Multi-Objective Service Similarity Metrics for more Effective Service Engineering Methods

Dionysis Athanasopoulos
Electronics, Information & Bioengineering Department
Politecnico di Milano, Italy
Email: dionysiscsuoi@gmail.com

Apostolos V. Zarras
Computer Science & Engineering Department
University of Ioannina, Greece
Email: zarras@cs.uoi.gr

***Abstract*—The usage of single-objective similarity functions in engineering tasks of service-oriented software may reduce their effectiveness, since a single similarity value can be misleading. A single value cannot be clearly interpreted, since it hides the values of its individual objectives. The state-of-the-art approaches, which propose service similarity functions, rely on single-objective functions exclusively. Going to a completely different direction, we propose the usage of multi-objective functions for calculating service similarity. We formally define such a function, and we provide preliminary results, which show that the effectiveness of a service-engineering task (esp., service organization) can be improved by using multi-objective functions.**

Keywords*—*Service-oriented engineering, service interface, service similarity function, multi-objective function.

I. Introduction

Service-oriented software follows the Service-Oriented Architecture (SOA) style, in which systems are designed as a composition of existing and reusable software functionalities. SOA style has been emerged as a promising solution to the rapid and low-cost development of large enterprise-scale software systems. The existing software functionalities used by SOA software are developed in-house (single-organization development), or by third parties (multi-organization development). Third-party functionalities are exposed by their providers (e.g., *Amazon*[1]) as services. Available services are usually accessible through the Web infrastructure by using the Web-service technology [1].

The functional characteristics of services are generally specified in terms of multiple facets [2] (e.g., interface, semantics). Among all different types of documents that specify the functional characteristics of a service, the only document that is always publicly available is the specification of its programmable interface. Thus, since the provision of all these documents is not the rule, we focus in this paper on the specification of service interface. As typically assumed by the state-of-the-art approaches (e.g., [3]), we also assume that different parts of service interface specify the functional characteristics of a service in terms of different facets. Details about which parts of service interface are related to different service description facets are given in Section III.

Motivation. A core part of the engineering of service-oriented software is the discovery, organization, and the selection of candidate services that meet the functional requirements of software. These services are used either for composing and releasing the final version of software, or for substituting currently used services in order to maintain it. The usage of effective service similarity metrics plays crucial role in the aforementioned tasks. To calculate the similarity of individual service facets (e.g., interface vs. semantics), different metrics (hereafter called *objectives*) are required, due to their different nature. To calculate the overall service similarity with respect to all objectives, the typical option in the literature is the usage of aggregation functions [4], integrating multiple objectives in only one.

However, the usage of a single-objective similarity function may negatively affect the outcome of the aforementioned tasks, by reducing their effectiveness, since a single service similarity value can be misleading. Specifically, *a single value cannot be clearly interpreted, because it hides the values of its individual objectives*. For instance, a high single value may not necessarily mean that both objectives have high values or equal single values may hide very different objective values. While the effectiveness reduction using single-objective functions has been verified in some research fields (e.g., [5]), it has not been investigated in the service engineering field.

Contribution. Going to a completely different direction from the state-of-the-art, we argue that multi-objective service similarity metrics are more effective than single-objective ones. To support this argument we propose a metric that employs multi-objective functions in the different parts of service interface document, which specify the functional characteristics of a service in terms of different facets. In this paper, we consider two objectives, one for assessing `semantic` and one for assessing `syntactic` (element/service) similarity[2]. In service similarity, each objective aggregates the corresponding objectives of the element similarity, guided by the element structure. In this way, there is a clear interpretation of the similarity results in each service facet. Also, the proposed metric can be extended with new objectives, without affecting the definitions of the remaining ones.

To demonstrate the benefits of measuring service similarity based on multi-objective functions, we compare the results produced by the service organization task that adopts a typical hierarchical clustering method [6], in two cases: (i) using the proposed multi-objective metric; (ii) using a hybrid single-objective metric, which adopts a hard-wired aggregation function for calculating the element similarity. We also show that

[1]http://aws.amazon.com

[2]We denote by this `font` semantic and by this `font` syntactic information.

the values of the single-objective metric can be misleading, compared to the values of the proposed multi-objective metric. The overall results are promising, indicating that the current research directions are somehow misguided.

