Extension de OWL-S avec les Standards de Qualité

Stéphane Jean[1] Francisca Losavio[2] Alfredo Matteo[2] Nicole Levy[3]
[1] Laboratoire d'Informatique Scientifique et Industrielle
LISI/ENSMA et Université de Poitiers
BP 40109, 86961 Futuroscope Cedex, France
jean@ensma.fr
[2] Centro ISYS, Escuela de Computación, Facultad de Ciencias
Universidad Central de Venezuela
francislosavio@gmail.com, almatteo@cantv.net
[3] Université de Versailles
St-Quentin en Yvelines, France
nlevy@prism.uvsq.fr

ABSTRACT

With the increasing amount of Web Services available on the Web, Web Services discovery issues are becoming increasingly important. Since current Web Services standard technologies (e.g, UDDI) only provide syntactic searches on Web Services descriptions (mainly their signatures), semantic discovery approaches based on ontologies (e.g, OWL-S) have been developed. These approaches lead to more precise descriptions of Web Services functionalities, but they provide few mechanisms to capture non functional aspects of Web Services collectively referred as Quality of Services (QoS). To fill this gap, some works have proposed Web Services discovery approaches based on QoS ontologies. However these approaches don't take into account existing standards about software quality and relationships that can be established between them. Yet, these standards could be used as a shared understanding between Services Providers and Customers and thus, would ease the Web Services discovery process. In this article we first propose an extension of OWL-S that makes it possible to describe QoS according to one or many quality standards. Then, we develop an approach exploiting this extension of OWL-S to ease Web Services discovery using quality criteria. This approach is based on an extension of SPARQL that facilitates the definition of Web Services discovery queries. It allows a user to find a Web Service even if it is described with quality criteria defined in an other standard that the one used to express its query.

Categories and Subject Descriptors

H.3.5 [**Information Storage And Retrieval**]: Online Information Services—*Web-based services*

JFO 2009 December 3-4, 2009, Poitiers, France

General Terms

Search process, Query formulation, Standardization

Keywords

Web Service Discovery, Quality of Service, Ontology, Software Quality Standard

1. INTRODUCTION

Avec l'avènement de la technologie des services Web (SW), la conception d'un nombre croissant d'applications informatiques repose sur l'utilisation de SW existants. Choisir les SW adéquats pour l'application en cours de conception n'est pas une chose aisée. Ceci nécessite en effet de trouver les SW qui satisfont les *besoins fonctionnels* de l'application en cours de développement et qui, autant que possible, respectent les autres besoins (*besoins non fonctionnels*) et en particulier le niveau de qualité requis. Dans cet article, nous traitons exclusivement des besoins non fonctionnels liés à la qualité. Dans le domaine des applications basées sur des SW, on parle de *Qualité de Service* (*QoS*), c'est à dire de l'ensemble des propriétés de qualité qu'un SW doit satisfaire.

Pour faciliter le processus de sélection de SW, il est nécessaire que ces services soient décrits le plus précisément possible. Les technologies actuelles basées sur SOAP (Simple Object Access Protocol), WSDL (Web Services Description Language) et UDDI (Universal Description Discovery and Integration) ne permettant qu'une description syntaxique de l'interface des SW, des approches sémantiques basées sur des ontologies ont été développées. En particulier, l'ontologie OWL-S [12] a été définie pour permettre une description sémantique des SW. Si OWL-S permet une description précise des fonctionnalités d'un SW, elle n'apporte que très peu d'éléments pour en décrire la qualité.

Plusieurs normes ont été définies par des institutions telles que l'ISO [7, 6] ou le W3C [1] pour définir précisément des critères de qualités. S'accorder sur un standard particulier permet aux fournisseurs et aux utilisateurs de SW de se baser sur un référentiel commun. Pourtant, même si de nombreuses approches ont été développées pour permettre de décrire la QoS et l'exploiter pour la recherche de services, peu se sont intéressées à la prise en compte des différents standards de qualité existants et des relations que l'on peut

établir entre eux. En conséquence, la diversité des normes disponibles rend difficile pour une personne qui recherche un SW d'évaluer la qualité d'un SW si celui-ci est décrit par une norme différente de celle qu'il connaît.

