Integrating Service Matchers into a Service Market Architecture[⋆]

Marie Christin Platenius, Steffen Becker, and Wilhelm Schäfer

Software Engineering Group, Heinz Nixdorf Institute,
University of Paderborn, Germany
{m.platenius,steffen.becker,wilhelm}@upb.de

Abstract. Service markets provide software components in the form of services. In order to enable a service discovery that satisfies service requesters and providers best, markets need automatic service matching: approaches for comparing whether a provided service satisfies a service request. Current markets, e.g., app markets, are limited to basic keyword-based search although many better suitable matching approaches are described in literature. However, necessary architectural decisions for the integration of matchers have a huge impact on quality properties like performance or security.

Architectural decisions wrt. service matchers have rarely been discussed, yet, and systematic approaches for their integration into service markets are missing. In this paper, we present a systematic integration approach including the definition of requirements and a discussion on architectural tactics. As a benefit, the decision-making process of integrating service matchers is supported and the overall market success can be improved.

Keywords: Service Matching, Service Markets, Software Architecture, On-The-Fly Computing.

1 Introduction

In the last decades, development turned from monolithic software products towards more flexible, component-based and service-oriented solutions. On *service markets*, service requesters can obtain software components that are provided in form of readily deployed services (Software-as-a-Service). Till date, there are only a few markets for this kind of services. However, following the example of markets for software products comparable to services, e.g., apps, we can expect service markets to rapidly increase in the future, too [12].

The more crowded service markets get, the more important becomes the quality and efficiency of the markets' service discovery mechanisms. While most established markets today are still limited to a relatively simple, keyword-based search, in academia, there is a mass of research for comprehensive *service matching* approaches, i.e., the analysis of whether the specifications of provided services satisfy a requested service [4,10] considering also structural, behavioral,

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

© Springer International Publishing Switzerland 2014

and quality properties. However, integrating a service matcher component implementing such a matching approach into an existing service market is complicated as there are different architectural possibilities with different consequences on market success. For example, integrating a service matcher into the requester's client provides the benefit of customizability but it may lead to a bottleneck that can slow down the whole discovery because many matching processes have to be performed sequentially. On the other hand, integrating a service matcher into the provider's system can lead to serious security problems allowing service providers to manipulate matching results but, depending on further parameters, a better performance may be attainable. Problems like these lead to the conclusion that a more systematic approach for the integration of service matchers on the architectural level is needed. However, until now, in literature, architectural decisions wrt. service matchers have rarely been discussed and there is no systematic approach for their integration into a market. Also applying classic software architecture decision-making methods has not been analysed wrt. service matchers and their influence on markets yet.

In this paper, we present a systematic approach for the integration of service matchers into a service market. This includes the definition of requirements and a discussion on architectural tactics based on these requirements. The contribution of this paper is an approach that can be used to integrate matchers into existing markets. Thereby, the general success of service markets, impacted by the use of service matching approaches, can be improved. An extended version of this paper including an application example has been published as technical report [9].

In the next section, we briefly summarize the foundations for this work. In Section 3, we derive requirements which we use to discuss integration tactics in Section 4. Section 5 deals with related work. The paper is concluded in Section 6.

2 Service Markets and Matching

In this paper, we use the following definition of a *service*: A service is a software component that is deployed and running on a service provider's platform. One example is Google Maps. Google Maps is a service offered by Google and it provides the functionality of querying and showing a map of some location. A *service market* allows trading, i.e., buying and selling, such services.

Although paradigms like Service-Oriented Architectures and Service-Oriented Computing have been investigated for several years now, there are not many established markets for services in the sense of readily deployed software components till date. In the area of web services, there has been the service registry standard UDDI but it has been officially discontinued years ago. However, along with emerging cloud providers, some more platforms to obtain web services for usage in the cloud appeared, e.g., Amazon Web Services [1]. Furthermore, there are markets for software products similar to services, like software components in the form of plug-ins (e.g., Eclipse Marketplace [14]) or apps. Schlauderer and Overhage analysed StrikeIron [13], Salesforce's AppExchange [11], and Google's Apps Marketplace (now Google Play [5]) as leading markets in 2010 [12].

