Effective and Flexible NFP-Based Ranking of Web Services

Matteo Palmonari, Marco Comerio, and Flavio De Paoli

University of Milano - Bicocca, viale Sarca 336, 20126 Milano, Italy
{palmonari,comerio,depaoli}@disco.unimib.it

Abstract. Service discovery is a key activity to actually identify the Web services (WSs) to be invoked and composed. Since it is likely that more than one service fulfill a set of user requirements, some ranking mechanisms based on non-functional properties (NFPs) are needed to support automatic or semi-automatic selection.

This paper introduces an approach to NFP-based ranking of WSs providing support for semantic mediation, consideration of expressive NFP descriptions both on provider and client side, and novel matching functions for handling either quantitative or qualitative NFPs. The approach has been implemented in a ranker that integrates reasoning techniques with algorithmic ones in order to overcome current and intrinsic limitations of semantic Web technologies and to provide algorithmic techniques with more flexibility. Moreover, to the best of our knowledge, this paper presents the first experimental results related to NFP-based ranking of WSs considering a significant number of expressive NFP descriptions, showing the effectiveness of the approach.

1 Introduction

Web Service (WS) discovery is a process that consists in the identification of the services that fulfill a set of requirements given by a user. Since more than one service is likely to fulfill the functional requirements, some ranking mechanisms are needed in order to provide support for the automatic or semi-automatic selection of a restricted number of services (usually one) among the discovered ones.

According to a gross-grain definition, the discovery process consists in first locating a number of WSs that meets certain functional criteria, and then identifying the services, among the discovered ones, that better fulfill a set of non-functional properties (NFPs) requested by actual users. The latter activity is called WS ranking and it is based on the computing of a degree of match between a set of requested NFPs and a set of NFPs offered by the discovered WSs. NFPs cover Quality of Service (QoS) aspects, but also other business-related properties, such as pricing and insurance, and properties not directly related to the service functionalities, such as security and trust.

The enrichment of WS descriptions based on WSDL by means of semantic annotation languages and ontologies (OWL-S, WSMO, SAWSDL) has been proposed to improve automation and precision of WS discovery and composition.

L. Baresi, C.-H. Chi, and J. Suzuki (Eds.): ICSOC-ServiceWave 2009, LNCS 5900, pp. 546–560, 2009.

Semantic annotations can be likewise exploited to support the description of NFPs and to improve ranking algorithms, as shown also by recent works such as [3,8,7,9,11,14]. Automated reasoning techniques based on semantic annotations are particularly suitable to mediate between different terminologies and data models considering the semantics of the terms used in the descriptions as defined by means of logical axioms and rules (e.g., at class-level, by making explicit that, in a given domain, the property *BasePrice* is equivalent to the property *ServicePrice*, or, at instance-level, by making explicit that a *fire insurance* is part of a *blanket insurance*). However, the crisp nature of matching-rules based on logical reasoning conflicts with the need to support ranking algorithms with more practical matching techniques; moreover, many reasoners show poor effectiveness when dealing with non trivial numeric functions (e.g., weighted sums) which are needed to manage more properties at the same time. As a consequence logic-based and algorithmic techniques need to be combined to provide for an effective and flexible approach to service ranking.

In this paper we present an effective and flexible approach to NFP-based ranking of Semantic WSs, which is based on PCM-compliant NFP descriptions. PCM (Policy Centered Meta-model) [6] is a meta-model that supports the description of the NFPs offered by a service, as well as requested by a user, by means of NFP expressions; NFP offers and requests are aggregated in sets called *Policies* to capture business scenarios by aggregating interdependent properties. A purpose of the PCM is to act as an intermediate and integrating meta-model that maps to significant subsets of popular languages (e.g., WSLA [10] and WS-Policy [17]).

The NFP-based WS ranking consists of a four-phase process: a **property matching phase** that identifies the NFPs in the offered policies that match with the NFP in the requested policy; a **local property evaluation phase** that computes a matching degree for each couple of matching NFPs; a **global policy evaluation phase** that computes a global matching degree between the requested policy and each offered policy; finally, services (and policies) are sorted according to their global matching degrees during a **policy ranking phase**. The ranking process has been tested by implementing the PoliMaR (Policy Matchmaker and Ranker) tool covering a significant set of NFP expressions for both requested and offered NFPs. Experimental results demonstrate the feasibility and the effectiveness of the approach.

The peculiar features of the proposed approach are the following:

- **expressivity**, by supporting rich descriptions of requested and offered NFPs addressing qualitative properties by mean of logical expressions on ontology values and quantitative properties by mean of expressions including ranges and inequalities;
- **generality**, by allowing semantic-based mediation in the matching phase with NFP descriptions based on multiple ontologies;
- **extensibility**, by supporting parametric property evaluation by customizing functions associated with operators;

– **flexibility**, by allowing incomplete specifications (i.e., unspecified properties and values in NFP requests and offers).

The paper is organized as follows: the problem of NFP-based WS ranking and the issues related to the NFP representations expressiveness are discussed in Section 2 through the introduction of a running example; Section 3 describes the PCM features and the ranking problem; Section 4 presents the approach to policy matchmaking and ranking; experimental results evaluating the scalability of the approach are discussed in Section 5; finally, the comparison with related works (Section 6) and concluding remarks (Section 7) end the paper.

