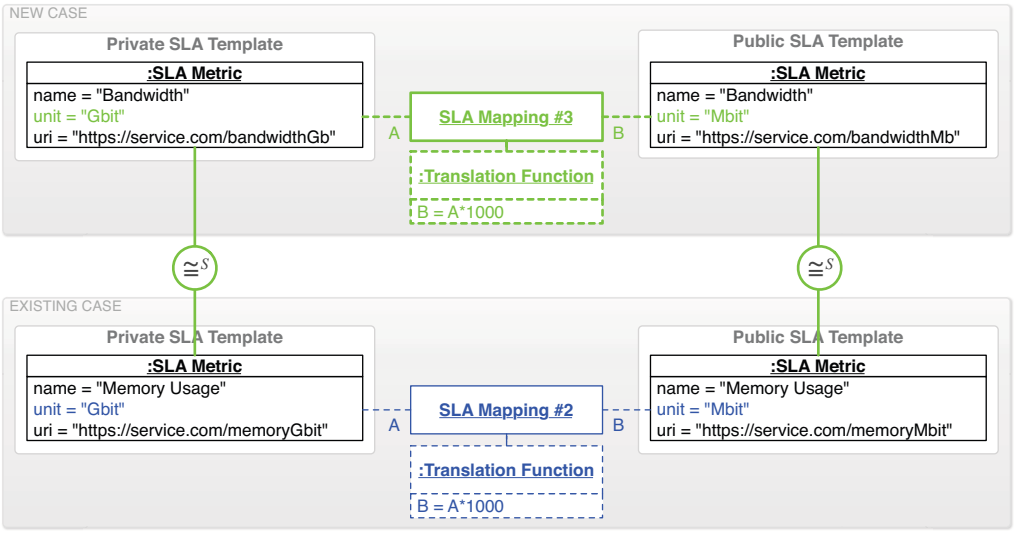


Fig. 2: Reusing a CBR case

note *semantical equivalence* of two SLA elements): While two cases describe SLA metrics for significantly different purposes (namely for measuring *memory usage* and *bandwidth usage* of a Cloud service), their similarity lies in the unit conversion from "Gbit" to "Mbit" and vice versa. Therefore, an SLA mapping with an associated translation function specified in the existing case can be reused in the new case to capture the unit conversion.

2) Similarity of element metrics: Element metrics describe the methods for measuring SLA element values. They can either directly specify a measurement source by stating its URI or indirectly reference and/or aggregate other SLA elements for retrieving their measurements. Composite aggregation of measurements is specified as a function of the metrics.

On the one hand, similarity of element metrics stating measurement sources is done analogously to element definition properties, since URIs may abstractly be seen as strings. Similarity of properties that reference individual SLA elements should only ensure equality of the referenced elements. On the other hand, computing similarity of the metric properties defined by functions, i.e., combinations of several QoS measurements, involves determination of the equality of the functions itself. Namely, due to the possible variations and transformations inside the functions (e.g., introduction of substitutions, change of the variable orders, utilization of different units, etc.), this process must involve several steps: (1) checking if all referenced SLA elements used in both functions are semantically equal; (2) ensuring that the both functions use the same variable assignment (i.e., two semantically equal SLA elements are assigned to the same variable); and (3) proving the logical equivalence of both formulas using an algebra solver. If one of these steps fails, functions are considered as non-equivalent. Otherwise, the elements are considered equivalent. In our framework, logical equivalence of metric functions is checked by Symia [34], a Java-based algebra solver library that facilitates parsing algebraic expressions, automatically detecting variables in expressions, and solving not only numeric, but also symbolic expressions (i.e., expressions containing abstract variables), which is of great importance in our context.

While the similarity of element definition properties and element metrics stating measurement sources is expressed as a continuous probability, the one of element metrics defined by functions can only be expressed by a binary classification since two functions can either be logically equal or not. This is expressed as a discreet value, i.e., 0 (non-equivalence) or 1 (equivalence) of a pair of functions.