In service markets, there are different roles, e.g., *service requesters* and *service providers*. Service requesters are interested in buying a service that fits to their requirements. Providers offer and sell services. Furthermore, there can be trusted third parties, e.g., a *market operator*, who provides and manages the market [12]. An actor can play several roles, e.g., intermediaries act as both requester and provider at the same time.

Service providers make their service offers available to requesters by publishing service specifications that help to *discover* their services. In most of today's markets, service specifications are either informal, describing a service's functional and non-functional properties using plain text mostly, or simple technical descriptions, like the Web Service Description Language (WSDL) [3], limited to the services' signatures. In academia, there are already a lot of approaches for more comprehensive but also machine-readable service specifications including expressive formalisms like protocols, ontological semantics, pre- and postconditions, and many more [8]. Such comprehensive specifications enable a service discovery taking into account technical, behavioral, as well as quality information, based on *service matching*. Service matching is the process of comparing the specification of a requested service, i.e., a *request*, to the specification of a provided service, in order to determine whether the provided service satisfies the request. It can be part of many different use cases, e.g., automated service composition, or used by service end-users. As an output, a matcher delivers a matching result which denotes how well a provided service specification matches a request. For example, a very simple specification of Google Maps could be `getMap(Location):Map`. Following the principles of simple signature matching approaches, this provided specification would achieve a good matching result with a request like `searchMap(City):Map` with `City` being a subtype of `Location`. There are many automated and much more complex matching approaches in literature. For a classification and an overview of recent surveys, refer to our earlier work [10].

3 Requirements for Matcher Integration

Figure 1 depicts an overview of the requirements collected for the integration of matchers into service markets. A dependency from A to B means that the fulfilment of B supports the fulfilment of A. Neither the requirements, nor the dependencies are meant to be a complete collection as we focussed on the ones that are most important in our context.

Due to page limitations, in this paper, we focus only on some of the requirements depicted in Fig. 1. For the complete list, refer to our technical report [9]. There, we also give an overview of the process and methodology we used to elicit the requirements. The requirements we selected are described in the following.

(*R5*) *Performance:* Performance refers to the time to perform one matching process, i.e., the time to determine how much a particular service satisfies the request. It needs to be high in order to gain a good efficiency of the overall discovery process (*R4*).

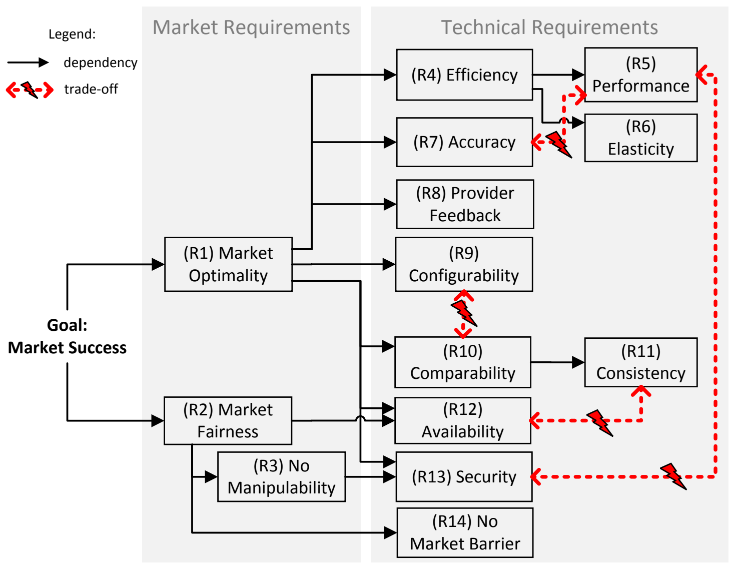

Fig. 1. Overview of the requirements for integrating a matcher into a market

(*R6*) *Elasticity:* Even if the performance of one matching process is good, the discovery's efficiency is still problematic when a huge amount of matching processes is required. Thus, similar to cloud computing systems, in service markets, the matching system needs to be elastic [6] in a way that it adapts to the amount of required matching processes.

