Market-Optimized Service Specification and Matching*

Svetlana Arifulina[1], Marie Christin Platenius[2], Steffen Becker[2], Christian Gerth[1], Gregor Engels[1], and Wilhelm Schäfer[2]

[1] Department of Computer Science, University of Paderborn, Germany
[2] Heinz Nixdorf Institute
University of Paderborn, Germany
{s.arifulina,m.platenius}@upb.de

Abstract. Various approaches in service engineering are based on service markets where brokers use service matching in order to perform service discovery. For matching, a broker translates the specifications of providers' services and requesters' requirements into her own specification language, in order to check their compliance using a matcher. The broker's success depends on the configuration of her language and its matcher because they influence important properties like the effort for providers and requesters to create suitable specifications as well as accuracy and runtime of matching. However, neither existing service specification languages, nor existing matching approaches are optimized in such way. Our approach automatically provides brokers with an optimal configuration of a language and its matcher to improve her success in a given market with respect to her strategy. The approach is based on formalized configuration properties and a predefined set of configuration rules.

Keywords: Service-Oriented Computing, Service Engineering, Service Specification, Service Matching, Service Brokers, Service Market.

1 Introduction

Many approaches in service engineering deal with emerging service markets, where service providers provide software services for trade [10,11]. In order to buy and use these services, service requesters have to discover services that satisfy their requirements. For this reason, various approaches introduce *service brokers*, who serve as intermediaries between requesters and providers [1]. Requesters and providers engage such brokers for a successful service discovery because the brokers have expertise in software services for certain markets [4]. For the discovery, a broker *matches* the requesters' requirements specifications to specifications of the provided services. For this, brokers use a special software called *matcher*. The goal of a matcher is based on specifications to determine

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

X. Franch et al. (Eds.): ICSOC 2014, LNCS 8831, pp. 543–550, 2014.

the extent, to which a provider's service complies with the requesters' requirements. Furthermore, providers and requesters often use different *specification languages*. Thus, the broker has to translate their specifications into her own language, which is supported by a certain matcher. This translation is out of the scope of this paper as it can be done automatically based on existing approaches [7].

In the market, different brokers compete with each other for customers [5]. Customers prefer a broker, who delivers most suitable services fast and with the least possible effort for them. Thus, in order to succeed in this competition, brokers can distinguish themselves by providing a fast and accurate service discovery with low effort for their customers. For that, brokers have to develop their own business strategies, which they adjust to the given market. A main part of this strategy is to find the configuration of a language and a matcher, which is optimal wrt. the service discovery and the customer's effort. Depending on the broker's strategy and the market characteristics, different configurations can be optimal because they are subject to multiple trade-offs. For example, a comprehensive specification language enables very accurate matching, but it requires quite a lot of effort for providers and requesters to create such detailed specifications. In contrast, simpler specifications can be matched much faster, but matching accuracy may suffer. Therefore, a broker becomes successful if she has several languages and matchers optimized according to her different strategies in the given market. However, there are too many different variations of languages and matchers to explore them manually as it is a tedious and error-prone task.

In this paper, we present a fully automated approach called `LM Optimizer`. `LM Optimizer` supports brokers to find an optimal "language-matcher configuration" (*LM Config*) in a service market. An LM Config refers to a pair of a service specification language and a corresponding matcher, working on instances of this language. Both the language and the matcher are configured in a certain manner. Configuration possibilities are determined by five kinds of configuration rules. Depending on the configuration, matching accuracy, matching runtime, and specification effort can be improved. LM Optimizer takes as an input market characteristics and the broker strategy described in the form of so-called configuration properties (CPs). Based on the given CPs, a configuration procedure applies well-defined configuration rules to configure a holistic service specification language and its matchers (provided as a part of LM Optimizer). As an output, the broker receives an LM Config optimal for the given CPs.

To sum up the contribution, our approach provides brokers with an optimal configuration of a language and a matcher customized for their business strategy and the given market. This allows brokers to obtain the best possible results in the service discovery. Thereby, our approach contributes to the development of successful service markets.

