Découverte Personnalisée de Services Web Sémantiques

OUHAB Abdelkrim, MALKI Mimoun

Laboratoire EEDIS, UDL de Sidi Bel Abbés, ALGERIE.
ouhabsba@gmail.com, malki@univ-sba.dz

Résumé. La découverte des services web sémantiques (SWSs) peut retourner un nombre important de services qui offrent des fonctionnalités similaires. La découverte personnalisée vise à complémenter la phase de découverte par une phase de sélection et de rangement en tenant compte des préférences de l'utilisateur. Nous proposons d'utiliser les propriétés non fonctionnelles (NFPs) pour personnaliser la découverte des SWSs. Dans cet article, nous présentons une ontologie pour la spécification des NFPs, les NFPs offertes (NFPo) et demandées (NFPr) sont des instances de cette ontologie. Dans notre approche, la sélection est basée sur des règles déclaratives énonçant les conditions pour la réussite du matching logique. Ces règles sont basées sur les descriptions ontologiques des individus représentant les NFPo et NFPr. L'utilisateur peut personnaliser la procédure de sélection selon ses préférences en exprimant ses propres règles de matching.

MOTS CLES : Découverte personnalisée de Service Web, Service Web Sémantique, Ontologie, Web Sémantique, NFP, OWL, SWRL.

1 Introduction

La découverte des SWSs est le processus qui permet de localiser tous les services web qui matchent les besoins fonctionnels du demandeur (requête) [1], à cause du nombre important de services qui offrent des fonctionnalités similaires, cette phase peut retourner un nombre important de services comme résultat. En fait, même si un SWS matche les fonctionnalités demandées par l'utilisateur, il peut encore être inacceptable par un autre utilisateur en considérant d'autres paramètres (par exemple, le coût est trop élevé ou la localisation n'est pas adéquate).

La découverte personnalisée des SWSs vise à complémenter la phase de découverte par une phase de sélection et de rangement en tenant compte des préférences de l'utilisateur. Ces préférences sont exprimées comme des contraintes numériques dans le cas des applications utilisant le QoS ou comme un ensemble de paramètres qui caractérisent l'environnement de l'utilisateur (par exemple la localisation) dans le cas des applications utilisant le contexte.

Dans cet article , nous utilisons les NFPs pour personnaliser la découverte des SWSs. Ces NFPs sont spécifiées en utilisant une ontologie générique qu'on a nommée *NFPservice*, les offres de NFPs (NFPo) et les préférences du demandeur (NFPr) sont des instances de cette ontologie. Dans notre approche, la sélection est basée sur des

règles déclaratives énonçant les conditions pour la réussite du matching logique. Ces règles sont basées sur la description ontologiques des individus représentant les NFPo et NFPr.
Notre approche présente les avantages suivants:

- Prise en charge des valeurs quantitatives (QoS) et qualitatives issues des ontologies de domaine (contexte).
- Les NFPs sont exprimées sous forme de contraintes en utilisant des opérateurs (> , < , =).
- L'ontologie *NFPservice* proposée peut être utilisée avec plusieurs descriptions fonctionnelles de SWSs (OWL – S [13], WSMO [14], SAWSDL [15]).
- La procédure de sélection peut être personnalisée sans changer le code : l'utilisateur peut exprimer ses propres préférences de sélection sous forme de règles. Le matching est réalisée par le raisonnement sur la base de connaissances construite à partir de ces préférences.

Cet article est organisé comme suit: la section 2 présente quelques travaux qui visent la personnalisation de la découverte des SWSs. Dans le section 3, nous présentons le modèle des NFPs et de sélection et dans le section 4 l'implémentation du modèle des NFPs en utilisant OWL-DL. La section 5 présentent les règles utilisées pour sélectionner les services et dans la section 6 nous présentons l'implémentation et l'expérimentation. Enfin nous finirons avec la conclusion et les perspectives dans la section 7.

2 État de l'Art

Plusieurs approches pour la personnalisation de la découverte des SWSs ont été proposées. La découverte des services web basée sur le contexte se focalise sur l'utilisateur et son environnement. Dans [2], les services web qui participent dans une composition sont sélectionnés en matchant leur contexte avec le contexte de l'utilisateur. Les éléments du contexte (représenté par un schéma XML) sont définis comme paires (nom, valeur), le matching des paramètres est basé sur l'équivalence logique alors que le matching des valeurs est basé sur l'inclusion. Dans [3], le contexte (localisation) est représenté par une ontologie générique qui peut connectée plusieurs ontologies de domaine. Le matching est basé sur la distance sémantique.