Dans cet article, nous proposons une extension de OWL-S pour permettre de rechercher des SW selon leur qualité exprimée selon un standard donné. Notre approche repose sur l'ontologie que nous avons proposée dans [11]. Cette ontologie permet de représenter les critères de qualité définis pour un domaine particulier (par exemple, les SW) tels qu'ils sont définis dans différents standards ainsi que les liens que l'on peut établir entre ces critères (par exemple, les critères qui sont équivalents dans deux standards). Nous montrons d'abord comment étendre OWL-S avec cette ontologie puis nous étudions l'intérêt de cette extension pour la recherche de SW. Pour faciliter cette recherche, nous proposons une extension du langage SPARQL [15] pour simplifier l'expression de requêtes portant sur la qualité de SW puis montrons comment nous pouvons exploiter les différentes relations qui peuvent être établies entre les critères de qualité provenant éventuellement de différents standards.

La suite de cet article est organisée comme suit. Dans la section suivante, nous synthétisons notre étude des différentes approches sémantiques existantes permettant de décrire et rechercher des SW par leur qualité. Dans la section 3, nous présentons l'ontologie sur laquelle repose notre approche et montrons comment nous pouvons l'utiliser pour étendre OWL-S dans la section 4. Dans la section 5, nous détaillons l'intérêt de cette extension pour la recherche de SW. La section 6 présente l'implémentation que nous avons réalisée pour valider notre approche et discute de ses avantages et de ses limitations. Finalement, nous concluons en section 7 et introduisons les perspectives de ces travaux.

2. TRAVAUX CONNEXES

De nombreuses approches basées sur des ontologies ont été proposées pour la QoS comme par exemple [9, 16, 5, 19]. Dans un état de l'art récent [17], Tran et al. montrent que les approches existantes se focalisent sur certains aspects de la représentation et du traitement de la QoS mais qu'aucune ne fournit une solution complète. Nous décrivons dans cette section les principaux travaux sur la définition d'ontologies pour la QoS qui s'appuient sur OWL-S comme l'approche proposée dans cet article. Nous indiquons pour chacune les aspects de la QoS sur lesquels elle se focalise.

OWL-Q [9] est une extension de OWL-S dont la particularité est de fournir un système complet et extensible pour associer des métriques aux propriétés de qualité. Ce système permet en effet de représenter des métriques simples, complexes (dérivées à partir d'autres métriques) et même d'ajouter de nouvelles métriques. De plus, il fournit des algorithmes complexes pour comparer ces métriques.

QOS-MO [16] permet de représenter la QoS de SW décrits en OWL-S. Sa particularité est de permettre de décrire l'interaction entre les fournisseurs et clients de SW. Cette ontologie définit en effet le concept `QoSContract` reliant `QoSOffered` et `QoSRequired` qui permet de représenter l'accord trouvé par le fournisseur et le client sur la qualité d'un SW.

onQoS [5] définit des éléments permettant d'associer des concepts de QoS aux profils OWL-S de SW. Cette ontologie propose notamment un puissant système de type de données pour décrire les valeurs des propriétés de qualité. L'évaluation de ces valeurs peut ensuite être faite à partir de métriques, échelles et règles de calcul.

DAML-QoS [19] propose une extension de DAML-S (devenu ensuite OWL-S) pour la QoS. Elle permet notamment de définir les contraintes qui peuvent être associées à des propriétés de qualité. Par exemple, le concept `QoSPrecondition` permet de définir les conditions qui doivent être remplies par le client de SW pour obtenir la qualité de service spécifiée par le fournisseur.

Nous notons qu'aucune des approches présentées précédemment ne s'est intéressée spécifiquement à la représentation des standards de qualité et des relations que l'on peut établir entre ces standards. En effet, même si ces ontologies peuvent être utilisées pour représenter les propriétés de qualité définies dans les standards, elles ne permettent pas de conserver leur origine comme, par exemple, le modèle de qualité (ensemble structuré de caractéristiques qui décrivent la qualité d'un produit logiciel [7]) du standard dans lequel elles ont été définies. De plus, même si certaines de ces ontologies ont proposé des relations pour lier des propriétés de qualité, elles ne se sont pas intéressées à l'établissement de correspondances entre les standards (équivalence, subsomption, etc.).

