

Matching of Incomplete Service Specifications Exemplified by Privacy Policy Matching

Marie Christin Platenius^{1(✉)}, Svetlana Arifulina², Ronald Petrlc²,
and Wilhelm Schäfer¹

¹ Heinz Nixdorf Institute, Paderborn, Germany
{m.platenius,wilhelm}@upb.de

² Department of Computer Science, University of Paderborn, Paderborn, Germany
{s.arifulina,ronald.petrlic}@upb.de

Abstract. Service matching approaches determine to what extent a provided service matches a requester's requirements. This process is based on service specifications describing functional (e.g., signatures) as well as non-functional properties (e.g., privacy policies). However, we cannot expect service specifications to be complete as providers do not want to share all details of their services' implementation. Moreover, creating complete specifications requires much effort. In this paper, we propose a novel service matching approach taking into account a service's signatures and privacy policies. In particular, our approach applies fuzzy matching techniques that are able to deal with incomplete service specifications. As a benefit, decision-making based on matching results is improved and service matching becomes better applicable in practice.

Keywords: Service discovery · Service matching · Fuzzy matching · Fuzziness · Uncertainty · Privacy policy matching

1 Introduction

Nowadays, service providers provide software components in the form of services in service markets. In order to buy and use those services, service requesters have to discover services that fit their requirements. For this reason, service brokers apply *service matching* approaches that determine to what extent a provided service's specification matches a requester's requirements [11]. Service matching is the most accurate if many different properties of a service are considered [6]. This means, not only functional properties including structural and behavioral information should be described and matched but also several non-functional properties. For example, in this paper, we focus on service specifications including signatures and privacy policies.

Current service matching approaches come with the problem that they assume specifications to be complete with all considered service properties specified in

This work was supported by the German Research Foundation (DFG) within the Collaborative Research Center "On-The-Fly Computing" (CRC 901).

detail. However, in the real world, we cannot expect service specifications to be perfect or complete for several reasons [12]. One of the reasons is that providers often do not know all information about their services. For example, quality properties (e.g., response time) are hard to judge if they depend on third-party servers. In addition, providers may not want to share everything in order to protect business interests. Furthermore, creating detailed and machine-readable specifications costs much effort. Especially if they are not used to it, it is hard to convince people to work on such specifications. Current matching approaches are not able to deal with such incomplete specifications and return false matching results in the presence of incompleteness. This leads to requesters rejecting actually fitting services or accepting unfitting services without knowing that the matching result is uncertain.

We propose a novel service matching approach that extends traditional signature and privacy policy matching by applying fuzzy matching techniques. In particular, our approach is able to deal with incomplete service specifications as it reflects the grade of incompleteness in a *fuzziness score*. The contribution of this paper is twofold: On the one hand, it provides an improved approach for service brokers that allows better decision-making during service selection. On the other hand, it represents a first step to bring service matching into practice, coping with real world assumptions like incomplete specifications.

The remainder of this paper is structured as follows. In Sects. 2 and 3, we describe the service specifications for signatures and privacy policies and how they are matched as a foundation. Section 4 motivates fuzzy matching and explains how we cope with it. Section 5 describes a concrete application of fuzzy matching to privacy policies. In Sect. 6, we briefly summarize related work and we conclude the paper in Sect. 7.

2 Service and Request Specification

As a running example, we consider a market where service providers offer different software services for university management. In order to find a suitable service, a service requester specifies her requirements, e.g., the requirements for a service that prints out an overview of grades. Such a request has to be matched to the specifications of the available provided services.

We specify services using our Service Specification Language (SSL) [14] because it has been optimized for efficient and accurate matching of comprehensive service specifications. It describes different service properties including structural information, behavioral information, and non-functional properties of a service. In this paper, we only focus on two aspects: signatures and privacy.

