A recommender mechanism for service selection in service-oriented environments

Konstantinos Tserpes *, Fotis Aisopos, Dimosthenis Kyriazis, Theodora Varvarigou

Department of Electrical and Computer Engineering, National Technical University of Athens, 9, Heroon Polytechniou Str, 15773 Athens, Greece

ARTICLE INFO

Article history:
Received 15 December 2010
Received in revised form 30 October 2011
Accepted 9 November 2011
Available online 25 November 2011

Keywords:
Service oriented architecture
Service selection
Quality of experience
Recommender systems

ABSTRACT

The emergence of service-oriented computing as a key-enabler of web applications and the subsequent increase in the web services available, has brought to the surface a major disadvantage of service oriented architecture: there is no consistent way for the consumer to select services based on non-functional, quality requirements. Consumers perceive quality through the prism of their own experience and it is important to see how their evaluation of the quality provided is mapped to the specific quality parameters offered by the provider. In order to achieve that, this work suggests to use consumer ratings of the latter parameters so as to create a consumer quality profile and a provider reputation. By correlating this profile with others, it is possible to identify similarities in the service ratings that will lead to a prediction of which service might be the most appropriate for a specific consumer. Based on this rationale, we devised a Service Recommender mechanism and introduced a slight modification in the service lifecycle to accommodate the new Service Recommendation protocol that supports the mechanism.

1. Introduction

Service selection remains a persisting problem in Service Oriented Architecture (SOA). The challenge does not lie only in the identification of services that implement a certain type of functionality but also in the selection of those services that provide the quality level that is requested from the consumers. Quality of Service (QoS) is reflected to various parameters that the provider can monitor during service invocation and use to evaluate quality levels. The consumers, not having access to the providers’ monitoring mechanisms nor to their infrastructure, need to trust the providers’ measurements and evaluate them against their own experience on the application. The latter is referred to as Quality of Experience (QoE).

This paper proposes a formal method to model the evaluation of QoS against QoE by allowing the consumers to rate the QoS measurements of the provider based on their experience. The history of ratings is then used for two purposes: (a) to rank the providers’ reputation based on their performance on certain QoS parameters, and; (b) to create a profile of the consumers so we can provide more accurate recommendations for service selection based on their characteristics. The authors advocate that through this process the providers that systematically fail to comply with their obligations against the consumers will be isolated, protecting the latter. Most importantly, this method allows for a more accurate selection of services that meet the consumers’ quality requirements.

The method to achieve that consists of two steps. The first step involves the setting up of a mechanism that collects the consumers’ evaluations (ratings) on the QoS parameters upon which the two parties had agreed before invocation. The second step involves the use of collaborative filtering techniques in the set of collected ratings for effective service recommendation and selection. Both steps introduce a slight modification to the typical service lifecycle management that is discussed later and the whole process will be referred to as the Service Recommendation (SR) protocol.

The proposed solution is based on an infrastructure that follows SOA principles. In SOA, service providers register the services in a service broker and the service consumers search and discover required services by a broker [1]. In the web services paradigm of SOA, the role of the broker is assumed by a web service registry, some times accompanied by supporting mechanisms for the service management. As such, the proposed infrastructure is allowing for service providers to register their web services in a Universal Description Discovery and Integration (UDDI) registry. Service consumers can then discover the services that they want to invoke based on their functional description, which is in turn captured using the Web Services Description Language (WSDL).

To each of the web service descriptions available, the providers attach a Service Level Agreement (SLA) template, i.e. a type of offer that defines the terms of service invocation and quantifies the QoS level that the providers offer. A service is invoked on behalf of the

* Corresponding author. Tel.: +30 2107722568; fax: +30 2107722569.
E-mail addresses: tserpes@mail.ntua.gr (K. Tserpes), fotais@mail.ntua.gr (F. Aisopos), dimos@mail.ntua.gr (D. Kyriazis), doravarv@mail.ntua.gr (T. Varvarigou).

0167-739X/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.future.2011.11.003
consumers only after the SLA terms have been agreed between the two sides. The SLA terms contain sets of parameters that define the delivered quality. We will refer to these interchangeably as QoS parameters or SLA terms for readability purposes, even though they are not necessarily always the same. Once the SLA terms are agreed, the providers give access to their resources through the web service interface and they monitor the QoS parameters. In the case that a violation occurs, the SLA clauses come into force for the respective party. This process is referred to as SLA negotiation and can be managed by the service broker.

Upon termination of the service use, the consumers are providing a score that indicates their level of satisfaction from each particular QoS parameter that the SLA contained. The ratings are stored in the service broker that owns the registry and the SLA repository creating a profile for both the providers and the consumers. With the use of collaborative filtering techniques, the system finds consumers whose rating vectors are closely correlated and implements an SR mechanism that proposes services that satisfy the non functional requirements of the consumers.