Certaines approches exploitent les propriétés existantes dans la description des SWSs. Dans [4], les paramètres utilisés pour la personnalisation sont groupés dans une ontologie générique, chaque paramètre est un sous concept du concept *ServiceParameter*s du profil OWL-S, les valeurs de ces paramètres sont des instances dans une ontologie de domaine, le matching utilisé est logique. Dans [5], L'algorithme de matching utilise plusieurs filtre pour sélectionner les services en se basant sur les propriétés non fonctionnelles représentées par la région géographique, le QoS, le nom du service et le nom du fournisseur.

Les approches utilisant le QoS sont nombreuses. Dans [6], les auteurs proposent OWL-Q une extension pour OWL-S et un algorithme basé sur les règles pour matcher deux métriques. Dans [7], une approche similaire est présentée, l'ontologie proposée pour le QoS peut être utilisée avec plusieurs descriptions fonctionnelles de SWSs.

L'utilisateur peut spécifier le modèle de matching. Dans [11], les auteurs présentent WSMO-QoS qui raffine la classe des propriétés non fonctionnelles dans WSMO, ils utilisent un algorithme de normalisation pour la sélection d'un service.

D'autres approches ne font pas de distinction entre FPs et NFPs. Dans [8], les auteurs présentent une upper ontologie qui décrit les offres, les requêtes et les services web possédant plusieurs configurations. Dans [9], les auteurs proposent une ontologie pour les WS-Policy et [10] propose une autre pour WS-Agreement, le matching dans ces deux derniers est basé sur ABLE Rule Language (ARL).

Nous présentons la différence entre les approches citées et notre approche (voir Table 1) en se basant sur les dimensions suivantes :

- D_FP_NFP : distinction entre FPs et NFPs.
- V_Qant : prise en charge des valeurs quantitatives .
- V_Qalt : prise en charge des valeurs qualitatives .
- Op : prise en charge des opérateurs (> , < , =).
- Alg_Per: l'algorithme de sélection peut être personnalisé sans changer le code.
- Rank: prise en charge du Ranking.
- Autres_descr: possibilité d'utiliser l'approche avec d'autres descriptions de SWSs comme OWL-S,WSMO ou SAWSDL.

Table 1. Comparaison de notre approche avec les autres.

	Notre approche	[2]	[3]	[4]	[5]	[6]	[7]	[11]	[8]	[9]	[10]
D_FP_NFP	+	+	+	+	+	+	+	+			
V_Qant	+				+	+	+	+	+	+	+
V_Qalt	+	+	+	+	+				+	+	+
OP	+					+	+	+		+	+
Alg_Per	+					+	+		+	+	+
Rank	+	+	+		+		+	+	+		
Autres_descr	+						+				

3 Les Modèles utilisés

3.1 Modèle des NFPs

Inspirés de [7], [8] et [9] , nous proposons un modèle conceptuel pour les NFPs qui pourra être utilisé pour la sélection des services web. Les propriétés non fonctionnelles d'un service sont modélisées par un ensemble de contraintes C (P, V, OP, poids) :

- P représente le paramètre à traiter, il peut être un concept dans une ontologie pour le QoS (Price, ResponseTime, etc.) ou un concept dans une ontologie de contexte (Location, etc.).
- V représente la valeur offerte ou demandée, elle peut être une valeur quantitative ou qualitative (instance ou concept dans une ontologie de domaine).

- Op est un opérateur relationnel (=, < ou >), il est utilisé uniquement dans la requête .Dans le cas d'une offre de NFPs la valeur V est interprétée comme la valeur minimale ou maximale supporté par le fournisseur du service.
- Le poids représente l'importance d'une contrainte de la requête.

Par exemple une contrainte de NFPr peut se présenter comme Cr1 (Price, 200, <, 2) ou comme Cr2 (Location, Algeria, =, 4) alors qu'une contrainte de NFPo peut se présenter comme Co1 (Price, 100, 'NULL', 'NULL') et Co2 (Location, Africa, 'NULL', 'NULL'). Dans ces exemples Price représente un concept dans l'ontologie QoS, Location un concept dans l'ontologie géographie (Geo) alors que Algeria et Africa sont des instances dans l'ontologie géographie.