Par ailleurs, au niveau des requêtes, nous notons que ces approches se sont essentiellement intéressées à des algorithmes permettant d'exploiter les types de données et métriques associés aux valeurs de propriétés de qualité. Ces travaux ne se sont pas penchés sur l'expression de requêtes utilisant des critères de qualité pour rechercher un SW dans un langage d'interrogation donné. Ces constatations nous ont amenés à développer une approche pour prendre en compte les standards de qualité dans la gestion de la QoS des SW.

3. ONTOLOGIE POUR LES STANDARDS DE QUALITÉ LOGICIELLE

L'approche de recherche de SW par la qualité que nous proposons dans cet article repose sur l'ontologie que nous avons proposée dans [11]. Nous en décrivons les principaux éléments dans cette section.

3.1 Standards de qualité logicielle

Définir précisément un critère de qualité comme par exemple la sécurité n'est pas une chose aisée. C'est pourtant nécessaire pour que fournisseurs et utilisateurs de SW puissent trouver un accord. Plusieurs standards ont été définis à cette fin. Ils donnent des définitions précises de critères de qualité et leur associent, lorsque c'est possible, des métriques qui permettent de caractériser précisément un SW. L'ontologie que nous utilisons présente la particularité de permettre d'intégrer les standards de qualité suivants.

Web Services Architecture (WSA) [1].

L'architecture de référence WSA a été définie par le Web Services Architecture Working Group pour garantir l'interopérabilité d'applications basées sur des SW. La spécification de cette architecture définit un ensemble de critères de qualité permettant d'évaluer la conformité de SW par rapport à cette architecture. Cette spécification définit sept critères de qualité : *Interoperability*, *Reliability*, *WWW Integration*, *Security*, *Scalability and Extensibility* et *Team Goals and Management and Provisioning*) qui sont ensuite raffinés en critères plus précis.

La norme ISO/IEC 9126-1 [7]

La norme ISO/IEC 9126-1 définit des propriétés de qualités qui peuvent être utilisées pour décrire un produit logiciel. Les propriétés de qualité sont représentées par un modèle de qualité constitué d'une hiérarchie de caractéristiques basée sur 6 principales caractéristiques : *Functionality*, *Reliability*, *Usability*, *Efficiency*, *Maintainability* et *Portability*. Ces caractéristiques sont ensuite raffinées jusqu'à atteindre des caractéristiques mesurables appelées attributs de qualité.

Contrairement à WSA, cette norme n'est donc pas spécifique au domaine des SW. Cependant, elle peut être adaptée à ce domaine en éliminant les caractéristiques non applicables et en ajoutant de nouvelles caractéristiques ou sous-caractéristiques.

La norme ISO/IEC 13236 [6]

La norme ISO/IEC 13236 porte sur le domaine de la QoS. Elle définit une terminologie et des concepts sur la QoS de manière à fournir un langage commun pour les clients et les fournisseurs de services distribués (par exemple, des SW mais pas obligatoirement). Par ailleurs, elle introduit un ensemble de caractéristiques, de métriques et de mécanismes pour permettre de caractériser, spécifier et gérer les besoins liés à la QoS.

La diversité des standards sur la qualité logicielle pose le problème de la variabilité de terminologie. Pour faciliter le traitement de ce problème, nous avons proposé dans [11], une ontologie pour représenter ces différents standards et les relier entre eux. Cette ontologie, sur laquelle repose l'approche de recherche de services proposée dans cet article, est décrite dans la section suivante.

3.2 Description de l'ontologie utilisée

Les principaux éléments de l'ontologie que nous avons proposé pour intégrer les différents standards de qualité logicielle sont présentés sur la figure 1.