In order to evaluate the usefulness and test the performance of the SR protocol and mechanism, an experiment in a real world business scenario (3D rendering) was executed. A reference implementation of the architecture that is proposed to support the SR mechanism was created and services were configured to be consumed by application domain experts (animators). The results show that the consumers can rely on such a decision support mechanism when they come across a plethora of service offers. The accuracy of the recommendations provides evidence that it is for their own benefit to contribute to the creation of a history of providers’ QoS evaluations using their own experience.

The SR protocol and mechanism and the architectural framework upon which it operates is presented in greater detail in the following sections. In particular, Section 2 discusses the literature references to the service recommendation as well as the recommender systems in general. Section 3 guides the reader through the service lifecycle management and the introduction of changes that the SR protocol brings to it. Section 4 presents the actual architectural framework that enables the abovementioned changes. Finally, Section 5 presents a proof of concept implementation and an experiment that confirms the usefulness of an SR mechanism. Finally, some items for discussion that yielded from this work are provided in Section 6.

2. Related work

The problem of service selection support has been tackled by various researchers in the recent past. In the frame of SOA, most works dealing with client-side quality evaluation perform recommendations during the service discovery/selection process, as for example in the research of Fudze and Abawajy [2]. Toward this direction, Hao et al. [3] presented a novel IR-Style mechanism for discovering and rating web services, with ratings here indicating each service relevancy, rather than its quality reputation. Reputation-based Service Selection in SOA has been studied in various works [4–8] while Grid Service brokering and accounting systems have been presented by Barmouta and Buyya [9] as well as Venugopal et al. [10], not focusing, however, on filtering based on the correlation of user experiences.

Most of the previous papers describe mechanisms for service selection integrated into the service discovery process that store and update QoS information in the UDDI registries [11], taking into account only objective factors described by service providers, with the exception of [4]. Subjective trust evaluations from clients that use the services were also introduced in the work of Wang et al. [12], where a QoS-supported recommender agent captures QoS information from a UDDI registry and also collects users’ appraisals in order to reach a specific recommendation for service selection through a Genetic algorithm (GA). The authors are not, however, correlating and grouping users as it is done in the current work, but they rather try to calculate each user’s trustability to evaluate his rating, which is a different approach.

This work can be investigated by numerous perspectives making it difficult to find a research studying the exact same topic. Perhaps the most interesting related approach on the same topic is the work of Stefan Schmidt et al. [13], where they deal with the topic of decision support for service selection in e-business environments using a fuzzy logic based framework. In this paper, the authors consider selection criteria such as trust, reputation and credibility, which have a fuzzy and variable nature by default [14]. The evaluation of the consumer is achieved through the integration of social reputation and trustworthiness factors. The main difference with this work is that the authors are making use of any kind of information that can be translated to the abovementioned selection criteria, rather than using only those that the consumer and the provider agree upon. This might have serious implications on the matchmaking process, as there is always a chance to evaluate a service provider in an unfair way or in a way that the consumer does not agree.

Another interesting work in the field of recommender systems and decision support is [15]. In this paper, the main concept is that the decision making process in e-commerce applications can be assisted by evaluating the past reputation, the trend and the confidence using fuzzy logic techniques. The most interesting part of this work is that it distinguishes the sources from which the three abovementioned criteria are deriving into two categories: the publicly accessible data (such as opinions, ratings, etc.) and the private/internal data (past transactions, review results, etc.). Using their proposed fuzzy logic mechanism, the authors evaluate the consumer value and correlating this information with the consumer risk and business risk, they calculate the consumer value factor. Similarly, the business value factor is calculated and the system attempts to match the two factors so as to optimize the business partnership. Again, the main difference is that the evaluation is not based on the criteria that both the negotiating parties agree.

On a different level, an instantiation of the problem that is studied here, is theoretically described in [16]. In the work of Park et al., consumer requirements are modeled using utility functions and consumers are operating in a non-cooperative multi-class QoS provision model. In this way, the authors are emulating the discrete offer model where the provider is offering a service at specific quality levels (Bronze, Silver, Gold service). Thus, the model used for resource sharing and arbitration at a router is used, in the context of QoS, as a way to model delivery of packaged network services by a service provider. Park et al., are studying the activities of the consumers in an e-business market as a network game in which the consumer policies reach a Nash equilibrium at a specific point of time. The presented model is not possible to be used in such an environment because consumers are acting selfishly, therefore, it is not possible to correlate their evaluations and trends. However, this model is more accurately approaching real market conditions in which consumers are usually forming cliques and trust groups, so as to maximize their cost/quality ratio.