3.2 Modèle de selection

Dans cet article, nous considérons uniquement le matching et le rangement des NFPo. Etant donné un ensemble de services web qui matchent les propriétés fonctionnelles, le matching entre n'importe quel NFPr (demande) et les NFPo (offres) est décomposé en un ensemble de matching simple entre une une seule contrainte demandée Cr(Pr,OP,Vr,poids) et une contrainte offerte Co(Po, Vo).

Pour calculer le degré de match des paramètres et des valeurs, on se base sur des notions bien connues de matching pour les SWSs comme "plugin" basé sur la subsomption et "exact" basé sur l'équivalence [23].

Calcul du degré de match des paramètres

Le calcul du degré de match des paramètres Po et Pr (en supposant qu'ils sont des concepts dans l'otologie OntoP) est réalisé par la fonction DoMp tel que :

$$DoMp(Po,Pr)=\begin{cases}1 & si \quad OntoP \models equivalentClass(Po,Pr)\\ 0.5 & si \quad OntoP \models subClassOf(Pr,Po)\\ 0 & sinon\end{cases} \qquad (1)$$

Calcul du degré de match des valeurs

Pour le calcul du degré de match des valeurs Vo et Vr, nous utilisons la fonction DoMv. Plusieurs cas se présentent

- Si Vo et Vr sont des valeurs numériques

$$DoMv(Vo,Vr)=\begin{cases}0 & si \quad ((Op="<")et(Vo\geq Vr))ou((Op=">")et(Vo\leq Vr))\\ & \quad ou((Op="=")et(Vo\neq Vr))\\ 1 & si \quad (Op="=")et(Vo=Vr)\\ 1-\dfrac{Vo-\min Vo_i}{\max Vo_i-\min Vo_i} & si \quad (Op="<")et(Vo<Vr)\\ 1-\dfrac{\max Vo_i-Vo}{\max Vo_i-\min Vo_i} & si \quad (Op=">")et(Vo>Vr)\end{cases} \qquad (2)$$

Si (Op = '<' et Vo<Vr) ou (Op = '>' et Vo>Vr) nous utilisons les formules (2) et (3) de l'algorithme de normalisation de [11] tel que $minVo_i$ ($maxVo_i$) représente le minimum (maximum) des valeurs prises par le paramètre Po dans les Co_i à traiter.

- Si Vo et Vr sont des concepts DoMv est définie comme DoMp
- Si Vo et Vr sont des instances dans l'ontologie OntoV

$$DoMv(Vo,Vr) = \begin{cases} 1 & si \quad OntoV \models sameAs(Vo,Vr) \\ 0.5 & si \quad OntoV \models prop(Pr,Po) \\ 0 & sinon \end{cases} \quad (3)$$

prop (Vr, Vo) est une propriété dans l'ontologie OntoV qui exprime en quelque sorte la relation de spécialisation. Par exemple pour les contraintes Cr2 (Location, Algeria, =, 4) et Co2 (Location, Africa, 'NULL', 'NULL') s'il existe un fait Locatedin (Algeria, Africa) dans l'ontologie de domaine géographie alors DoMv=0,5.

On calcule le degré de macth pour une contrainte comme suit :

$$DoMc = DoMp * DoMv \quad (4)$$

Enfin on calcule le degré de macth DoMs pour une NFPo contenant n contraintes Co_i (Po_i, Vo_i) ayant comme degré de match $DoMc_i$ et w_i comme poids tel que $1 \leq i \leq n$ comme suit :

$$DoMs = \sum_{i=1}^{n} DoMc_i * w_i \quad (5)$$

4 La Représentation de l'ontologie

Pour l'implémentation du modèle des NFPs, nous utilisons OWL (Web Ontology Language)[16]. Pour la formulation déclarative des directives de matching sous la forme de règles, nous avons besoin de primitives additionnelles de modélisation non fournis par OWL. Nous nous servons de SWRL [17] (Semantic Web Rule Language). Dans ce travail nous nous limitons à *DL-safe rules* [18] qui un fragment décidable de SWRL et qui est supporté par des moteurs d'inférence tels que KAON2[1]. Le modèle présenté dans la section 3.1 est représenté par une ontologie générique nommée *NFPservice* (voir figure 1). Les NFPo et NFPr sont des instances de l'ontologie générique *NFPservice.* Chaque NFPo est représentée par un fichier .owl, NFPr est représenté par le fichier *request.owl* qui contient en plus les règles de matching.