2 Problem Context and Motivation

The problem of ranking a set of services can be defined as follows: given a set of service descriptions $S = \{s_1, ..., s_n\}$, and a specification R of non-functional requirements, define a sorting on S based on R. In this paper we assume that a set of services, namely *eligible services*, are identified by a discovery engine on the basis of their functional properties (FPs); the non-functional property descriptions of the eligible services form the set S to be ranked.

As discussed in [6], the distinction between FP and NFP is often ambiguous and no rules are available to qualify a property as FP or NFP. From our point of view this is a consequence of the fact that functional or non-functional is not an intrinsic qualification of a property, but it depends on the application domain and context. For example, the service location could be classified as a FP for a logistic service and as a NFP for a payment service. Moreover, from the requester perspective, the classification of requested properties as FP or NFP might be of little interest and counterintuitive. The requested properties represent the user preferences and could be mandatory or optional. In this paper, we adopt the proposal described in [1]. From the requester perspective, we considered hard and soft constraints to distinguish between the properties that are specified as mandatory or optional in a service request. From the provider perspective, we consider FPs those properties of a service that strictly characterize the offered functionality (e.g., service location for a shipment service) and NFPs those properties that do not affect or affect the offered functionality only marginally (e.g., service location for a payment service). Then, in order to support the matching between requested and offered properties, FPs and NFPs are mapped with hard and soft constraints respectively.

To illustrate the main aspects that need to be covered when dealing with NFP-based ranking, let us consider a running example based on the discovery scenario in the logistic domain presented in [1]. The scenario derives from an analysis of the logistic operator domain conducted within the Networked Peers for Business (NeP4B) project[1] and has inspired one of the current discovery scenarios in the Semantic Web Service Challange[2]. In this scenario, several logistic operators offer

[1] http://www.dbgroup.unimo.it/nep4b
[2] http://sws-challenge.org/wiki/index.php/Scenarios

one or more services (e.g., freight transport, warehousing) each one characterized by offered NFPs. A set of relevant NFPs in this domain are: (i) *payment method*: how the user can perform the payment; (ii) *payment deadline*: the maximum number of days that the user can wait to perform the payment after the service fulfilment; (iii) *insurance*: the type of compensation in case of failure applied to the service; (iv) *base price*: the amount of money to be paid to get the service; (v) *hours to delivery*: the number of hours required for the service fulfilment.

A freight transport service provider can specify the following NFP offered by its service: *"I offer a service that performs freight transportation in 24-48 hours with a base price equal to 100 Euros. I accept carriage paid payment within 45 days and I offer a blanket insurance on the transportation".*

Users in this context might want to formulate quite rich requests to identify the best service according to their own stated criteria. An example of user request, written in natural language, is the following: *"I am interested in a service to perform a freight transportation in one or two days with a price less than or equal to 120 Euros. Moreover, I would like to use a service allowing, at least, a 15-days postponed payment with carriage paid or carriage forward payment method. Finally, I prefer a service offering a fire insurance or any insurance type that includes it".*

A detailed discussion about the expressiveness of languages and models needed to represent NFPs in order to support WS discovery can be found in our previous work [6]. Here, we just observe that: NFPs may refer to either numerical values (e.g., 120) or world objects (e.g., *fire insurance*); some values can be undefined in the requests (e.g., in a lower bound expression such as *price less than or equal to 120 Euros*) or even in the offered NFPs (e.g., in a range expression such as *24-48 hours*); a user expresses constraints on different NFPs at a same time and may want to express preferences about what should be considered more important.

3 PCM-Compliant NFP Descriptions and Policy Ranking

The Policy Centered Meta-model (PCM) has been developed to address NFP representation and service ranking. In the PCM, requested or offered NFPs are grouped into policies; offered policies are associated with services and defined by an applicability condition for the properties composing them. As a result, the problem of ranking a set S of n WSs can be reformulated as the problem of ranking a set P of k policies on the basis of a requested policy RP, with $k \geq n$ since a service can be offered with more policies.

The PCM is defined by a language-independent conceptual syntax, whose semantics is defined by an ontology. Two concrete syntaxes of the PCM are provided in OWL and WSML. Since the implementation of the ranker presented in the paper uses the WSML language, in the following we will use a WSML-like notation with small variants to shorten the descriptions. In this paper we provide for a brief description of the PCM by means of examples, focusing on the elements that are more relevant in the ranking process. The reader can refer to [6] for details and formal definitions.