A signature describes the inputs and outputs of a service's operation. For example, the signature presented in the middle part of Fig. 1 describes the operation `printOverview` of the provided service `OverviewOfGradesPrinter` with two input parameters (`grades` of type `List` and `id` of type `ID`) and one output parameter (`diagram` of type `Diagram`). All data types used in the signature are defined by ontological concepts that represent knowledge of different domains

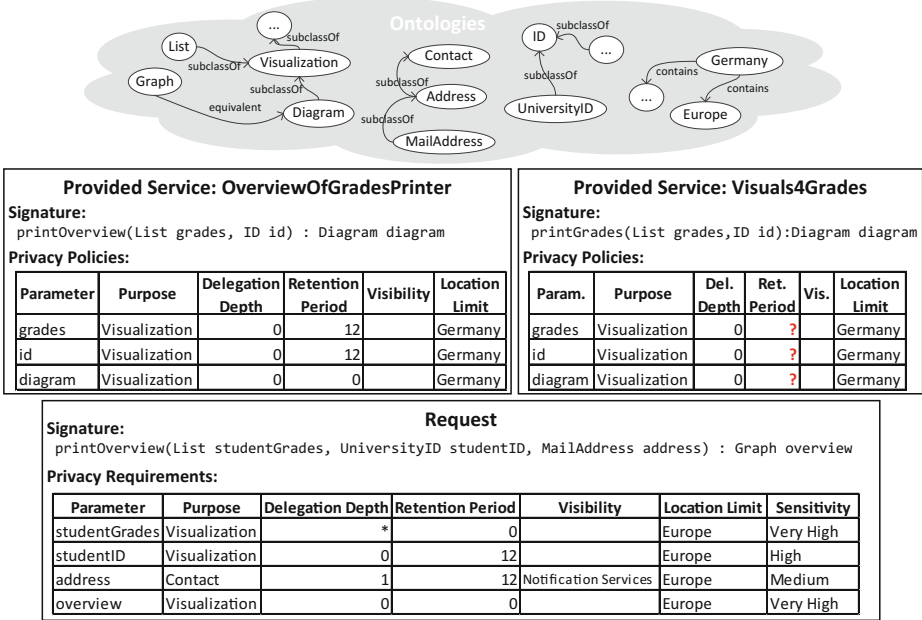


Fig. 1. Example ontologies, a specification of a provided service, and a request

including concepts and relations between these concepts. For example, the upper part of Fig. 1 depicts extracts of ontologies for the domains of visualization, contact, university, and locations.

Our privacy specifications are based on an integration of several existing languages. Most parts follow the principles of the privacy policy model by Costante et al. [4,5] including further elements from the approaches presented by Kapitsaki [8] and Tbahrithi et al. [16]. We adapted, united, and extended these approaches and integrated them into SSL. In particular, we connected the privacy specification to the above described signature specification.

The middle part of Fig. 1 depicts the privacy specification of **OverviewOfGradesPrinter** in a tabular notation. Each row in this table represents a privacy policy. Each privacy policy refers to a parameter from the service’s signature and specifies what kind of usage regarding the data corresponding to this parameter is permitted. In our example specifications, every parameter of the signature is covered: the two input parameters **grades** und **id** and the output parameter **diagram**. The columns represent the different restrictions that can be specified for a policy:

Purpose defines the reason for the collection of the corresponding data and its usage [4]. For example, the policies for all depicted parameters are related to the purpose of **Visualization**. All other kinds of usage of **grades**, **id**, and **diagram** are not allowed for **OverviewOfGradesPrinter**. All terms used as

purpose have to be defined in an ontology. A parameter may also appear in several policies with different purposes.

Delegation Depth refers to the amount of levels (i.e., services) a parameter may be forwarded to. This becomes relevant if a service is used as part of a composition. The privacy policy for `OverviewOfGradesPrinter` is very strict with respect to delegation as it allows no delegation of the given data at all. The default range for delegation depth is $[0,10]$ [4]. Additionally, the values `undefined` and `infinity` are supported.

Retention Period defines for how long (here: how many months) the service will store given data. The more privacy-critical a parameter is, the more critical it is to store it. However, storage may be necessary in order to perform certain purposes. For example, for hotel reservation services, much data has to be stored for billing purposes. In our example, the `OverviewOfGradesPrinter` stores the `grades` as well as the `id` for 12 months. This is due to the provided functionality to create visualizations of comparisons of the grades of different terms. For retention period, we support values in $[0,100]$ as well as `undefined` and `infinity`.

Visibility allows providers to restrict a policy to a set of service providers, service categories, or even specific services. For example, one can allow certain data to be visible only to services provided by Google, only to notification services, or only to the Google Alert Service.

