# Integrating Semantic Web Services Ranking Mechanisms Using a Common Preference Model

José María García[a,*], Martin Junghans[b], David Ruiz[a], Sudhir Agarwal[b], Antonio Ruiz-Cortés[a]

[a] *University of Seville, Spain*
[b] *Karlsruhe Institute of Technology, Germany*

**Abstract**

Service ranking has been long-acknowledged to play a fundamental role in helping users to select the best offerings among services retrieved from a search request. There exist many ranking mechanisms, each one providing *ad hoc* preference models that offer different levels of expressiveness. Consequently, applying a single mechanism to a particular scenario constrains the user to define preferences based on that mechanism's facilities. Furthermore, a more flexible solution that uses several independent mechanisms will face interoperability issues because of the differences between preference models provided by each ranking mechanism. In order to overcome these issues, we propose a Preference-based Universal Ranking Integration (PURI) framework that enables the combination of several ranking mechanisms using a common, holistic preference model. Using PURI, different ranking mechanisms are seamlessly and transparently integrated, offering a single façade to define preferences using highly expressive facilities that are not only decoupled from the concrete mechanisms that perform the ranking process, but also allow to exploit synergies from the combination of integrated mechanisms. We also thoroughly present a particular application scenario in the SOA4All EU project and evaluate the benefits and applicability of PURI in further domains.

*Keywords:* Semantic Web Services, Service Retrieval, Service Ranking, Systems Integration, Preference Models

## 1. Introduction

Service Oriented Computing [1] has brought a paradigm shift where complex applications are now built upon software components – services – that can be reached over the Internet. Although most services are exploited within large corporations nowadays, the number of publicly available services is envisioned to increase in the future [2]. Precisely, the combination of service oriented computing and semantic technologies, which is known as Semantic Web Services (SWS), provides tools to properly manage that amount of available knowledge [3]. Service retrieval is a key process in this vision, where SWS offerings are retrieved from repositories with respect to a concrete user request [4].

In this current scenario, where service repositories are being actively developed [5, 6] in order to cope with the growth in the number of services, ranking mechanisms have been long-acknowledged to be required for the selection of the best retrieved offerings with respect to certain user-defined preferences. Although there exist several approaches on SWS ranking, there is a lack of a generic and flexible preference model that offers a comprehensive collection of facilities to define user preferences related to the services to be selected. In turn, each ranking mechanism provides an *ad hoc* preference model that constrains the expressiveness of user preferences, which are tightly coupled with the underlying ranking mechanism applied.

However, in order to allow the expression of complex preferences for end users, they should be provided with more flexibility to define preferences, so a service retrieval and ranking system may integrate several ranking mechanisms, providing a higher number of facilities to state user preferences. Nevertheless, interoperability issues between preference models may appear, as they cannot be easily combined, and potential synergies may remain unexploited.

In this article, we present a preference-based ranking integration framework named PURI, which provides a solution to previously identified issues. Our integrated ranking solution gives both developers and end users control on how the ranking process should be performed, because user-specified preferences can combine every facility that the integrated ranking mechanisms provide, seamlessly integrating them and making the most of each ranking approach, according to the user's particular needs. The main contributions of our proposed integrated ranking solution

*Corresponding author. Address: ETS Ingeniería Informática. Av. Reina Mercedes, s/n. 41012 Sevilla (Spain). Tel.: (+34) 9545 59814. Fax: (+34) 9545 57139

*Email addresses:* josemgarcia@us.es (José María García), martin.junghans@kit.edu (Martin Junghans), druiz@us.es (David Ruiz), sudhir.agarwal@kit.edu (Sudhir Agarwal), aruiz@us.es (Antonio Ruiz-Cortés)

*URL:* http://www.isa.us.es/josemaria.garcia (José María García)

can be summarized in the following four key features:

1. Our solution provides a **simple** entry point for the end user to define preferences for service ranking using a common preference model based on an upper ontology that is sufficiently expressive to **combine** more specific preference models from concrete ranking mechanisms.

2. The proposed framework provides a high **expressiveness** and **flexibility** because it allows developers to integrate any available ranking mechanisms, provided that they adapt their models to be defined in terms of our common preference model.

3. The resulting service retrieval and ranking system offers a **lightweight** and **integrated** solution, as it does not add a noticeable penalty on the performance of the integrated ranking mechanisms, whose results are combined without altering their accuracy.

4. Our proposal has been validated in different scenarios, including a use case within the SOA4All EU FP7 project[1] that is presented in this article, along with a discussion on the **applicability** of PURI to several domains. The discussed use case shows a successful application of PURI that provides a holistic integrated ranking that combines three different ranking approaches from SOA4All.

The rest of this article is organized as follows. Firstly, in Section 2 we identify the challenges in SWS ranking to motivate our proposal. In Section 3 related work on service retrieval approaches and preference models is discussed. Then, our SWS retrieval and integrated ranking proposal is presented in Section 4. A concrete application of this proposal is discussed in Section 5, contextualizing it in the SOA4All EU FP7 project, which serves the purpose of validating our presented solution. Finally, Section 6 enumerates the conclusions of this work.

## 2. Challenges in Semantic Web Service Ranking

In the following, we motivate our proposal by presenting an example that allows the identification of some challenges in the SWS ranking field that need to be addressed. Let a sample user request be informally defined as:

> *"I want to look for services that can send some SMS messages to my friends, preferring the cheaper ones though the number of messages they can deliver at the same invocation should also be fair enough."*

In this request, the user not only states the desired functionality (to send SMS messages), but the preferences about some service properties. Concretely, the user is looking for services with the *lowest* possible price per message, but also considering that the number of simultaneous messages allowed to send in a single request should be *fair*. After retrieving the compliant services with respect to functionality, a ranking process has to be performed in order to rank the result list in terms of the user preferences, simplifying the selection of the best service for the user. These preferences have to be expressed in terms of some model of a particular ranking mechanism that has to be chosen in order to perform that process.

On the one hand, the lowest price preference can be modeled using ranking approaches that allow definition of the desired tendency of a given attribute, usually a non-functional property (NFP) of a service, such as price per message in the example. Thus, retrieved services should be ranked according to the price value in ascending order. NFP-based simple ordering proposals [7] or multi-criteria ranking approaches [8, 9] that offer simpler preference modeling and more efficient ranking mechanisms can be directly applied to evaluate this kind of preference. In turn, more expressive approaches, such as those based on utility functions [10, 11] or fuzzy logics [12, 13], are less suitable because they exhibit a lower ranking performance, in general, though they provide more complex preference modeling facilities.

On the other hand, in order to model the preference on the number of simultaneously sent messages, we need to define what is considered to be a *fair* number for the user. For instance, a user may specify that a fair number of messages is a value between 3 and 5. In this case, a desired tendency definition (as in the price preference discussed before) cannot be used because services are preferred if the NFP value is around the desired interval, instead of preferring a minimum or maximum value. In turn, a fuzzy based ranking mechanism offers means to express this more complex preference, provided that a fuzzy membership function defines to which extent an NFP value can be considered to be *fair*. Furthermore, mechanisms based on utility functions can be also applied, as a fuzzy membership function can be considered as a particular case of a utility function.

Desirably, both preferences should be defined and combined using a unique preference model, so a single ranking mechanism should be chosen to help the user to select the best service [14]. However, if a multi-criteria, tendency based ranking mechanism is chosen, the second preference on the number of messages cannot be properly described. In turn, a utility function or fuzzy based ranking approach allows to define both preferences, however it may also imply more modeling effort to express the price preference as well as a lower global performance compared to a tendency based approach. From the user's perspective, it would be more valuable if they could flexibly choose between expressiveness and performance for each preference description.

[1] `http://www.soa4all.eu`

Consequently, in order to rank services according to a combination of both preferences, different ranking approaches could be used correspondingly for price and number of messages preferences (*e.g.* a tendency based and a fuzzy logic based approach), although the user then has to define each part of the preference using a different model. However, as preference models cannot be directly combined at the conceptual level, results from each ranking mechanism have to be manually analyzed so that the user can come up with a global rank. In consequence, there exists an actual challenge on **how to combine several preference models** so that ranking results obtained from corresponding ranking mechanisms can be automatically integrated, transparently returning a global rank to the user. This broad challenge can be divided into three more specific challenges or issues that should be addressed in order to provide a complete solution:

(C1) **Expressiveness when defining preferences.** When using a particular ranking mechanism, users are constrained to define their preferences using the specific model offered by that mechanism only. In the previous example, if users choose one particular mechanism to rank, they will be able to define their preferences using either NFP tendencies or fuzzy rules. A more expressive model is necessary to define preferences using different facilities offered by existing mechanisms.

(C2) **Interoperability of preference models.** As preference models and associated ontologies are mostly coupled with their corresponding ranking formalisms, each preference model provides different facilities to define preference terms that cannot be combined in general. In other words, there are interoperability issues between preference models, mainly because there is no common model to define and combine preferences.

(C3) **Integration of ranking mechanisms.** The interoperability issues at the conceptual level are also reflected at the implementation level. In order to combine preferences from different ranking mechanisms, they have to be integrated in a seamless and efficient way so that a single entry point for the ranking process is presented to the user.

PURI framework accounts for these issues, providing a solution that fulfills the identified challenges, as discussed in Section 4.

## 3. Related work

In this section we review existing solutions in SWS ranking with respect to their provided degree of expressiveness (C1), interoperability (C2), and integrability (C3) in order to analyze how they deal with our identified challenges and to what extent they present the associated issues. Table 1 sums up our review, comparing the analyzed proposals. Works are presented in chronological order, showing the evolution with respect to the addressing of the identified challenges. Basically, earlier proposals are characterized by low expressiveness and low integrability, while later ones provide more support for those issues. Interoperability remains similar between analyzed works, as they all use semantic approaches for service ranking, taking advantage of the inherently better interoperability provided by ontologies and semantic definitions.

In particular, expressiveness (C1) is measured amongst proposals as the amount of facilities their associated preference models offer, and the complexity of preferences that can be described with them. A *low* expressiveness in Table 1 means that ranking mechanisms do not provide a separate model to let users define their own preferences, as in early works [15, 16, 17], which compute a ranking based on matching or similarity degrees between service offers and requests described using a SWS framework such as OWL-S or WSMO. Note that there is also a more recent proposal [5] that does not provide a preference model, because it is primarily aimed at solving interoperability issues between SWS definitions.

Precisely, SWS definitions are extended by several proposals in order to provide more facilities to express preferences. [18] presents different extensions that allow for the definition of NFP properties and measurements, that can be used by its ranking mechanisms. However, as the user cannot control how NFP values are used to rank SWSs (*i.e.* they cannot express preferences with respect to those NFPs), we assign to this proposal a low degree of expressiveness. Similarly, other proposals present fully-fledged NFP models, but low expressiveness for preferences, because they rely on externally defined preference models instantiated by the user, as in [19, 7].