The following is an example of a section of a service description in the context of logistics operators.

```
< ONTOLOGY HEADING: namespace declaration, ontology import...>

instance premiumPolicy memberOf pcm#Policy
    pcm#ServiceReference hasValue
    ''http://www.itis.disco.unimib.it/research/ontologies/WSSouthItalyOrdinaryTransport.wsml''
    pcm#hasCondition hasValue premiumCondition
    pcm#hasNfp hasValue [off.BasePrice1 memberOf nfpo#BasePrice]
    pcm#hasNfp hasValue [off.PaymentDeadline1 memberOf nfpo#PaymentDeadline]
    pcm#hasNfp hasValue [off.HoursToDelivery1 memberOf nfpo#HoursToDelivery]
    pcm#hasNfp hasValue [off.PaymentMethod1 memberOf nfpo#LogisticPaymentMethod]
    pcm#hasNfp hasValue [off.Insurance1 memberOf nfpo#LogisticInsurance]
    ...
```

The term *instance* introduces the name of the instance of the ontology, and *memberOf* specifies the class it belongs to. The namespace *pcm#* is for the PCM ontology and *nfpo#* for a domain-specific NFP ontology extending the PCM[3]. A policy is identified by a URI and associated with one or more WSs by *ServiceReference*. A *PolicyCondition* defines the requirements a client profile should fulfill to select that policy (e.g., the *premiumPolicy* is for frequent clients that subscribed for a significant number of shipments per years); NFPs are represented in the PCM by *PolicyNfp*s and are expressed in terms of, possibly external, ontologies (e.g., *nfpo#BasePrice*). A NFP is specified by means of a *NfpExpression* that is characterized by a *ConstraintOperator* and by a set of attributes that depends on the constraint operator type. Different examples, referred to *premiumPolicy*, are synthetically represented on the right-hand side of Figure 1.

To explicitly take into account the requestor perspective, PCM introduces the concept of *RequestedPolicy* that is composed of *Request*s stating what values are acceptable for a certain property, and expressing the relevance of each required property. *Request*s are therefore defined extending *PolicyNfp*s with the property *hasRelevance*, whose range is a rational within [0..1]. The requests formulated in the scenario in Section 2 are collected in the *LOReqPolicy1* in the left-hand side of Figure 1.

PCM makes distinction between qualitative and quantitative NFP expressions. Qualitative expressions refer to objects (their values are instances of given domain ontologies) and are further classified in *SetExpression*s and *CustomExpression*s. Quantitative expressions assume numeric values, whose measurement units is specified by a *unit* term; quantitative expressions are further classified into *SingleValueExpression*s and *RangeExpression*s.

Figure 2 shows the properties that characterize each class of NFP expressions, the respective ranges, and a set of *built-in* constraint operators, which are also exploited by the ranker proposed in this paper (the set of operators is extensible by mean of standard ontology import mechanisms). As for *SetOperator*s, PCM

[3] All ontologies are available on-line at`http://www.itis.disco.unimib.it/research/ontologies`

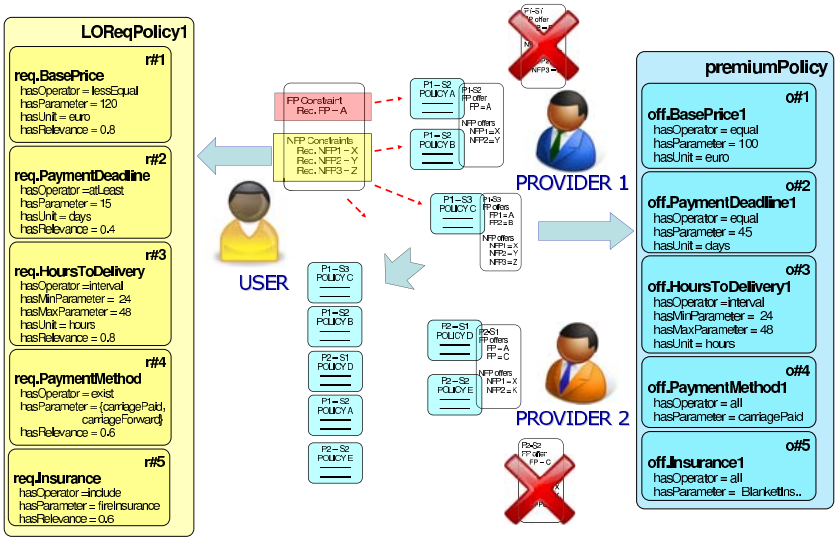

Fig. 1. The scenario revisited according to the PCM

introduces (i) the two standard logical operators *all* and *exist* with their logical meanings, and (ii) the operator *include*. Intuitively, a *include*-based request (e.g., *I need an insurance including fire insurance*) asks for values that *logically* include the selected values (e.g., *a blanket insurance*); logical inclusion is looked up by exploring hierarchical properties of different nature (e.g., *part-of*, *topological inclusion*). The set of *CustomOperator*s allows domain experts to introduce other operators to deal with object values. As an example, a request based on *semanticDistance* operator may ask for values that are semantically close to the specified one.

	Expression Class	Attribute	Range
Qualitative	Set Expression	pcm:hasOperator	pcm:SetOperator // {all, exist, include}
		pcm:hasParameters	URI *(instance)*
	Custom Expression	pcm:hasOperator	pcm:CustomOperator // e.g. semanticDistance
		pcm:hasParameters	-
Quantitative	Single Value Expression	pcm:hasOperator	pcm:BinaryOperator // {≥↑, ≥↓, ≤↑, ≤↓}
		pcm:hasParameter	Lit *(numerical value)*
		pcm:hasUnit	pcm:Unit
	Range Expression	pcm:hasOperator	pcm:TernaryOperator // {interval}
		pcm:hasMinParameter	Lit *(numerical value)*
		pcm:hasMaxParameter	Lit *(numerical value)*
		pcm:hasUnit	pcm:Unit

Fig. 2. Characterization of the four NFP Expression classes

As for quantitative expressions, PCM defines a set of operators that supports the most common clauses for numeric values (e.g., inequalities and ranges). Beside the standard binary operator = (*equal*), and ternary operator *interval* that fixes a minimum and a maximum value, new operators have been introduced to increase expressiveness of inequalities. These operators are: (i) $\geq\uparrow$ (*greaterEqual*) to specify a lower bound, so that the highest possible value is better; (ii) $\geq\downarrow$ (*atLeast*) to specify a lower bound, so that the lowest possible value is better; (iii) $\leq\downarrow$ (*lessEqual*) to specify an upper bound, so that the lowest possible value is better; (iv) $\leq\uparrow$ (*atMost*) to specify an upper bound, so that the highest possible value is better. Observe that binary operators are followed by one parameter and ternary operators by two parameters.