[1] http://kaon2.semanticweb.org/

Pour des raisons de simplicité, nous présentons l'ontologie *NFPservice* informellement par l'intermédiaire d'un diagramme de classes UML, où les classes UML correspondent aux concepts OWL, les associations UML aux propriétés objet, l'héritage UML à la relation de subconcept et les attributs UML aux propriétés datatype de OWL [19].La propriété *hasConstraint* permet de relier une instance du concept *NFP* à plusieurs contraintes (instances de *Constraint*) , chaque contrainte est reliée à un paramètre (instance de *Parameter*) dont la propriété *hasParameterType* pointe vers un concept dans une ontologie .La propriété *hasValue* permet de stocker la valeur du paramètre , cette valeur peut être de type xsd(string,int,float,…) comme elle peut être un concept ou une instance dans une ontologie , dans ces deux derniers cas *hasValue* contiendra l'URI du concept ou de l'instance. La propriété *hasValueType* relie la contrainte à une instance (owl ou xsd) du concept *ValueType*, nous utilisons cette propriété pour décider quel type de comparaison doit être effectué pour matcher deux valeurs Vo et Vr : comparaison numérique ou comparaison basée sur le raisonnement logique. La propriété *hasOperator* relie une contrainte à une instance (less_than, equals_to , greater_than) du concept *Operator* , cette propriété est utilisée pour savoir si le demandeur préfère la valeur minimale ou maximale parmi les valeurs offertes .La propriété *hasResult* permet de relier une contrainte à une instance (exact , plug_in) du concept *Result* et la propriété *hasDoM* permet de stocker le degré de match pour une contrainte (DoMc). Les valeurs de ces deux dernières propriété sont inférées par les règles *DL-safe rules.* La valeur de la propriété *vMatch* =1.0 dans l'instance exact et 0.5 da l'instance plug_in, ces valeurs peuvent être modifier par le demandeur , par exemple en attribuant la valeur 0 à *vMatch* dans l'instance plug_in une contrainte qui a un match plug_in aura automatiquement un DoMc = 0.

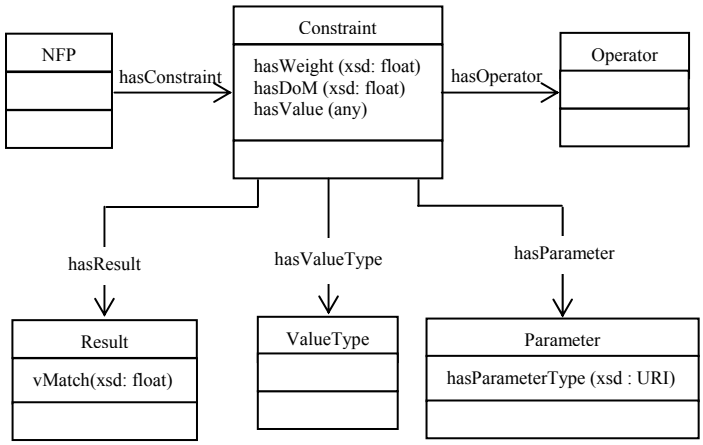

Fig. 1. Ontologie générique *NFPservice*

Pour l'interconnexion de la description NFPo avec la description fonctionnelle FPo nous nous sommes inspirés de [20], nous stockons une correspondance entre l'URI de NFPo et l'URI de FPo dans une base de données composée d'une seule table

nommée *Table_NFP*, cette solution nous permet d'utiliser la description NFPo avec plusieurs descriptions fonctionnelles comme OWL-S,WSMO ou SAWSDL.

5 Sélection De Service

Le modèle de sélection présenté dans la section 3.2 est implémenté en utilisant des règles *DL-safe rules*. L'utilisation des règles au lieu d'écrire du code java nous permet d'assurer une certaine flexibilité. L'utilisateur peut adapter la procédure de sélection selon ses préférences en exprimant ses propres règles de matching. Dans l'exemple de la section 3.2 ,pour la propriété locatedin un utilisateur peut avoir une règle qui spécifie que locatedin(Algeria , Africa) vaut 1, alors qu'un autre peut attribuer la valeur 0.5 ou même 0.