In turn, [20] allows to define relative weights directly in WSMO user requests, so that we consider they provide a *medium* expressiveness, offering some simple facilities to define preferences. Furthermore, [9] and more recently [8] propose extensions to annotate WSMO user requests with both relative weights and NFP tendencies, allowing the user to express if a concrete NFP is preferred to yield the lowest or the highest possible value. A *high*er expressiveness is provided in [11], as the authors extended [8] by facilities to express preferences as utility functions.

The flexibility and expressiveness of utility functions are also provided by some proposals [22, 10], though this kind of preference can be considered to be too complex to be defined by a regular user [25]. Thus, other approaches offer more user-friendly facilities to combine these preferences, define NFP-based policies [23, 24], and to express qualitative preferences [21].

Concerning interoperability (C2), early approaches [15, 16, 17] do not provide a separate semantic model to define user preferences, so they cannot be combined and reused with models proposed by the rest of the proposals, producing a *low* interoperability. Furthermore, the preference

Table 1: Summary of the related works and their capabilities with respect to the challenges.

| Proposal | C1 | C2 | C3 |
|---|---|---|---|
| Early approaches (2003-04) [15, 16, 17] | Low | Low | Low |
| Zhou *et al.* (2004) [18] | Low | Medium | Low |
| Maximilien & Singh (2004) [19] | Low | Medium | Low |
| Dobson *et al.* (2005) [7] | Low | Medium | Low |
| Vu *et al.* (2006) [20] | Medium | Medium | Medium |
| Wang *et al.* (2006) [9] | Medium | Medium | Low |
| Siberski *et al.* (2006) [21] | High | Low | Medium |
| Kritikos *et al.* (2006) [22] | High | Medium | Low |
| Toma *et al.* (2007) [8] | Medium | Medium | Low |
| Lamparter *et al.* (2007) [10] | High | Medium | Medium |
| García *et al.* (2008) [11] | High | Medium | Low |
| Carenini *et al.* (2008) [23] | High | Medium | High |
| Palmonari *et al.* (2009) [24] | High | Medium | Medium |
| Pedrinaci *et al.* (2010) [5] | Low | High | Medium |

model in [21] described as a SPARQL extension also offers a low degree of interoperability.

In turn, most analyzed approaches have a *medium* interoperability as they semantically define their models using separate ontologies or extensions to existing ones. However, some transformations and mappings [26] are needed in order to fully combine different models in a single preference definition, as discussed in Section 2. Nevertheless, [5] provides a higher interoperability due to the fact that its SWS definition model is based on Linked Data principles, enabling easy reuse and combination of ontologies [27].

With respect to integrability (C3), we measure how difficult it is to integrate each ranking mechanism with other ranking implementations. Proprietary and difficultly extensible proposals are rated to have a *low* integrability, because their underlying ranking mechanisms are very different, particularized for each retrieval scenario they try to solve, and their APIs are not extensively documented. Proposals that provide components that are easier to integrate, because they offer hybrid architectures [20, 24] or SPARQL endpoints [21, 10, 5] are rated with *medium* degree of integrability. Finally, [23] is *high*ly integrable because it defines a component-based, hybrid architecture that allows the user to customize the service retrieval and ranking processes.

## 4. Our proposal

As discussed above, there are several interoperability issues between ranking mechanisms that constrain the flexibility of semantic service ranking systems, which tend to be designed eclectically, *i.e.* allowing the application of a single ranking mechanism that provides a limited set of facilities to define preferences. In the following we present an integrated solution to semantic service retrieval and ranking that offers a common, highly expressive preference model that allows for the integration of several ranking mechanisms into the service retrieval and ranking system.

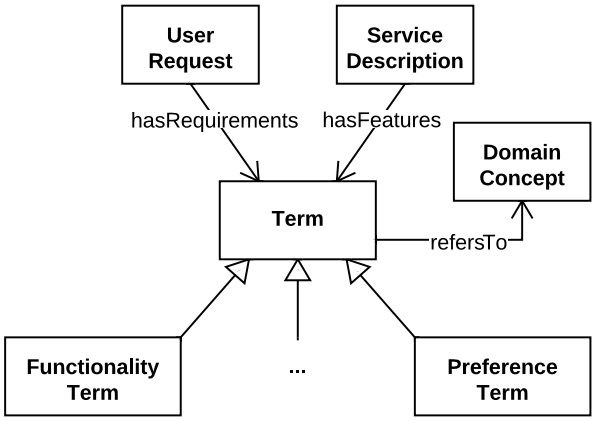


Figure 1: User requests and service descriptions upper ontology.

### *4.1. System Architecture Overview*

In order to integrate both service retrieval and ranking mechanisms, we first need to define what we consider as user requests and service descriptions. Figure 1 presents an upper ontology that simply interprets both user requests and service descriptions as a set of terms that are related to a certain concept from a domain ontology. Each term may specify different aspects from service descriptions and user requests, such as functionality and user preferences. Thus, a functionality term of a service description may specify its inputs, outputs, pre-conditions and effects of a service, *e.g.* described by OWL-S [28] or WSMO [29]. In turn, a user request will specify similar terms to define its desired functionality. Therefore, the related domain concepts consist on the types of input and output parameters, and those involved in pre-conditions and effects.

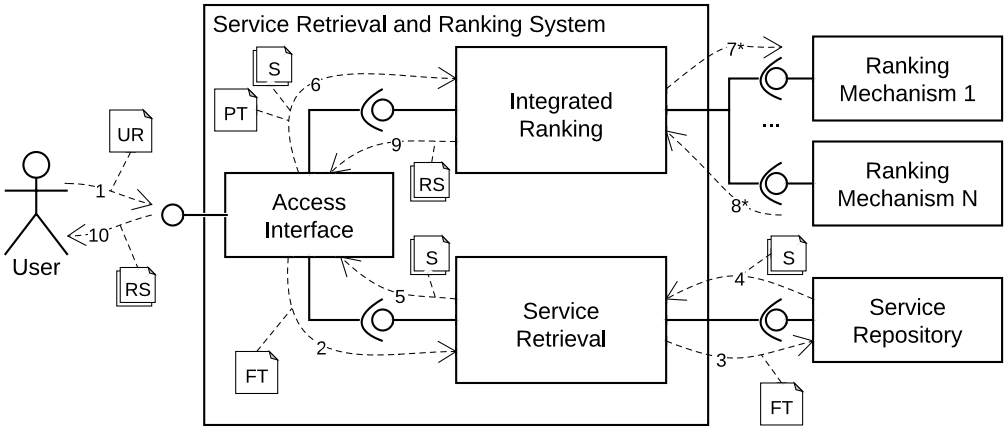


Figure 2: Integrated service retrieval and ranking system architecture.

Furthermore, preference terms pertaining to a user request give information about how to rank the retrieved services to obtain the best one. Those preferences usually refer to domain concepts that represent NFP, whose actual values are also included within NFP terms individuals of service descriptions. In the rest of this section, service descriptions and user requests are described in more detail, presenting the different involved terms.

Applying our interpretation of a user request to the service retrieval and ranking scenario, we can separate terms depending on their belonging class, so that functionality terms can be forwarded to the service retrieval component, while preference terms will be solely used by the integrated ranking [30]. Figure 2 showcases the hybrid architecture of our proposed service retrieval and ranking system.

Firstly, a user sends a request (UR) to the system (Step 1). The access interface analyzes the request and differentiates functionality (FT) and preference terms (PT) depending on their belonging class. Then, the service retrieval component searches in a service repository in order to retrieve matching services (S) with respect to functionality-related terms (Steps 2, 3, and 4). The access interface component passes this set of retrieved services (Step 5) to the preference-based integrated ranking component (Step 6) that analyzes the previously identified preference terms and the corresponding ranking mechanisms are executed (Step 7) and combined (Step 8). Finally, the combined results from the integrated ranking mechanisms are returned as the response to the user (Steps 9 and 10), who obtains a ranking of the services (RS) that provide the desired functionality ordered according to the specified preferences. The following subsections discuss each main component in detail, as the access interface component simply routes relevant terms from the user request to both retrieval and ranking components.

For instance, Listing 1 shows a simplified user request in RDF Turtle syntax [31] for a SMS messaging service to illustrate the workflow of our proposed architecture. Thus, the access interface analyzes the `ex:request`, routing the functionality term `ex:sendSMS` and its descriptive statements to the retrieval component as they define the required functionality of a desired service with an operation to send an SMS message[2]. Then, that component retrieves a set of SMS messaging services, as requested. After that, `ex:userPreference` is further analyzed by the integrated ranking component, ranking the previous set of services with respect to user preferences. Section 4.3 explains in detail the analysis of preferences for the integrated ranking process.

### *4.2. Service Retrieval based upon Functional and Non-Functional Properties*

Service descriptions consist of functional and non-functional terms in order to express respective service properties. Each service property is represented by a key-value pair. Retrieval focuses on the matchmaking of the service functionality described by property term with a complex value structure. Services not only provide information but may also cause changes in the information state of the service provider. Therefore, we consider a state-based model of the behavior. A service execution is a sequence of states and state transitions that denote information change. The behavior of atomic services is described by the functionality terms comprising input and output parameters, pre-condition, and effects. Services with an atomic behavior are distinguished by only two possible interactions: the consumption of input parameters at service invocation and the return of outputs at service completion. In addition, we allow for a classification of services in functionality classes. Formal definition of classes allow for automatic and correct classification of services, which can be exploited as an indexing structure in order to speed up the retrieval process [32].

[2]The sample `telco` ontology that models several concepts of the SMS messaging domain can be found at `http://www.isa.us.es/soa4all-integrated-ranking/downloads/sms.owl`

Listing 1: An excerpt of a sample user request.

```
@prefix puri: <http://www.isa.us.es/soa4all-integrated-ranking/onto#> .
@prefix posm: <http://www.wsmo.org/ns/posm/0.1#> .
@prefix telco: <http://www.semanticweb.org/ontologies/SMS.owl#> .
@prefix ex: <http://example.org/onto#> .

ex:request a puri:UserRequest ;
   puri:hasRequirements ex:sendSMS , ex:userPreference .
ex:sendSMS a posm:Operation , puri:FunctionalityTerm ;
   puri:refersTo telco:Message .
   #... further statements specifying service inputs, outputs, and other information
ex:userPreference rdf:type puri:PreferenceTerm .
   #... further statements specifying the user preference (see Listing 3)
```

*Service Descriptions.* For the description of services we use ontologies based on the logical formalism of description logics [33]. This allows us to use existing ontology reasoners to reason about Web service properties while not forcing a global set of property names. The vocabulary used for the description of service properties is inherited from WSMO-Lite [34], SA-WSDL [35], and POSM[3]. In addition, a domain ontology that provides the means to model service resources in the description shall be given.

An ontology individual w of class posm:Service represents a service $w$ in the respective service description ontology. The service $w$ has operations with each of it described by input and output parameters, as well as pre-conditions and effects. POSM provides the concept of message parts in order to model the parameters in simple part-whole relations[4]. Pre-condition and effects are logical axioms describing the start and the end state of a service execution respectively. In our implementation, the domain ontology is modeled in the WSML-Core ontology language. Logical axioms are expressed in the WSML-Flight language [36]. Furthermore, services and operations can have multiple NFPs modeled as ontology object or data type properties. Listing 2 shows an excerpt of a service description example of a SMS messaging service (prefixes are the same as in Listing 1), where `posm:hasOperation` is a sub-property of `puri:hasFeatures`. NFPs can be added as additional featured terms that define, for instance, the price and the number of messages sent simultaneously (see Listing 4).