Location Limit restricts privacy policies with respect to a location. For example, for the `OverviewOfGradesPrinter`, the provided data may only be processed in Germany. This also holds for delegated data. The location limit field can contain one or more terms defined in an underlying ontology of locations. Thus, the granularity of values (e.g., cities vs. countries vs. continents) depends on the ontology used.

We consider specifications that specify all properties listed above as complete. If any property is not specified, a specification is *incomplete*. For example, the provider of the second service in Fig. 1, `Visuals4Grades`, did not provide any information about retention periods.

A requester specifies her privacy requirements with the same concepts (see lower part of Fig. 1). In addition, she specifies a column *Sensitivity*. Thereby, the requester can define the importance of each requirement. She can choose between “very low”, “low”, “medium”, “high”, “very high”, and “mandatory”.

Please note that Fig. 1 only shows extracts of the complete specifications that can contain much more information, e.g., more detailed information about the service’s behavior.

3 Service Matching

In order to decide to what extent provided services satisfy a given request and to find out which one fits best, the services’ specifications are matched to the request. Matching is performed using a stepwise matching process. In

general, our matching process includes comparisons of structural requirements, behavioral requirements, and requirements for different non-functional properties [12]. In this paper, we focus on two steps of the matching process: Signature Matching and Privacy Matching. The lower part of Fig. 1 illustrates an exemplary specification of the request from our running example. In the following, we explain how this request is matched to the provided specification of `OverviewOfGradesPrinter`.

3.1 Ontological Signature Matching

Our signature matching approach follows the principles of established signature and semantic matching approaches [9, 10, 15]. In addition, our matcher is configurable and can take into account different parts of a signature, e.g., operation names, parameter types, parameter names, optional parameters, and exceptions. In this paper, we focus on signature matching based on parameter types.

The parameters are matched based on the rules for contravariance and covariance. This means, two signatures match if (a) the set of input parameters of the provided service's signature is a subset of the input parameters of the requested service's signature and (b) the set of output parameters of the requested service's signature is a subset of the output parameters of the provided service's signature. To achieve this, the signature matcher compares all parameter types of the two signatures pairwise, applying bipartite graph matching. Each data type pair is matched based on the referenced ontologies. Two parameter types match if (a) the input parameter type in the requested service's signature is either equivalent or more specific (i.e., a subclass) than the corresponding input parameter type in the provided service's signature and (b) the output parameter type of the provided service's signature is either equivalent or more specific than the output parameter type in the requested service's signature. For example, in the ontologies depicted in Fig. 1, the concept `UniversityID` is more specific than the concept `ID` and `Graph` is equivalent to `Diagram`. For our running example, this means, the matcher determines the following matching pairs:

- requester's input `studentGrades` and provider's input `grades`
- requester's input `studentID` and provider's input `id`
- requester's output `overview` and provider's output `diagram`

As all output parameters of the requester signature found a match and all input parameters of the provider signature found a match, the two signatures match 100 %. This result, including the determined parameter pairs, is an input to the subsequent privacy matching step.

3.2 Privacy Policy Matching

The privacy policies of a provided service match the privacy requirements of a request if the provided policies are more strict than the requested ones. For this purpose, the matching algorithm iterates over the requirements (rows) of

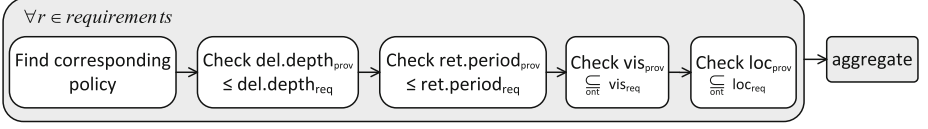


Fig. 2. Privacy matching algorithm

the requester’s specification and checks all the columns as depicted in Fig. 2: (1.) Find a corresponding policy in the provider’s specification based on the parameter pairs created during signature matching and the specified purposes. (2.) Match delegation depth and (3.) retention period: both checks succeed if the provider’s value is lower or equal than the requester’s value. (4.) Match visibility and (5.) location limits: both succeed if the provider’s values are a subset of the requester’s values, using an ontology to find relations between values. At the end, the final matching result is aggregated taking into account the requester’s sensitivity specifications; the final result is a value between 0 (does not match at all) and 1 (matches perfectly).