3) Final decision on element similarity: To make a final decision on semantic equality of SLA elements, the similarities of their individual properties have to be combined to an overall value representing the probability of element equality. For this purpose, we utilize machine learning methods. As input features for classification, we use the equality probabilities of the individual element properties that are precalculated using the string similarity measurements, CBR-enhanced similarity detection, and algebra solver. Classification is done separately for each SLA element in the hierarchical order (i.e., from the SLA metric level up to the SLO level). In our evaluation, we utilize the Support Vector Machines (SVM) method for implementing the classification strategies for the individual SLA elements.

SVM is a concept in statistics and computer science for a set of related supervised learning methods that analyze data and recognize patterns in such way that they take a set of input data and predict, for each given input, which of two possible classes forms the input. In our scenario, SVM uses a training phase in which it builds an internal model for prediction of the final binary classifier for the semantic equality of SLA elements.

The internal SVM model uses the equality probabilities of the individual SLA element properties as the input features and produces a binary output with *semantically equal* and *semantically different* as possibly classes for the given input.

4) Automatic generation of SLA mappings: Once the equal SLA elements from two different SLA templates have been matched, the SLA mappings must be created to bridge the discovered differences. SLA mappings are automatically created and recommended to the user.

Generation of SLA mappings for incompatible element definition properties is straightforward, since they only map already identified differences in string values. On the other hand, generation of more complex mappings between incompatible element metrics involves creation of suitable translation functions, which can later be used to translate measurements defined by one SLA metric into the other and vice versa. In the simplest case, two semantically equal SLA metrics represent the same measurement defined in the same unit, but differ in their structural specification. In this situation, it is enough to create an SLA mapping referencing the SLA metrics and defining a translation function for converting a measurement from one SLA template into another. In more complex cases, two semantically equal SLA metrics additionally differ in their units. In this case, generation of the translation function involves knowledge about how the units of both metrics are mathematically related. This knowledge can be obtained from the previously used SLA mappings by utilizing CBR methods. If an equivalent unit translation has been used in a similar case, knowledge about specification of the translation function can be transferred to the new case. If a similar case does not exist, the user must manually define the mapping.

After receiving the recommended SLA mappings, a user submits his feedback to the recommendation component stating the correctness of the recommended data. In case of a negative feedback, a user can state whether the SLA mapping was incorrect or an SLA element was wrongly matched. This data is then used for the learning process of the CBR and SVM methods. Note that dealing with the contradictory user feedback is out of the scope of this paper.

B. Automatic Provider Selection

Automatic provider selection assumes assessing semantic equality of two SLA templates and requires matching of equivalent SLA elements from those templates. Given a user's private SLA template and a set of public SLA templates, the matching process starts by iterating through all public SLA templates and checking the similarity of all SLA elements contained by the user's private SLA template and the currently examined public SLA template. For each pair of SLA templates, the overall equivalence probability is computed as the relative number of matched SLA elements in the total number of elements of both SLA templates. Finally, the public SLA template with the highest equivalence probability is chosen as the optimal offering, but only if its equivalence probability exceeds a predefined threshold, which is in our simulation set to 70%. Otherwise, a user is notified that the appropriate service offering currently does not exist in the market.

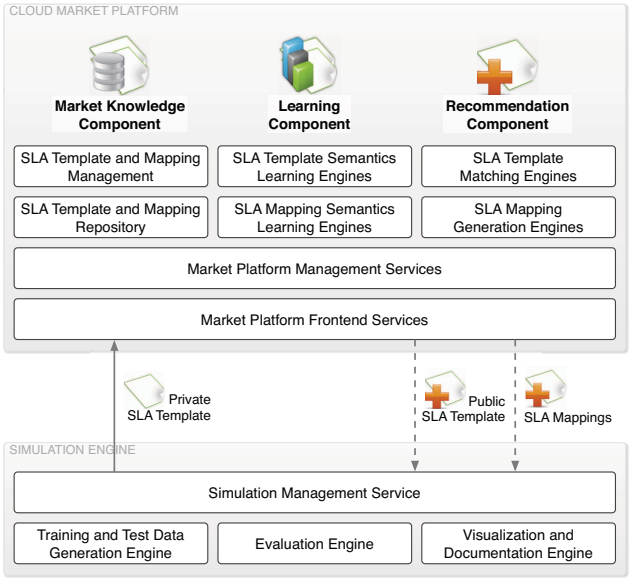


Fig. 3: Simulation environment

V. Evaluation

A. Simulation Environment

For the simulation purposes, we designed a prototype integrating our framework for autonomic generation and management of SLA mappings into a simulated market environment. Fig. 3 depicts our framework comprising two core components: a *cloud market platform* and a *simulation engine*.