Listing 2: Simplified description of a sample SMS service.

```
ex:smsService1
   a posm:Service , puri:ServiceDescription ;
   posm:hasOperation ex:sendBatchSMS .
ex:sendBatchSMS
   a posm:Operation , puri:FunctionalityTerm;
   puri:refersTo telco:Message .
```

[3]The Procedure-Oriented Service Model (POSM) ontology is a lightweight approach to the structural description of procedure-oriented Web services, compatible with WSMO-Lite annotation. See `http://www.wsmo.org/ns/posm/0.1` for details.

[4]See `http://www.w3.org/2001/sw/BestPractices/OEP/SimplePartWhole` for details on partonomic modeling

*Service Retrieval.* The identification of desired services from a repository that satisfy functional and non-functional constraints is a filtering step that typically precedes service ranking. A service request is expressed in a constraint specification formalism that also reflects the property-based structure of service descriptions, but differs as it specifies constraints over desired property value sets (as opposed to service descriptions that specify concrete property values) [37]. Property names are defined in ontologies. The chosen ontology language may support the specification of sets and ranges of values, otherwise they can be modeled by individuals and classes, too. Thus, a semantics of a service request corresponds to a set of services such that each service in this set fulfills the constraints specified in the request.

The matchmaker compares each service description with the user request by checking for each property whether its value is in the desired set of values for the property. The desired functionality is described by queries over the elements inputs $I_R$, outputs $O_R$, pre-condition $\phi_R$, and effects $\psi_R$, where $I_R$, $O_R$, $\phi_R$, and $\psi_R$ are logical expressions describing a set of desired values. For example, $I_R$ specifies the set of desired input parameter combinations. Each such combination corresponds to a possible model of the query $I_R$ [38]. The functionality matchmaking consists of checking whether the inputs $I_w$, outputs $O_w$, preconditions $\phi_w$, and postconditions $\psi_w$ of service $w$ are contained in $I_R$, $O_R$, $\phi_R$, and $\psi_R$ respectively. That is, functionality $(I_w, O_w, \phi_w, \psi_w)$ of a service $w$ fulfills the desired functionality $(I_R, O_R, \phi_R, \psi_R)$ specified in a request $R$ if $I_w \models I_R$, $O_w \models O_R$, $\phi_w \models \phi_R$, and $\psi_w \models \psi_R$. Analogously, NFPs of a service are verified by checking whether their values are in the respective sets of desired values.

### *4.3. Preference-Based Integrated Ranking*

In order to provide users a high flexibility when defining service preferences, several ranking mechanisms can be integrated into the previously discussed service retrieval and ranking scenario. However, as each mechanism usually provides an *ad hoc* preference model, there is a need for a generic, common definition framework that allows the user to combine preferences to be evaluated by different available mechanisms, as discussed in Section 2.

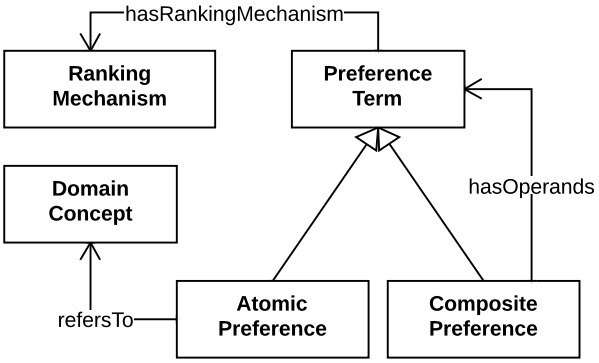


Figure 3: Upper ontology for user preferences.

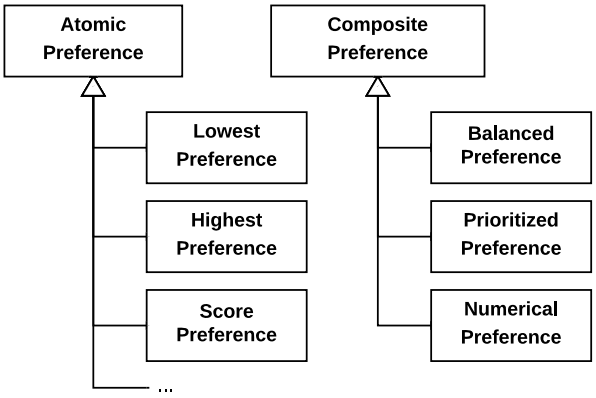


Figure 4: Selected preference terms from the model.

Our preference-based integrated ranking proposal manages those two levels of integration, providing (1) a common preference model that integrates preference definitions from each mechanism, and (2) a framework that integrates corresponding ranking implementations depending on that model. In the following both the common model and integration framework are presented.

*4.3.1. An Upper Ontology to Model Preferences*

Our proposed preference model, which is an adaptation of the model introduced in [39], is used as an upper ontology so that individual preference terms from each integrated ranking mechanism are expressed using common concepts from a single preference ontology, enabling interoperability between rankers. Figure 3 presents a UML representation of the upper ontology of our preference model, where the user basically can express atomic preferences using different preference terms that are handled internally by the corresponding ranking mechanism. Then, composite preferences can be used to compose those terms, defining the relationship between previously expressed atomic preferences. Note that composite preferences are also handled by a ranking mechanism that offers facilities to combine simpler atomic preferences.

In particular, atomic preferences are related to a domain-specific concept that usually represents a NFP that should be optimized to fulfill the user preference over it. Figure 4 outlines some of the available preference terms from our model. Thus, a `Lowest` (a `Highest`) preference means that the user prefers a lower (higher) NFP value. A `Score` preference expresses that the NFP value will be evaluated using a scoring function. This function returns a real value between 0 and 1, which expresses to what extent the referred NFP value is preferred against others. Additionally, the preference model supports other facilities to express preferences, not only including quantitative but also qualitative preferences [39]. However, we only focus on the described preference terms in this article, since they are applied in Section 5. A formal description of both atomic and composite preferences of our model is presented in [40]

Concerning composite preferences, our proposed model contemplates three facilities that may be extended to refine their semantics. First, a `Balanced` preference $P$ combines preference terms $P_1, \ldots, P_n$ using the *Pareto-optimality principle*, which considers that combined preferences $P_i$ are equally important for the user, balancing their fulfillment. Therefore, a service $w_l$ is considered better than another service $w_m$ with respect to $P$ if $w_l$ is better than $w_m$ with respect to any $P_i$ such that $w_l$ is not worse than $w_m$ with respect to the rest of the combined preferences $P_j$ with $i \neq j$.

Second, a `Prioritized` preference combines preferences in importance order, *i.e.* if a list of preference terms $P_1, P_2, \ldots, P_n$ is combined using a prioritized preference, services will be ranked first in terms of $P_1$. Then, services that cannot be compared (or are equally preferred) using $P_1$ are ranked in terms of $P_2$, and so on.

Finally, a `Numerical` preference is the combination of preferences using a real function to obtain a numerical score value for each service. Consequently, services are ranked in terms of their score values. However, this composite preference can only combine quantitative preferences that can be evaluated to a score value by their corresponding ranking mechanism.

The adaptation of this common preference model to a concrete service selection scenario enables interoperability between ranking mechanisms, using it as a preference meta-model, which provides a higher expressiveness compared to each mechanism isolated. Consequently, the first two challenges concerning the expressiveness and the interoperability (C1 and C2 in Section 2) are addressed by the application of this upper model, while the integration (C3) is solved by the PURI framework described below. See also Section 5.2 for a particular example.

*4.3.2. PURI: A Framework to Integrate Ranking Mechanisms*

Our integrated preference based ranking solution uses preferences defined in terms of the presented model in order to rank a set of matching services. As described above,

each preference term is handled by a particular ranking mechanism. In order to correctly call each mechanism, compose the results, and manage in general the integrated ranking process, we propose the use of the PURI framework, which is described in the following.

PURI stands for Preference-based Universal Ranking Integration framework and provides facilities to integrate several ranking mechanisms by using the previously described common preference model. Essentially, its integrated ranking solution takes a set of retrieved services and a user preference, which is evaluated in order to obtain the needed combination of ranking mechanisms that have to be invoked such that a given set of services is ordered according to the preferences. The returned service ranking is interpreted as a strict partial order, because some services may not be able to be compared to each other.

**Algorithm 1:** Function $rank(S, u) : S'$

**Input**: A set of services $S$ and the user preference instance $u$
**Output**: A strict partially ordered set of services $S'$

```
1  Instantiate R such that hasRankingMechanism(u, R)
2  if AtomicPreference(u) then
3      S' ← doAtomicRank(S,u) of R
4  else if
   BalancedPreference(u) ⊔ NumericalPreference(u)
   then
5      foreach u_i such that hasOperands(u, u_i) do
6          S_i ← rank(S, u_i)
7      S' ← compose(u, S_1, ..., S_n)
8  else if PrioritizedPreference(u) then
9      S' ← S
10     i ← 1
11     while S' is not completely ordered and
       hasOperands(u, u_i) do
12         S' ← rank(S', u_i)
13         i ← i + 1
14 return S'
```

The core functionality of the integrated ranking process is implemented by the `rank` function presented in Algorithm 1. Within this process, PURI dynamically instantiates the appropriate ranking mechanism for a given user preference $u$, as denoted by its corresponding `hasRankingMechanism` relation (line 1). If $u$ directly corresponds to an `AtomicPreference`, the `doAtomicRank` function of the instantiated ranking mechanism is executed (line 3) and the partially ordered set $S'$ of services is returned (line 14).

If $u$ is a composite preference, each of its $n$ components are obtained from the `hasOperands` relation. In this case, PURI applies two different workflows depending on the concrete type of $u$ in order to optimize the global execution time of composed mechanisms:

1. `Balanced` and `Numerical` preference terms fork the ranking execution in order to evaluate their component preferences in parallel (line 6). After all the corresponding `rank` calls finish their execution, results are properly composed in terms of the concrete composite preference $u$, as described in Section 4.3.1, to obtain a strict partial order (line 7). This final composition is performed in constant time (`Numerical`) or linear time (`Balanced`) on the number of component preferences, and it does not affect the accuracy of results obtained from the integrated ranking mechanisms.

2. `Prioritized` preference terms sequentially evaluate each component term, so that the execution is terminated as soon as the evaluation of the corresponding component term returns an ordering over all the retrieved services, or there is no more component terms to be evaluated (line 11). As with previous composite preferences, this evaluation does not affect the accuracy of results obtained separately from the composed mechanisms.