For our running example, these five steps work as follows:

(1.) Based on the preceding signature matching and an ontology-based matching of the purposes, we know that the **studentGrades** policy of the request corresponds to the **grades** policy of the provider’s specification. Furthermore, the **studentID** policy of the request corresponds to the **id** policy of the provider’s specification and the **overview** policy corresponds to the **diagram** policy. As the parameter **address** has not been assigned during signature matching, there has to be no corresponding policy for the requester’s **address** requirement. As a consequence, this requirement is ignored as always holds: since **address** is not taken as an input, its privacy requirements cannot be violated. (2.) In our example, all requirements match with respect to delegation depth as the provider’s specification is very strict in this context ($0 \leq *$, $0 \leq 0$, $0 \leq 0$). (3.) Retention period matches for **studentID** ($12 \leq 12$) and for **overview** ($0 \leq 0$). However, for **studentGrades**, the requester’s retention period is stricter ($12 \geq 0$). This is considered a mismatch. (4.) The visibility restrictions match if the visibility restrictions of the provider are a subset of the allowed visibilities of the requester. For our running example, this is the case for the three policies to be considered. (5.) Furthermore, these policies also match regarding the location limit because the locations specified in the provider’s policies are part of the locations specified by the requester ($Germany \leq Europe$). This relation is defined in the underlying ontology (see upper part of Fig. 1).

Due to the retention period mismatch in one of the policies, the final result for our example must be less than 1. The concrete value is calculated as follows: Each policy starts with a result of 1. This result is reduced with each mismatch depending on the difference. For example, because of the retention period mismatch above, we calculate the result for the **studentGrades** policy as follows: $result_{policy} = 1 - \min(1, 0.5 + ((providerRetentionPeriod - requesterRetentionPeriod)/100)) = 1 - \min(1, 0.5 + ((12 - 0)/100)) = 0.38$.

The value 0.5 is part of the matcher’s configuration and denotes the minimal impact of a retention period mismatch. By configuring this value, the broker can customize how much each of the properties contribute to the privacy matching result. The difference is divided by 100 (maximum value for retention period) for normalization. The results for the policies are aggregated to a final result for the whole privacy aspect. For this, we use a weighted average about the results of each policy multiplied with a multiplier corresponding to its sensitivity level. The higher the sensitivity level, the larger is the multiplier (“very low” = 0.2, “low” = 0.4, “medium” = 0.6, “high” = 0.8, “very high” = 1). For policies with the sensitivity “mandatory”, any kind of mismatch results in a matching result of 0. According to these values, for our example, we have the following aggregation function: $finalresult = (result_{studentGrades} \cdot 1 + result_{studentID} \cdot 0.8 + result_{overview} \cdot 1) / (1 + 0.8 + 1) = 0.78$. In addition to the final, numerical matching result, the requester can be provided with a detailed log about the matching process including a list of all mismatches.

4 Fuzziness in Service Matching

Current service matching approaches work fine if we count on several assumptions including that the providers share all information about their services and specify them in detail. As discussed in Sect. 1, there are several reasons for doubting this. However, when dropping these assumptions, uncertainty or *fuzziness* can be induced in service matching. Induced fuzziness leads to the fact that current matching approaches potentially deliver false results or even cannot be applied at all. In our earlier work, we describe different fuzziness sources [12, 13]. In this paper, we focus on *provider-induced fuzziness*, i.e., the case of incomplete specifications.

Coping with fuzziness in service matching leads to the problem that we require some kind of measurement in order to enable assessing the risk for fuzziness of a matching result. As an example, consider Fig. 1. With current matching approaches, both services, `OverviewOfGradesPrinter` and `Visuals4Grades`, would receive a very similar matching result considering the depicted request because in both cases, retention periods do not match. However, in the first case, we are certain that there is a mismatch, while, in the second case, we are uncertain, and the service could potentially match completely. For a requester, the second service would be a better choice as it matches at least as good as the other one, but, probably, even better. Because of such uncertainties, we propose that matchers should not only determine and output the matching result but also a *fuzziness score* reflecting the confidence a requester can have regarding the delivered matching result.

There are some requirements such fuzziness scores should satisfy:

- Normalization: For the requester, it should always be clear whether a returned fuzziness score is low or high. This means, the maximum possible fuzziness score and the minimum possible fuzziness score should be known.