The **Cloud market platform** represents the basic infrastructure for autonomic management of the Cloud market and integrates the actual implementation of our framework for autonomic creation and management of SLA mappings. In particular, it comprises: (1) frontend services, which form the basic interface for submission of users' requirements and offers for services to the market; (2) market platform management services responsible for management of supply and demand in the market (including pricing and allocation mechanisms); (3) a market knowledge component responsible for storing and managing public and private SLA templates as well as SLA mappings; (4) a learning component responsible for capturing characteristics of the data and learning to automatically recognize complex patterns and make intelligent decisions based on the data concerning the semantics of SLA elements and SLA templates; and (5) a recommendation component responsible for making final decisions on SLA matching and provider selection and recommendation of equal SLA elements, SLA mappings, and SLA templates.

The **simulation engine** is responsible for automating the simulation process. In particular, it comprises: (1) a training and test data generation engine, implementing a configurable interface for semi-automatically generation of training and test

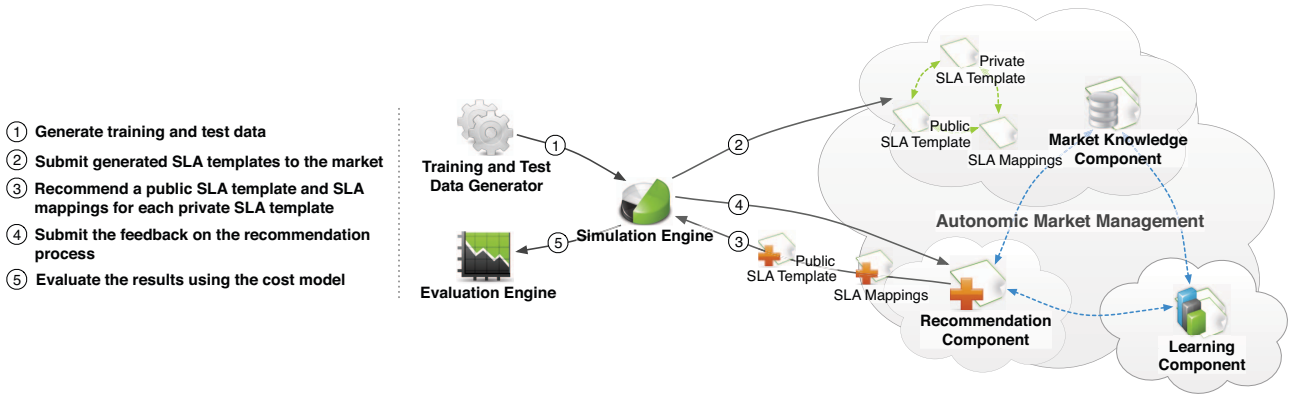


Fig. 4: Simulation testbed

data sets in compliance with predefined generation policies; (2) an evaluation engine, providing methods for assessing the results of the simulation by using an adequate cost model; and (3) a visualization and documentation engine, facilitating graphical visualization of the evaluation results as well as documentation for later analysis.

B. Simulation Process

The simulation is conducted in three main steps (Fig. 4): generation of demand and supply in the market, recommendation of the optimal offerings on the market to the service buyers (including the recommendation of SLA mappings between the differing SLA templates), and the evaluation of the recommendation results based on the simulated user feedback.