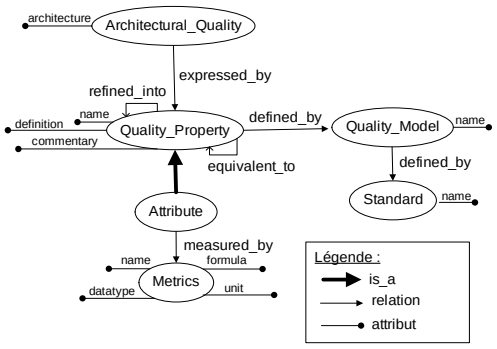

Figure 1: Extrait de l'ontologie pour intégrer les standards de qualité logicielle

- **Architectural_Quality** : représente un style d'architecture pour une famille d'applications. Le style d'architecture défini est caractérisé par un ensemble de propriétés de qualité. SOA est un exemple de style d'architecture.
- **Quality_Property** : représente une caractéristique de qualité. Cette caractéristique peut être raffinée par une ou plusieurs autres caractéristiques de qualité qui peuvent oui ou non être mesurées. La sécurité est un exemple de propriété de qualité. Elle est définie par les différents standards présentés précédemment. Elle est par exemple raffinée dans le standard WSA par `authorization` et `authentification`, deux propriétés de qualité mesurables (appelées attributs). La relation `equivalent_to` permet d'établir des correspondances entre propriétés de qualité définies dans plusieurs standards. Par exemple, si on se réfère aux définitions données par les standards WSA et ISO 9126-1, `confidentiality` (ISO) est équivalent à `authorization` (WSA). Pour simplifier la figure, nous n'avons pas montré toutes les relations que l'on peut établir entre propriétés de qualité. Ces relations seront détaillées dans la section 5.3.
- **Attribute** : représente une propriété de qualité qui peut être mesurée. Par exemple, l'`authentification` définie par WSA indique si oui ou non l'application repose sur une identification des utilisateurs. Elle peut ainsi être mesurée par une valeur booléenne.
- **Metrics** : représente une métrique ayant une valeur appartenant à un type de données. Cette valeur peut être caractérisée par une unité de mesure et être calculée à partir d'une formule. Par exemple, la protection selon le standard ISO 13236 est mesurée par la probabilité de défaillance. C'est une métrique qui prend ses valeurs dans le type `real` (valeurs comprises entre 0 et 1) qui n'a pas d'unité de mesure ni de formule de calcul (sa valeur est donc spécifiée par l'utilisateur). Notons que nous n'avons pas détaillé la représentation des unités de mesure. Si notre approche est implantée avec le modèle d'ontologies PLIB [13], celui-ci permet nativement la modélisation des unités de mesure. Avec le modèle d'ontologies OWL [3] qui n'a pas ce type de constructeur, nous pouvons nous appuyer sur une ontologie qui les définit (par exemple, la `Measurement Units Ontology` [1]).
- **Quality_Model** : représente un ensemble de propriétés de qualité qui permettent de spécifier la qualité d'un logiciel ou d'une brique logicielle.
- **Standard** : représente une norme établie définissant un ensemble de critères sur la qualité logicielle. ISO 9126-1 et WSA sont des exemples de standards qui peuvent être utilisés dans le domaine des SW.

La figure 2 reprend les exemples [2] évoqués précédemment en montrant un exemple d'instanciation de cette ontologie. Dans cet exemple, l'architecture SOA est caractérisée par des propriétés de qualité venant du standard ISO 13236 et du standard WSA. Pour simplifier la figure nous n'avons pas indiqué les modèles de qualité et les standards de qualité auxquels sont attachées les propriétés ; nous les avons seulement préfixées par le nom du standard auquel elles appartiennent. Ces propriétés de qualités peuvent être raffinées en attributs. C'est par exemple le cas de la sécurité définie dans le standard ISO 13236 qui est raffinée en deux attributs `protection` et `confidentiality` dont les mesures sont respectivement une valeur réelle et une valeur booléenne.

1. `http://idi.fundacionctic.org/muo/muo-vocab.html`
2. Les exemples utilisés tout au long de cet article ont été volontairement simplifiés et adaptés par rapport aux standards présentés dans la section 3, pour faciliter la compréhension du lecteur.

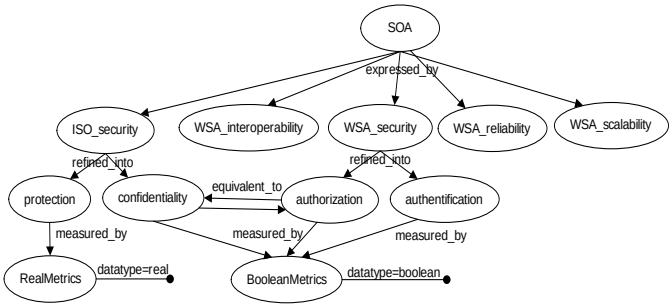

Figure 2: Exemple d'instanciation

Nous montrons maintenant comment nous pouvons utiliser cette ontologie pour étendre l'ontologie OWL-S afin de faciliter la recherche de SW en fonctions de besoins non fonctionnels.