On a different note, recommender systems over the Internet in general, have been thoroughly studied by various research groups ([17–20] and others). Specifically, collaborative filtering techniques are frequently used to support online recommender systems. Distinctive examples can be seen in the works of Herlocker et al. [21], Schaefer et al. [22] and Burke [23]. Most of the above use the Pearson correlation coefficient technique on mono-dimensional ratings, evaluating user preferences on simple products such as movies, rather than web services.
3. Service management lifecycle

The SR protocol is comprised of two interdependent processes “injected” to the SOA service management lifecycle [24], i.e. service publication, discovery, negotiation, selection, invocation and monitoring. The first process is the evaluation of the service QoS after a service is invoked and the second is the actual provision of recommendations during the service selection phase, based on the history that is created on the first phase. This section presents each step of the lifecycle, focusing on the differences that the SR protocol infuses. As it can be expected, the SR protocol proposes changes: (a) in the service selection phase, in which the SR mechanism is intervening proposing services and (b) by introducing a new phase, i.e. the service evaluation phase.

3.1. Service publication

Service publication is the initiating phase of the lifecycle. Providers may register their services to service repositories belonging to service brokers by submitting a description of their properties. In this way, the services become available to consumers when they request them following the publish/subscribe pattern. Even though several approaches have been proposed for modeling the service properties so they can be easily registered (with the most prominent probably being the Web Service Modeling Ontology (WSMO) [25]), no model has been widely adopted by the web service industry that captures the non-functional properties. This is mainly attributed to the fact that non-functional properties are application-oriented, making it practically impossible to enumerate and even quantify them. On the contrary, a model for capturing the functional properties has been well established during the past years: the Web Services Description Language (WSDL) [26].

3.2. Service discovery

Once the services are published to the service repository, the consumers can subscribe and retrieve those that meet their requirements. In order to achieve that, they submit their requirements in the form of a list of properties that they would desire the services to have.

The commonly used implementation of a service repository is the Universal Description Discovery and Integration (UDDI) registry [27]. Web services are indexed in UDDI registries as WSDL descriptions that model the interface and implementation of web services. The functional properties of each service can then be queried using the UDDI inquiring API and a Model, an XML description of interfaces used in UDDI, that can also capture notions of WSDL.

The result of this query from the consumer results in a list of WSDL links that correspond to the services that match the functional criteria. However, the consumer, needs to make a selection from this list based on a different set of criteria. These criteria are the non-functional or QoS requirements that in combination to the service cost, will lead to the final decision for which service to be invoked. This process is described in what follows.

3.3. Service selection

In order for the consumers to pick one service out of a multitude that meet their functional requirements, they also need to filter out the list according to their QoS requirements and will to pay for it. In this phase, the provider and the consumer negotiate over the QoS that the provider will deliver so as to end up with the most appropriate service.

In SOA, QoS is defined and measured through SLAs, thus, this phase of the lifecycle can incorporate the SLA negotiation subphase. SLA negotiation [28,29] is a process that enables a service provider and a service consumer to agree on the terms upon which the resources of the provider will be used by the consumer during the service invocation. The outcome of this process is an SLA document that is used for the monitoring and enforcement of the agreed terms.

An SLA document typically includes the following details: details about the two sides that enter to an agreement, the lifetime of the SLA during which the consumer can use the service, metrics that quantify and measure service quality, clauses for the case of a violation from any of the two sides and finally, billing information for the use of the service.

The part that essentially captures the notion of quality is the part of the quality metrics. This part explains what are the thresholds that the provider needs to maintain in order to guarantee a level of QoS and avoid violating the SLA. For example, such common metrics are the number of CPUs, the memory or the disk space volume that is going to be made available for the consumer through the service. The combination of these metrics (e.g. presented in a vector) provides a precise measurement of the QoS level.

All these details are captured in a structured document and the terms are negotiated between the two sides. The prevalent negotiation model is the “discreet offer” model [30]. The concept behind the discreet offer negotiation model is that the provider creates fixed offers which are captured in a type of pre-SLA document, called SLA template, which contains all the details that define an offer but the consumers’ details. As such, a service provider may create, for example, three SLA templates for a service, each one with different quality terms and billing information. In this way, the provider can scale the delivered quality in three discreet levels: “Gold”, “Silver” and “Bronze”.

If the consumers require to invoke the particular service, they have to select one of these SLA templates and inform the provider. The latter will then produce the complete SLA document, the details of which (such as the QoS parameters or SLA terms) are sent to the consumer. The SLA terms in the document are then used by another component that deals with monitoring the service usage and enforcing the SLA terms in case of SLA term violation.

Summarizing the above, in the phase of service selection, the list of services available that was produced in the previous phase, is presented to the consumer in the form of a list of SLA templates that are linked to these services. The consumers evaluate the offers and select one of the SLA templates. By doing so, they manage not only to select a specific service that meets their functional requirements but also the one that is closer to their non-functional requirements too.

This paper proposes an innovation at this point of the service lifecycle management. The proposal involves the use of an SR mechanism to assist the consumer in selecting the appropriate SLA template of a specific provider out of a group of alternatives, based on the collective experience of the consumers as will be discussed in greater detail in the following sections.