The formal discussion of the relationships between the PCM and other well-recognized languages such as WS-Policy and WSLA is out of the scope of this paper. However, we can show that significant sections of WSLA and WS-Policy descriptions, and in particular, the significant subset for WS ranking, are PCM compliant, making our ranking techniques applicable to these languages.

WSLA is used by service providers and service consumers to define service performance characteristics. The commitment to maintain a particular value for a NFP (i.e., *SLAParameter*) is defined in the *Service Definition* section of a WSLA specification through the *Service Level Objective*s. A *Service Level Objective* is defined by an *Expression* based on quantification-free first order logic; the language includes ground predicates and logic operators, and easily maps to predicate logic. The simplest form of a logic expression is a plain predicate that can be mapped to a *PolicyNfp* characterized by an expression where the constraint operator and the parameter represent the *Type* (e.g., $=$, $\leq$) and the *Value* (e.g., numerical values) of the WSLA expression, respectively. Complex WSLA expressions are mapped as follows: implications are deleted and the resulting WSLA expression is put into a *disjunctive normal form* (a disjunction of conjunctions of ground predicates) exploiting standard techniques for predicate logic; each WSLA conjunction C is then represented by a *Policy* P composed of *PolicyNfp*s representing the WSLA predicates in C; finally, for all the WSLA conjunctions C containing a predicate occurring in the head of an implication, an applicability condition representing the body of the implication is created for the *Policy* representing C.

WS-Policy is the most cited standard for enriching WSDL files with NFP specifications. A WS-Policy specification is an unordered collection of zero or more *policy alternatives* defined as *assertions* stating behaviors or requirements or conditions for an interaction. A WS-Policy alternative can be mapped to a *Policy*. WS-Policy assertions can be mapped to a *PolicyNfp* specification. "*And*" aggregations of WS-Policy assertions can be mapped to sets of *PolicyNfp*s in a *Policy*. "*Or*" aggregations of WS-Policy assertions can be mapped to multiple *PolicyNfp*s. WS-Policy specifications of nested policies need more articulated descriptions of ontology values (parameters) by means of *CustomExpression*s, which is not straightforward but it is supported by the WSML/OWL data-model.

4 Policy Matchmaking and WS Ranking: Combining Semantics and Algorithms for Policy Evaluation

The WS ranking process is composed of four phases: (i) **property matching phase**: for each *Request*, identify the set of *PolicyNFP*s to be evaluated; (ii) **local property evaluation phase**: for each identified *Request/PolicyNFP* couple, evaluate how the offered property satisfies the requested one - results are in range [0, 1]; (iii) **global policy evaluation phase**: for each policy, evaluate the results of the previous phase to compute a global satisfaction degree - results are values in range [0, n]; (iv) **policy ranking phase**: policies are ranked according to their global satisfaction degree.

The ranking process has been implemented in the PoliMaR tool. Figure 3 shows the components of the tool and their connection to external tools. As discussed above we assume that: (i) a number of PCM compliant policies are stored into an ontology repository; (ii) the eligible services are used by the *Ontology Loader* to make the reasoner load the knowledge needed to perform the ranking process; (iii) if NFPs are specified according to another model, the *PCM Wrapper* is used to transform the original descriptions into PCM-based descriptions.

The Matching Evaluator. The property matching phase is performed by the matching evaluator. According to the approach based on decoupling the matching phase from the evaluation phase, the matching evaluator has two goals: (i) discover the *PolicyNFP*s that match against the *Request*s; and (ii) retrieve all the data concerning these NFPs to support the other components in the evaluation tasks.

A mediator-centric approach is used to achieve these goals, according to the WSMO asset that exploits different kinds of mediators to solve semantic mismatches. In this case, the mediation is defined by logic programming rules. A first set of rules mediates among the possibly different ontologies on which offered and requested NFPs are based on. These rules retrieve a set of matching couples exploiting subclass relations. The following example of matching rule

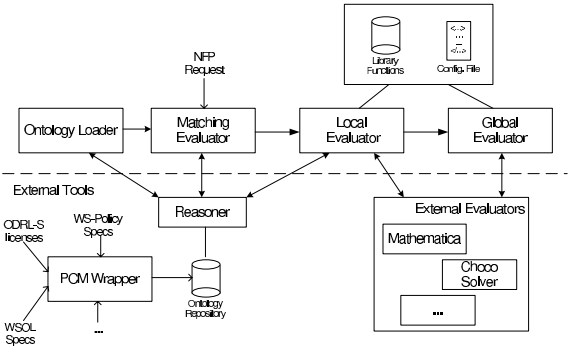

Fig. 3. The overall architecture of the PoliMaR tool

specifies that a request and an offer match if they belong to specific subclasses of *PolicyNfp*.

```
axiom BasePriceMatching
    definedBy
        matchCouple(?request,?nfp,baseprice) :–
            (?request memberOf nfpo#BasePriceRequest) and
            (?nfp memberOf nfpo#BasePrice) or
            (?nfp memberOf nfpo#ServicePrice)
```