The simulation process is conducted as follows. In the first step, the training and the test data is randomly created in a semi-automatically fashion so that it meets requirements specified in data generation policies. These policies are forwarded to the data generation engine to ensure automatic generation of training and test datasets that meet specific properties of real datasets such as a clear separation between semantically different and semantically equal SLA templates or data distributions within the datasets. Moreover, the data generation engine contains a set of rules for creating instances of SLA elements based on characteristics of real-world examples as well as rules for modeling syntactic differences between semantically equal SLA elements that can be found in real-world scenarios. The result of this process is the collection of triples containing a public SLA template from the set of all public SLA templates, a private SLA template from the set of all private SLA templates that is semantically equal to the associated public SLA template, and the pre-calculated SLA mappings between the two templates.

The second step of the simulation process involves submission of the previously created SLA templates and SLA mappings to the market. Namely, for each triple, the simulation engine submits the private SLA template to the market platform prototype. The prototype's recommendation component then automatically analyzes the submitted private SLA template and tries to match it with an existing public SLA template stored in the market repository contained by the market knowledge component. As already described in Section IV, this process finds the best matching public SLA template and returns it back to the simulation engine. Together with the public SLA template, the recommendation returns the SLA mappings between the discovered equivalent elements of the two SLA templates, as described in Section IV.

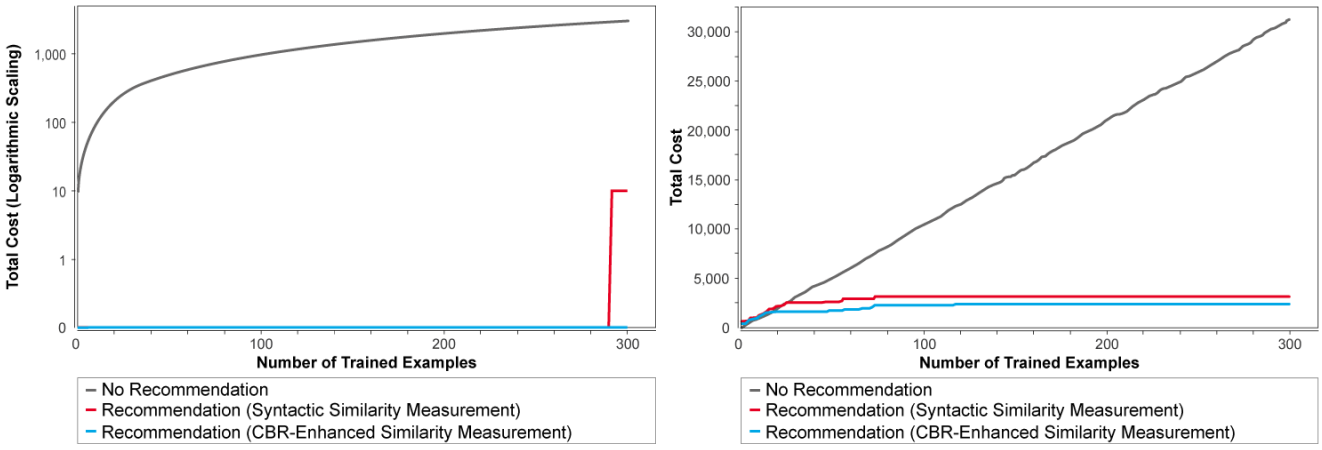
For the evaluation of the recommendation algorithm, the simulation engine analyzes correctness of the recommendation by comparing it to the precalculated public SLA template and SLA mappings. If the recommended public SLA template is equal to the precalculated public template, the recommendation has been correct; otherwise an incorrect recommendation has been detected. Analogically, the correctness of the recommended SLA mappings is checked. In case of an incorrect recommendation, the framework reports the mistake to the market platform's recommendation component, thus simulating the negative user feedback. This feedback is then automatically forwarded to the learning component.

C. Simulation Setup

For the evaluation of our approach, we define two simulation setups: for evaluating the process of provider selection (i.e., public SLA template recommendation) and for evaluating the methods for SLA mapping recommendation. Table I summarizes the simulation settings.