- Comparability: Fuzziness scores calculated for different services need to be comparable. For example, if the requester chooses between two services based on matching results of $m_{serviceA} = 90\%$ and $m_{serviceB} = 95\%$ and fuzziness scores of $f_{serviceA} = 10\%$ and $f_{serviceB} = 50\%$, it should make sense to choose *serviceA* because its matching result is very close to the best available matching result while it is the less fuzzy one.
- Severity: The severity of fuzziness should be reflected in the fuzziness score. For example, if fuzziness occurs in a part of the specification that is irrelevant for the matching result, it should be ignored.

Our idea is to let each matching step return one fuzziness score and to aggregate these at the end to get a final fuzziness score. Thereby, the fuzziness measurements can leverage the specifics of each matching step best in order to fulfill the mentioned requirements, as we will show in the next section.

5 Fuzzy Matching of Incomplete Privacy Policies

In the example of privacy, a provider has to provide policies by law. However, this still does not mean that she also provides a formal representation as required for matching and that this representation includes all properties used during the matching process. In this section, we apply the fuzzy matching concepts introduced in Sect. 4 to privacy matching based on incomplete specifications.

5.1 Considering Incompleteness During Matching

In the presence of incomplete specifications, the matching approach has two possibilities to take into account missing values. (1) The optimistic case is to consider all missing information on the provider side as the most strict case, e.g., retention period = 0. This potentially increases the number of false positives. This means, the matcher returns a high matching result although, in reality, the service does not match. (2) The pessimistic case is to substitute missing information with the least strict case, e.g., retention period = *. This can lead to false negatives because, in uncertain cases, the matcher might return low matching results although, in reality, the service would be a good match. The choice whether to use a pessimistic or an optimistic approach depends on the broker's strategy and the risk affinity of the requesters.

5.2 Measuring Fuzziness Scores

There are different ways to measure incompleteness in privacy policies such that the returned fuzziness score satisfies the requirements mentioned above. In this paper, we describe one selected algorithm. The algorithm takes the mapping of requirements to corresponding provided policies, as determined during signature matching, as an input. It builds upon the privacy matching algorithm as it iterates through the specified concepts in similar ways.

For each requirement, the algorithm checks, whether one of the requested properties is not specified at the provider’s side. In general, the more values are missing at the provider’s side, the higher is the fuzziness score. If nothing is missing, the fuzziness score is 0. If everything is missing (the provider didn’t specify privacy policies at all), the fuzziness score is 1.

The fuzziness score for privacy matching is an aggregation of fuzziness scores for the different policies. A policy’s score grows with respect to (a) the requester’s sensitivity for this policy and (b) the weight assigned according to the severity of the provider’s restrictions. The rationale for taking sensitivity into account is as follows: if a policy cannot be checked due to incompleteness and the requester’s sensitivity regarding this policy is very high, the fuzziness is critical; if his sensitivity for this policy is rather low, fuzziness is less critical. The rationale for taking the severity into account is as follows: if a requester allows an infinite delegation depth (or retention period) for a certain policy, it does not matter if the provider’s delegation depth for this policy is not specified; but if a requester is very strict and specified an allowed delegation depth of 0, fuzziness induced by the missing provider’s delegation depth is much more critical.

As an example, consider the specification of **Visuals4Grades** and the request depicted in Fig. 1. Note that the provider did not specify any retention periods. The fuzziness score of the **grades** policy will be calculated as follows: $f_{grades} = 1 \cdot 1 = 1$, with the first 1 being a constant assigned to the requester’s sensitivity value “very high” (see Sect. 3.2) of the corresponding **studentGrades** policy and the second 1 being a constant corresponding to the minimum value 0 that the requester specified for retention period. Accordingly, the fuzziness score of the **id** policy will be calculated as follows: $f_{id} = 0.8 \cdot 0.8 = 0.64$, with 0.8 being the constant assigned to the sensitivity “high” and 0.8 being a constant corresponding to a medium low value 12 the requester specified for retention period in this policy. As a result, the fuzziness value for the **grades** policy is lower because the requester required a very strict policy for **grades**, while she specified a less strict policy with respect to retention period for **id**.

As we see in this example, the requirements *Severity* as well as *Comparability* are satisfied. As a foundation for *Comparability* and with all fuzziness values determined by this approach being normalized to a value between 0 and 1, the requirement *Normalization* is satisfied as well.