3.4. Service invocation (binding) and monitoring

When consumers conclude to one SLA template, they communicate that decision to the provider. Then, the latter creates an SLA document which contains all the necessary information for executing and monitoring the service as explained in the previous section. At the same time, the provider makes the necessary resources available for the agreed time period during which the SLA will remain active. The consumers can now invoke the service within that time period, using the (technical) details provided by the provider.
This process is also known as “service binding”. Now, the consumers have to ensure that they will not violate their part of the SLA. If they do, then the provider will immediately monitor that in its resources and put the SLA clauses into force.

On the other hand, the providers also need to abide by the terms of the SLA that concern their side. However, there is no objective way for the consumers to know that they did, since they either own or control the resources which are monitored, adding a bias in the measurements. The consumers can only rely on their experience on judging the quality of the service output (QoE).

A basic innovation of this paper is that it allows the consumers to quantify and aggregate their QoE against the providers’ QoS, once the SLA is terminated. This is explained in the next chapter.

3.5. Service evaluation

The service evaluation phase is an addition to the typical SOA service management lifecycle that is imposed by the use of our SR mechanism. As such, upon SLA termination, the consumers are asked by the service broker to rate the SLA terms by indicating their satisfaction on the delivery of the QoS levels from the provider. This allows for the creation of a history of ratings from multiple consumers that can be used to evaluate the providers’ reputation as well as to enable the recommendation of services (SLA templates) in the service selection phase.

4. SR protocol architecture

This section presents the architecture of the framework that can support the notion of the SR mechanism using the consumer evaluation ratings, i.e. implement the SR protocol. The framework is designed upon SOA principles and leverages on existing web service tools and standards, implementing all the necessary processes to accommodate the SR protocol.

The architecture focuses on two stages of the service management lifecycle as it was presented in the previous section: Service Selection and Service Evaluation. This section provides a more detailed view on how the SR mechanism is operating in the frame of a SOA in order to assist consumers in the service selection. For readability purposes, the description of the system is not emphasizing on the service publication and monitoring phases which are out of this paper’s context.

4.1. Assisted service selection

Fig. 1 presents the flow of logic in the architecture for selecting services.

In what follows, we briefly introduce the involved components. As a general comment, it is worth stating that all the components depicted in Fig. 1, except from the provider and consumer, belong to the service broker.

4.1.1. System components

Consumer: The consumer component is a client on the consumer side that provides interfaces for the expression of consumer requirements for service discovery and for handling the SLA negotiation process.

Service manager: The core component on the service broker that manages requests from consumers and contacts providers on behalf of them. The service manager communicates and orchestrates the operation of all service broker components in order to eventually provide a service end point to the consumer.

Service registry: A registry in which service descriptions are “pushed” by service providers, to make them available to consumers, allowing for service discovery. A UDDI registry is the most common such registry.

SLA template repository: A database (usually an object-oriented, XML database) that keeps the SLA templates for each service the providers register. For each service, there might be multiple SLA templates which essentially represent the providers’ distinct offers. The quality terms in the SLA templates of a service usually vary in the quality they provide and the associated cost and can be divided into three discreet offers: Gold, Silver and Bronze.

Provider: The service provider’s interface to its SLA negotiation mechanism.

Service recommender (SR): A component, given a list of SLA templates and a consumer ID, that provides a ranked list of these templates based on the particular consumer’s history of rating on the specific QoS parameters that pertain to these SLAs. Each time the lifetime of the SLA expires (i.e. the provider stops providing resource access to the consumer), the consumers are asked to evaluate the level of satisfaction from the quality provided and rate QoS parameters upon which they had agreed with the providers (see Fig. 2). These ratings are kept in the SR so as to form groups based on similarity. Whenever a consumer asks for a service that relates to a set of QoS parameters, the SR checks within the most similar consumer groups to identify those that have actually evaluated these QoS parameters. Analyzing the correlation of those groups, the system is able to propose a rank of the services available to the consumers.

4.2. Flow of logic

As Fig. 1 depicts, the flow starts when the consumer side makes a serviceQuery to the proposed service broker for a web service. It is assumed that the consumer is registered to the service broker’s components before making such a call. For readability purposes, this process is not described here. The query includes the functional properties (FunctionalProps) of the requested service, following the specifications of a UDDI query that will be then relayed to the UDDI registry.

In a typical publish-and-inquire scenario in UDDI, the provider publishes its business; registers a service under it; and defines a binding template with technical information on its web service. The binding template also holds reference to one or several tModels, which represent abstract interfaces implemented by the web service. The tModels might have been uniquely published by the provider, with information on the interfaces and URL references to the WSDL document. Therefore, the customer inquiry purpose is to find an implementation of a known interface. In other words, the client has a tModel ID and seeks binding templates referencing that tModel [31].

The query results are a serviceList, i.e. a list of web service IDs (bindingTemplate IDs) that the service manager can handle. In fact, the service manager transforms the serviceList to an XML query and relays it to the SLA template repository so as to retrieve the list of SLA templates (SLATemplateList) associated to these web services.