A second set of rules is defined to retrieve the data related to the set of matched NFPs. The reasoner exploits standard mechanisms of variable binding to explore the PCM-compliant ontologies and retrieve the information for each NFP. Moreover, retrieval of such data is not straightforward because non monotonic rules are exploited to put results in a kind of normal form (e.g., some quantitative properties might be defined through binary operators in some policies and ternary operators in other policies). Formally, the results of the matching evaluator are provided by executing a query and consist in a table with all the relevant information necessary for the next phases. An example is sketched in Table 1.

The Local Evaluator. The local evaluator takes a result table, like the one shown in Table 1, as input. We call *matching couple* every couple <Request, PolicyNfp> in the table. The output of the local evaluator is a *local satisfaction degree* (*LD* for short) for each couple. A LD is expressed by a value in the range [0..1], where 0 means "no match" and 1 means "exact match". In our approach, the matching degree for each matching couple is calculated by a function that takes the form $e\left(cop_r, cop_o, norm\left(v_r\right), norm\left(v_o\right)\right)$, where cop_r and cop_o are the requested and offered constraint operators; $norm\left(v_r\right)$ and $norm\left(v_o\right)$ are the requested and offered normalized values (i.e., values after a unit conversion when necessary). Observe that values of qualitative properties are objects in the ontology, which means that a default *object* unit can be considered; moreover, a qualitative property can refer to a set of objects (a property can assume multiple values).

The set of local evaluation functions is stored in a *Library Functions*. Links between functions and constraint operators are defined by a *configuration file* to supply a flexible and extensible solution. This is a crucial advantage to address

Table 1. A fragment of the table displaying the matching phase results

Policy/Req.Policy	NFP	Operator	MinParameter	MaxParameter	Unit	Relev.
LOReqPolicy1	**req.BasePrice**	lessEqual	120	null	euro	0.8
premiumPolicy	**off.BasePrice1**	equal	100	null	euro	-
goldPolicy	**off.BasePrice2**	interval	80	150	euro	-
...	**...**	...	...	...	...	-
...	...	...	...	...	...	...
LOReqPolicy1	**req.Insurance**	include	fireInsurance	null	null	0.6
premiumPolicy	**off.Insurance1**	all	blanketInsurance	null	null	-
silverPolicy	**off.Insurance3**	all	fireInsurance	null	null	-
	...	**...**	...	...	...	-

the development of effective tools. In the current implementation, a number of functions for matching qualitative and quantitative properties have been developed; the configuration file allows for links to new operators or to new tailored functions. Qualitative and quantitative NFPs need to be handled in a different way. The quantitative local evaluation functions currently in use have been introduced in [4]. As for qualitative local evaluation functions, the reasoner needs to be recalled to exploit inference mechanisms based on the NFP domain ontologies in use. In particular, we considered the *all* operator in the *PolicyNfp*s and the operators *all*, *exist* and *include* in the *Request*s.

The operators *all* and *exist* have standard logical meaning; basic inferences based on identities need to be considered for both the operators (e.g., when a service ships to "Italy" and the request ask for a service shipping to "Italia"). Let V be the set of requested values and O the set of offered values. For the *Request*s based on the *all* operator, we evaluate a LD d within the range [0..1]. If $V \subseteq O$, then $d = 1$; If $V \cap O = \emptyset$, then $d = 0$. If $V \not\subseteq O$ and $V \cap O \neq \emptyset$, then $d = |V \cap O|/|V|$. For the *Request*s based on the *exist* operator, the LD d can assume the value 0 or 1. If $V \cap O = \emptyset$, then $d = 0$. If $V \cap O \neq \emptyset$, then $d = 1$.

*Request*s specified through an *include* operator need to consider specific dependencies among the values specified in the *PolicyNfp*s. In the running example discussed in Section 2 the insurance ontology defines the *fireInsurance* as a *partOf* of the *blanketInsurance*. Therefore, policies offering a *blanketInsurance* satisfies requests asking for services that offer *fireInsurance*. A mediator-centric approach is used. In the rule ontology, where mediation rules are stored, the axiom for the example states that the *partOf* relation among insurance is to be considered as an inclusion relation (see the listing below).

```
axiom insuranceInclusion
    definedBy
        include(?X,?Y) :-
            (?X memberOf ins#Insurance) and (?Y memberOf ins#Insurance) and ins#partOf(?X,?Y)
```

The local evaluation function for inclusion operators expands the set O of offered values according to the transitive closure for the inclusion relations involving offered and requested values. Then, LD is calculated as for the *all* operator.