4. EXTENSION DE OWL-S POUR DÉCRIRE LA QUALITÉ DE SERVICE SELON DES STANDARDS

OWL-S [12] est une ontologie qui a été définie par des chercheurs de plusieurs organisations pour permettre de déclarer et décrire des SW. L'objectif de cette ontologie est de permettre la découverte et l'invocation dynamique de SW ainsi que leur composition.

Si cette ontologie permet de définir précisément les caractéristiques fonctionnelles d'un service, nous montrons dans la section suivante qu'elle présente des limitations pour en décrire les aspects non fonctionnels.

4.1 Limitations de OWL-S pour décrire la qualité d'un SW

Dans OWL-S, un service est décrit selon les trois axes suivants.

- **ServiceProfile** : décrit les fonctionnalités rendues par le service en terme d'entrées, sorties, préconditions, paramètres et résultat.
- **ServiceModel** : décrit le fonctionnement du service en indiquant comment les résultats sont produits étape par étape.
- **ServiceGrounding** : décrit comment accéder au service. En particulier, il permet de préciser le protocole et le format des messages.

La documentation de OWL-S indique que le `ServiceProfile` peut être utilisé pour décrire la qualité d'un SW. Le `ServiceProfile` est en effet associé à un ensemble de `ServiceParameter` qui permettent de noter un service par des couples (critère, valeur).

Cependant, cette modélisation très simple pose plusieurs limitations. D'abord, elle ne permet pas de décrire les propriétés de qualité pour, par exemple, indiquer la définition qu'en donne le standard qui l'a définie. Cet aspect est néanmoins important pour faciliter l'interaction avec l'utilisateur. Ensuite, elle ne permet pas de faire des liens entre ces propriétés pour, par exemple, les raffiner ou établir des correspondances entre elles. Pourtant, ces liens peuvent ensuite s'avérer utiles pour exploiter la qualité du service ainsi décrite. Par exemple, le travail présenté dans [2] montre que ces relations peuvent être utiles pour déterminer les critères de qualité à réaliser pour un audit. Nous montrerons dans la section 5.3 qu'elles peuvent aussi être utilisées pour la découverte de SW. Enfin, la modélisation proposée par OWL-S pour représenter la QoS ne permet pas d'associer aux propriétés de qualité des métriques complexes, par exemple, à base de formule ou d'unités de mesure. Pourtant la plupart des critères de qualité nécessite ce type de modélisation.

Pour pallier à ces limitations, nous proposons d'étendre OWL-S avec l'ontologie présentée dans la section précédente.

4.2 Extension de OWL-S proposée

L'extension que nous proposons est présentée sur la figure 3. En OWL-S, le `ServiceProfile` caractérise les fonctionnalités du service en précisant des éléments comme ses entrées (`ServiceInput`) et ses sorties (`ServiceOutput`). Comme nous l'avons indiqué précédemment, il est également décrit par un ensemble de `ServiceParameter` qui possèdent un nom (`serviceParameterName`) et une valeur (`sParameter`). Notre extension de OWL-S repose sur la spécialisation de `ServiceParameter` par la classe `Quality_Property` de notre ontologie. Le reste de notre ontologie reste inchangée si ce n'est que la propriété `name` de `Quality_Property` disparaît puisqu'elle est maintenant héritée de la classe `ServiceParameter` et que la classe `Architecture_Quality` est supprimée puisqu'on s'intéresse ici à un type d'architecture spécifique (SOA).

Notons que dans notre modèle seules les instances de la classe `Attribut` ont une valeur. Pour les instances de `Quality_Property` la valeur de cette propriété est une valeur anonyme (un noeud blanc en RDF).