Impact. Service-oriented software engineering tasks can exploit such a metric via examining in which facets the compared services are similar and aggregating in a proper way the final results of the service similarity objectives. In this way, the engineering tasks can integrate the results of the distinct service similarity objectives within a *composite* metric, while the state-of-the-art approaches aggregate the objectives of element similarities into a hybrid one, as discussed in Section II. In general, a composite multi-objective metric is more flexible than a single-objective one, since the former offers alternative orderings of (simultaneously or sequentially calculated) objectives. In this way, the effectiveness of engineering tasks, which depend on the proposed similarity metric, is usually improved.

The rest of the paper is structured as follows: Section II describes the related state-of-the-art approaches. Section III defines the proposed multi-objective metric. Section IV evaluates the effectiveness of the metric. Finally, Section V concludes this work and discusses its future directions.

II. State-of-the-art

Studying the state-of-the-art approaches, which propose service similarity functions based on the service interface exclusively, we interestingly observe that they all rely on single-objective functions, as described below.

In detail, some of these approaches initially represent their input services as a set of elements, usually connected in a hierarchical structure. In these representations, an element may be characterized by information, which belong to different facets. For instance, the definition, `element name= 'price' type= 'float'`, includes the attribute `name`, which is mainly related to the service semantics, while the attribute `type` is related to the service-interface syntax. Based on these representations, the related approaches define different objectives for the different parts of an element. The element similarity is assessed by integrating these objectives into a single hybrid one by typically using hard-wired aggregation functions, such as the sum [7], [8] and the weighted sum [3], [9], [10], [11], [12].

There are also approaches that exploit only one kind of the information, included in the service interface document (e.g., only service semantics [13], [14], [15]).

Finally, in all these approaches, the most similar pairs of elements are determined either by comparing them as two flat sets of elements [7], [8], [3], [9], [10], [11], [13], or by matching their structure (e.g., XML schema structure [14]).

III. Multi-Objective Service Similarity

The *`semantic`* objective of the proposed function is based on the part of service interface, which implicitly defines service semantics and is the *`name`* of elements. The remaining information in the service-interface document concerns the signatures of service interface and is considered `syntactic` information. Alternative definitions of the *`semantic`* and the `syntactic` information could be considered. Prior to defining the proposed function, we firstly specify the model, with which we represent a service interface.

This model does depend on the version of the language in which a service interface is specified. In general, the interface of a Web service is specified in the Web Services Description/Definition Language (WSDL), which has been released in two versions, in 1.1 [3] and 2.0 [4]. Both versions of WSDL are XML[5]-based languages.

A. Service Interface Representation

Independently of the specification language, a service interface si can be represented by a tree model, in which the root node is the interface, characterized by its *`name`* and the set of its operations (Table I (Eq., 1)). An operation is characterized by its *`name`*, its (possibly empty) input and output messages (Table I (Eq., 2)). A message comprises its *`name`* and the (possibly empty) set of its message types (Table I (Eq., 3)). A message type is characterized by its *`name`* and its XML type (Table I (Eq., 4)), which in turn consists of its *`name`* and the set of its built-in XML data-types (Table I (Eq., 5)). Even if a complex XML type is recursively defined by other XML types forming a tree structure, we keep only the built-in data-types, which are the leaves of this tree structure by assuming that they include the substantial portion of *`semantic`* and `syntactic` information. The internal nodes of this tree structure can also contribute in the examined *`semantic`* and `syntactic` information. However, we leave this issue as future work.