Consequently, the SLATemplateList is relayed to the SR component, along with the consumerID that uniquely defines the consumer that initiated the process. The SR is expected to reply with rankedSLATemplateList, i.e. the list of the SLA templates given, ranked based on similar consumers’ past evaluation on the potential services.

The operating principle behind the SR is based on the assumption that in a set of consumers with a potentially very large population, it is possible to recognize similarities in the behavior of distinct groups. This allows for the formation of subgroups that can be examined locally and recognize a generic, single strategy that characterizes the behavior of each one of them. In this way, it
becomes possible to predict a single consumer’s behavior by identifying the group’s strategy. This prediction can act as a recommendation for service selection and this principle is exploited by our SR mechanism.

The SR mechanism presents similarities with the e-Bay’s feedback system. A consumer is relying on other consumers’ opinion when they want to invoke a service. The main difference is that in this case, the consumer does not know the service’s rating in order to decide but instead the proposed system is collecting the ratings and uses them in order to calculate a personalized rating prediction on behalf of the consumer for every QoS parameter in the SLA.

Having our SR mechanism embedded in a service broker, rather than in the provider or the consumer side, is very important because it helps in avoiding a very basic shortcoming of feedback systems: that the design of the rating policy influences both the level of trust and efficiency of an e-Bay-like marketplace and therefore systems where ratings are kept secret serve to limit strategic ratings and are proved more effective in enhancing trust [32].

Continuing with the flow, the consumers use the rankedSLATemplateList in order to examine their options. They can either take into account or disregard the SR recommendation in order to select the SLA template of their preference. Once they select it, the Service Manager proposes it to the provider. If the provider accepts it, the provider component sends the final SLADocument. This document can then be used for service monitoring and SLA evaluation. Upon SLA termination, either because of a contract breach or because of SLA lifetime expiry, the Service Manager requests the consumer to evaluate the specific QoS parameters that were defined in the SLA (Fig. 2). This happens by asking the consumers to rate each of the QoS parameters, expressing the feeling of how the thresholds promised by the provider were maintained. The Ratings are then relayed to the SR component, which stores them and batch processes them in order to be able to produce even more efficient recommendations. The information that the SR needs to maintain, apart from the ratings, is the evaluated SLA terms, the consumer identity and the web service details (including the service provider). For simplifying this process, the Service Manager relays to the SR the whole SLA document incorporating the consumer ratings.

For that purpose, the authors created a slightly modified version of WS-Agreement [33], a specification used for modeling the service description as well as the QoS and billing terms (guarantee terms) of the SLA documents. WS-Agreement was chosen over other models such as WSLA [34] and SLAng [35] because it has been widely adopted and it is the only one that is still supported by an open community (Open Grid Forum). The modified version includes a new attribute “rating” under each “Guarantee Term” element of the WS-Agreement specification as depicted on Table 1.
Thus, the SR registry stores the whole WS-Agreement document along with the ratings. The benefits from extending WS-Agreement rather than using a far simpler data model for the SR mechanism are multiple: (a) WS-Agreement is a universal specification, widely adopted by many web service providers; (b) there are minimum processing requirements from the consumer client and are limited to the addition of the rating attributes to each SLA term; (c) the SR mechanism adopts the accounting scheme that is already used in the WS-Agreement (the Agreement-Initiator ID equals to a consumer ID), and; (d) the service details (e.g. the service provider ID) are already included in the document, so it only remains to map them to a UDDI record by simply cross-checking.

Having described the generic architecture and rationale of the SR mechanism, the document proceeds with presenting some details on the experiment the authors conducted using a reference implementation of this architecture.

5. Reference implementation

In the frame of this work, a reference implementation was built in order to experiment with the feasibility of the architecture and to finetune some technical details. The implementation included two major decisions: (a) what is the recommendation technique that is going to be used; and, (b) what is the service oriented environment that will provide tools for the service lifecycle management. These two decisions are discussed in more detail in the two following sections.

5.1. Recommendation technique

There are various ways to leverage on the collected ratings in order to effectively select services. Some techniques include memory-based collaborative filtering techniques and especially on correlation methods [36], vector similarity, inverse user frequency and case amplification [37]. In principle, these techniques model consumer ratings for a service as vectors. In order to examine whether the ratings of different consumers are close to each other, the abovementioned methods suggest to first calculate the correlation degree between them. Vector correlations have been extensively studied [38,39] and applied in various fields, like geophysics [40].

By finding the rating vectors that are highly correlated, one can assume that their owners (consumers) share a common view on the topic they rated. Therefore, it is possible to use the ratings of one consumer as predictions for the ratings of the other, in services that they have not commonly evaluated. In this experiment, the authors have correlated multi-dimensional vectors representing ratings for different SLA terms using collaborative filtering techniques.