La procédure de sélection peut être décomposée en deux phases principales, la première phase détermine les contraintes qui matchent les contraintes de NFPr, la seconde phase consiste à trier le résultat obtenu à partir de la première phase (voir algorithme 1). La requête NFPr est extraite à partir du profil utilisateur ou elle est générée automatiquement à partir des informations fournies par l'utilisateur. La base de connaissance est construite en important les $NFPo_i$.owl et request.owl dans KB.owl

---

Algorithme 1: Algorithme de sélection

Entrées :
ListeFP : liste des URI des services remplissant la fonctionnalité demandée.
NFPr : représente les NFP demandées.
Sorties :
ListeServiceMatch : liste des services ($URIservice_i$, $DoMs_i$) remplissant NFPr triée par $DoMs_i$

1- Pour chaque service i dans ListeFP extraire son URI_NFPo_i à partir de Table_NFP
2- Construire la base de connaissances KB = NFPr + $NFPo_i$ + ontologies relatives
3- Formuler la requête SPARQL
ListeNFPintermédiaire = SELECT ?S ?C ?P ?PT ?TypeMatchP ?DoMc ?DoMs
WHERE {
?S hasConstraint ?C.
?C hasParameter ?P.
?P hasParameterType ?PT.
?C hasDoM ?DoMc.
?C hasResult ?TypeMatchP.
EVALUATE ?DoMs: = compute_DoMs (?DoMc, ?S)}
4- Construire ListeNFP (URI_NFPo_i, $DoMs_i$) à partir de ListeNFPintermédiaire triée par $DoMs_i$
5- Extraire les $URIservice_i$ correspondant aux URI_NFPo_i à partir de Table_NFP
6- Return ListeServiceMatch ($URIservice_i$, $DoMs_i$)

5.1 Matching

Cette phase consiste à comparer chaque contrainte Co représentée par un paramètre Po et sa valeur Vo d'une NFPo avec chaque contrainte Cr représentée par un paramètre Pr et sa valeur Vr de NFPr. Le résultat obtenu dans cette phase est un ensemble de contraintes Co qui matchent.

Le moteur d'inférence exécute chaque règle dont les critères sont vérifiés et infère automatiquement d'autres faits. Chaque fait inféré asserte un match (exact ou plug_in) entre une Co et une Cr (algorithme1 etape2). Nous avons spécifié différentes règles suivant la valeur de la propriété *hasValueType.* Le matching des valeurs numériques est réalisé par plusieurs règles et dépend de la propriété *hasOperator.* Par exemple la règle suivante permet de matcher deux contraintes numériques Co(Po,Vo) et Cr(Pr,Vr) avec un degré de match exact (Po et Pr sont des concepts dans l'ontologie *OntoP.owl*)

$$\begin{aligned} hasResult(?Co, exact) \leftarrow\ & hasParameter(?Co, ?Po), hasParameterType(?Po, ?PTo), \\ & hasValueType(?Co,xsd), hasValue(?Co,?Vo), hasConstaint(request, ?Cr), \\ & hasOperator(?Cr,less\_than), hasParameter(?Cr,?Pr), hasValue(?Cr, ?Vr), \\ & hasParameterType(?Pr,?PTr), hasValueType(?Cr,xsd), \\ & (?Vo < ?Vr), exact(?PTo, ?PTr) \end{aligned} \quad (6)$$

$$exact(x, y) = \begin{cases} true & si \quad OntoP \models equivalentClass(x, y) \\ false & sinon \end{cases} \quad (7)$$