The Global Evaluator. The global evaluator takes the set of LDs evaluated for each matching couple as input, and provides a *global satisfaction degree* (*GD* for short) as output. GD provides information about how much a *Policy* matches a *RequestedPolicy* and it is computed by taking into account the relevance associated with each *Request* in the *RequestedPolicy*. Different global evaluation functions can be defined and stored into the *Library Functions*. A possible function is the weighted sum of the LDs, where weights are the relevance values of the corresponding Requests. The global evaluator ranks the *Policies* according to their GD.

Observe that our approach is tolerant w.r.t. the incompleteness of the NFP specifications (i.e., *Requests* whose matching *PolicyNfps* are not specified in a *Policy*). In fact, the more *Requests* in the *RequestedPolicy* match with some

PolicyNfps for a given *Policy*, the greater the GD is; however, the evaluation does not crash when a *Request* in the *RequestedPolicy* does not match with any *PolicyNfps* of a given *Policy*.

5 Experimental Results

The current version of the PoliMaR tool has been implemented using Java JDK 1.6.0 update 11 for Linux 64 bit and provides all the components described in Figure 3 except for the PCM Wrapper. All the ontologies are represented in the WSML language. The ranker uses KAON2 (v2007-06-11) as ontology repository and reasoner and the Wsml2Reasoner API (v0.6.1) to communicate with the reasoner. PoliMaR is now part of the GLUE2 discovery engine [2] available at http://glue2.sourceforge.net.

The current implementation of PoliMaR has been tested to evaluate the scalability and the efficiency of the matching and evaluation components. The evaluation activity has been performed using an Intel Core2 Q6700 2.66 Ghz with 2GB RAM and Linux kernel 2.6.27 64 bits. Due to the lack of large and accessible sets of NFP descriptions to derive PCM-based descriptions, the experiment has been carried out starting from a set of randomly generated descriptions that consists of about 500 policies. The generated test set is a combination of the properties discussed in Section 2 to form policies that are described according to the *NFPO* ontology; constraint operators and parameters are selected randomly according to the ranges specified in the *NFPO*.

The *RequestedPolicy* (TRP) represented in Figure 1 was used as testbed. It is composed of three quantitative requests $r\#1$, $r\#2$, and $r\#3$; and two qualitative requests $r\#4$, and $r\#5$. Observe that $r\#4$ and $r\#5$ are based on two different constraint operators, namely *all* and *include*, that require different reasoning tasks for the evaluation. The performed tests were:

- TEST 1: Measurement of the overall execution time in the cases of single and multiple file storage;
- TEST 2: Analysis of the execution-time distribution between reasoning and algorithmic computation for single file storage;
- TEST 3: Analysis of the execution-time distribution among the ranking phases (matching, local and global evaluation) for single file storage;
- TEST 4: Measurement and comparison of the overall execution time with increasing complexity in the requested policy for single file storage.

TEST 1 (Figure 4a) highlights that: (i) the multiple file approach is efficient only for small numbers of policies. Moreover, the KAON2 reasoner was able to manage at most 136 WSML files containing a policy each; (ii) the required time increases exponentially for the multiple-file approach and polynomially for the single-file approach; (iii) there is an amount of time (approximately 5 seconds) that is independent of the input. It represents the time required to invoke the reasoner through the WSML2Reasoner API. The conclusion that can be driven from this first set of tests is that semantic tools available today make the single

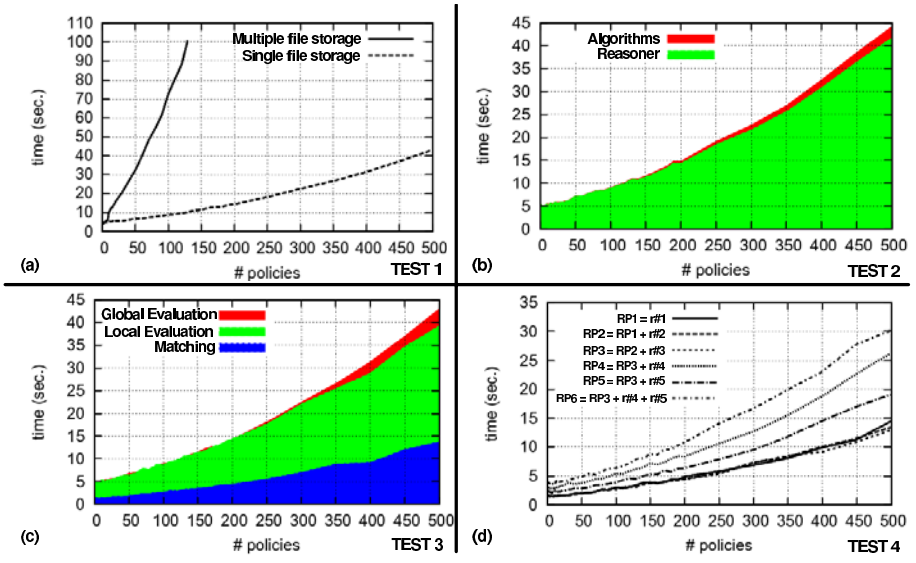

Fig. 4. Evaluation Tests

file approach compulsory. Ongoing research on large scale distributed reasoning might overcome this limit in the future.

TEST 2 (Figure 4b) highlights that the bottleneck for our evaluation is represented by the reasoner: the time required for the evaluation of quantitative NFPs and the global evaluation phase is very short.