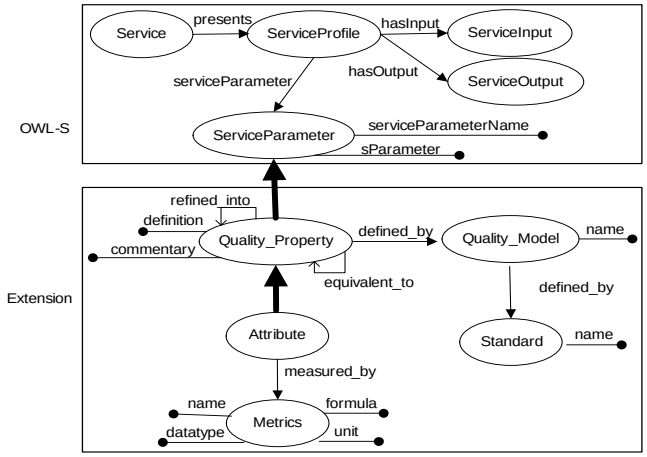

Figure 3: Extension de OWL-S proposée

Ayant présenté les limitations de OWL-S pour la description de la qualité de SW, nous avons montré comment nous pouvons l'étendre avec notre ontologie pour les standards de qualité logicielle. Nous montrons maintenant l'intérêt de cette extension en nous focalisant sur l'aspect recherche de service.

5. RECHERCHE DE SW SELON DES CRITÈRES DE QUALITÉ DES STANDARDS

L'extension de OWL-S que nous proposons peut s'avérer particulièrement utile pour les différents acteurs impliqués dans la diffusion et l'utilisation de SW. Elle peut, par exemple, être utilisée pour permettre à un administrateur d'un annuaire de Services Web de choisir le ou les standards de qualité qui permettront de décrire les services de l'annuaire. Les fournisseurs de services pourront alors saisir leurs services en renseignant les valeurs de propriétés de qualité associées à ce standard. Pour faciliter cette saisie notre extension pourrait être utilisée pour afficher la définition et les commentaires sur la propriété de qualité dont la valeur doit être renseignée.

Dans cet article, nous nous intéressons à l'utilité de cette extension pour un utilisateur qui recherche un SW. Nous commençons par décrire les capacités d'interrogation qu'offrent cette extension et relevons leurs insuffisances.

5.1 Capacités d'interrogation

OWL-S étant une ontologie, nous pouvons utiliser un langage d'interrogation d'ontologies tel que SPARQL [15] ou OntoQL [8] pour rechercher un service.

Exemple. La requête SPARQL [3] suivante permet de rechercher les services qui prennent en entrée une carte de crédit et respecte le critère `confidentiality` défini dans la norme ISO 9126-1.

```
SELECT ?service
 WHERE {
  ?service rdf:type Service
  ?service presents ?profile
  ?profile hasInput <http://cordio.bs/CreditCard>
  ?profile serviceParameter ?parameter
  ?parameter defined_by ?qualityModel .
  ?qualityModel defined_by <http://www.iso.org/ISO9126-1>
  ?parameter serviceParameterName "confidentiality"
  ?parameter sParameter ?value
  FILTER (?value = true) }
```

Explication. Le premier triplet de la clause `WHERE` introduit la variable `?service` pour parcourir l'ensemble des services. Cette variable est projetée dans la clause `SELECT` pour retourner les services qui satisfont les autres triplets. Les propriétés nécessaires à la requête étant définies sur le profil du service, le deuxième triplet utilise la propriété `presents` pour introduire la variable `?profile` sur le profil du service. Le troisième triplet permet ensuite de vérifier que le service prend bien en entrée une carte de crédit. Le quatrième triplet introduit la variable `?parameter` sur une propriété de qualité du service. Les deux triplets suivants permettent de vérifier que cette propriété de qualité fait partie d'un modèle de qualité défini par la norme ISO 9126-1. Les trois suivants vérifient que cette propriété de qualité a pour nom `confidentiality` et qu'elle est satisfaite (valeur `true`).

Cet exemple montre que notre extension de OWL-S permet de rechercher un service en fonction de critères de qualité définis dans un standard particulier. Des requêtes plus complexes pourraient également être écrites pour retourner l'ensemble des valeurs de propriétés de qualité des services retournés avec la définition donnée par les standards de chacune de ces propriétés. De même on pourrait écrire des requêtes utilisant les métriques associées aux propriétés de qualité pour, par exemple, tester l'unité de mesure dans laquelle une valeur est définie.