TABLE I. Definition of the service interface representation.

$$si := \Big(\mathit{name} : String,\ ops : OPs\Big) \mid OPs = \{op_i : OP\} \quad (1)$$

$$OP := \Big(\mathit{name} : String,\ in : MSG,\ out : MSG\Big) \quad (2)$$

$$MSG := \Big(\mathit{name},\ mts : MTs\Big) \mid MTs = \{mt_i : MT\} \quad (3)$$

$$MT := \Big(\mathit{name},\ xt : XT\Big) \quad (4)$$

$$XT := \Big(\mathit{name},\ bts : BTs\Big) \mid BTs = \{\mathtt{bt_i} : \mathtt{anyType}^{\,6}\} \quad (5)$$

B. Multi-Objective Service Similarity Function

The proposed function $\mathcal{F}$ (Table II (Eq., 1)) accepts as input a pair of service interfaces and calculates the values of its *`semantic`* and `syntactic` objectives. Both objectives traverse in parallel the tree representations of the compared service interfaces. The hierarchical traversal of all layers (interface, operation, message, message type, and XML type) of the service interface representation is common in both objectives. The objectives differ in the considered part (*`semantic`* vs. `syntactic`) of service interface representation.

Service interface similarity. In this layer, both objectives calculate their values by firstly identifying the set C_{OPs} of

[3]http://www.w3c.org/TR/wsdl
[4]http://www.w3.org/TR/wsdl20/
[5]http://www.w3.org/TR/REC-xml
[6]http://www.w3.org/TR/xmlschema-2

TABLE II. DEFINITION OF MULTI-OBJECTIVE SERVICE SIMILARITY FUNCTION.

$$\mathcal{F}(si_1 : SI, si_2 : SI) := \Big[\mathcal{F}_{si}(\texttt{'sem'},\ si_1,\ si_2),\ \mathcal{F}_{si}(\texttt{'syn'},\ si_1,\ si_2)\Big] \tag{1}$$

$$\mathcal{F}_{si}(obj : OBJ,\ si_1 : SI,\ si_2 : SI) := \frac{\mathcal{F}_n(obj,\ si_1.name,\ si_2.name) + f_{OPs}(obj,\ si_1.ops,\ si_2.ops)}{2} \mid OBJ := \texttt{'sem'} \vee \texttt{'syn'} \vee \texttt{'single'} \tag{2}$$

$$f_{OPs}(obj : OBJ,\ ops_1 : OPs,\ ops_2 : OPs) := \frac{\sum_{k=1}^{|C_{OPs}|} \mathcal{F}_{op}(obj,\ ops_1.op_i,\ ops_2.op_j)}{|C_{OPs}|} \Big| (op_i,\ op_j) \in C_{OPs} \wedge i,j \in [1, |C_{OPs}|] \tag{3}$$

$$\mathcal{F}_{op}(obj : OBJ,\ op_1 : OP,\ op_2 : OP) := \frac{\mathcal{F}_n(obj,\ op_1.name,\ op_2.name) + \frac{\mathcal{F}_{msg}(obj,\ op_1.in,\ op_2.in) + \mathcal{F}_{msg}(obj,\ op_1.out,\ op_2.out)}{2}}{2} \tag{4}$$

$$\mathcal{F}_{msg}(obj : OBJ,\ msg_1 : MSG,\ msg_2 : MSG) := \frac{\mathcal{F}_n(obj,\ msg_1.name,\ msg_2.name) + f_{MTs}(obj,\ msg_1.mts,\ msg_2.mts)}{2} \tag{5}$$

$$f_{MTs}(obj : OBJ,\ mts_1 : MTs,\ mts_2 : MTs) := \frac{\sum_{k=1}^{|C_{MTs}|} \mathcal{F}_{mt}(obj,\ mts_1.mt_i,\ mts_2.mt_j)}{|C_{MTs}|} \Big| (mt_i,\ mt_j) \in C_{MTs} \wedge i,j \in [1, |C_{MTs}|] \tag{6}$$

$$\mathcal{F}_{mt}(obj : OBJ,\ mt_1 : MT,\ mt_2 : MT) := \frac{\mathcal{F}_n(obj,\ mt_1.name,\ mt_2.name) + \mathcal{F}_{xt}(mt_1.xt,\ mt_2.xt)}{2} \tag{7}$$