5.3 Case Study

We implemented our specification and matching approaches including the presented algorithm to measure incompleteness within an Eclipse-based tool-suite. Using this implementation, we evaluated the concepts presented in this paper together with privacy experts from the CRC 901 “On-The-Fly Computing” [17]. Our evaluation question was: how well do the results returned by our approach support a requester in deciding for a service. For this, we performed our fuzzy privacy matching approach on 21 pairs of example specifications and requests from the university management domain. For all these pairs, we first determined the matching results manually by expert knowledge, based on complete

specifications. After that, we ran our approach on incomplete versions of these specifications. After that, the experts' matching results based on full knowledge were compared to the results the matcher determined for the incomplete specifications. Thereby, we could determine correct matches (calculated matching result was equal to the expert's result, i.e., a true positive or a true negative) as well as mismatches (calculated matching result was not equal to the expert's result, i.e., a false positive or a false negative). Precision turned out to be 0.714 and recall was 1. Furthermore, we measured fuzziness scores and compared them to the matching results. The case study's results showed that the fuzziness scores for true positives and true negatives were rather low while the results for false positives and false negatives were comparatively high in 95 % of all cases. Based on a selection strategy that does not select services with a high (>0.5) fuzziness score, true positives have been reduced. Accordingly, the precision has been increased to 0.857. This result suggests to answer the evaluation question with "yes" as the determined fuzziness scores can contribute to an improved decision-making. In the future, we plan to evaluate our approach more extensively, taking into account more examples and more evaluation questions.

6 Related Work

Many approaches about service matching have already been published. The novelty of our approach is the combination of privacy matching and ontological signature matching considering fuzzy techniques to handle incomplete specifications. Thus, approaches considered related to our work are, on the one hand, privacy matching approaches, and, on the other hand, matching approaches that are able to handle incomplete service specifications.

There are many privacy matching approaches. The approach closest to our approach is the approach presented by Costante et al. [4, 5], which is also a foundation for our work. For example, the concepts of purpose, delegation depth, retention period, visibility, and sensitivity are based on their work. However, we added our own notions of ontology references and the connections to signatures and signature matching. In addition, their approach focuses on service composition, while we only focus on matching and provide additional concepts of fuzzy matching. Costante's original approach does not deal with incomplete specifications. The same differences hold for the works of Kapitsaki [8] and Tbahriti [16]. Further similar approaches are semantic matching approaches for WS-Policy specifications [3, 18]. Although these specifications describe rather security-related properties [18] or other quality properties [3] instead of privacy, the presented matching approaches are based on ontologies, as well. However, these approaches do not address the challenge of matching incomplete specifications and quantifying fuzziness occurring during fuzzy matching.

The challenge of matching incomplete service specifications has not been focussed in literature up to now. In our earlier work [13], we surveyed fuzzy matching approaches in general and came to the conclusion that the quantification of fuzziness, especially in the presence of incomplete specifications, is still

an open issue. There are some approaches that consider incomplete knowledge at the provider's side, e.g., [1] or [19], however, they do not quantify and return this incompleteness. Instead, the matching result is "adulterated" and does not provide the requester with information about uncertainty or its severity. Furthermore, these approaches deal with rather simple specifications without considering different characteristics of quality properties like our privacy specification does. Other approaches deal with incomplete specifications by trying to extract them either from a service's implementation or from other parts from the specification. For example, Strawberry synthesizes protocols [2]. This is no substitute for measuring and returning fuzziness scores and it probably does not work for all kinds of service properties, but it could reduce the induced fuzziness. Thus, it would be interesting to combine such approaches with ours. Similarly, component adaption approaches (e.g., [7]) could be used to enforce requested properties, e.g., delegation depth, in case of uncertainty.

To conclude, although there are many service matching approaches that can be used for privacy policy matching and service matching in general, the challenge of matching incomplete specifications and reflecting incompleteness is the matching result is new.

7 Conclusions

In this paper, we propose a service matching approach for combined signatures and privacy policies matching using fuzzy matching techniques. In particular, in addition to returning matching results, our approach measures and returns fuzziness score representing the degree of fuzziness induced by incomplete service specifications. This fuzziness score reflects the extent and criticality of the occurring fuzziness and, thereby, improves the service requester's decision-making. In general, the benefit is a novel service matching approach doing first steps to make service matching applicable in practice by reducing the assumptions existing service matching approaches make.