Nous pouvons cependant remarquer que ces capacités de recherche présentent les deux principales limitations suivantes que nous avons choisies de traiter dans cet article.

- Les requêtes sont écrites en encodant l'extension que nous avons proposée d'OWL-S. De plus, l'utilisation des propriétés de qualité dans la requête se fait différemment des autres propriétés (retour d'abord de la valeur via la propriété `sParameter`, puis test de cette valeur).

- Le retour des requêtes ne prend pas en compte les relations que l'on peut établir entre les propriétés de qualité. Par exemple, la requête précédente ne retournera pas les services satisfaisant la propriété `authorization` de l'ISO 9126-1 même si celle-ci est déclarée comme étant équivalente à la propriété `confidentiality` du standard WSA.

Nous détaillons maintenant l'approche que nous proposons pour pallier à ces insuffisances.

5.2 Extension de SPARQL pour la recherche de SW selon des critères de qualité

Comme nous l'avons indiqué précédemment, la syntaxe de SPARQL n'est pas adaptée à la recherche de services. Plus précisément, elle pose les trois problèmes suivants :

- La dépendance de la requête avec l'extension de OWL-S. Par exemple, la requête de l'exemple précédent est écrite est indiquant que les propriétés de qualité sont liées à un profil de service par la propriété `serviceParameter`. Ainsi, l'écriture d'une telle requête nécessite d'abord une bonne connaissance de notre extension de OWL-S. De plus, une telle requête se retrouve fortement couplée à notre extension et donc à ses possibles évolutions.

- Les propriétés de qualité sont utilisées de manière différente des autres propriétés de l'ontologie. Pour tester la valeur d'une propriété de qualité, il est en effet nécessaire de commencer par rechercher sa valeur par la propriété `sParameter` avant de pouvoir la tester. Par soucis d'homogénéité et de simplicité, il serait préférable de pouvoir les utiliser comme toute autre propriété.

- Le mélange des besoins fonctionnels et non fonctionnels. La clause `WHERE` de notre exemple contient en effet à la fois des triplets pour les besoins fonctionnels (celui portant sur la propriété `hasInput`) et des triplets pour les besoins non fonctionnels (les triplets suivants). Pour rendre la requête plus lisible et modulaire, il serait préférable de bien distinguer ces deux types de triplets.

Pour pallier à ces inconvénients, nous proposons une extension du langage SPARQL.

3. Par mesure de concision, les espaces de noms sont omis dans les requêtes.

5.2.1 Syntaxe de l'extension proposée

Nous souhaitons modifier le langage SPARQL pour faciliter la recherche de SW selon des critères de qualité. Ce besoin étant très spécifique, il serait bon que cette extension soit modulaire et par mesure d'homogénéité conserve une syntaxe similaire à celle de SPARQL. Pour répondre à ces deux besoins, nous proposons d'ajouter une nouvelle clause nommée `QUALITY` au langage SPARQL. Cette clause présente une syntaxe [4] similaire à la clause `WHERE` :

```
QualityClause  ::= QUALITY '{' TriplesQuality+ '}'
TriplesQuality ::= VarOrTerm PropQualityOrProp Object
```

Explication. La clause `QUALITY` est composée d'un ensemble de triplets (sujet, prédicat, objet). Le sujet peut être une variable ou un terme (`VarOrTerm`) tels qu'ils sont définis dans la syntaxe SPARQL. En particulier, ceci permet d'utiliser une propriété de qualité comme sujet d'un triplet. L'objet du prédicat peut être une propriété de l'ontologie ou une propriété de qualité (instance de la classe `Quality_Property` ou de sa sous-classe `Attribute`). L'objet d'un prédicat est identique à la définition donnée dans la grammaire de SPARQL.