$$\mathcal{F}_{xt}(obj : OBJ,\ xt_1 : XT,\ xt_2 : XT) := \frac{\mathcal{F}_n(obj,\ xt_1.name,\ xt_2.name) + f_{BTs}(obj,\ xt_1.bts,\ xt_2.bts)}{2} \tag{8}$$

$$f_{BTs}(obj : OBJ,\ bts_1 : BTs,\ bts_2 : BTs) := \frac{\sum_{k=1}^{|C_{BTs}|} \mathcal{F}_{bt}(obj,\ bts_1.bt_i,\ bts_2.bt_j)}{|C_{BTs}|} \Big| (bt_i,\ bt_j) \in C_{BTs} \wedge i,j \in [1, |C_{BTs}|] \tag{9}$$

$$\mathcal{F}_{bt}(obj : OBJ,\ \texttt{bt}_1 : \texttt{anyType},\ \texttt{bt}_2 : \texttt{anyType}) := \begin{cases} 0, & \text{if } obj = \texttt{'sem'} \\ \mathcal{F}_{bt}(\texttt{bt}_1,\ \texttt{bt}_2), & \text{otherwise} \end{cases} \tag{10}$$

$$\mathcal{F}_n(obj : OBJ,\ name_1 : String,\ name_1 : String) := \begin{cases} 0, & \text{if } obj = \texttt{'syn'} \\ \begin{cases} Lin(name_1, name_2), & \text{if } name_1, name_2 \in \text{WordNet} \\ Levenshtein(name_1, name_2), & \text{if } name_1, name_2 \notin \text{WordNet} \\ 0, & \text{otherwise} \end{cases}, & \text{otherwise} \end{cases} \tag{11}$$

their most similar pairs of service operations. These pairs are found by solving the assignment problem of the maximum weighted matching in a bipartite graph [16]. The nodes of the graph correspond to service operations and the edges to similar operations. Following, both objectives calculate the average similarity of the set of the most similar operations, f_{OPs} (Table II (Eq., 3)). The overall (*semantic* or syntactic) service similarity, $\mathcal{F}_{si}$ (Table II (Eq., 2)[7]), equals to the average of the similarity of their *names* (taken into account only in the *semantic* objective) and the (*semantic* or syntactic) similarity of their operations. The way, in which the proposed function calculates the similarity of *names*, is explained in the following.

Service operation similarity. To calculate the similarity between two operations, $\mathcal{F}_{op}$ (Table II (Eq., 4)), both objectives calculate the average of the similarity of the operation *names* and of the average similarity of their input and output messages.

Operation message similarity. The similarity between two messages, $\mathcal{F}_{msg}$ (Table II (Eq., 5)), equals to the average of the similarity of their *names* and of the similarity of their message types. To calculate the latter similarity, both objectives firstly identify the set C_{MTs} of the most similar pairs of message types, by solving again the assignment problem. Following, the similarity of the set of the message types, f_{MTs} (Table II (Eq., 6)), equals to the average similarity of the previously identified most similar message types.

[7]The equation is parameterized with regard to the used objective.

Message type similarity. The similarity between two message types, $\mathcal{F}_{mt}$ (Table II (Eq., 7)), equals to the average of the similarity of the message-type *names* and the similarity of the XML types of the message types.

XML type similarity. The similarity between two XML types, $\mathcal{F}_{xt}$ (Table II (Eq., 8)), equals to the average of the similarity of their *names* and the similarity of their built-in data-types. To calculate the latter similarity, both objectives firstly identify the set C_{BTs} of the most similar pairs of built-in data-types, by solving once more the assignment problem. Following, the similarity of the set of the built-in data-types, f_{BTs} (Table II (Eq., 9)), equals to their average similarity.