Pour l'exemple de la section 3.2, cette règle va produire un nouveau fait *hasResult (Co1,exact)* dans la base de connaissance KB qui asserte que Co1 est un exact match. Pour prendre en charge les autres opérateurs (equals_to , greater_than) ainsi que le degré de match plug_in , nous utilisons d'autres règles. Pour les contraintes dont la valeur de *hasValueType* est "owl" , d'autres règles ont été spécifiées. Par exemple la règle suivante permet de matcher deux contraintes Co(Po,Vo) et Cr(Pr,Vr) avec un degré de match exact (Vo et Vr sont des individus dans l'ontologie *OntoV.owl*)

$$\begin{aligned} hasResult(?Co, exact) \leftarrow\ & hasParameter(?Co, ?Po), hasParameterType(?Po, ?PTo), \\ & hasValueType(?Co,owl), hasValue(?Co,?Vo), hasConstaint(request, ?Cr), \\ & hasOperator(?Cr,equals\_to), hasParameter(?Cr,?Pr), hasValue(?Cr, ?Vr), \\ & hasParameterType(?Pr,?PTr), hasValueType(?Cr,owl), \\ & same\_individu(?Vo, ?Vr), exact(?PTo, ?PTr) \end{aligned} \quad (8)$$

$$same\_individu(x, y) = \begin{cases} true & si \quad OntoP \models sameAs(x, y) \\ false & sinon \end{cases} \quad (9)$$

Dans l'exemple de la section 3.1 (Cr2 et Co2) les valeurs de l'attribut Location ne représentent pas le même individu dans l'ontologie Geo.owl. Dans ce cas le modeleur peut spécifier le comportement de matching adapté aux besoins du client.

La règle 10 peut être utilisée pour réaliser le matching de ces deux valeurs. la contrainte Co2 va avoir un degré de match plugin puisque Algeria se trouve dans Africa selon l'ontologie Geo.

$$\begin{aligned} hasResult(?Co, plug\_in) \leftarrow\ & hasParameter(?Co,?Po), hasParameterType(?Po,?PTo), \\ & hasValueType(?Co,owl), hasValue(?Co,?Vo), hasConstaint(request,?Cr), \\ & hasOperator(?Cr,equals\_to), hasParameter(?Cr,?Pr), hasValue(?Cr,?Vr), \\ & hasParameterType(?Pr,?PTr), hasValueType(?Cr,owl), \\ & is\_located\_in(?Vr,?Vo), exact(?PTo,?PTr) \end{aligned} \quad (\mathbf{10})$$

$$is\_located\_in(x,y) = \begin{cases} true & si \quad Geo \models locatedin(x,y) \\ false & sinon \end{cases} \quad (\mathbf{11})$$

Les prédicats *islocatedin , same_instance , plugin et exact* sont implémentés comme fonction *builtin* en utilisant un appel séparé à un moteur d'inférence.

5.2 Rangement des services

Dans cette phase les services sont rangés suivant leur degré de match (DoMs). Pour calculer le DoMs nous calculons le degré de match (DoMc) pour chaque contrainte Co issue de la phase de matching en utilisant la méthode citée dans la section 3.2. Le calcul du DoMc est réalisé par la règle suivante :

$$\begin{aligned} hasDoM(?Co,?DoMc) \leftarrow\ & hasResult(?Co,?R), hasResultValue(?Co,?RV), \\ & hasValueType(?Co,?VT), hasOperator(?Co,?OP), hasValue(?Co,?Vo), \\ & hasParameter(?Co,?Po), hasParameterType(?Po,?PTo), \\ & computeDoMc(?R,?RV,?VT,?OP,?Vo,?PTo,?DoMc) \end{aligned} \quad (\mathbf{12})$$

Enfin la requête SPARQL (algorithme 1) va générer pour chaque NFPo matchant NFPr un ensemble de tuples détaillé qui va être exploiter pour afficher le résultat final à l'utilisateur sous différentes formes.

6 Implémentation et Expérimentation

L'approche proposée à été implémentée en Java .Comme moteur d'inférence nous utilisons KAON2 parce qu'il supporte les règles *DL-safe rules* et le langage d'interrogation pour RDF SPARQL [22]. Pour le matching de fonctionnalité nous utilisons l'API OWLS-MX [21]. Les ontologies utilisées sont stockées en tant que fichiers .owl dans le serveur web Apache Tomcat[2].

Pour des raisons de comparaison nous avons utilisé les données de test de [11], ces données représentent quatre services et une requête avec sept critères de qualité *: Price, Transaction, Time Out, Compensation Rate, Penalty Rate, ExecutionDuration*

[2] http://tomcat.apache.org/

et Reputation (voir Table 2).On a introduit ces paramètres comme concepts dans l'ontologie *"QoS middle ontology"* [12].