(*R10*) *Comparability:* If different services are matched to the same request, services with the same matching result should satisfy the request equally well. Similarly, services with better matching results should satisfy the requester more than services with lower matching results. This has to hold, even and especially, if those services are offered by different providers.

(*R11*) *Consistency:* Matching results are only comparable if they are consistent. Dynamic markets, where providers can appear and disappear or change their offers at any time, can lead to situations in which several versions of a service or a service decription are available. It has to be avoided that this dynamics leads to inconsistent matching results between different providers so that comparability can be ensured.

(*R13*) *Security:* Matching results have to be secure so that they cannot be manipulated by any service provider. For example, if aspects like reputation of a service are matched, providers have to be prevented from cheating in a way that they claim to have a better reputation than they actually have.

Figure 1 shows trade-offs between requirements by dotted arrows annotated with a flash symbol. Because of such conflicting requirements, there may not be one general best solution for all service markets. For a description of these trade-offs, please refer to our technical report [9]

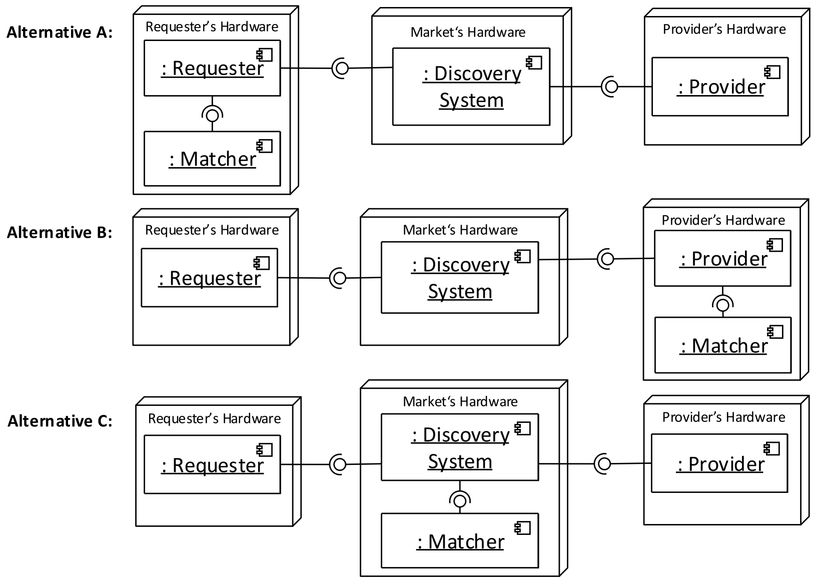

Fig. 2. Architectural alternatives for matcher integration

4 Integrating a Matcher Based on Architectural Tactics

When integrating the matcher into a market, there are different alternatives, as depicted in Fig. 2. In Alternative A, the matcher is integrated into the requester's system and, therefore, deployed on the requester's hardware. Here, the requester's system accesses the discovery system to get the specifications of the provided services and forwards them to the matcher. Alternative B lets the providers deploy the matcher on their own resources. In this case, the discovery system forwards the request to the providers, where each provider matches its service specifications against the request. In Alternative C, the matcher is part of the discovery system and deployed on the market operator's resources. Instead of the market operator, this role can also be played by another trusted third-party which is part of the market. The specifications of provided services could still be located at the providers, or, alternatively, stored on the market operator's resources, too. As we can see, the question of where to integrate the matcher is related to the question of who deploys the matcher.