Parameter	Value
Number of public SLA templates	
- Public SLA template recommendation	100
- SLA mapping recommendation	3
Number of private SLA templates per public SLA template	
- Public SLA template recommendation	3
- SLA mapping recommendation	100
Number of SLOs per SLA template	5-7
Number of SLA parameters per SLA template	5-7
Number of SLA metrics per SLA template	7-10
Maximal hierarchy level for nested SLA metrics	5

TABLE I: Simulation settings



(a) SLA template recommendation (total cost development)

(b) SLA mappings recommendation (total cost development)

Fig. 5: Evaluation results

To evaluate the provider recommendation process, we generate a set of 100 public SLA templates and 3 semantically equal private SLA templates per one public SLA template. SLA templates contain between 5 and 7 SLOs as well between 5 and 7 SLA parameters. The number of SLA metrics per SLA template is in range of 7 to 10. Note that the number of SLA metrics is larger than the number of SLOs in oder to allow nesting of SLA metrics.

For evaluation of the SLA mapping recommendation process, the focus is on the syntactic differences between elements of private SLA templates and semantically equal elements of corresponding public SLA templates. Therefore, only a small number of public SLA templates is necessary, but with a large variety of corresponding private SLA templates. For this purpose, we generate 3 public SLA templates with 100 semantically equal private SLA templates for each of the public templates. The settings for the structure of individual SLA templates is equivalent to the setup for public SLA template recommendation.

D. Formalization of the Cost Model

When initially joining a Cloud market, users need to execute several manual steps to find an adequate Cloud service and establish a contract for its usage. Manual execution of these steps involves a high effort for market participants. In this section, we build a simple cost model to quantitatively assess this effort. This model will then be used in our further discussions on the benefits of the approach presented in this paper.

When joining the market, market participants need to search and select an adequate provider service offering that best matches their own requirements. Since in our vision Cloud services traded on the market are described by public SLA templates, market participants must associate their private SLA template with the best-matching public SLA template. In our cost model, we mark the cost of manually searching for the best-matching public SLA template with the variable a. This cost is common for the traditional Cloud markets in which SLA templates are not automatically recommended to the participants, but also when a template has been recommended that is not the best-matching for the user. On the other hand, a market participant has no cost if he must not manually search for an appropriate public SLA template, which is in case when a public SLA template is automatically and correctly recommended to the user.

After selecting an adequate provider, market participants need to specify SLA mappings between their private SLA template and the chosen public SLA template to bridge any possible syntactical differences. In our cost model, we mark the cost of manually creating a new SLA mapping with the variable c. On the other hand, a user has no cost if he must not create an SLA mapping, which is in the case when a mapping is not necessary or the correct SLA mapping was automatically provided to the user. Since the recommended SLA mappings may also be incorrect, a user may have an additional cost d to delete the recommendation, i.e., to send a negative feedback. Note that in some cases a user may have to both delete the recommendation and create a new SLA mapping, which results in the total cost of $c + d$.

The presented cost model is in our simulations used to compare the benefits of automatically creating SLA templates and SLA mappings in Cloud markets. The values of the variables a, c and d used in our simulations are depicted in Table II. Note that, considering the presented model, the absolute values of the costs are less important than their relative difference.

Cost	Cost description	Value
a	Manual association of a public SLA template	10
c	Manual creation of a new SLA mapping	10
d	Identification of an incorrectly recommended mapping	10

TABLE II: The cost model

E. Evaluation Results

In this section, we evaluate our approach by measuring the cost for market participants of adapting incorrect recommendations of public SLA templates and SLA mappings and compare it to the situation in which they have to search for optimal services and create SLA mappings manually. Furthermore, we mutually compare two recommendation strategies based on different learning methods: classification with input features based only on string similarity metrics and classification with input features based on string similarity metrics as well as those based on CBR knowledge.

Fig. 5a illustrates the evaluation results for automatic provider selection, i.e., recommendation of public SLA templates. It depicts the total cost in logarithmic scale of manual selection of public SLA templates achieved by different numbers of trained examples. As it can be seen, the usage of learning methods for matching SLA elements and for automatically associating and recommending public SLA templates significantly reduces the cost of manual association of public SLA templates by market participants, even when using less advanced learning methods, such as the classification method with input features based only on string similarity metrics. However, when comparing this "simple" classification method to the one with additional input features based on CBR knowledge, we can notice only a slight improvement in the cost reduction when compared to the latter case. This is due to the very small number of equivalent SLA elements that could not be matched by less advanced learning methods in comparison to the number of elements that could be matched by such methods, resulting in only a minor influence on the actual value of the overall equivalence probability between two SLA templates. Hence, even less advanced learning methods are able to show good results for automatic provider selection since not every single semantically equal pair of SLA elements between a public and a private SLA templates must be identified correctly, but even an approximation of the average similarity probability over all SLA elements of both templates is able to give enough information for making correct decisions about their semantical equivalence in most cases.