For instance, the ranking of a set of services $S$ is computed for the preference described in Section 2, which composes a `LowestPreference` and a `Score Preference` using a `BalancedPreference` (see Listing 3). Algorithm 1 is executed as follows: PURI firstly detects that the topmost preference $u$ is a composite one, specifically a `BalancedPreference`. Consequently, it recursively calls the `rank` function (line 6) over each operand $u_1$ and $u_2$, with both being atomic ones. Thus, for $u_1$, which is a `LowestPreference`, the associated ranking mechanism (the tendency based mechanism described in Section 2) is instantiated in line 1 and executed over $S$ in line 3. The other atomic preference $u_2$ is similarly evaluated, and the results obtained from the two corresponding ranking mechanisms (ordering $S_1$ computed by the tendency based mechanism and ordering $S_2$ by the fuzzy logic based one, respectively) are then composed in line 7 as defined by the `Balanced` composite preference, in order to obtain the global ordering $S'$ of the original set of services $S$ returned in line 14.

Concerning the selection of the appropriate ranking mechanism for a given preference term, the PURI framework presents a protected variation point [41]. Therefore, domain experts and developers, who hold the responsibility for the adaptation of PURI to a particular scenario, have to design and implement a concrete ranking mechanism selection strategy. In our use case application discussed in Section 5, we explicitly define at design time, within the adapted preference model, which specific ranking mechanism have to be instantiated to evaluate each concrete preference term. Thus, the evaluation of the `hasRankingMechanism` relation performed in line 1 of the

Algorithm 1 always returns a unique, predetermined ranking mechanism in our use case.

However, if another application scenario provides various ranking mechanisms that can be used to evaluate the same preference term, developers responsible for the concrete adaptation of PURI to this scenario have to implement a concrete strategy to select the appropriate mechanism. For instance, Quality-of-Service (QoS) properties of available mechanisms can be monitored so that the best one can be chosen at run time, depending on current QoS values of selected properties, such as response time or availability [42]. Consequently, in this case, line 1 of the Algorithm 1 may return a different ranking mechanism for each `rank` call over the same preference term.

There are some key features that the framework offers for developers to extend and adapt PURI to their particular needs. First, ranking mechanisms can be dynamically registered into a factory that is utilized to transparently instantiate them when needed to evaluate specific preferences. Second, each ranking implementation can be adapted to handle several preference terms from the upper model, which in turn can be also extended to fulfill particular requirements of each scenario. Third, the default implementation for composite preferences handles the aggregation of atomic quantitative and qualitative preferences automatically, because every preference will finally produce a strict partial order over services [39, 40]. Finally, developers can also change the PURI implementation of the returned ranking, provided that it successfully represents a strict partial order.

Consequently, a PURI adaptation that already provides an integrated ranking system can be also extended, integrating additional ranking mechanisms to support other preference facilities or provide higher performance. Although the use case application described in Section 5 focuses on integrating three different mechanisms, another ranking implementation may be added seamlessly, provided that its preference model is mapped to our common model (as discussed in Section 5.2), and a corresponding adapter is implemented, so that PURI can properly access and integrate that mechanism with the existing ones.

For instance, in order to integrate a Constraint Programming (CP) based ranking mechanism that allows the user to define preferences by defining utility functions [11], we have to identify utility functions definitions as instances of `ScorePreference` terms from our common model. Concretely, utility functions are directly mapped as the real function associated to a `ScorePreference`. Concerning the adapter, we register its implementation in PURI as a new ranking mechanism that is able to evaluate a `ScorePreference`. Consequently, this adapter has to translate real functions defined within `ScorePreference` terms to a Constraint Satisfaction Optimization Problem that is internally handled by the original CP-based ranking mechanism as described in [11]. The obtained results are again handled by the ranking adapter so that it returns to PURI a strict partial order that may be transparently combined with other ranking mechanisms results.

## 5. SOA4All Integrated Ranking: A Use Case Application

In the SOA4All FP7 European project[5], a fully-fledged, semantically-enhanced infrastructure to describe, search, compose, and execute services is proposed to offer effective, scalable, and usable solutions in an envisioned world of billions of available services [2]. The service retrieval and ranking scenario proposed in SOA4All provides three different ranking approaches, namely objective, multi-criteria and fuzzy ranking mechanisms [43]. Each approach provides different user interfaces and preference expressiveness depending on the applied ranking mechanism. As a consequence, a SOA4All user cannot combine preferences from the three ranking mechanisms offered.

Comparing the three approaches with the challenges identified in Section 2, we conclude that, though they provide different levels of expressiveness, there exist interoperability issues between them that prevent their integration. Furthermore, users cannot choose which ranking mechanism (or combination of them) should be applied to different service requests, depending on the expressiveness and performance needed, for instance.

In order to take full advantage of the three developed ranking mechanisms in SOA4All, the PURI framework was applied to this scenario, so an integrated ranking was implemented using those mechanisms, adapting and extending the previously discussed preference model and developing a single user interface to perform the service retrieval and ranking scenario as a whole, allowing the definition of preferences based on the adapted common model. In the following, we introduce the SOA4All ranking mechanisms and show how we applied the PURI framework in order to integrate them into a single service retrieval and ranking solution.

### *5.1. SOA4All Ranking Mechanisms*

In the subsequent paragraphs we give an overview of the three ranking methods developed in SOA4All. As we show at the end of this section, each approach has benefits and drawbacks in comparison to the other approaches.

#### *5.1.1. Ontology-based Feature Aggregation for Multi-valued Ranking*

The first approach ranks services based on objective features of Web services that can be automatically crawled and monitored [44]. For WSDL services, three independent ranking values are calculated. The values are based on (i) crawl meta-data like the number of related documents, (ii) verboseness of WSDL documents (esp. documenting parts), and (iii) monitoring data like availability and response time. These values are then combined with

[5] `http://www.soa4all.eu`

equal weights to one global rank. For Web APIs, a confidence score of a Web page describing a Web API is taken into account, only.

The global rank of services is independent of the user preferences and can be directly derived from the individual scores. Objective preferences can be applied for typical Web service meta-data such as the related documents score, since it is mostly valid to prefer services with a high number of documents on the Web strongly related to them over services with less related documents. Further, the WSDL metrics rank favors services with comments and descriptions in their WSDL service descriptions and the monitoring rank promotes services with high availability. The confidence score of a Web API denotes the likelihood of a Web resource to be a Web API [44].

*Related Documents Rank.* This rank is based on the crawl meta-data that is delivered by the crawler, and is calculated based on the following information: (i) How many related documents does a service have? We need to check the document annotations that belong to a service and then count the unique documents that are tied to the annotations (as more annotations can refer to the same document). (ii) How is the document related to a specific service? Documents can be direct inlinks or direct outlinks of the WSDL documents that describe a service, or the connection to the service can be determined by a term vector analysis of the documents and the service. In a first step, the number of related documents per service is calculated. Considering the number of document annotations only is not adequate because one document might have several annotations (*e.g.* a document that has a `DirectOutLink` annotation and a `TermVectorSimilarityAssociation`). Therefore, all `DirectInLink`, `DirectOutLink`, and `TermVectorSimilarityAssociation` annotations are first extracted in order to obtain the identifiers of all documents that correspond to the annotations. The number of multiple occurrences of the same document is counted, and then stored using the `hasNumberOfRelatedDocuments` property of the seekda Ranking Ontology. The single values that are used for the single kinds of related documents to calculate the temporary rank are currently experimental. These might be changed on a frequent basis until we discover the values that seem optimal for our needs. The final rank is stored for each service using the `hasRelatedDocsRank` property of the seekda Ranking Ontology.

*WSDL Metrics Rank.* This rank is based on metrics extracted from the WSDL documents. Currently, the metrics takes into account (a) the documentation of the service element, and (b) the documentation of the operations. The rank computation puts more importance on the documentation of the single operations than on service documentation, because it is more likely that operations contain useful information regarding the functionality provided by the operation as well as regarding their invocation. The final rank is stored for each service using the `hasWSDLMetricRank` property of the seekda Ranking Ontology.

*Monitoring Rank.* This rank is based on the liveliness information of a service, *e.g.* is the server reachable, does it correctly implement the SOAP protocol, etc. This liveliness information is delivered by seekda on a weekly basis. The availability score is a number between 0 and 1. Its value is set depending on the endpoint check result. For example, the score is 0 for read timeouts or errors, and 1 if based on the resulting payload (e.g., XML fault), it is very likely that communication with the WSDL service over SOAP messages is functioning. Other score values are set mostly based on the HTTP response code to reflect situations such as pages that are not found, or pages that require a login or an authentication, etc. The average service availability score is derived from scores for different time periods: last week, last month and last 6 months. It was assumed that the long-time availability of a service is more relevant than only the short-time availability over one week. It is important to note that this rank does not state anything about whether the functionality that the service announces is correctly implemented or not. The calculated rank is stored for each service using the `hasMonitoringRank` property of the seekda Ranking Ontology.

*Web API Confidence Score.* The ranking of Web APIs is derived from the seekda Web API confidence score. This score is calculated based on two classifiers within the crawler that check whether a Web resource might be a Web API or not. First, a classifier based on a support vector machine trained on a given data set is used for automatic classification of Web documents. Second, the Web API Evaluator performs structural and term vector analyses of the Web API resources, documentation Web pages, and related Web documents. The Web API Evaluator calculates indicators for the likelihood of the three Web resource categories of describing a Web API. The Web API rank is derived from the combination of the both confidence scores whereas more importance is given to the score of the SVM classifier. The applied weighting of the classifiers was determined by earlier evaluation results.

Obviously, services with higher values for this score are preferred. The global rank aggregates the individual values with equal weights. Then, it is normalized to the interval $[0, 1]$ and is finally added as an additional service property to the service description. For Web APIs, the Web API rank is at the same time the global rank of the service.

#### *5.1.2. Multi-criteria Ranking based on Non-Functional Properties*

The second approach bases the service ranking on user defined preferences [8]. NFPs in offers and requests are specified by means of logical rules using terms of given

NFP ontologies. The NFP model of descriptions is as follows.

**Class** NonFunctionalProperty
hasAnnotations **type** Annotation
hasDefinition **type** Axiom

NFPs specified in the user request and service descriptions are formalized by means of logical rules using terms from NFP ontologies. The logical rules used to model NFPs of services are evaluated, during the ranking process, by a reasoning engine. Additional data is required during this process: (i) which NFPs the user is interested in, (ii) the importance of each of these NFPs, (iii) how the list of services should be ordered (i.e., ascending or descending), and (iv) concrete instance data. The NFP values obtained by evaluating the logical rules are sorted and the ordered list of services is built.

First, a set of terms containing NFPs and their associated importance is extracted from the user request. If no importance is specified, the default value is consider to be 0.5, which stands for a moderate interest in the respective NFP. The importance is a numeric value ranging from 0 to 1, where 1 encodes the fact that the user is extremely interested in the NFP and 0 encodes the opposite. Instance data from the goal is extracted and a knowledge base is created. The last step in extracting relevant information for the ranking process is to identify how the results should be ordered, either ascending or descending.

Once the preprocessing is completed, each service is assessed in order to determine whether the NFPs specified in the user request are available in the service description. If this is the case, the algorithm extracts the corresponding logic rules and evaluates them using a reasoning engine that supports WSML rules (e.g. IRIS[6]). A quadruple structure containing the computed value and its importance for each service and NFP is built. An aggregated score is computed for each service by summing the normalized values of NFPs weighted by importance values. The results are collected in a set of tuples, where each tuple contains the service id and the computed score. Finally, the scores are ordered as specified by the user and the final list of services is returned.