The paper is organized as follows: In the next section, we introduce a running example. Section 3 presents an overview of our approach, while its details are explained in Section 4 and 5. Section 6 briefly presents related work and Section 7

draws conclusions. A longer version of this paper, including an evaluation, has been published in form of a technical report [2].

2 Running Example

As a running example, we use a service market for `University Management` (UM). In this market, customers request software services that facilitate management tasks at a university. For that, they engage a broker, as shown in Fig. 1. In the market, three services are available: `Course Manager`, `Exam Manager`, and `Room Manager`. The broker matches the given requirements to the specifications of the provided services, in order to find the most suitable one.

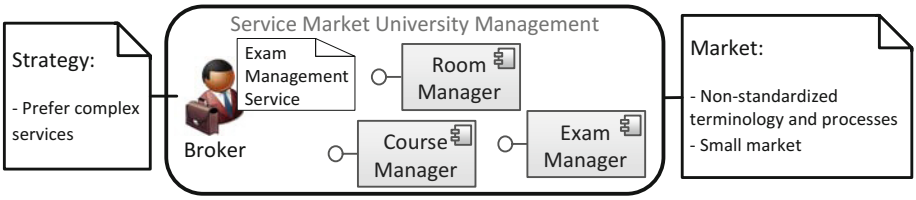

Fig. 1. Example service in the UM market

As shown in Fig. 1, the UM Market has certain characteristics that influence matching. For example, university structure and its management varies significantly from university to university. Therefore, neither terminology nor processes are standardized for services and their specifications in this market. In addition, the market is small because there are currently only few offers. According to the broker's strategy, this specific broker wants to trade only complex services because she expects the most profit from them. As a result, the broker needs an optimal LM Config in order to become as successful as possible.

3 Overview: LM Optimizer

We call our approach for finding an optimal LM Config `LM Optimizer`. `LM Optimizer` is fully automated in order to solve the high effort of finding an optimal LM Config manually, as elaborated in Section 5.2. Figure 2 provides an overview of our approach. `LM Optimizer` takes `configuration properties` as an input and delivers an `optimal LM Config` as an output. The configuration properties represent characteristics of the given market and properties of the broker strategy. As an output, we obtain an optimal LM Config consisting of a configured specification language and a set of matchers for it. The matching results delivered by the matcher (based on the service specifications in the configured language) are optimal for the given configuration properties.

`LM Optimizer` consists of three parts: (1) A holistic service specification language called `SSL`. (2) A set of matchers called `SSL Matchers`, which realize

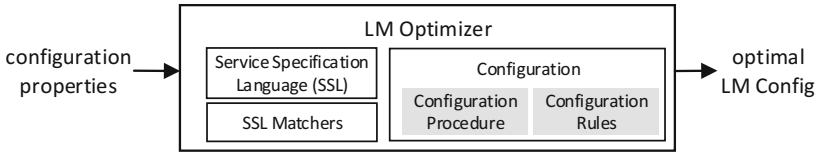

Fig. 2. Overview of the Approach

matching for this language. (3) `SSL` and `SSL matchers` are configured by the `Configuration`, which is responsible for obtaining an optimal LM Config for the given configuration properties. It consists of a set of `configuration rules` and a `configuration procedure`. The configuration procedure applies the configuration rules for the given configuration properties, in order to configure the SSL and its matchers optimally. SSL and its matching are described Section 4, while the configuration part is explained in Section 5.

4 Service Specification Language and Matching

Our `service specification language`, or `SSL`, is a holistic service specification language designed as a part of `LM Optimizer` (see Section 3). SSL consists of eight different *service specification aspects* which describe different structural, behavioral, and non-functional properties of a service. Each service specification aspect consists of a set of *language constructs* which describe a certain part of the corresponding service aspect. SSL can be used for comprehensive service specification in many different markets. For each aspect in the SSL, there is one matcher that is able to compare specifications of that aspect. A matcher may implement different matching approaches. Each matcher can run as a matching step in a so-called matching process. The technical report [2] explains in detail the SSL, properties of matching steps, and the matching process.