Each of the three alternatives has different benefits and drawbacks and, in particular, a different impact on the fulfilment of our requirements. Table 1 lists these benefits and drawbacks with respect to the requirements collected in Section 3. A minus in the table means that it is difficult to satisfy the corresponding requirement, whereas a plus means that it is easier to satisfy it. Similarly to the architectural tactics described by Bachmann et al. [2], the table depends on bound and free parameters: Bound parameters are already fixed because their assignment is the same for all service markets. Free parameters (highlighted in italics) need

Table 1. Where to integrate the matcher?

	Alternative A: Requester	**Alternative B: Provider**	**Alternative C: Market Operator**	**w**
R5	- (no caching)	+	+	*?*
	depends on kind of requesters/providers			
R6	*depends on number of providers and requests at a time*			*?*
R10	+	- (conflict with *R9* and *R13*)	- (conflict with *R9*)	*?*
R11	- (not insurable)	+	- (not insurable)	*?*
R13	+	- (high risk for manipulation)	+	*?*

further assumptions to be assigned, i.e., the evaluation can have a different result depending on the properties of a concrete service market. Furthermore, there is a column about weights in order to allow influencing the overall evaluation by assigning special priorities to some requirements. Weights can be positive (e.g., +1), if a requirement is particularly important, or negative (e.g., -1), if it is less important compared to the other ones. Similar to free parameters, weights depend on properties of concrete service markets, too. Thus, they are not yet assigned in this general version of the table.

The table only covers the requirements selected in Sec. 3. For the complete table, refer to our technical report [9]. In the following, we describe the evaluation shown in Table 1.

(*R5*) Performance Regarding the performance, Alternative A seems not to be a good solution. Matchers located at the service provider or a market operator provide the possibility to cache matching results and benefit from it when similar requests are received. This possibility is not available for matchers deployed at the requester's because it would only pay off, if one requester repeatedly states similar requests, which is not the case if the discovery scenario works well and the requester already gets a satisfying result after her first request. Furthermore, if the matcher runs on hardware with high computation power, e.g., compute centers, this could speed up the matching process, too. In contrast, if the matcher runs on a mobile device, a matching process takes longer. Which role can provide the more appropriate resources depends (amongst others) on the domain. For example, requesters in some technical domain could be assumed to have better hardware available than the typical hotel booking service user, whereas providers can be expected to have access to more computation power than the requesters in both cases.

(*R6*) Elasticity For Alternative A, the amount of matchers increases with each new requester and, thereby, also with the amount of stated requests. This is good if there are many requests but only few service offers. For Alternative B, the amount increases with each new provider, and, thereby, also with the amount of service offers. If we have a market with many providers but only few requesters, Alternative B is preferable. Alternative C suffers from the fact that the market operator has to provide or pay a cloud infrastructure in order to provide an elastic matching architecture.

(***R10***) **Comparability** Comparability is in conflict with configurability. This especially holds for Alternative B and C: if providers use different matching configurations, the matching results for services of different providers are not comparable for the requester. However, if the requester has the matcher and the possibility to configure it, this is no problem as all services from different providers are matched with the same configuration. Comparability among different requests is not needed as matching on service markets is one-sided, i.e., we search an optimal allocation of services to requests but not the other way around because software services are immaterial and (almost) not capacity-constrained. In addition, comparability is also influenced by security as, in the case of manipulation of matching results, comparability cannot be ensured. This is a disadvantage of Alternative B because it is most susceptible for manipulation (see *R13*). All in all, for *R10*, Alternative A is the best solution.

(***R11***) **Consistency** Regarding consistency, Alternative A as well as Alternative C both have the disadvantage that it is not necessarily ensured that they match the specification describing the provider's current offer. Compared to this, the provider has a better chance to ensure consistency because the provider manages the specifications herself.