This approach, i.e. correlation methods for generic recommender mechanisms, is widely used. In general, since the wide spread of the Internet and the blooming of the electronic commerce, collaborative filtering algorithms were used a lot for recommender systems. Correlation techniques however, stand first amongst the choices of the engineers for the development of decision support systems, as they are reliable, fast and easy to be implemented. In particular, [41] presents the efficiency of the Pearson product–moment correlation coefficient method in service selection decision support. The current paper, makes a step ahead of that work, presenting the architecture that is able to support the service selection decision support.

5.2. Experiment

The SR mechanism for the service selection process is implemented as a supporting mechanism on an actual tested service enablers service provisioning on a large scale. It concerns the phase of selection in the service management lifecycle and it only requires intervention in the way the UDDI registry is operating and a modification of the consumer client component so as to comply with the new interfaces.

In order to test our reference implementation, we realized a real-world scenario: the rendering of 3D frames to videos. In this scenario, the consumer is an animator, who creates a 3D model, usually in the form of a wireframe and submits it to a renderer. The renderer, is a software that uses the wireframe as a set of instructions so as to turn the 3D model into 2D images with 3D photorealistic effects adding lights, textures, shadows, etc. This experiment extends the initial experiment explained in [41] by increasing the number of consumers and SLA offers available and emphasizing on the user experience point of view, i.e. how the recommended service actually satisfies the consumer.

5.2.1. Infrastructure

Extending the existing infrastructure [41,42] for providing 3D rendering as a service, we set up an experiment to measure the impact in the user experience of the SR protocol and mechanism.

The set up included 5 rendering services with a web service interface being hosted to a GRIA [43,44] middleware. Each rendering service uses a different rendering software and have equal access rights to the resources. Thus, the rendering services are acting as distinct service providers.

The service broker configuration comprised an instance of the architecture described in Section 4. Even though GRIA was selected as the reference architecture, the general concept is middleware-independent and could be implemented in various SOA environments. The selection of GRIA was a convenient solution because it implements the discreet offer SLA negotiation model, accommodating all the necessary components for that purpose. The SR component described above, is built and deployed as an extra component, coupled to a UDDI registry instance that is called during the service discovery and the selection process.

5.2.2. Experimental setup

The SLA templates that were created for the purpose of the experiment included fixed service provider details that were...
pointing to web services implementing the 3D image rendering algorithm. Also, their lifetime was set to one job execution (in contrast with setting a time period) with no constraints on the consumer side, so as to allow the animators to execute their jobs without worrying about SLA violations. GRIA was set so as to allow the execution to continue, even in the case of SLA violation, as it is the consumer who should be able to evaluate that and not the providers or the broker.

Regarding the quality-related terms in the SLA, in order to simplify testing and evaluation and make it easier for the animators to evaluate the QoS, we avoided using resource-level terms in our SLAs and emphasized on application-level terms. For example, we did not include the number of CPUs or the memory volume that the animator is allowed to use, allowing the resource scheduler of GRIA to allocate them based on demand. Instead, the animators identified a number of quality metrics that would steer them from one rendering service to another. These metrics were adopted as QoS parameters in the SLA and they are the following: response time, availability, frame errors rate and price. The reason for the selection of these terms, is because they were requested from real consumers (animators) and they can be measured by both sides. Moreover, they can be controlled from the provider side with appropriate mechanisms. However, it is important to mention that in practice, the terms are usually resource-level, especially in generic service providers acting as service platforms (e.g. cloud providers).

The terms are explained as such:
- **Response time**: The time it takes for the service to render a video and return it to the consumer.
- **Availability**: The times that the service was available when it was requested against the times it was not.
- **Frame errors rate**: Depending on the complexity of the 3D model, some renderers may return videos with missed/over-written frames or errors in the audio.
- **Price**: The price of the services for the consumer. Even though this is not a QoS parameter, this is used as an evaluation criterion from the animators, combining quality provided with price (in case price is not following the market's trends). In reference to the statement in Section 1 about the use of the terms, “QoS parameter” and “SLA term” suffices to say that price, is an SLA term but not a QoS parameter, but it will be dealt as such in this document.

Based on these, for each one of five service providers defined in the current experiments, three SLA templates were created with their QoS parameters varying so that they could be categorized to Gold, Silver and Bronze offers. Therefore, 5 × 3 = 15 service instances were created. Each one of seven (7) animators was requested to use and evaluate the service instances. They were allowed to use any input they desired with the limitation that they will use the same input for each type of SLA template (Bronze, Silver, Gold). Furthermore, they were requested that the produced video would not be longer than 3 min. The reason that this “rule-of-thumb” limitation was imposed was because we needed to ensure that the infrastructure would not be overloaded by the calls of animators introducing problems that could be out of the providers’ control.

After the invocation of each service, a web form (built-in to the GRIA client) was presented to the animators asking them to rate each of the QoS parameters agreed in the SLA in a scale of 1–5 (1 = not satisfied and 5 = fully satisfied). This scale is equivalent to a typical five-level Likert scale. By completing this step, the SR component created an initial dataset with ratings that helped us deal with the cold start problem.