Built-in data-type similarity. Concerning the syntactic similarity between two built-in data-types, $\mathcal{F}_{bt}$ (Table II (Eq., 10)), it is usually calculated in the literature based on a statically defined similarity table. In this paper, we adopt the table detailed in [9], where the built-in data-types are organized into five groups: the *Integer*, *Real*, *String*, *Date*, and *Boolean* groups (e.g., the *Integer* group consists of the integer, byte, short, and long data-types). Data-types of the same group are characterized by the highest possible similarity degree (equal to one). Data types of different groups are not always considered completely dissimilar, but their similarity degree equals to the complement of the information loss that occurs if a casting from the one data-type to the other can be applied. Underline that we do not provide in Table II a specific formula for the metric $\mathcal{F}_{bt}$, because the value of this metric is directly given in the aforementioned similarity table.

Regarding the *semantic* similarity between two built-in data-types, it apparently equals to zero.

Name similarity. Concerning the similarity between two *names*, it is usually calculated in the literature either by comparing the *names* as a sequence of string characters or by comparing their meaning. In the former way, a variety of string similarity metrics [17], [18] have been used in the literature. In the latter way, the relatedness of the concepts of an ontology, to which the *names* belong, is calculated. An ontology can be domain-specific, i.e., includes concepts related to an application domain, or general-purpose, i.e., includes all the concepts of a human language. A metric that uses domain-specific ontologies tends to give more effective results, as commented in [9], though, they are not usually available.

Inspired by [19], we propose the metric $\mathcal{F}_n$ (Table II (Eq., 11)), which firstly calculates the similarity of two *names* by comparing their meaning through using the general-purpose ontologies of *WordNet* [20]. Among the six available similarity metrics [21], which exploit *WordNet*, the proposed metric is based on *Lin*'s metric because it is one of the most efficient and effective ones. If only one of the *names* is contained in ontologies, then the proposed metric considers them completely dissimilar. If both *names* are not contained in ontologies, then the proposed metric compares them as a sequence of string characters and calculates their similarity using *Levenshtein*'s metric [22]. Among all the string-based metrics, the proposed metric uses *Levenshtein*'s metric since it gives the best results in XML schemas, as evaluated in [23].

IV. Emerging Results

Our evaluation includes two parts. Firstly, we demonstrate that single-objective functions can be misleading when measuring service similarity. Secondly, we show how the use of multi-objective functions can improve the effectiveness of a service clustering method that enables the service organization. Prior to discussing the results of our evaluation, we present the experiment setup.

Experiment setup. We compare our multi-objective similarity metric against a single-objective one. The latter metric follows the same hierarchical way of calculating similarity with our multi-objective metric. The main difference between the two metrics is that in the single-objective metric the similarity of built-in data-types equals to the average of their *semantic* and syntactic similarities. The used single-objective function is also defined in Table II (Eq., 2-11).

For the comparison of the two metrics, we use the services, StockQuotePrice (si_1), StockQuoteRoundedPrice (si_2), Date (si_3), and Calculator (si_4), whose WSDL documents are available at this location[8]. The first (resp., second) service returns the (resp., rounded) price of an input product, the third service gives the date of an input product order, and the fourth service calculates the square root of a given integer number. We use these services, since they cover different cases of service similarities.

[8]http://www.cs.uoi.gr/~dathanas/allServices.zip

Single-objective values are misleading. To realize the first part of our evaluation, we calculated the values of the single- and the multi-objective functions for the pairs (si_1, si_2), (si_1, si_3), and (si_1, si_4). Ideally, the services of (si_1, si_2) have high similarities in both objectives. The services of (si_1, si_3) have high *semantic* and low syntactic similarity. Finally, the services of (si_1, si_4) have low *semantic* and high syntactic similarity. Based on the results, presented in Table III, the multi-objective function correctly reflects the ideal service similarity in both facets. On the contrary, the single-objective values is misleading in (si_1, si_3) and (si_1, si_4), since it hides the zero and the very high values, respectively, of the syntactic objective.

TABLE III. Similarity values of the used Web services.