Fig. 5b illustrates the evaluation results for SLA mapping recommendation. It depicts the total cost of manual creation of SLA mappings achieved by different numbers of trained examples. The simulation starts with a high number of incorrect recommendations during the first 5 to 10 recommendation iterations. This is due to the initialization of SVM with random weights at the beginning of the simulation process and adjusting its weight function automatically over the following training iterations. After the SVM reaches a good approximation for the weight function, its predictions are relatively stable. The evaluation result shows that the usage of learning methods for matching SLA elements and for automatic generation and recommendation of SLA mappings significantly reduces the cost of manual creation of SLA mappings by market participants. When comparing the classification method with the input features based only on string similarity metrics to the one with additional input features based on CBR knowledge, we see that the latter can even reach a significantly lower cost than the former. This is due to the fact that CBR is able to detect complex difference patterns such as synonyms or abbreviations and therefore is able to create correct SLA mappings in such cases, while string similarity metrics are only able to detect small differences in a small number of characters and consider all other cases as semantically different. In our simulation we could show that the reductions in cost reached through the use of CBR are up to 30 percent after 20 iterations in comparison to the overall cost incurred when applying only string similarity metrics. However the concrete difference in cost between both methods ultimately depends on the degree of reoccurring patterns found in real-world scenarios, which may vary in different application domains.

Summarizing, as shown in the evaluation results, our approach for autonomic management of SLA mappings is able to significantly reduce market participants' cost of manual provider selection and manual creation of SLA mappings. The scope of reductions in these costs between the initial situation without recommendation and the approach proposed in this paper depends on the used SLA model, the assumptions made on the characteristics of SLA element specifications, and on the learning and reasoning strategies used. The evaluation has shown that even less advanced learning strategies are able to facilitate good results in automatic provider selection, while automatic creation of SLA mappings requires more sophisticated learning techniques since any inaccurate reasoning directly leads to a wrong recommendation and thus incurs unwanted cost for market participants. For manual provider selection, our approach was able to reduce market participants' cost by nearly 100 percent for all examples in comparison to its original amount without automatic recommendation. In other words, the automatically recommended provider is in almost all cases the correct one, i.e., the same provider that market participants would have chosen by manual selection. For automatic creation of SLA mappings, after a certain training period, our approach was able to reduce the market participants' cost by nearly 100 percent for examples that reused already trained knowledge in comparison to its original amount without automatic recommendation. As it could be seen in the evaluation results, the learning strategy based on CBR needs extensive training to be able to turn its advantages to account, but outperforms the less advanced learning strategy that we have tested in our evaluation since it is able to identify complex patterns and changes in SLA element and SLA mapping specifications.

VI. Conclusion and Future Work

In this paper, we demonstrated an approach for automatic SLA matching and provider selection in Grid and Cloud computing markets with the goal of reducing the cost for manual creation of SLA mappings and search for optimal services for market participants. Using our simulation-based evaluation, we showed the significant benefits of our approach.

In our future work, we will consider applying different similarity metrics and use machine learning methods to automatically determine the influence of each individual metric to the overall matching result and to automatically adapt the set of metrics to specific characteristics of individual properties of SLA elements. In addition, information about characteristics of semantically equal SLA elements will be clustered according to the application context, in order to consider differences in specifications of requirements in distinct application domains. Finally, the obtained knowledge will be autonomically revised and outdated information may be withdrawn, in order to ensure better compliance with the actual evolution of market participants requirements. In addition to the proposed learning strategies, we will analyze the usage of already existing external data sources for detection of semantically equal properties, such as synonym, spelling, and unit conversion dictionaries.