(***R13***) **Security** Security becomes a problem, in particular, if the provider deploys the matcher. In this case, it is hard to keep the provider from manipulating matching results. In contrast, the requester or the (trusted) market provider can be assumed not to be interested in faking the results.

As we can see, there is no obvious answer to the question of where to integrate the matcher. Each alternative has its advantages and disadvantages and some aspects depend on the concrete environment, e.g., the market's size. For this reason, the table needs to be adapted to concrete application scenarios. For an example of such an adaption refer to our technical report [9].

5 Related Work

Even though there is a lot of research for the single areas of software architectures, service matchers, and service market mechanisms, the integration of service matchers into a market has not been addressed on the architectural level, yet. For example, Klusch [7] as well as Dong et al. [4] give overviews of semantic service discovery architectures and classifies them into centralized and decentralized architectures. However, the alternatives are not discussed with respect to requirements for a matcher's integration nor taking into account market mechanisms. Similarly, the architecture description of web services by the W3C [15] distinguishes between a centralized server-sided scenario (similar to our Alternative C) and an index or peer-to-peer client-sided matching scenario (Alternative A) However, only the requirements wrt. the dynamics and the scalability of the environment are taken into account.

6 Conclusions

In this paper, we presented a systematic approach for integrating a service matcher into a service market. This approach includes the definition of requirements and a discussion on architectural tactics in order to enable a more informed decision-making regarding architectural alternatives. Both practitioners and researchers benefit from this paper. In practice, our results can be used to integrate service matchers into existing or emerging service markets and thereby supporting both requesters and providers by achieving their goals. In research, this paper also represents a first attempt to bridge the gap between markets and the mass of existing matching approaches in literature.

References

1. Amazon Web Services. Website, `aws.amazon.com` (last access: June 2014)
2. Bachmann, F., Bass, L., Klein, M.: Deriving architectural tactics: A step toward methodical architectural design. Technical report, Software Engineering Institute, Carnegie Mellon University, CMU/SEI-2003-TR-004 (2003)
3. Chinnici, R., Moreau, J.-J., Ryman, A., Weerawarana, S.: Web Services Description Language Version 2.0 Part 1: Core Language. Technical report (2007)
4. Dong, H., Hussain, F.K., Chang, E.: Semantic Web Service matchmakers: state of the art and challenges. In: Concurrency and Computation: Practice and Experience, vol. 25, pp. 961–988. Wiley Online Library (2012)
5. Google. Google Play - Website, `play.google.com/` (last access: June 2014)
6. Herbst, N.R., Kounev, S., Reussner, R.: Elasticity: What it is, and What it is Not. In: 10th Int. Conf. on Autonomic Computing. USENIX (2013)
7. Klusch, M.: Semantic web service coordination. In: CASCOM: Intelligent Service Coordination in the Semantic Web, pp. 59–104. Springer (2008)
8. O'Sullivan, J., Edmond, D., ter Hofstede, A.H.M.: Service description: A survey of the general nature of services. Distributed and Parallel Databases Journal (2002)
9. Platenius, M.C., Becker, S., Schäfer, W.: Integrating Service Matchers into a Service Market Architecture. Technical Report tr-ri-14-340, Heinz Nixdorf Institute (2014)
10. 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: 16th Int. Symposium on Component-based Software Engineering. ACM (2013)
11. Salesforce.com, Inc., Salesforce AppExchange, `appexchange.salesforce.com` (last access: June 2014)
12. Schlauderer, S., Overhage, S.: How Perfect are Markets for Software Services? An Economic Perspective on Market Deficiencies and Desirable Market Features. In: Proc. of the 19th European Conf. on Information Systems (2011)
13. StrikeIron. StrikeIron - Website, `http://www.strikeiron.com` (last access: June 2014)
14. The Eclipse Foundation. Eclipse Marketplace, `marketplace.eclipse.org` (last access: June 2014)
15. W3C. Web services architecture, `w3.org/TR/ws-arch` (last access: June 2014)