The novelties of the second ranking approach are the combined use of ontological representation of NFPs with multiple NFP dimensions and the possibility to justify the computed ranking by the provision of provenance information [43].

*5.1.3. Fuzzy Logic Based Ranking Approach*

The third service ranking mechanism advances the expressiveness of user preferences from the second approach. The fuzzy logic based ranking mechanism features the following abilities: (i) express vagueness while formulating preferences using linguistic terms instead of crisp values, (ii) assign crisp property values to different categories

[6] `http://www.iris-reasoner.org`

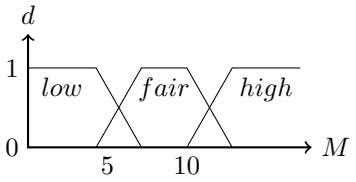


Figure 5: Example of membership functions that define the membership degree $d$ per category for the number $M$ of messages.

by specifying overlapping fuzzy set membership functions that model these categories, and (iii) create complex preferences constructed by the combination of simple terms.

User preferences are expressed by fuzzy rules, which are an efficient way to compute an approximative solution, often more efficiently than classical approximation methods [45]. Intuitively, a fuzzy rule describes which combination of property values a user is willing to accept to which degree. A rule head assigns a linguistic term that represents the degree of acceptance to the distinguished linguistic variable `acceptance`. The degree of acceptance holds for the service if the condition in the rule body holds. Similarly, a rule body contains expressions that assign categories, represented by linguistic terms and modeled by fuzzy sets, to service properties. Conjunctions, disjunctions, and negations of these expressions are also allowed in the body.

In our example, the rule

if `numberOfMessages` = *fair*
then `acceptance` = *super*

is part of the intended preference. Preferences contain those NFPs that a service should fulfill in order to be accepted for further consideration. Fuzzy set membership functions specify different levels (categories) of acceptance. The linguistic term *fair* is specified by a fuzzy set representing a range of number of messages (instead of a crisp value). A corresponding fuzzy set definition for this property is given in Figure 5. Three membership functions defining the terms *low*, *fair*, and *high* are specified by three overlapping trapezoid-shaped fuzzy sets with varying number of messages in horizontal and the degree of membership $d$, between 0 and 1, in the vertical direction. The linguistic terms allow users to refer to the sets instead of using values in the rule base. If the given fuzzy set definitions cannot be reused for a preference, users may customize or create them according to their needs with a Web based visual interface.

The process of computing a fuzzy logic based rank of services is composed of four main steps: *a*) fuzzification, *b*) inferencing, *c*) aggregation, and *d*) defuzzification. In step *a*), crisp NFP values of service descriptions are fuzzified, *i.e.* their membership in corresponding categories modeled by fuzzy sets are computed. During the inferencing step *b*), a degree of fulfillment for each fuzzy rule defined in the preference is computed. Then, the fuzzy set of the head of each rule is chopped at the level equal to the degree of

fulfillment of the rule's premise. In step *c*), the chopped fuzzy sets of the rule heads are aggregated. The aggregated set denotes the service rank as another fuzzy set, which is finally defuzzified to a value between 0 and 1 in step *d*). A more detailed description of the computational process along with an example was provided in [43].

*Membership degree.* In step *a*) and *b*), the degree of membership of a service property and the degree of fulfillment of a fuzzy rule, respectively, are computed as follows. Let $O$ be the concept that represents the acceptance and let $O$ be divided into $k$ categories $O_1, ..., O_k$. Then, there exist (at most) $k$ rules in the rule base $\{R_1, ..., R_j | j \leq k\}$, where any rule $R_i$ has $O_i$ in the rule head. Each rule is processed in the inferencing step *b*) as follows. For each $R_i$, the degree $d^w_{O_i}$ to that a service $w$ fulfills the body of $R_i$ is determined. The membership degree $d^w_{O_i}$ of the whole premise is derived from the degrees of the individual terms by using the following semantics suggested by Zadeh in [46]. Let $\mu_A$ and $\mu_B$ denote two membership functions, then

$$(\mu_A \wedge \mu_B)(w) \equiv min\{\mu_A(w), \mu_B(w)\},$$
$$(\mu_A \vee \mu_B)(w) \equiv max\{\mu_A(w), \mu_B(w)\},$$
$$\neg\mu_A(w) \equiv 1 - \mu_A(w).$$

The membership of $w$ to $\mu$ is calculated by obtaining the value of $y$ for $x$, which lies on the line passing through $(x_1, y_1)$ and $(x_2, y_2)$. As each membership function is described by several lines, the value of $y$ is only obtained from the line that passes through $(x, y)$ between the line's end points $(x_1, y_1)$ and $(x_2, y_2)$.

$$y = \frac{y_2 - y_1}{x_2 - x_1}(x - x_1) + y_1$$

At the end of step *b*), a new membership function $\mu^w_{O_i}$ is computed from $d^w_{O_i}$ and the fuzzy set $O_i$ representing the linguistic term in the rule head. Therefore, the part of the membership function $\mu^w_{O_i}$ that is higher than $d^w_{O_i}$ is cut.

In the aggregation step *c*), the fuzzy sets $\mu^w_{O_i}$ computed in step *b*) are aggregated into one fuzzy set $\mu^w_O$, which represents the solution of the problem. The solution $\mu^w_O$ simply takes the maximum of all the $\mu^w_{O_i}$ functions. Figure 6 illustrates the result of step *b*) for a rule base with two rules for the *good* and *super* acceptance categories. The degree of fulfillments of the rule bodies are 0.5 and 0.8 respectively. The membership function $\mu^w_O$ shown in Figure 7 is the result of the aggregation of both functions $\mu^w_{O_i}$ in Figure 6.

In the final step *d*), the fuzzy set $\mu^w_O$ is defuzzified to obtain a crisp value between 0 and 1, which represents the actual rank [12]. Again, while there are various defuzzification techniques available in the literature, the center of gravity method is perhaps the most widely used. According to [47], there is no systematic procedure for choosing a defuzzification strategy. Hence, other strategies like the max criterion, mean of maximum, or center of area method

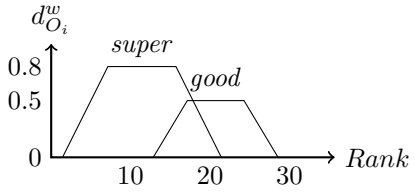


Figure 6: Example set of membership functions generated in step *b*) from two fuzzy if-then rules.

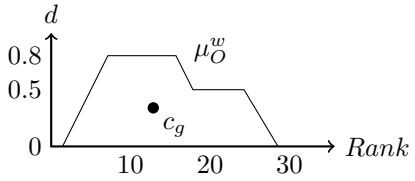


Figure 7: Aggregated membership function derived from the example in Figure 6. $c_g$ denotes the center of gravity.

are also applicable. In our approach, the $x$ coordinate of the center of gravity (illustrated by $c_g$ in the example of Figure 7) determines the overall acceptance of a service $w$ and is computed as follows.

$$x = \frac{\sum_{i=1}^{n} x_i y_i}{\sum_{i=1}^{n} y_i}$$

*5.1.4. Comparison of Ranking Methods*

The first approach (objective) is clearly distinguished by its simplicity. It is similar to Google Web site ranking as the global rank can be computed off-line and independently from user preferences. Therefore, the ranking can be further exploited by other components like other ranking or retrieval mechanisms if top-$k$ algorithms are applied. That is, the $k$ most promising services (with a high global rank) are processed exclusively or in a privileged manner such that results can be delivered faster. The downside of the first mechanism is its limitation to a given set of properties that are observable by the crawler as well as the lacking ability of user customization, *i.e.*, the expressiveness of available preferences is constrained.

The second ranking method (multi-criteria) overcomes the shortcoming of the previous one by providing a preference model and taking any ontologically defined NFPs into account. It provides users simple means to express preferences on ascending and descending orderings with weighted aggregation into a global rank. However, this approach relies on the assumption of independent property preferences. That is, it cannot be expressed that a user accepts a higher price if a high quality is offered, for instance. The limited expressiveness of the preference model is therefore the motivation for the third method.

Fuzzy if-then preferences have a higher expressiveness. Dependencies between different desired properties as well as desired fuzzy value ranges can be specified. Further, users can express rather vague preferences by fuzzy sets.

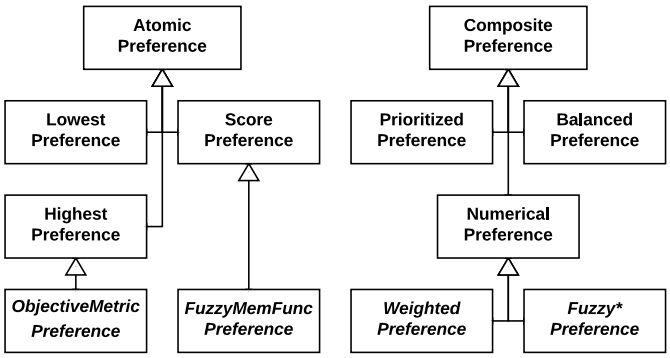


Figure 8: SOA4All adaptation of the preference model.

On the downside of this third approach is the increased computational effort that is required to compute a ranking, and the complexity of the preference definition.

As a conclusion, each SOA4All ranking mechanism serves a particular purpose depending on the level of expressiveness and flexibility the user needs when defining preferences for service ranking. For instance, in order to model the example discussed in Section 2, the multi-criteria ranking can be used to define and efficiently evaluate the price preference, whereas the fuzzy logic based approach is useful to express the preference on the number of messages. However, the combination of those two preferences is not possible in this use case, as it has already been identified in Section 2 as a challenge in SWS ranking. In this scenario, our PURI framework can be applied to overcome the associated issues and effectively combine SOA4All ranking mechanisms, as described in the following.

### *5.2. Preference Model Adaptation*

Before instantiating the PURI framework to provide an integrated ranking solution for the SOA4All use case, the different preference models offered by each ranking mechanism have to be integrated into our previously presented common preference model. Therefore, similarities between our preference model facilities and those provided by SOA4All ranking mechanisms have to be identified. Figure 8 shows the extended preference model that supports SOA4All facilities, where new preference terms with respect to the basic model shown in Figure 4 are depicted in italics.

In the first case, objective multi-valued ranking is based on metrics and derived ranks that can be used to order the retrieved services, where the resulting ranking should present services with higher ranking values for the chosen metric at its top. Consequently, we simply interpret a preference on a concrete `ObjectiveMetric` as a particular case of a `HighestPreference` that constrains the domain concept that can be referred to the available monitored metrics. For instance, a request may contain a preference where the user prefers services with higher global rank.