Matching with the SSL delivers matching results of high accuracy as diverse properties of services are compared. However, the performance of the matching is rather low because comparing all these service properties requires much computation time. Furthermore, writing specifications, which cover all aspects, costs providers and requesters a lot of effort. Thus, we propose a mechanism to configure the SSL and its matchers so that only service properties are considered, which are essential for matching in the given market according to a certain broker strategy. Configuration possibilities for the matchers include ordering, adaption, and weighting of matching steps. Correspondingly, accuracy, runtime, and effort can be balanced in an optimal way for this market and the broker strategy.

5 Configuration

In this section, we describe the configuration performed by LM Optimizer. This includes a set of configuration rules presented in Section 5.1 and a configuration procedure presented in Section 5.2.

5.1 Configuration Rules

LM Optimizer uses configuration rules to find an optimal LM Config. Configuration rules contain the knowledge of how certain configuration properties (CPs) influence service matching with SSL and its matchers. Thus, they determine the configuration of the SSL and its matchers for an optimal LM Config.

CPs formalize the properties of a market or the broker's strategy. One benefit of CPs is that the configuration knowledge condensed in the configuration rules is defined over a formal notation, which can be applied to every market. Thereby, we leverage market knowledge using a systematic, repeatable approach. Furthermore, we allow the broker to control the choice of an optimal LM Config by explicitly setting the relevant CPs. For these CPs, the broker describes her target market and her strategy by assigning concrete values to them.

Configuration rules are grouped in five types: (1) *Selection of specification aspects* – serves to select specification aspects needed in an optimal LM Config. (2) *Selection of language constructs* – helps to select language constructs needed within individual aspects. These two rule types are applied for configuring the SSL. (3) *Configuration of matching process* – are applied to determine an optimal order of matchers based on matcher dependencies and their runtime. (4) *Configuration of matcher properties* – sets certain matcher properties, e.g., a concrete algorithm of a matcher suitable for the current configuration. (5) *Configuration of aggregation of matching results* – puts the focus at certain matching steps, which are weighted higher during the aggregation in the final matching result.

Table 1 shows some example configuration properties and rules. For an extended list of rules as well as a complete overview of all configuration possibilities, refer to our technical report [2].

In order to understand the rationales of the example rules, let us consider the CP `Standardization`. A CP serves as a basis for the rules and has a range defined as a set of values, which can be assigned to that CP. The CP has the range of `true` for standardized terminology and processes, `terminology_only` for a standardized terminology, `processes_only` for standardized processes, and `false` for no standard for both. Well-established terminology can replace behavioral specifications of single operations because the semantics of the used names is commonly understandable. If the processes are standardized, matching of the order of operations is not needed as the behavior of equally named services is understood in the same way. Thus, according to `Rule 1`, `Signatures` are needed in an optimal LM Config but no `Pre- and Postconditions` or `Protocols`.

In `Rule 2`, the internal behavior of a service should be matched additionally to `Signatures`. For that, we can use `Pre- and Postconditions`. `Protocols` have to be considered as well, in order to match different orders of service operations in the non-standardized processes. `Rule 3` is used to select language constructs within the `Signatures` aspect. Due to the lack of standardization in the market, matching cannot rely on either operation or parameter names. Thus, they should not be considered for matching.