In the future, we want to learn more about how to improve our fuzzy matching concepts by applying them to other service aspects. For example, incompleteness at the provider side could occur if the service's reputation is matched: If a service is new in the market, then there are only few or no ratings.

Acknowledgments. We would like to thank Shafi Vijapurwala for contributing to our algorithm for measuring incompleteness in privacy specifications.

References

1. Bacciu, D., Buscemi, M., Mkrtchyan, L.: Adaptive fuzzy-valued service selection, pp. 2467–2471. ACM (2010)
2. Bertolino, A., Inverardi, P., Pelliccione, P., Tivoli, M.: Automatic synthesis of behavior protocols for composable web-services. In: Proceedings of the the 7th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering. ACM (2009)

3. Chaari, S., Badr, Y., Biennier, F.: Enhancing web service selection by QoS-based ontology and WS-policy. In: Proceedings of the 2008 ACM Symposium on Applied Computing, pp. 2426–2431. ACM (2008)
4. Constante, E., Paci, F., Zannone, N.: Privacy-aware web service composition and ranking. In: IEEE 20th International Conference on Web Services (ICWS), pp. 131–138. IEEE (2013)
5. Costante, E., Paci, F., Zannone, N.: Privacy-aware web service composition and ranking. *Int. J. Web Serv. Res. (IJWSR)* **10**(3), 1–23 (2013)
6. Cubo, J., Pimentel, E.: On the service discovery using context-awareness, semantic matching and behavioural compatibility. In: IEEE 15th International Conference on Computational Science and Engineering. IEEE (2012)
7. Gay, R., Mantel, H., Sprick, B.: Service automata. In: Barthe, G., Datta, A., Etalle, S. (eds.) FAST 2011. LNCS, vol. 7140, pp. 148–163. Springer, Heidelberg (2012)
8. Kapitsaki, G.M.: Reflecting user privacy preferences in context-aware web services. In: IEEE 20th International Conference on Web Services (ICWS). IEEE (2013)
9. Klusch, M., Kapahnke, P.: The iSeM matchmaker: A flexible approach for adaptive hybrid semantic service selection. *J. Web Semant. Sci. Serv. Agents World Wide Web* **15**, 1–14 (2012)
10. Moser, O., Rosenberg, F., Dustdar, S.: Domain-specific service selection for composite services. *Trans. Softw. Eng.* **38**(4), 828–843 (2012)
11. Papazoglou, M.P., Van Den Heuvel, W.-J.: Service oriented architectures: approaches, technologies and research issues. *VLDB J.* **16**, 389–415 (2007)
12. Platenius, M.C.: Fuzzy service matching in on-the-fly computing. In: Proceedings of the 9th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (2013)
13. Platenius, M.C., von Detten, M., Becker, S., Schäfer, W., Engels, G.: A survey of fuzzy service matching approaches in the context of on-the-fly computing. In: Proceedings of the 16th International ACM Sigsoft Symposium on Component-Based Software Engineering (2013)
14. SSE Development Team. Service Specification Environment - Website. <http://googl/E7QjPN>. Last Access May 2014
15. Stroulia, E., Wang, Y.: Structural and semantic matching for assessing web-service similarity. *Int. J. Coop. Inf. Syst.* **14**(04), 407–437 (2005)
16. Tbahrithi, S.-E., Medjahed, B., Ghedira, C., Benslimane, D., Mrissa, M.: Respecting privacy in web service composition. In: 2013 IEEE 20th International Conference on Web Services (ICWS), pp. 139–146. IEEE (2013)
17. University of Paderborn. Collaborative Research Center “On-the-Fly Computing” (CRC 901). <http://sfb901.uni-paderborn.de>. Last Access Apr 2014
18. Verma, K., Akkiraju, R., Goodwin, R.: Semantic matching of web service policies. In: Proceedings of the Second Workshop on SDWP, pp. 79–90 (2005)
19. Wang, P.: QoS-aware web services selection with intuitionistic fuzzy set under consumer’s vague perception. *Expert Syst. Appl.* **36**(3), 4460–4466 (2009)

Advances in Service-Oriented and Cloud Computing
Workshops of ESOCC 2014, Manchester, UK, September
2-4, 2014, Revised Selected Papers
Ortiz, G.; Tran, C. (Eds.)
2015, XI, 287 p. 81 illus., Softcover
ISBN: 978-3-319-14885-4