Concerning the NFP-based multi-criteria ranking mechanism, its own preference model allows to define the NFP of interest, which is identified as the `DomainConcept` that a preference refers to in our model as depicted in Figure 3. Depending on the desired ordering, the preference can be considered as a `Lowest` or a `Highest` one in the common model, which is extended by including an associated operand that can be used to define the relative importance as a float value. This importance value can be used itself to compose several ascending or descending preferences, because it is used in the normalization and aggregation stage of this ranking mechanism. Therefore, we added a composite numerical preference called `WeightedPreference` that can combine several `Lowest` or `Highest` preferences provided that they define a corresponding importance value. Note that objective multi-valued ranking metrics can be also used as the referred NFP so that both ranking mechanisms can be easily combined using a `WeightedPreference`.

Finally, the fuzzy logic based ranking mechanism provides fuzzy rules and membership functions as the basic constructs to define preferences. On the one hand, a fuzzy rule is interpreted as a specialization of a `NumericalPreference` whose combining function is defined by the fuzzy ranking algorithm. A `FuzzyRulePreference` contains a premise (rule body) and a conclusion (rule head). Premises may contain fuzzy logic negations, disjunctions and conjunctions that are also considered to be specializations of `NumericalPreferences`[7] as they can combine different fuzzy set membership functions. A conclusion also contains a fuzzy set membership function that is interpreted as the fuzzy score of the rule. Furthermore, rules can also be combined in a `FuzzyGoalPreference`, that is interpreted as a particular `NumericalPreference`, too.

On the other hand, fuzzy set membership functions are considered atomic preferences because they provide means to obtain a fuzzy score value depending on the value of a referred domain concept, such as price or number of messages. In our common model, there exists a generic preference called `ScorePreference` that is defined after a real function that computes the score used to rank services [39]. Therefore, we model fuzzy membership functions as a particular case of a `ScorePreference` (denoted as `FuzzyMemFuncPreference` in Figure 8), whose scoring function is precisely that membership function.

For instance, the example discussed in Section 2 can be modeled using the adapted common preference model as follows. In that example there are two atomic preferences that can be modeled using (1) a `LowestPreference` on the price per message, as provided by the NFP-based multi-criteria ranking mechanism; and (2) a fuzzy membership function (`FuzzyMemFuncPreference`), which models the

[7] In order to simplify Figure 8, all fuzzy preference composite constructors (rules, goals, negations, disjunctions and conjunctions) are denoted as `Fuzzy*Preference`.

preference on the *fair* number of simultaneously sent messages, evaluated by the fuzzy logic based approach. Furthermore, both atomic preferences can be composed using a `BalancedPreference` (directly implemented by PURI), so that they are considered equally important for the user. A partial representation of this example is presented in Listing 3, deliberately omitting `hasRankingMechanism` values as they have been described previously, as well as prefixes already defined in previous listings. Note that the fuzzy membership function has to be included within a fuzzy goal, which is also partially represented.

Listing 3: Instance of the user preference from Section 2.

```
ex:userPreference
   a puri:BalancedPreference ;
   puri:hasOperands ex:pricePreference ,
       ex:numberOfMessagesPreference .
ex:pricePreference
   a puri:LowestPreference ;
   puri:refersTo telco:UnitCost .
ex:numberOfMessagesPreference
   a puri:FuzzyGoalPreference ;
   puri:hasOperands ex:superRule ,
       ex:goodRule .
ex:superRule a puri:FuzzyRulePreference ;
   puri:hasOperands ex:superRuleBody ,
       ex:superRuleHead .
ex:superRuleBody
   a puri:FuzzyMemFuncPreference ;
   puri:refersTo telco:NumberOfMessages ;
   puri:hasScoringFunction
       ex:fairMembershipFunction .
```

The PURI framework analyzes this preference and the referred NFP values of service descriptions (*e.g.* `telco:UnitCost` and `telco:NumberOfMessages` as in Listing 4) in order to rank a set of previously retrieved services. For instance, Listing 4 describes the NFP values of `ex:smsService1` service from Listing 2.

Listing 4: NFP terms of a service description.

```
@prefix wl: <http://www.wsmo.org/ns/wsmo-lite#> .

ex:smsService1
   a posm:Service , puri:ServiceDescription ;
   puri:hasFeatures ex:priceService1 ,
       ex:messagesService1 .
ex:priceService1
   a wl:NonFunctionalParameter ,
       puri:NonFunctionalTerm ;
   puri:refersTo telco:UnitCost ;
   telco:hasAmount "0.03"^^xsd:double .
ex:messagesService1
   a wl:NonFunctionalParameter ,
       puri:NonFunctionalTerm ;
   puri:refersTo telco:NumberOfMessages ;
   telco:hasAmount "10"^^xsd:integer .
```

In our example, a preference over the number of message that a service is able to send with one invocation is expressed in the fuzzy goal `ex:numberOfMessagesPreference`. Two categories *good* and *super* for the objective function are used and formulated as follows.

if `numberOfMessages` = *low* or `numberOfMessages` = *high*
then `acceptance` = *good*

if `numberOfMessages` = *fair*
then `acceptance` = *super*

*Super* acceptance is preferred over *good* acceptance. In order to determine the degree of acceptance, the specified membership function for the service property `NumberOfMessages` allows to determine the membership of each service to each category (*low*, *fair*, or *high*). For the given example offer, the membership degree $d$ for each category is $d(low) = 0$, $d(fair) = 1$, $d(high) = 0$. Therefore, the example offer falls into the acceptance category *super* of the objective function which is specified in the preference of Listing 3.

### *5.3. Applying PURI to SOA4All Integrated Ranking Implementation*

Starting from the model adaptation discussed in the previous paragraphs, the actual implementation of the SOA4All integrated ranking approach involves the application and extension of the PURI framework to develop a holistic solution to service retrieval that allows the combination of several ranking mechanisms, exploiting synergies and providing a single, unified user interface to the SOA4All service retrieval scenario.

First of all, each ranking mechanism interface was adapted to PURI ranking API. Essentially, each adapter supports the corresponding preference terms that are handled by each ranking mechanism. Using a dynamic instantiation, PURI is able to identify which adapters have to be used to rank a set of retrieved services in terms of a user provided preference. Furthermore, PURI is also responsible to orchestrate those adapters in order to combine ranking results from different ranking mechanisms in the event that composite preferences are specified by the user.

The developed SOA4All integrated ranking was deployed as a Web service itself, so that it could be easily integrated within the global SOA4All service retrieval and ranking solution and is simultaneously offered as a standalone component which allows to define preferences based on the discussed common model for the three ranking mechanisms proposed in SOA4All.

Finally, in order to apply our holistic solution to the SOA4All use case, it is necessary to put together both service retrieval and integrated ranking implementations, integrating both components using a common user interface to the global SOA4All service retrieval system. A user can first enter criteria in order to filter the result set. A set of functionality classes from a tree-structured hierarchy can be selected. Multiple selections are interpreted such that retrieved services are member of all selected classes. Furthermore, the user may refine the desired service functionality with logic expressions describing inputs, outputs, pre-conditions, and effects. The desired values of NFPs can be constrained, too. Based on these requirements, the

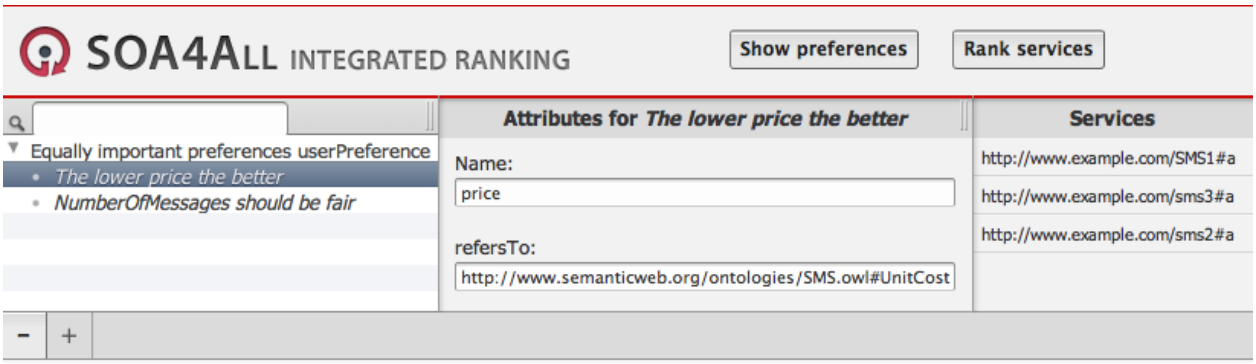


Figure 9: Screenshot of the preference definition user interface.

SOA4All retrieval component is able to determine the set of matching service descriptions.

In a second step, the user may specify preferences in order to rank services. Therefore, as depicted in Figure 9, the Web-based interface guides the user in expressing preferences with minimal knowledge about the syntax. The preference type (see Figure 4) is chosen from a predefined list and the referred NFP concept as well as operands are entered in dedicated text fields. Finally a name is assigned to the preference that allows to construct composite preference structures more conveniently. Upon submit, the services are presented to the user in the ranked order. The prototype implementation of the integrated ranking component can be reached at `http://www.isa.us.es/soa4all-integrated-ranking/`.

### *5.4. Discussion*

In order to evaluate the validity of our proposal, we analyze its effectiveness in fulfilling the challenges identified in Section 2. Table 2 presents a qualitative analysis of the SOA4All Integrated Ranking implementation, though we discuss in the following how each challenge can be also analyzed in more generic scenarios.

Concerning the expressiveness of each proposal (C1), the integrated ranking approach using PURI allows the user to define preferences using any of the facilities provided by each other ranking mechanism. Consequently, it offers a higher expressiveness even when compared to the SOA4All fuzzy based approach, because of the possibility to combine other mechanisms[8] provided by our user-centric, strict partial order interpretation of preferences [48].

In addition to the SOA4All PURI application that validates our proposal, we have also evaluated that our common preference model provides a high flexibility and expressiveness by applying it to additional validation scenarios. Thus, our model was successfully applied to describe different discovery scenarios described by the research community within the SWS Challenge[9]. In particular, we validated an early version of our common model using the Shipment Discovery scenario [11], in addition to the complete description of the more complex Logistics Management scenario presented in [39].

In turn, the interoperability degree (C2) has been measured by evaluating both the kind and number of preferences that can be combined. Although in the SOA4All application each ranking mechanism uses semantic models to define preferences, the lack of an upper model made their interoperation at the description level difficult, showing a medium interoperability in Table 2. Using our common preference model, facilities from any ranking mechanism can be composed together, exploiting its synergies and providing the user with more control over the service retrieval and ranking process.

In particular, we evaluated what kind of combinations are enabled by our model. Because of its inductive definition, the number of preferences that can be combined together is not limited. Composite preferences are composed of several preference terms that can be either atomic or composite preferences, allowing the user to express complex desires that combine an arbitrary number of preferences. Thus, we performed experiments that generated random combinations of preferences with an arbitrary depth, showing that the execution time of the analysis performed by PURI on every combination is at most linear with respect to the number of preference terms, regardless of their tree structure and depth. However, it is worth noting that real user preferences only contains two to six terms, in general [49].