Table 2. Données utilisées pour l'expérimentation

Paramètre	**Price**	**Transaction**	**TimeOut**	**ComRat**	**PenRat**	**Execu**	**Repu**
requête	30	1	80	0,4	0,8	120	4,0
Poids	4	0	0	2	1	1	2
Opérateur	<	=	=	>	>	<	>
NFPo1	25	1	60	0,5	0,5	100	2,0
NFPo2	40	1	200	0,8	0,1	40	2,5
NFPo3	28	1	140	0,2	0,8	200	3,0
NFPo4	55	1	180	0,6	0,4	170	4,0

Nous avons développé cinq ontologies NFPo1.owl, NFPo2.owl, NFPo3.owl, NFPo4.owl et request.owl qui sont des instances de l'ontologie générique *NFPservice*. Les résultats obtenues sont conformes avec [11] car le service 1 est sélectionné (voir figure 2). La différence entre les degrés de match des services dans [11] et nos résultas peut s'expliquer par : (1) une contrainte de service qui ne matche aucune contrainte de la requête a un degré de match dans [11] alors que dans notre approche elle est éliminée durant la procédure de match par les règles, (2) dans notre approche si l'opérateur utilisé est ' = ' et Vo ≠ Vr pour un paramètre alors le DoMv=0 alors que dans [11] il peut avoir une valeur > 0.

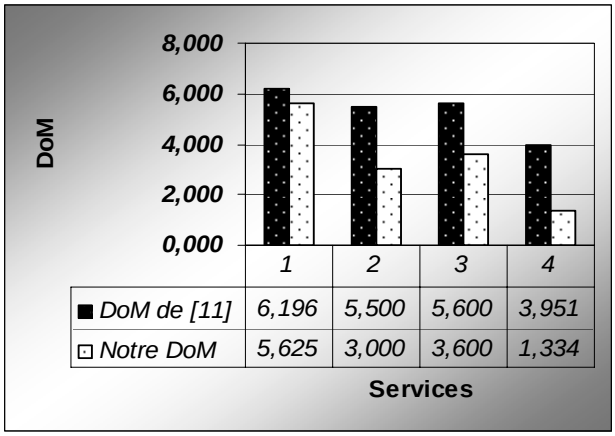

Fig. 2. Résultats de l'expérimentation

7 Conclusion et Perspectives

Dans ce travail, nous avons proposé de personnaliser la découverte des SWSs en utilisant les propriétés non fonctionnelles (NFPs). Ces NFPs sont spécifiées en utilisant une ontologie générique qu'on a nommée *NFPservice*, les offres de NFPs (NFPo) et les préférences du demandeur (NFPr) sont des instances de cette ontologie . Nous avons proposée aussi un algorithme de sélection basé sur des règles déclaratives énonçant les conditions pour la réussite du matching logique. L'utilisateur peut personnaliser la procédure de sélection selon ses préférences en exprimant ses propres règles de matching.

Comme perspectives, nous projetons d'appliquer notre méthode sur un plus grand nombre de propriétés non fonctionnelles dont les valeurs sont de différents types (numériques, individus ou concepts d'ontologies) avec un nombre important de services afin de faire une étude comparative effective.

Nous travaillons actuellement sur l'introduction de l'unité de mesure dans l'ontologie NFPservice ainsi que la spécification de règles pour la conversion des valeurs d'une unité de mesure vers une autre.