REFERENCES

[1] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica *et al.*, "Above the Clouds: A Berkeley View of Cloud Computing," UCB/EECS-2009-28, EECS Department, University of California, Berkeley, Tech. Rep., 2009.

[2] R. Buyya, C. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality for Delivering Computing as the 5th Utility," *Future Generation Computer Systems*, vol. 25, no. 6, pp. 599–616, 2009.

[3] M. Risch, I. Brandic, and J. Altmann, "Using SLA Mapping to Increase Market Liquidity," in *Service-Oriented Computing ICSOC/ServiceWave 2009 Workshops*, ser. Lecture Notes in Computer Science, vol. 6275. Springer, 2010, pp. 238–247.

[4] I. Breskovic, C. Haas, S. Caton, and I. Brandic, "Towards Self-Awareness in Cloud Markets: A Monitoring Methodology," in *9th IEEE International Conference on Dependable, Autonomic and Secure Computing (DASC 2011)*, 2011.

[5] I. Breskovic, M. Maurer, V. Emeakaroha, I. Brandic, and J. Altmann, "Towards Autonomic Market Management in Cloud Computing Infrastructures," in *International Conference on Cloud Computing and Services Science (CLOSER 2011)*, 2011.

[6] I. Breskovic, M. Maurer, V. Emeakaroha, I. Brandic, and S. Dustdar, "Cost-Efficient Utilization of Public SLA Templates in Autonomic Cloud Markets," in *4th IEEE International Conference on Utility and Cloud Computing (UCC 2011)*, 2011.

[7] K. Kritikos and D. Plexousakis, "Requirements for QoS-Based Web Service Description and Discovery," *Services Computing, IEEE Transactions on*, vol. 2, no. 4, pp. 320–337, 2009.

[8] V. Tran and H. Tsuji, "A Survey and Analysis on Semantics in QoS for Web Services," in *Advanced Information Networking and Applications, 2009. AINA'09. International Conference on*. IEEE, 2009, pp. 379–385.

[9] N. Chudasma and S. Chaudhary, "Service Composition Using Service Selection with WS-Agreement," in *Proceedings of the 2nd Bangalore Annual Compute Conference*. ACM, 2009, p. 21.

[10] H. Muñoz, I. Kotsiopoulos, A. Micsik, B. Koller, and J. Mora, "Flexible SLA Negotiation Using Semantic Annotations," in *Service-Oriented Computing. ICSOC/ServiceWave 2009 Workshops*. Springer, 2010, pp. 165–175.

[11] K. Fakhfakh, S. Tazi, K. Drira, T. Chaari, and M. Jmaiel, "Implementing and Testing a Semantic-Driven Approach Towards a Better Comprehension Between Service Consumers and Providers," in *Advanced Information Networking and Applications Workshops (WAINA), 2010 IEEE 24th International Conference on*. IEEE, 2010, pp. 183–188.

[12] I. Brandic, D. Music, S. Dustdar, S. Venugopal, and R. Buyya, "Advanced QoS Methods for Grid Workflows based on Meta-Negotiations and SLA-Mappings," in *Workflows in Support of Large-Scale Science, 2008. WORKS 2008. Third Workshop on*. IEEE, 2008, pp. 1–10.

[13] I. Brandic, D. Music, and S. Dustdar, "Service Mediation and Negotiation Bootstrapping as First Achievements Towards Self-adaptable Cloud Services," in *Grids and Service-Oriented Architectures for Service Level Agreements*, P. Wieder, R. Yahyapour, and W. Ziegler, Eds. Springer, 2010, pp. 119–132.

[14] M. Maurer, V. Emeakaroha, I. Brandic, and J. Altmann, "Cost-Benefit Analysis of an SLA Mapping Approach for Defining Standardized Cloud Computing Goods," *Future Generation Computer Systems*, 2011.

[15] N. Oldham, K. Verma, A. Sheth, and F. Hakimpour, "Semantic WS-Agreement Partner Selection," in *Proceedings of the 15th international conference on World Wide Web*. ACM, 2006, pp. 697–706.

[16] L. Green, "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*. ACM, 2006, pp. 185–194.