Finally, at implementation level, we analyzed the integrability degree (C3). In the SOA4All scenario, both multi-criteria and fuzzy logic based ranking mechanisms cannot be easily integrated because of the different underlying formalisms (*i.e.* they offer a low integrability degree), though the objective ranking can be transparently integrated with the service retrieval component, retrieving services already ordered by the computed global rank, that can be further processed by any of the other ranking

[8] See Section 5.1 for a comparison of their corresponding degrees of expressiveness

[9] `http://sws-challenge.org`

Table 2: Comparison between eclectic and holistic SOA4All service ranking.

| Ranking Approach | C1 | C2 | C3 |
|---|---|---|---|
| Objective Ranking | Low | Medium | Medium |
| Multi-criteria Ranking | Medium | Medium | Low |
| Fuzzy based Ranking | High | Medium | Low |
| Integrated Ranking | High | High | High |

mechanisms.

However, the integrated ranking approach implemented using PURI is not only able to integrate the three available ranking mechanisms into a unique service retrieval and ranking system, but also to orchestrate the ranking execution workflow in terms of the concrete preferences defined by the user, providing a high integrability (C3). Furthermore, as the integrated ranking solution is based on the internal ranking mechanisms, the integrated ranking performance depends on those mechanisms. As we measured in our SOA4All application, the workflow orchestration provided by the framework does not add a significant penalty to the total execution time, because it redirects ranking requests to relevant mechanisms and subsequently combines the results of internal rankers. Latter step takes linear time with respect to the number of results of each composite preference. If preferences are composed using `PrioritizedPreference` terms, the overall execution time may be less as some services may not need to be compared with respect to second and subsequent composed preferences. Moreover, as preferences composed using `BalancedPreference` terms are evaluated in parallel, the whole composition performed by PURI usually takes less time than the individual evaluation of composed preferences by their corresponding ranking mechanisms, subsequently.

In addition to the previous validation of our proposal within our use case scenario, we also performed a reusability analysis of PURI, in order to evaluate the ease-of-use and extensibility of our proposed framework. Table 3 present a subset of the metrics we computed in order to analyze the reusability of a framework according to [50]. We evaluated both PURI framework and the adaptation described in this section to SOA4All project, although in Table 3 we only represent values from the multi-criteria ranking adaptation in order to better analyze the cost of integrating a mechanism.

According to the measured values, the number of methods (NOM), attributes (NAT), and parameters per method (NPM) for each class in both cases are low, offering a streamlined, understandable architecture that helps developers to easily extend the framework and implement new adaptations. Concerning actual number of lines of code in methods (MLC) and their algorithmic complexity (WMC), our PURI framework obtain low values showing that it has been designed to be simple enough to facilitate its reusability. The adaptation is inherently more complex, since it deals with the concrete ranking mechanism implementation. However, in both cases the maximum McCabe cyclomatic complexity is 9, below the recommended threshold [51]. Finally, both PURI framework and the evaluated adaptation do not present lack of cohesion, and coupling degrees are contained, especially in the case of the adaptation, which only depends on the PURI framework and the concrete ranking mechanism being adapted.

Nevertheless, our proposal presents two particular limitations. On the one hand, in order to extend the integrated ranking system by adding other ranking mechanisms, a proper adapter has to be implemented, possibly extending the preference model so that facilities provided by new mechanisms are integrated into the common model. According to the presented metrics, this extension is not costly. On the other hand, if several ranking mechanisms can evaluate the same preference term, the user cannot specifically state which concrete mechanism should be used to rank with respect to that term, as that is pre-defined at design time as discussed in Section 4.3.2. While the former issue can be solved at design time by solution developers, the latter can be considered a particular instance of a service ranking. Using this interpretation, the different ranking mechanisms should be described as candidate services, so that they could be ranked according to the user preferences on them. For instance, a user may prefer to rank services using more expressive ranking mechanisms instead of faster ones.

## 6. Conclusions

Current service retrieval systems have to perform a subsequent ranking so that they can return an ordered list of services in terms of defined preferences, in order to obtain the best service that fulfills the request. However, ranking mechanisms are coupled with *ad hoc* preference models that constrain the expressiveness of user preferences. Furthermore, these models are not interoperable in general, so a service retrieval system cannot combine several ranking mechanisms to provide more flexible and expressive facilities to define preferences.

Our proposal solves these identified issues of the service retrieval scenario by providing a common and highly expressive semantic preference model that enables the integration of different ranking mechanisms adapting the PURI framework, which is presented in this article. Consequently, our contribution offers a series of features that can be summed up as follows:

- **Flexibility**. The integration of any ranking mechanism using a common preference model allows end users to choose which preference facilities need for each request, without knowing the underlying ranking mechanisms that will be instantiated to actually rank retrieved services.

Table 3: Software metrics for PURI framework and its SOA4All adaptation.

| | PURI | | | | PURI - SOA4All Adaptation | | | |
|---|---|---|---|---|---|---|---|---|
| Metric | Total | Mean | Dev. | Max | Total | Mean | Dev. | Max |
| NOM | 43 | 4.30 | 2.45 | 10 | 33 | 4.71 | 2.86 | 10 |
| NAT | 9 | 0.90 | 0.70 | 2 | 5 | 0.71 | 0.45 | 1 |
| NPM | – | 1.05 | 0.85 | 3 | – | 0.79 | 0.80 | 3 |
| MLC | 209 | 4.75 | 5.43 | 18 | 460 | 16.43 | 18.51 | 54 |
| WMC | 81 | 8.10 | 5.41 | 18 | 91 | 13 | 13.48 | 44 |
| LCM | – | 0.10 | 0.20 | 0.50 | – | 0 | 0 | 0 |
| AFC | – | 3.50 | 2.06 | 7 | – | 1 | 0 | 1 |
| EFC | – | 2.50 | 0.50 | 3 | – | 3.50 | 2.50 | 6 |

NOM=number of methods in classes and interfaces; NAT=number of attributes in classes; NPM=number of parameters per method; MLC=number of lines in methods; WMC=weighted sum of McCabe cyclomatic complexity for all methods in a class; LCM=lack of cohesion of methods; AFC=afferent coupling; EFC=efferent coupling.

- **High expressiveness**. The preference model offers a comprehensive set of preference terms that can be easily combined and adapted in order to define complex preferences to rank services.

- **Ease of use**. Users do not need to access each ranking mechanism separately if they want to combine their results. A single entry point is provided in our solution that allows users to define their preferences and process them to rank the retrieved services.

- **Lightweight**. PURI provides a lightweight integration solution that does not add any noticeable performance penalty to the ranking process performed by each mechanism alone.

- **Reusability**. Our proposed framework can be extended and applied to other scenarios with low effort, enabling a seamless integration of new ranking mechanisms.

Furthermore, we have presented a validation of our proposal contextualized in the SOA4All EU FP7 project, in addition to other validation scenarios where PURI have been successfully applied, such as public administration service infrastructures and proposed scenarios from the SWS Challenge. Particularly, in SOA4All, we have integrated three different ranking mechanisms, namely objective multi-valued ranking, NFP-based multi-criteria ranking, and fuzzy logic based ranking. Furthermore, our solution to this scenario provides a single user interface to define requirements and preferences, simplifying their definition and offering a unique entry point for the whole service retrieval and ranking system, no matter what ranking mechanisms will be needed in the process.

Finally, we have also identified other scenarios where PURI can be useful to provide a holistic, integrated ranking approach. Specifically, we successfully adapted our framework to the PLATINA-FAST service trading system that is being implemented for Regional Administration in Andalusia, Spain. Furthermore, PURI is being adopted as the ranking framework for a series of local and European research project proposals from different domains, such as service agreements, dynamic configurators, and e-learning platforms.

Having successfully validated and applied PURI to the service ranking domain, we further plan to investigate how our approach can be applied to the more general ranking problem in the domain of information retrieval. Any structured information featuring property descriptions can be considered for a ranking. For instance, Web sites like the ones returned upon submitting a Web search query (*e.g.*, to Google Web search) can be ranked with PURI based on user specific preferences on properties like popularity, degree of connectivity, etc.

We have already identified the requirements to adapt PURI to more general information retrieval domains. In principle, the preference model does not need to be modified to support different domains, because it is already generic. Conversely, the API should be modified to accept any item that is able to be ranked. As a consequence, ranking mechanisms have to be adapted to allow the ranking of those domain-dependent items, and to return a properly typed ranking result, too. Finally, PURI abstract factory should be changed to accept multiple ranking mechanisms adapted to the specific domain to that PURI is going to be applied to.

## Acknowledgements

This work has been partially supported by the European Commission (FEDER) and Spanish Government under CICYT projects SETI (TIN2009-07366) and TAPAS (TIN2012-32273), by the Andalusian Government under projects ISABEL (TIC-2533) and THEOS (TIC-5906), by

the EU FP7 IST project 27867 SOA4All, and by the EC FP7 Network of Excellence 215483 S-CUBE.

The authors would like to thank the reviewers for their invaluable opinions and recommendations that improved this work substantially, and Rafael Z. Frantz for his helpful support and revisions.

[1] M. Papazoglou, D. Georgakopoulos, Service-Oriented Computing, Communications of the ACM 46 (10) (2003) 25–28.

[2] J. Davies, M. Potter, M. Richardson, S. Stincic, J. Domingue, C. Pedrinaci, D. Fensel, R. González-Cabero, Towards the Open Service Web, BT Technology Journal 26 (2).

[3] A. M. Fardin, N. B. Naser, N. M. Ali, Empower Service Directories with Knowledge, Knowledge-Based Systems 30 (0) (2012) 172–184.

[4] K. Sycara, M. Paolucci, A. Ankolekar, N. Srinivasan, Automated Discovery, Interaction and Composition of Semantic Web services, Journal of Web Semantics: Science, Services and Agents on the World Wide Web 1 (1) (2003) 27–46.

[5] C. Pedrinaci, D. Liu, M. Maleshkova, D. Lambert, J. Kopecký, J. Domingue, iServe: a Linked Services Publishing Platform, in: Ontology Repositories and Editors for the Semantic Web Workshop, Vol. 596 of CEUR Workshop Proceedings, 2010.

[6] N. Steinmetz, H. Lausen, M. Brunner, Web Service Search on Large Scale, in: L. Baresi, C.-H. Chi, J. Suzuki (Eds.), Proceedings of the 7th International Conference on Service-Oriented Computing and 2nd Service Wave, Vol. 5900 of Lecture Notes in Computer Science, 2009, pp. 437–444.

[7] G. Dobson, R. Lock, I. Sommerville, QoSOnt: a QoS Ontology for Service-Centric Systems, in: Proceedings of the 31st Euromicro Conference on Software Engineering and Advanced Applications, IEEE Computer Society, 2005, pp. 80–87.

[8] I. Toma, D. Roman, D. Fensel, B. Sapkota, J. Gomez, A Multi-criteria Service Ranking Approach Based on Non-Functional Properties Rules Evaluation, in: B. Krämer, K.-J. Lin, P. Narasimhan (Eds.), Proceedings of the 5th International Conference on Service-Oriented Computing, Vol. 4749 of Lecture Notes in Computer Science, Springer, 2007, pp. 435–441.