Références

1. V. Tsetsos, C. Anagnostopoulos, S. Hadjiefthymiades: Semantic Web Service Discovery: Methods, Algorithms and Tools, Semantic Web Services: Theory, Tools and Applications, (Ed. J. Cardoso), IDEA Group Inc., 2007, ISBN 978-1-59904-047-9.
2. E. Mussi: Flexible and Context-Aware Processes in Service Oriented Computing. Doctoral Dissertation. Politecnico di Milano Dipartimento di Elettronica eInformazione , 2007.
3. R.Brusch : ServiceMatching with Contextualised Ontologies. Dissertation Doktors der Naturwissenschaften, Fakult¨at f ¨ur Elektrotechnik, Informatik undMathematik der Universit¨at Paderborn, 2006.
4. J. Irena Ziembicki: Distributed Search in Semantic Web Service Discovery. Thesis Master of Mathematics in Computer Science. University of Waterloo, Ontario, Canada, 2006.
5. J. Javier Samper, F. Javier Adell, Leo van den Berg, and J. José Martinez: Improving Semantic Web Service Discovery. JOURNAL OF NETWORKS, ISSN: 1796-2056, VOL. 3, NO. 1, JANUARY 2008, Page(s): 35-42.
6. K. Kritikos and D. Plexousakis: OWL-Q for Semantic QoS-based Web Service Description and Discovery. Proceedings of the 6th International Semantic Web Conference and the 2nd Asian Semantic Web Conference ,pages 123- 137.BEXCO,Bussan KOREA 11Nov 2007.
7. Le-Hung Vu, F. Porto, M. Hauswirth, K. Aberer: An extensible and Personalized Approach to QoS-enabled Service Discovery. 11th Intl. Database Engineering and Applications Symposium, IDEAS, Banff, Canada, September 2007.
8. S. Lamparter, A. Ankolekar, R. Studer, S. Grimm: Preference based selection of highly configurable web services. In: roceedings of the 2007 International Conference on the World Wide Web 2007. pp. 1013-1022.
9. K.Verma, R.Akkiraju, and R.Goodwin: Semantic matching of Web service policies. Second International Workshop on Semantic and Dynamic Web Processes, 2005, pp. 79–90.

10. N. Oldham, K. Verma, A. Sheth, and F. Hakimpour: Semantic WS-agreement partner selection. In WWW '06: Proceedings of the 15th international conference on World Wide Web, pages 697–706, New York, NY, USA, 2006. ACM Press.
11. X. Wang, T. Vitvar, M. Kerrigan and I. Toma: A QoS-aware Selection Model for Semantic Web Services. In Proceedings of the 4th International Conference on Service Oriented Computing (ICSOC 2006), Chicago, USA, December 4-7, 2006.
12. E. Michael Maximilien, Munindar P. Singh: A Framework and Ontology for Dynamic Web Services Selection, SEPTEMBER • OCTOBER 2004, Published by the IEEE Computer Society.
13. D. Martin, M. Burstein, J. Hobbs, O. Lassila, D. McDermott, S. McIlraith, S. Narayanan, M. Paolucci, B. Parsia, T. Payne, E. Sirin, N. Srinivasan, K. Sycara, OWL-S: Semantic Markup for Web Services, W3C Member Submission 22 November 2004.
14. D. Roman, H. Lausen, U. Keller, J. de Bruijn, C. Bussler, J. Domingue, D. Fensel, M. Hepp, M. Kifer, B. König-Ries, J. Kopecky, R. Lara, E. Oren, A. Polleres, J. Scicluna, M. Stollberg : D2v1.4. Web Service Modeling Ontology (WSMO), WSMO Working Draft 16 February 2007.
15. R. Akkiraju , B. Sapkota: Semantic Annotations for WSDL and XML Schema Usage Guide". W3C Working Group Note 28 August 2007
16. D.L. McGuinness , F. van Harmelen: OWL Web Ontology Language Overview W3C Recommendation 10 February 2004.
17. I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean. SWRL: A semantic web rule language combining OWL and RuleML, 2004. W3C Submission.
18. B. Motik, U. Sattler, and R. Studer: Query answering for OWL-DL with rules. Journal of Web Semantics: Science, Services and Agents on the WWW, 3(1):41–60, 2005.
19. S. Brockmans, R. Volz, A. Eberhart, and P. Löffler: Visual modelling of OWL DL ontologies using UML. In Proc. of the 3rd Int. Semantic Web Conf., Hiroshima, Japan, 2004
20. V. Diamadopoulou, Y. Panagis, E. Sakkopoulos, and Ch. Makris (2006): Techniques to support Web Service selection and consumption with QoS characteristics, in the J. Network and Computer Applications, Elsevier Science, Vol 32, No 2, pp. 108-130 (impact factor: 1.265).
21. M. Klusch,B. Fries,K. Sycara (2006): Automated Semantic Web Service Discovery with OWLS-MX. Proceedings of 5th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS), Hakodate, Japan, ACM Press. Best Paper Award Nominee.
22. E. Prud'hommeaux, A. Seaborne: SPARQL Query Language for RDF W3C Recommendation 15 January 2008.
23. M. Paolucci, T. Kawamura, T. R. Payne, and K. P. Sycara: Semantic matching of web services capabilities. In 1st Int. Semantic Web Conference, pages 333–347, Sardinia, Italy, 2002.