In the sequel, one (1) more service animator was requested to use and evaluate, under the same conditions as the previous, 10 of the service instances (only Bronze and Silver offers) which were available. Then, the animator was asked to select one of the remaining 5 Gold SLA templates based on the experience he collected from using the same services but under different SLAs. The concept behind this step was to allow the animator to form an opinion about each service provider and then use this opinion to evaluate them in a more demanding case.

After running the service instance, the animator was asked to evaluate it. Then the selection was compared to the one that the SR was producing using the Pearson product-moment correlation coefficients [41]. If the selected service instance did not match the recommended one, the animator was asked to invoke the latter, using the same configuration (input file and settings) and then evaluate it.

This was repeated 7 times, each time, placing another one from the 7 animators to the position to select one of the 5 service instances based on their experience. Each iteration required the creation of new SLA offers and the re-evaluation from the rest of the animators of the new 15 SLA templates. With the response time of the submitted jobs ranging from 1 to 6 min, and an average of 3 min, each animator spent about (6 × 15 + 10) × 3 = 300 = 5 h in the experiment in a timespan of 7 days (1 iteration per day). Fig. 3 graphically illustrates the experiment scenario during this first phase.

The results showed that in 4 out of 7 cases, the animator selected the same service provider as the one that the SR was proposing, whereas in 2 out of the 3 remaining cases, the animators’ evaluations to the recommended service instance were closer to the predicted ones (from the SR) rather than the ratings in their initial selection.

This outcome indicates that the experience of others is useful in modeling the experience of a service consumer. From a user experience point of view, the proposed system is beneficial in understanding the consumers’ preferences. However, this does not imply that the consumer selection is the optimum, thus, we still need to show that the SR outcome is indeed an accurate prediction of the best option for the consumer, i.e. that the experience of similar consumers can actually guide for an optimized match between QoE and QoS.

In fact, a closer examination of the results provides a proof of concept. Table 2, presents the ratings of animator 1 for Bronze and
Table 2
Actual and predicted ratings for animator 1.

<table>
<thead>
<tr>
<th>Animator</th>
<th>SLAT1</th>
<th>SLAT2</th>
<th>SLAT3</th>
<th>SLAT4</th>
<th>SLAT5</th>
<th>SLAT1</th>
<th>SLAT2</th>
<th>SLAT3</th>
<th>SLAT4</th>
<th>SLAT5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bronze SLA templates</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Silver SLA templates</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Gold SLA templates</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

Table 3
Correlation matrix for animator 1. Cor (1, x) indicates the correlation coefficient between animator 1 and animator X. There is one value for each QoS parameter. The values range from [-1, 1], with 0 implying no correlation and 1 and -1 implying high correlation.

<table>
<thead>
<tr>
<th>Animator</th>
<th>QoS1</th>
<th>QoS2</th>
<th>QoS3</th>
<th>QoS4</th>
</tr>
</thead>
<tbody>
<tr>
<td>QoS1</td>
<td>0.707214</td>
<td>0.596371</td>
<td>0.342097</td>
<td>0.50703</td>
</tr>
<tr>
<td>QoS2</td>
<td>0.297398</td>
<td>0.210172</td>
<td>0.278793</td>
<td>0.874886</td>
</tr>
<tr>
<td>QoS3</td>
<td>0.393732</td>
<td>0.218017</td>
<td>0.117871</td>
<td>0.30567</td>
</tr>
<tr>
<td>QoS4</td>
<td>0.118937</td>
<td>-0.38059</td>
<td>-0.30893</td>
<td>0.786246</td>
</tr>
</tbody>
</table>

Silver SLAs and the respective predicted values for the Gold SLAs from the SR mechanism. Commonly numbered SLAs are linked to the same service provider. One would expect that the animators' previous experience with the provider should depict his expected behavior when he comes to select a service instance. Indeed, this is the case in most of the situations, with the predicted ratings being close to the ones provided. An interesting case, however, is portrayed with the highlighted cell in which even though the consumer has rated the provider highly for the QoS parameter (4 and 5), the predicted value is significantly low (2.599404).

This ostensible failure of the system can be explained if we also observe Table 3 which presents the correlation between animator 1 and the rest for each of the QoS parameters. For the QoS parameter 3 in which the failure appeared, the consumer who affects animator 1 the most, is animator 7. The latter rated the particular service instance (SLA1) for the Gold SLA and for the specific QoS parameter (QoS3), really low, with a rate of 1. Therefore, his opinion is weighted more than the rest in estimating the rating of animator 1, leading to a low predicted value. In fact, the actual value that animator 1 gave when executing the service instance, was 3 out of 5, a value closer to the prediction rather than his own average rating.

Furthermore, one would expect that ratings should improve for the same service as we are moving from Bronze, to Silver, to Gold SLA templates. However, this is not the case, because the templates in our experimental setup indicate only changes in the consumer requirements and not changes in the provider performance. This is due to the fact that the monitoring and the SLA enforcement mechanisms of the GRIA middleware were practically de-activated so as to indeed provide a quality differentiation between services that it is not affected by resources being allocated to avoid SLA violations.