Service pair	*Single-objective*	*Multi-objective* $\mathcal{F}$ = [*'sem'*, 'syn']
(si_1, si_2)	0.89	[*0.90*, 0.99]
(si_1, si_3)	0.76	[*0.76*, 0.00]
(si_1, si_4)	0.29	[*0.28*, 0.99]

Note that in these results the value of the used single-objective function is very close to that of the *semantic* objective, since the amount of the *semantic* information, used by the proposed metric, is higher than that of the syntactic information. However, this fact does not subvert our observation that the values of the single-objective may be misleading. Even if the portions of *semantic* and syntactic information are equal, similar single-objective values would probably conceal very different values in the individual objectives.

Effectiveness improvement in service organization. We indicatively use a service-engineering method that depends on a service similarity function. In particular, we focus on a method that organizes similar services into groups. To this end, we apply the typical hierarchical bottom-up clustering method [6] to form groups of *semantically* and syntactically similar services. We executed this method by giving as input to it the previous service interfaces, si_1-si_4, along with one more, called Clock (si_5), which returns the current time. We use it since it is highly syntactically and *semantically* similar with si_3.

The used clustering method initially considers that all service interfaces form singleton clusters and following, repeatedly merges cluster pairs until no more clusters can be formed (zero similarity), or only one cluster remains. The output of the method is a hierarchy of clusters. Usually, useful clusters are determined at the lowest levels of this hierarchy. We executed the clustering method twice, one time for the single-objective and one for the multi-objective function. In the case of the multi-objective function, the method merges clusters whose similarity is greater than other clusters in both objectives. The results are presented in the dendrograms of Figure 1, which depict the clusters produced at each clustering iteration. Note that the ideal clusters are the following: (si_1, si_2), which corresponds to the most similar service interfaces in both objectives, and (si_3, si_5), which corresponds to the second most similar pair of service interfaces in both objectives.

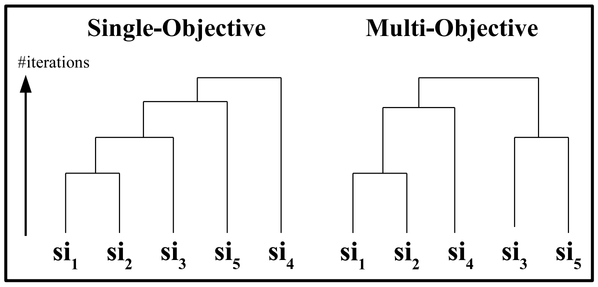

Fig. 1. The results of the clustering method for each similarity function.

Based on the dendrograms, we observe that using the multi-objective function is more effective than using the single-objective function. In the case of the multi-objective function, the method correctly merges, si_1 and si_2, in its first step. In its second step, the method correctly merges, si_3 and si_5. In the case of the single-objective function, the method forms in its first step the same cluster with that in the case of the multi-objective function. However, in its second step, the method wrongly merges the previous cluster with si_3. This faulty step is because the single-objective function hides the high value of the `syntactic` objective for (si_3, si_5) and the low of the same objective in (si_1, si_3) and (si_2, si_3).

V. Conclusions and Future Work

To sum up, we argued that single-objective service similarity metrics can be misleading and compromise the effectiveness of related service-engineering methods. To support our argument, we proposed a multi-objective metric and we evaluated it against a single-objective one. Our preliminary results showed that the effectiveness of service-engineering methods can be improved by using multi-objective metrics, indicating that the proposed research direction is promising. However, there is room for further research to this direction.

Possible future work includes more sophisticated multi-objective metrics, which consider the structure of the input/output XML types. Additional objectives, which measure similarity in terms of other service facets (e.g., business protocols, quality of service) can be examined. Finally, from a broader perspective, the adoption of multi-objective metrics in various phases of the service-engineering life-cycle is the ultimate challenge, which involves modeling the issues involved as multi-objective optimization problems.

References

[1] T. Erl, *Service-Oriented Architecture: Concepts, Technology, and Design*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2005.

[2] J. Walkerdine, J. Hutchinson, P. Sawyer, G. Dobson, and V. Onditi, "A faceted approach to service specification," in *International Conference on Internet and Web Applications and Services*, 2007.