[9] X. Wang, T. Vitvar, M. Kerrigan, I. Toma, A QoS-Aware Selection Model for Semantic Web Services, in: A. Dan, W. Lamersdorf (Eds.), Proceedings of the 4th International Conference on Service-Oriented Computing, Vol. 4294 of Lecture Notes in Computer Science, Springer, 2006, pp. 390–401.

[10] S. Lamparter, A. Ankolekar, R. Studer, S. Grimm, Preference-based Selection of Highly Configurable Web Services, in: C. L. Williamson, M. E. Zurko, P. F. Patel-Schneider, P. J. Shenoy (Eds.), Proceedings of the 16th International Conference on World Wide Web, ACM, 2007, pp. 1013–1022.

[11] J. M. García, I. Toma, D. Ruiz, A. Ruiz-Cortés, A Service Ranker based on Logic Rules Evaluation and Constraint Programming, in: 2nd Non-Functional Properties and Service Level Agreements in Service Oriented Computing Workshop, Vol. 411 of CEUR Workshop Proceedings, Dublin, Ireland, 2008.

[12] S. Agarwal, M. Junghans, B. Norton, J. M. García, Second Service Ranking Prototype, Deliverable 5.4.3, SOA4All (2011).
URL `http://www.soa4all.eu/file-upload.html?func=fileinfo&id=261`

[13] H. Jin, X. Ning, W. Jia, H. Wu, G. Lu, Combining Weights with Fuzziness for Intelligent Semantic Web Search, Knowledge-Based Systems 21 (7) (2008) 655–665.

[14] M. Dastani, N. Jacobs, C. M. Jonker, J. Treur, Modelling User Preferences and Mediating Agents in Electronic Commerce, Knowledge-Based Systems 18 (7) (2005) 335–352.

[15] L. Li, I. Horrocks, A Software Framework For Matchmaking Based on Semantic Web Technology, in: Proceedings of the 12th International Conference on World Wide Web, ACM Press, 2003, pp. 331–339.

[16] B. Benatallah, M.-S. Hacid, C. Rey, F. Toumani, Semantic Reasoning for Web Services Discovery, in: Workshop on E-Services and the Semantic Web, 2003.

[17] K. Sycara, M. Paolucci, J. Soudry, N. Srinivasan, Dynamic Discovery and Coordination of Agent-Based Semantic Web Services, IEEE Internet Computing 8 (3) (2004) 66–73.

[18] C. Zhou, L. Chia, B. Lee, DAML-QoS Ontology for Web Services, in: IEEE International Conference on Web Services, 2004, pp. 472–479.

[19] E. Maximilien, M. Singh, A Framework and Ontology for Dynamic Web Services Selection, IEEE Internet Computing 8 (5) (2004) 84–93.

[20] L. H. Vu, M. Hauswirth, F. Porto, K. Aberer, A Search Engine for QoS-enabled Discovery of Semantic Web Services, International Journal of Business Process Integration and Management 1 (4) (2006) 244–255.

[21] W. Siberski, J. Z. Pan, U. Thaden, Querying the Semantic Web with Preferences, in: I. F. Cruz, S. Decker, D. Allemang, C. Preist, D. Schwabe, P. Mika, M. Uschold, L. Aroyo (Eds.), Proceedings of the 5th International Semantic Web Conference, Vol. 4273 of Lecture Notes in Computer Science, Springer, 2006, pp. 612–624.

[22] K. Kritikos, D. Plexousakis, Semantic QoS Metric Matching, in: Proceeding of the 4th IEEE European Conference on Web Services, IEEE Computer Society, 2006, pp. 265–274.

[23] A. Carenini, D. Cerizza, M. Comerio, E. D. Valle, F. de Paoli, A. Maurino, M. Palmonari, A. Turati, GLUE2: A Web Service Discovery Engine with Non-Functional Properties, in: Proceedings of the 6th IEEE European Conference on Web Services, IEEE Computer Society, 2008, pp. 21–30.

[24] M. Palmonari, M. Comerio, F. de Paoli, Effective and Flexible NFP-Based Ranking of Web Services, in: L. Baresi, C.-H. Chi, J. Suzuki (Eds.), Proceedings of the 7th International Conference on Service-Oriented Computing and 2nd Service Wave, Vol. 5900 of Lecture Notes in Computer Science, Springer, 2009, pp. 546–560.

[25] A. Ruiz-Cortés, O. Martín-Díaz, A. D. Toro, M. Toro, Improving the Automatic Procurement of Web Services Using Constraint Programming, International Journal of Cooperative Information Systems 14 (4) (2005) 439–468.

[26] C. R. Rivero, I. Hernández, D. Ruiz, R. Corchuelo, Generating SPARQL Executable Mappings to Integrate Ontologies, in: Proceedings of the 30th ER International Conference on Conceptual Modeling, 2011, pp. 118–131.

[27] C. Bizer, T. Heath, T. Berners-Lee, Linked Data - The Story So Far, International Journal on Semantic Web and Information Systems 5 (3) (2009) 1–22.

[28] D. Martin, M. Burstein, J. Hobbs, O. Lassila, D. McDermott, Others, OWL-S: Semantic Markup for Web Services, Tech. Rep. 1.2, DAML (2006).
URL `http://www.ai.sri.com/daml/services/owl-s/1.2/`

[29] D. Roman, U. Keller, H. Lausen, J. de Bruijn, R. Lara, M. Stollberg, A. Pollers, C. Feier, C. Bussler, D. Fensel, Web Service Modeling Ontology, Journal of Applied Ontology 1 (2005) 77–106.

[30] J. M. García, D. Ruiz, A. R. Cortés, O. Martín-Díaz, M. Resinas, An Hybrid, QoS-Aware Discovery of Semantic Web Services Using Constraint Programming, in: B. J. Krämer, K.-J. Lin, P. Narasimhan (Eds.), Proceedings of the 5th International Conference on Service-Oriented Computing, Vol. 4749 of Lecture Notes in Computer Science, Springer, 2007, pp. 69–80.

[31] Turtle - Terse RDF Triple Language, W3C Team Submission (2008).
URL `http://www.w3.org/TeamSubmission/turtle/`

[32] S. Agarwal, M. Junghans, Meaningful Service Classifications for Flexible Service Descriptions, in: Proceedings of The 7th IEEE 2011 World Congress on Services (SERVICES 2011), IEEE, Washington DC, 2011, pp. 85–86.

[33] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, P. F. Patel-Schneider (Eds.), The Description Logic Handbook: Theory, Implementation, and Applications, Cambridge University Press, 2003.

[34] T. Vitvar, J. Kopecký, J. Viskova, D. Fensel, WSMO-Lite Annotations for Web Services, in: Proceedings of the 5th European Semantic Web Conference on The Semantic Web: Research and

Applications, ESWC'08, Springer, 2008, pp. 674–689.
[35] J. Farrell, H. Lausen, Semantic Annotations for WSDL and XML Schema, Recommendation, W3C (2007).
URL http://www.w3.org/TR/sawsdl/
[36] J. de Bruijn, D. Fensel, M. Kerrigan, U. Keller, H. Lausen, J. Scicluna, Modeling Semantic Web Services: The Web Service Modeling Language, Springer, 2008.
[37] M. Junghans, S. Agarwal, Web Service Discovery Based on Unified View on Functional and Non-functional Properties, in: Proceedings of the 4th IEEE International Conference on Semantic Computing, IEEE, 2010, pp. 224–227.
[38] M. Junghans, S. Agarwal, R. Studer, Towards Practical Semantic Web Service Discovery, in: L. Aroyo, G. Antoniou, E. Hyvönen, A. ten Teije, H. Stuckenschmidt, L. Cabral, T. Tudorache (Eds.), ESWC (2), Vol. 6089 of Lecture Notes in Computer Science, Springer, 2010, pp. 15–29.
[39] J. M. García, D. Ruiz, A. Ruiz-Cortés, A Model of User Preferences for Semantic Services Discovery and Ranking, in: L. Aroyo, G. Antoniou, E. Hyvönen, A. ten Teije, H. Stuckenschmidt, L. Cabral, T. Tudorache (Eds.), ESWC (2), Vol. 6089 of Lecture Notes in Computer Science, Springer, 2010, pp. 1–14.
[40] J. M. García, D. Ruiz, A. Ruiz-Cortés, An Intuitive and Formal Description of Preferences for Semantic Web Service Discovery and Ranking, Tech. Rep. ISA-12-TR-07 (Dec 2012).
URL http://www.isa.us.es/sites/default/files/josemgarcia-tr-soup_0.pdf
[41] C. Larman, Design - Protected Variation: The Importance of Being Closed, IEEE Software 18 (3) (2001) 89–91.
[42] H. Q. Yu, S. Reiff-Marganiec, Non-Functional Property Based Service Selection: A Survey and Classification of Approaches, in: F. de Paoli, I. Toma, A. Maurino, M. Tilly, G. Dobson (Eds.), NFPSLA-SOC'08, Vol. 411 of CEUR Workshop Proceedings, CEUR-WS.org, 2008.
[43] I. Toma, S. Steinmetz, H. Lausen, S. Agarwal, M. Junghans, First Service Ranking Prototype, Deliverable 5.4.1, SOA4All (2009).
URL http://www.soa4all.eu/file-upload.html?func=fileinfo&id=143
[44] N. Steinmetz, H. Lausen, Ontology-Based Feature Aggregation for Multi-valued Ranking, in: A. Dan, F. Gittler, F. Toumani (Eds.), ICSOC/ServiceWave Workshops, Vol. 6275 of Lecture Notes in Computer Science, Springer, 2009, pp. 258–268.
[45] L. A. Zadeh, Outline of a New Approach to the Analysis of Complex Systems and Decision Processes, IEEE Trans. on Systems, Man, and Cybernetics SMC-3 (1973) 28–44.
[46] G. J. Klir, B. Yuan (Eds.), Fuzzy Sets, Fuzzy Logic, And Fuzzy Systems: Selected Papers by Lotfi A. Zadeh, World Scientific Publishing Co., Inc., River Edge, NJ, USA, 1996.
[47] C. Lee, Fuzzy Logic in Control Systems: Fuzzy Logic Controller. II, Systems, Man and Cybernetics, IEEE Transactions on 20 (2) (1990) 419 –435. doi:10.1109/21.52552.
[48] W. Kießling, Foundations of Preferences in Database Systems, in: Proceedings of the 28th International Conference on Very Large Data Bases, Morgan Kaufmann, 2002, pp. 311–322.
[49] S. Holland, M. Ester, W. Kießling, Preference Mining: A Novel Approach on Mining User Preferences for Personalized Applications, in: N. Lavrac, D. Gamberger, H. Blockeel, L. Todorovski (Eds.), PKDD, Vol. 2838 of Lecture Notes in Computer Science, Springer, 2003, pp. 204–216.
[50] K. Erni, C. Lewerentz, Applying Design-Metrics to Object-Oriented Frameworks, in: Proceedings of METRICS '96, IEEE, 1996.
[51] T. J. McCabe, A Complexity Measure, IEEE Trans. Software Eng. 2 (4) (1976) 308–320.