Example rules configuring the matchers are `Rule 4` – `Rule 6`. `Rule 4` states to decrease all matching thresholds for a market with a small size by 0.2 resulting in

Table 1. Example configuration rules

Rule no.	Rule type	Rule definition
Rule 1	Selection of specification aspects	Standardization = true → select Signatures
Rule 2	Selection of specification aspects	Standardization = false → select Signatures & Pre-/Postconditions & Protocols
Rule 3	Selection of language constructs	Standardization = false → do not consider Operation and parameter names in Signatures
Rule 4	Configuration of matching process	Market size = small → configure: decrease all thresholds by 0.2
Rule 5	Configuration of matcher properties	Standardization = true → use string similarity matching
Rule 6	Configuration of aggregation of matching results	Privacy important = true → configure privacy weight: multiply with 2

more services returned. Since in a small market, the probability of a perfect match is rather low, we can receive more matching results by decreasing the thresholds. In the matching process, matchers with a higher threshold are moved to the beginning because after their execution fewer services have to be matched. Thus, this decreases the runtime of the matching process. `Rule 5` sets the matching algorithm to string similarity matching if the market has standardized terminology and processes. This allows sparing the runtime because names in the standardized market are reliable for matching. `Rule 6` configures the aggregation of matching results by increasing the weight of the privacy matching result by the multiplicity of 2 if privacy is important for the broker in this market.

We rely on the knowledge of a broker to assign reasonable range values to her CPs. As future work, we plan to introduce measurable metrics for market properties, which will allow setting the range values at least semi-automatically.

5.2 Configuration Procedure

In this section, we present the part of LM Optimizer responsible for the configuration of the SSL and its matchers. It applies the configuration rules to a set of CPs assigned with concrete values given as input by the broker.

The configuration procedure configures the SSL by building a view on a subset of its specification aspects. Each aspect is also reduced to a subset of its language constructs. Thus, the whole aspects like `Signatures` or their language constructs, e.g., `parameter names`, can be omitted. Matching of an SSL configuration is limited to aspects and constructs defined in this configuration. We show three different example configurations in the technical report [2].

There are two phases in the configuration procedure (their order is important as the configuration of matching steps depends on the preceding selection):

1. Language Configuration: In this phase, the necessary service aspects are selected by applying the rule types *Selection of specification aspects* and *Selection of language constructs* described in Section 5.1.

2. Matcher Configuration: In this phase, for each selected language aspect, a corresponding matcher is added as a matching step in the matching process. The matching process is configured by ordering the matching steps. Then, the matching algorithms and the aggregation of results are configured.

LM Optimizer is a fully automated approach as a manual approach cannot cope with the high theoretical complexity of the configuration procedure. SSL consists of 8 aspects, where each one has 1 to 10 language constructs to configure with the mean number of 4.4 language constructs per aspect. The mean number of configurations per service aspect is 148. Including the possibility to arbitraryly combine these configurations, the number of all possible SSL configurations becomes $3.4 \cdot 10^{10}$. Then, we compute the number of possible matching configurations considering that thresholds and weights can be changed in a certain rate within a given interval. The number of configurations is exponential in this case. Currently, LM Optimizer supports 128 different language configurations and 256 different matching configurations enabling the choice between $1.6 \cdot 10^4$ LM Configs. These numbers are based on the extended list of rules from our technical report [2] also containing an evaluation of our approach.

6 Related Work

In the following, we briefly discuss the approaches mostly related to our work. We also explain why they do not solve the problem we stated in this paper.

Two comprehensive service specification approaches established in academia are the Unified Specification of Components (UnSCoM) framework [9] and the Unified Service Description Language (USDL) [3]. These two aim at comprehensive description and matching of a variety of existing service aspects and language constructs for them. In comparison to our approach, the authors of these two approaches do not provide any configuration possibilities either of the languages or of the corresponding matchers. Furthermore, neither the languages nor the matchers are optimized for any market characteristics or broker strategies.

Di Ruscio et al. propose a framework called `BY`ADL to create a customized architectural description language (ADL) by tailoring it, e.g., for domain specifics or operations like analysis or visualization [6]. The authors tailor an ADL by extending it with another language. In comparison, we configure the SSL by view building. In addition, we also configure the operation of matching.

Furthermore, there are some configurable service matchers [12]. However, their configuration possibilities are limited to different signature matching strategies and not selected automatically. Similarly, there are matchers that configure their aggregation strategies (but no other features) automatically [8]. Furthermore, their configuration only influences the matcher but never the specification language, and thereby the considered language constructs, as in our approach.