[17] G. Dobson and A. Sanchez-Macian, "Towards Unified QoS/SLA Ontologies," in *Services Computing Workshops, 2006. SCW'06. IEEE*. IEEE, 2006, pp. 169–174.

[18] F. Giunchiglia and P. Shvaiko, "Semantic Matching," *The Knowledge Engineering Review*, vol. 18, no. 03, pp. 265–280, 2003.

[19] F. Giunchiglia, M. Yatskevich, and P. Shvaiko, "Semantic Matching: Algorithms and Implementation," in *Journal on Data Semantics IX*. Springer, 2007, pp. 1–38.

[20] M. Bilenko, R. Mooney, W. Cohen, P. Ravikumar, and S. Fienberg, "Adaptive Name Matching in Information Integration," *Intelligent Systems, IEEE*, vol. 18, no. 5, pp. 16–23, 2003.

[21] M. Bilenko and R. Mooney, "Adaptive Duplicate Detection Using Learnable String Similarity Measures," in *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2003, pp. 39–48.

[22] A. Elmagarmid, P. Ipeirotis, and V. Verykios, "Duplicate Record Detection: A Survey," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 19, no. 1, pp. 1–16, 2007.

[23] A. Doan, J. Madhavan, R. Dhamankar, P. Domingos, and A. Halevy, "Learning to Match Ontologies on the Semantic Web," *The VLDB Journal*, vol. 12, no. 4, pp. 303–319, 2003.

[24] S. Boßung, J. Grundy, and J. Hosking, "Semi-independent Discovery of Mapping Rules to Match XML Schemas," *Department of Computer Science, The University of Auckland*, p. 71pp, 2003.

[25] A. Boukottaya, C. Vanoirbeek, F. Paganelli, and O. Khaled, "Automating XML Documents Transformations: A Conceptual Modelling Based Approach," in *Proceedings of the First Asian-Pacific Conference on Conceptual Modelling*, vol. 31. Australian Computer Society, 2004, pp. 81–90.

[26] H. Köpcke, A. Thor, and E. Rahm, "Evaluation of Entity Resolution Approaches on Real-World Match Problems," *Proceedings of the VLDB Endowment*, vol. 3, no. 1-2, pp. 484–493, 2010.

[27] P. Christen, "Automatic Record Linkage Using Seeded Nearest Neighbour and Support Vector Machine Classification," in *Proceeding of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2008, pp. 151–159.

[28] H. Zhao and S. Ram, "Entity Identification for Heterogeneous Database Integration – A Multiple Classifier System Approach and Empirical Evaluation," *Information Systems*, vol. 30, no. 2, pp. 119–132, 2005.

[29] M. Maurer, I. Brandic, and R. Sakellariou, "Simulating Autonomic SLA Enactment in Clouds Using Case Based Reasoning," in *ServiceWave 2010*, ser. Lecture Notes in Computer Science, vol. 6481. Springer, 2010, pp. 25–36.

[30] H. Ludwig, A. Keller, A. Dan, R. King, and R. Franck, "Web Service Level Agreement (WSLA) Language Specification," IBM Corporation, Tech. Rep., 2003, http://www.research.ibm.com/wsla/WSLASpecV1-20030128.pdf.

[31] A. Andrieux, K. Czajkowski, A. Dan, K. Keahey, H. Ludwig, T. Nakata, J. Pruyne, J. Rofrano, S. Tuecke, and M. Xu, "Web Services Agreement Specification (WS-Agreement)," *Forum American Bar Association*, vol. 192, no. GFD.107, pp. 1–81, 2007.

[32] D. Lin, "An Information-Theoretic Definition of Similarity," in *Proceedings of the 15th International Conference on Machine Learning*, vol. 1. San Francisco, 1998, pp. 296–304.

[33] W. Cohen, P. Ravikumar, and S. Fienberg, "A Comparison of String Metrics for Matching Names and Records," in *KDD Workshop on Data Cleaning and Object Consolidation*, vol. 3. Citeseer, 2003, pp. 73–78.

[34] "Symja - A Java Computer Algebra System," http://code.google.com/p/symja/, accessed: March 2012.