TEST 3 (Figure 4c) highlights that: (i) the time required for the execution of the global evaluation phase does not influence significantly the evaluation time; (ii) the time used for the local evaluation phase is twice as long as the time for the matching phase.

TEST 4 has been executed considering six combinations of the single requests forming the TRP with increasing degrees of complexity. The first three requested policies were composed of quantitative requests only: $RP1$ was composed of $r\#1$ (written $RP1 = r\#1$); $RP2 = RP1 + r\#2$; $RP3 = RP2 + r\#3$. The next three requested policies considered also qualitative requests: $RP4 = RP3 + r\#4$ (the *all* constraint operator is used); $RP5 = RP3 + r\#5$ (the *include* constraint operator is used); $RP6 = RP3 + r\#4 + r\#5$ (both *all* and *include* are used).

The results of TEST 4 (Figure4d) highlight that: (i) the number of quantitative NFP constraints marginally affects the evaluation time (RP1, RP2 and RP3 show similar evaluation times); (ii) the evaluation of a qualitative NFP expressed with the *all* operator requires more time than one expressed with the *include* operator. Considering a Policy Repository with 500 policies, the introduction of an *all* constraint determines an increment of 12 seconds, instead the inclusion of an *include* constraint determines an increment of 4 seconds.

6 Related Work

Many "non-semantic" approaches to NFP specification and monitoring exist. The most relevant among them are WSLA [10] and WS-Policy [17]. The mappings between these languages and the PCM have been discussed in Section 3. Current standards for semantic descriptions of services (e.g., WSMO [5] and OWL-S [15]) cover only marginally the specification of NFPs. They basically adopt attribute-value descriptions. A comparison of PCM with the several proposals (e.g., [8,11,9,14]) that try to overcome this current limitations, was discussed in [6]. Relevance and applicability conditions are distinctive characteristics of the PCM. The definition of combined offers can also be considered a distinguishing characteristic of PCM, since only [11] and [9] provide limited support by allowing association of more QoS offers with an OWL-S service profile.

Considering the classification proposed in [19], our approach to NFP-based ranking of WSs can be classified as a *policy-based* solution for NFP descriptions of Web Services that allows for *ontology-based preference modeling*. Similar approaches are presented in [7,3,13,16,18,12,8]

An hybrid architecture for WS selection based on DL reasoners and Constraint Programming techniques is proposed in [7]. An extension of WS-Policy with ontological concepts to enable QoS-based semantic policy matching is presented in [3]. Approaches for the WS selection based on the normalization of QoS values are described in [13,16]. A NFP-based service selection approach that modifies the Logic Scoring Preference (LSP) method with Ordered Weighted Averaging (OWA) operators is proposed in [18]. A framework for WS selection that combines declarative logic-based matching rules with optimization methods is described in [12]. A WSMO-based hybrid solution to WS ranking based on the usage of axioms for requested and offered NFPs is defined in [8].

The comparison with these approaches is carried out focusing on the features described in Section 1: (i) *expressive NFP descriptions*; (ii) *semantic-based mediation*; (iii) *parametric NFP evaluation*; (iv) *tolerance to unspecified NFP*; (v) *experimental results*. Table 2 reports the results of the comparison (*yes/no* is used to show whether the approach achieves the requirement, and *low, average, high* to indicate at what level the approach reaches the requirement). The result of the comparison is that, among the considered approaches, only [12] presents an evaluation activity executed on a large number of policies. Test activities

Table 2. Comparison of NFP-based Web service ranking approaches

	Expr. Desc.	**Mediation**	**Param. Eval.**	**Unspec. NFP**	**Experiment**
Garcia et al. 2007 [7]	low	no	no	no	no
Chaari et al. 2008 [3]	low	no	no	no	no
Liu et al. 2004 [13]	low	no	no	yes	low
Wang et al. 2006 [16]	low	no	no	yes	low
Yu et al. 2008 [18]	low	no	no	yes	low
Lamparter et al. 2007 [12]	low	yes	no	yes	high
Garcia et al. 2008 [8]	high	yes	yes	no	low
Our Approach	**yes**	**yes**	**yes**	**high**	**high**

demonstrates that the semantic service selection described in [12] is more efficient. However, that approach is based on simpler NFP descriptions. Only the *equal* operator is allowed in the definition of qualitative NFP and quantitative NFPs defined as range of values are not considered. Moreover, the approach is less extensible and flexible since the algorithms are hard-coded.

The consideration of high-expressive NFP descriptions and the definition of parametric NFP evaluations are provided only by [8]. The proposed exploitation of axioms support complex and conditioned NFP definitions (e.g., if the client is older than 60 or younger than 10 years old the invocation price is lower than 10 euro). Our proposal differs for four different aspects. First, we support assertions about properties with undefined values by specifying them with a range of possible guaranteed values. This supports the evaluation of offers with some unspecified values. Second, we decouple the evaluation of policy/request matching and applicability conditions. This supports information retrieval without forcing the requester to know and specify all the information required to evaluate the applicability conditions. In case of incomplete requests, the user is involved to evaluate the actual applicability conditions. Third, our descriptions can be obtained by wrapping existing specifications defined in WSLA and WS-Policy. Fourth, our approach has been tested against a significant set of NFP descriptions.