Having analyzed that, we proceeded with investigating whether the recommended selection was in fact, the best option (Phase B, Fig. 4). In order to examine that, the animator that acted as the selector each time, was also asked to execute the remaining service instances (3 or 4 depending on whether their initial selection matched the recommended) under the same conditions. Therefore, 35 vectors of ratings were produced (5 for each Gold SLA Template of the 7 animators) to be compared against the 35 estimated vectors from the SR mechanism.

The comparison results for the first animator, showed that in 4 cases out of the 35, the mean squared error (MSE) exceeded the threshold of 0.05. In fact, for the first animator, the MSE for the 35 service instances follows the pattern depicted in Fig. 5. Together with the rest of the 6 iterations of that phase of the experiment, we captured a ~11.4% of the occasions in which the MSE exceeded the value 0.05. However, the results of this experiment serve mainly as an evaluation to the application of the selected collaborative technique in QoS parameters' ratings. A different selection of algorithm could potentially yield different results. In fact, in [41], this is investigated in depth, with a mean relative error reaching the value of 7.3% for a dataset of 500 users, for 4 services and 4 parameters.

On a side note, examining the experiment results from an architecture point of view, it is important to mention that the SR mechanism has infused an overhead in the response time of a UDDI registry implementation. Indeed, the computation of Pearson correlation coefficients for l customers and n services available has a time complexity of O(n + P) [45]. Given that we calculate those coefficients for m quality parameters, the total complexity of the correlation, and thus the recommendation equals to O(m∗(n+P)). The prediction algorithm complexity seems to be deterrent for the application of such a mechanism, however, the calculations take place offline, greatly improving the system's performance.
of the consumer will have to pay for each service he/she consumed, therefore there should be little to no incentive for conducting such an activity. Another open issue is the rating maturity. As time goes by, ratings’ importance may be decreased having less impact to the overall rating over time. This is considered as future work, that is, to extend the SR mechanism so as to include weighted rating vectors and a “timestamp”. For each rating vector, the bigger the difference between current datetime and the timestamp, the greater the decrease of the weight of the ratings will be.

As a final remark, it needs to be stressed that the proposed SR mechanism is operating as a decision support system and under no circumstances does it become a replacement to human decision.

Acknowledgment

We would like to thank STEFI S.A.(http://www.stefi.gr) for the invaluable help that they provided us in conducting the experiments.

References


Mean Squared Error

Fig. 5. Mean squared error for the 35 service instances.

6. Conclusions and discussion

This paper presents a Service Recommender mechanism that enables the provision of better guarantees that the selected service will comply with the quality requirements of the consumer. Its implementation requires the extension of the typical service selection process in the service lifecycle management and the introduction of a new phase: that of the service evaluation. These changes are captured in what we call the SR protocol.

The SR mechanism uses the ratings of the consumer on the SLA terms of the service they have used so as to build a profile of the consumer. The profiles indicate the level of consumer satisfaction on specific SLA terms as well as in the service as a whole and can be exploited in order to identify similarities in the way that consumers perceive quality based on their experience. The fact that some profiles are similar is a fact that can be used so as to recommend services to one another.

The implementation of the SR mechanism is also applicable for “open” service-oriented environments where the service hosting is not necessarily centrally managed by a middleware. It is implemented as an extension of the UDDI registry and uses a slightly modified version of the WS-Agreement specification to model the SLAs and capture the negotiation process including the evaluation phase. A reference implementation is created using a real world application scenario: 3D Rendering services are provided to animators who evaluate their instances. The SR mechanism implements a collaborative filtering technique with positive results in predicting the service instance that the animators would select as the one that will satisfy their requirements.

From this experience with the animators, we had to abstract the SLA terms to high level terms that the animators understand. That is, they did not feel comfortable evaluating terms such as number of CPUs or allocated memory. However, it is known that in cloud infrastructures which become the hosts for multiple services, it is not possible for each application service to accommodate their (high level) performance model and therefore, only low level terms are used for that purpose. Having said that, it is the sincere opinion of the authors that one cannot know if the consumers will be able and willing to rate SLA terms (especially of cloud SLAs). However, the authors believe (and believe that this paper proves) that by doing that, it will be for their own benefit. So, there is motivation, which is important, but in fact life has proven that there are always more “leechers” than “seeders”.

A case of special interest for the SR protocol is when a malicious consumer wants to promote or harm a service provider by swarming the SR mechanism with fixed ratings regarding the provider. However, a large number of consumers and services could reduce the contribution of an individual to the system. Also, the malicious vectors could differentiate from the norm

of honest ratings leading to small rating correlations, rendering this contribution insignificant. Finally, the evaluation phase comes after the service invocation which implies that the malicious consumer will have to pay for each service he/she consumed, therefore there should be little to no incentive for conducting such an activity.

Fig. 5. Mean squared error for the 35 service instances.