[3] G. Spanoudakis and A. Zisman, "Discovering services during service-based system design using uml," *IEEE Transactions on Software Engineering*, vol. 36, no. 3, pp. 371–389, 2010.

[4] G. Beliakov, A. Pradera, and T. Calvo, *Aggregation Functions: A Guide for Practitioners*, ser. Studies in Fuzziness and Soft Computing. Springer, 2007, vol. 221.

[5] K. Praditwong, M. Harman, and X. Yao, "Software module clustering as a multi-objective search problem," *IEEE Transactions on Software Engineering*, vol. 37, no. 2, pp. 264–282, 2011.

[6] O. Maqbool and H. A. Babri, "Hierarchical clustering for software architecture recovery," *IEEE Transactions on Software Engineering*, vol. 33, no. 11, pp. 759–780, 2007.

[7] Y. Wang and E. Stroulia, "Flexible interface matching for web-service discovery," in *International Conference on Web Information Systems Engineering*, 2003, pp. 147–156.

[8] J. Wu and Z. Wu, "Similarity-based web service matchmaking," in *IEEE International Conference on Services Computing*, 2005, pp. 287–294.

[9] P. Plebani and B. Pernici, "Urbe: Web service retrieval based on similarity evaluation," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 11, pp. 1629–1642, 2009.

[10] K. Elgazzar, A. E. Hassan, and P. Martin, "Clustering WSDL documents to bootstrap the discovery of web services," in *IEEE International Conference on Web Services, Miami, Florida, USA*, 2010, pp. 147–154.

[11] D. Athanasopoulos, A. Zarras, P. Vassiliadis, and V. Issarny, "Mining service abstractions," in *International Conference on Software Engineering*, 2011, pp. 944–947.

[12] Z. Cong and A. F. Gil, "Efficient web service discovery using hierarchical clustering," in *Agreement Technologies - International Conference, Beijing, China*, 2013, pp. 63–74.

[13] F. Liu, Y. Shi, J. Yu, T. Wang, and J. Wu, "Measuring similarity of web services based on wsdl," in *International Conference on Web Services*, 2010, pp. 155–162.

[14] Y. Hao, Y. Zhang, and J. Cao, "Wsxplorer: Searching for desired web services," in *International Conference on Advanced Information Systems Engineering*, 2007, pp. 173–187.

[15] X. Dong, A. Halevy, J. Madhavan, E. Nemes, and J. Zhang, "Similarity search for web services," in *International Conference on Very Large Data Bases*, 2004.

[16] R. Burkard, M. Dell'Amico, and S. Martello, *Assignment Problems*. USA: Society for Industrial and Applied Mathematics, 2009.

[17] S. V. Rice, H. Bunke, and T. A. Nartker, "Classes of cost functions for string edit distance," *Algorithmica*, vol. 18, no. 2, pp. 271–280, 1997.

[18] G. Navarro, "A guided tour to approximate string matching," *ACM Computing Surveys*, vol. 33, no. 1, pp. 31–88, 2001.

[19] F. Giunchiglia, P. Shvaiko, and M. Yatskevich, "S-match: an algorithm and an implementation of semantic matching," in *European Semantic Web Symposium*, 2004, pp. 61–75.

[20] G. A. Miller, "Wordnet: a lexical database for english," *ACM Communications*, vol. 38, no. 11, pp. 39–41, 1995.

[21] T. Pedersen, S. Patwardhan, and J. Michelizzi, "Wordnet: : Similarity - measuring the relatedness of concepts," in *National Conference on Innovative Applications of Artificial Intelligence*, 2004, pp. 1024–1025.

[22] V. Levenshtein, "Binary Codes Capable of Correcting Spurious Insertions and Deletions of ones," *Problems of Information Transmission*, vol. 1, pp. 8–17, 1965.

[23] W. W. Cohen, P. D. Ravikumar, and S. E. Fienberg, "A comparison of string distance metrics for name-matching tasks," in *International Workshop on Information Integration on the Web*, 2003, pp. 73–78.