7 Conclusions

In this paper, we presented a fully automated approach `LM Optimizer` that supports service brokers to create a language-matcher configuration that is optimal for a given service market as well as a broker's strategy. Using this configuration, a broker can distinguish herself from other brokers competing for customers of their service discovery commissions. Thereby, LM Optimizer supports brokers

to be most successful. Three different case studies showed that LM Optimizer returned configurations with a high quality.

In the future, we want to extend our approach by extending the configuration rules and the configuration procedure in order to integrate more rules that are sophisticated. Furthermore, we want to apply our approach to other activities of a service broker in addition to service matching. For example, we expect tools for service certification or quality prediction to be configurable in similar ways.

Another field of future work is to adapt a broker's language matcher configuration already in use. Since service markets are changing rapidly [10], a broker needs to adapt the used matching approaches continuously. For that, the broker can use our approach to configure the most suitable language and the matcher for the current changed marker characteristics or the changed strategy. This allows the broker to remain competitive by being flexible to its changing characteristics.

References

1. Alonso, G., Casati, F., Kuno, H., Machiraju, V.: Web Services: Concepts, Architectures and Applications, 1st edn. Springer (2010)
2. Arifulina, S., Platenius, M.C., Gerth, C., Becker, S., Engels, G., Schäfer, W.: Configuration of Specification Language and Matching for Services in On-The-Fly Computing. Tech. Rep. tr-ri-14-342, Heinz Nixdorf Institute (2014)
3. Barros, A., Oberle, D. (eds.): Handbook of Service Description: USDL and Its Methods. Springer, New York (2012)
4. Benatallah, B., Hacid, M.S., Leger, A., Rey, C., Toumani, F.: On automating web services discovery. The VLDB Journal 14(1), 84–96 (2005)
5. Caillaud, B., Jullien, B.: Chicken & Egg: Competition among Intermediation Service Providers. The RAND Journal of Economics 34(2), 309–328 (2003), http://www.jstor.org/stable/1593720
6. Di Ruscio, D., Malavolta, I., Muccini, H., Pelliccione, P., Pierantonio, A.: Developing next generation adls through mde techniques. In: Proceedings of the ICSE 2010, USA, vol. 1, pp. 85–94 (2010)
7. Kappel, G., Langer, P., Retschitzegger, W., Schwinger, W., Wimmer, M.: Model Transformation By-Example: A Survey of the First Wave. In: Düsterhöft, A., Klettke, M., Schewe, K.-D. (eds.) Conceptual Modelling and Its Theoretical Foundations. LNCS, vol. 7260, pp. 197–215. Springer, Heidelberg (2012)
8. Klusch, M., Kapahnke, P.: The iSeM Matchmaker: A Flexible Approach for Adaptive Hybrid Semantic Service Selection. Web Semantics: Science, Services and Agents on the World Wide Web 15(3) (2012)
9. Overhage, S.: UnSCom: A Standardized Framework for the Specification of Software Components. In: Weske, M., Liggesmeyer, P. (eds.) NODe 2004. LNCS, vol. 3263, pp. 169–184. Springer, Heidelberg (2004)
10. Papazoglou, M.P., Traverso, P., Dustdar, S., Leymann, F.: Service-Oriented Computing: A Research Roadmap. International Journal of Cooperative Information Systems 17(2), 223–255 (2008)
11. Schlauderer, S., Overhage, S.: How perfect are markets for software services? an economic perspective on market deficiencies and desirable market features. In: Tuunainen, V.K., Rossi, M., Nandhakumar, J. (eds.) ECIS (2011)
12. Wei, D., Wang, T., Wang, J., Bernstein, A.: Sawsdl-imatcher: A customizable and effective semantic web service matchmaker. Web Semantics: Science, Services and Agents on the World Wide Web 9(4), 402–417 (2011)