7 Concluding Remarks

This paper represents an effort toward the development of feasible and practical ranking tools. The proposed solution overcomes some limits of the current approaches by combining high expressivity in NFP descriptions with a rich and extensible set of operators and evaluation functions. Experimental results show the effectiveness of the approach when dealing with a significant number of policy specifications, even if some desirable improvements emerged as necessary to reach high efficiency and increase performance.

Currently, our approach assumes the availability of PCM-based descriptions but we are working to fill these limitations by developing a wrapper to retrieve data from heterogeneous service descriptions defined using different languages and formats (e.g., WSLA and WS-Policy, but also RDF or generic XML files) and use them to define PCM-based Policies to be processed by PoliMaR. Moreover, our current research focuses on performance improvements by means of caching strategies for qualitative property evaluation. Future work will deal with the development of tools to support users in writing NFP descriptions and evaluation functions.

References

1. Carenini, A., Cerizza, D., Comerio, M., Della Valle, E., De Paoli, F., Maurino, A., Palmonari, M., Sassi, M., Turati, A.: Semantic web service discovery and selection: a test bed scenario. In: proc of the Int. Workshop on Evaluation of Ontology-based tools and the Semantic Web Service Challenge (EON&SWS-Challenge) (2008)

2. Carenini, A., Cerizza, D., Comerio, M., Della Valle, E., De Paoli, F., Maurino, A., Palmonari, M., Turati, A.: Glue2: a web service discovery engine with non-functional properties. In: Proc. of the Eur. Conf. on Web Services, ECOWS (2008)
3. Chaari, S., Badr, Y., Biennier, F.: Enhancing web service selection by qos-based ontology and ws-policy. In: Proc. of the Symp. on Applied computing, SAC (2008)
4. Comerio, M., De Paoli, F., Maurino, A., Palmonari, M.: Nfp-aware semantic web services selection. In: Proc. of the International Enterprise Distributed Object Computing Conference (EDOC), Annapolis, USA, pp. 484–492 (2007)
5. de Bruijn, J., Lausen, H., Polleres, A., Fensel, D.: The web service modeling language: An overview. In: Sure, Y., Domingue, J. (eds.) ESWC 2006. LNCS, vol. 4011, pp. 590–604. Springer, Heidelberg (2006)
6. De Paoli, F., Palmonari, M., Comerio, M., Maurino, A.: A Meta-Model for Non-Functional Property Descriptions of Web Services. In: Proc. of the Int. Conference on Web Services (ICWS), Beijing, China (2008)
7. García, J.M., Ruiz, D., Ruiz-Cortés, A., Martín-Díaz, O., Resinas, M.: An hybrid, qos-aware discovery of semantic web services using constraint programming. In: Krämer, B.J., Lin, K.-J., Narasimhan, P. (eds.) ICSOC 2007. LNCS, vol. 4749, pp. 69–80. Springer, Heidelberg (2007)
8. Garcia, J.M., Toma, I., Ruiz, D., Ruiz-Cortes, A.: A service ranker based on logic rules evaluation and constraint programming. In: Proc. of the Non Functional Properties and Service Level Agreements in SOC Workshop, NFPSLASOC (2008)
9. Giallonardo, E., Zimeo, E.: More semantics in qos matching. In: Proc. of Int. Conf. on Service-Oriented Computing and Application, SOCA (2007)
10. Keller, L.H., The, A.: wsla framework: Specifying and monitoring service level agreements for web services. J. Netw. Syst. Manage. 11(1), 57–81 (2003)
11. Kritikos, K., Plexousakis, D.: Semantic qos metric matching. In: Proc. of the Eur. Conf. on Web Services (ECOWS), pp. 265–274 (2006)
12. Lamparter, S., Ankolekar, A., Studer, R., Grimm, S.: Preference-based selection of highly configurable web services. In: Proc. of the Int. Conf. on World Wide Web (WWW), pp. 1013–1022 (2007)
13. Liu, Y., Ngu, A.H., Zeng, L.Z.: Qos computation and policing in dynamic web service selection. In: Proc. of the Int. World Wide Web conference on Alternate track papers and posters (WWW-Alt), New York, NY, USA (2004)
14. Maximilien, E., Singh, M.P.: A framework and ontology for dynamic web services selection. IEEE Internet Computing 08(5), 84–93 (2004)
15. OWL-S. Semantic Markup for Web Services (2003), `http://www.daml.org/services/owl-s/1.0/owl-s.html`
16. Wang, X., Vitvar, T., Kerrigan, M., Toma, I.: A qos-aware selection model for semantic web services. In: Dan, A., Lamersdorf, W. (eds.) ICSOC 2006. LNCS, vol. 4294, pp. 390–401. Springer, Heidelberg (2006)
17. Ws-Policy. Web Service Policy 1.2 - Framework (2006), `http://www.w3.org/Submission/2006/SUBM-WS-Policy-20060425/`
18. Yu, H.Q., Reiff-Marganiec, S.: A method for automated web service selection. In: Proc. of the Congress on Services (SERVICES), pp. 513–520 (2008)
19. Yu, H.Q., Reiff-Marganiec, S.: Non-functional property based service selection: A survey and classification of approaches. In: Proc. of the Non Functional Properties and Service Level Agreements in SOC Workshop, NFPSLASOC (2008)