Cette extension permet de ré-écrire la requête de l'exemple précédent de la manière suivante :

```
SELECT ?service
 WHERE {
  ?service rdf:type Service
  ?service presents ?profile
  ?profile hasInput <http://cordio.bs/CreditCard> }
 QUALITY {
  ?profile confidentiality "true"
  confidentiality defined_by ?qualityModel .
  ?qualityModel defined_by <http://www.iso.org/ISO9126-1>
 }
```

Explication. Notre extension permet maintenant de distinguer clairement l'expression des besoins fonctionnels (dans la clause `WHERE`) de l'expression des besoins non fonctionnels (dans la clause `QUALITY`). Le premier triplet de la clause `QUALITY` montre qu'une propriété de qualité peut être utilisée dans cette clause comme une propriété de l'ontologie. Les deux suivants restent similaires à la requête initiale.

5.2.2 Sémantique de l'extension proposée

Si la syntaxe des clauses `WHERE` et `QUALITY` sont similaires, leur interprétation est par contre différente. La règle suivante montre l'interprétation d'un triplet `(s, p, o)` où `p` est une propriété de qualité :

$$(\texttt{s p o}) \wedge (\texttt{p type Quality\_Property}) \Rightarrow (\texttt{s serviceParameter ?param}) . (\texttt{?param serviceParameterName p}) . (\texttt{?param sParameter o})$$

Explication. Si `p` est une propriété de qualité, le triplet `(s,p,o)` doit être interprété comme un ensemble de triplets. Les deux premiers triplets recherchent `p` parmi les propriétés de qualité en utilisant son nom (valeur de la propriété `serviceParameterName`). Le troisième indique que l'objet du triplet (`o`) correspond à la valeur de la propriété `sParameter`.

Notons qu'une interprétation similaire est réalisée si une propriété de qualité est utilisée comme sujet d'un triplet. L'exemple suivant illustre la règle précédente.

Exemple. L'utilisation de la propriété de qualité `confidentiality` comme une propriété usuelle en SPARQL est interprétée de la manière suivante :

$$(\texttt{?profile confidentiality "true"}) \Rightarrow (\texttt{?profile serviceParameter ?param}) . (\texttt{?param serviceParameterName "confidentiality"}) . (\texttt{?param sParameter "true"})$$

Les autres règles permettent d'exploiter les métriques qui peuvent être associées à un attribut de qualité. La règle suivante montre l'interprétation d'un triplet `(s, p, o)` où `o` est une valeur caractérisée par un type de données (notation `^^datatype`). Nous supposons ici que `p` est une propriété de qualité (plus précisément un attribut) interprétée selon la règle précédente.

$$(\texttt{s p o\^\^type}) \Rightarrow (\texttt{s p o}) . (\texttt{p measured\_by ?m}) . (\texttt{?m datatype type})$$

Explication. Si la valeur de l'objet `o` est donnée selon un certain type de données, on réalise la comparaison en vérifiant que la propriété est bien mesurée selon ce type de données. Pour cela, le triplet initial est interprété en trois triplets. Le premier triplet fait la comparaison de la valeur littérale (sans type de données). Les deux suivants font la comparaison par rapport au type de données (propriété `datatype`) défini sur la métrique de la propriété de qualité (retrouvée par la propriété `measured_by`).

Exemple. Voici l'interprétation de l'attribut `confidentiality` caractérisé par le type de données `xsd :boolean`.

$$(\texttt{?profile confidentiality "true"\^\^xsd : boolean}) \Rightarrow (\texttt{?profile confidentiality "true"}) . (\texttt{confidentiality measured\_by ?m}) . (\texttt{?m datatype "xsd : boolean"})$$

Pour le traitement des unités de mesure, nous utilisons une approche fréquemment utilisée [5] qui consiste à utiliser les unités de mesure comme un type de données (selon la même notation `^^unit`). Ainsi une règle similaire à la précédente permet d'interpréter un triplet `(s, p, o)` où `o` est une valeur caractérisée par une unité de mesure.

$$(\texttt{s p o\^\^unitOfMeasure}) \Rightarrow (\texttt{s p o}) . (\texttt{p measured\_by ?m}) . (\texttt{?m unit unitOfMeasure})$$

Explication. La règle est similaire à la précédente mais se base cette fois-ci sur l'unité de mesure donnée par la propriété `unit`. Pour l'interpréteur de requête, la distinction

4. La syntaxe présentée est simplifiée et repose sur les notations utilisées pour la grammaire de SPARQL définie à l'adresse `http://www.w3.org/TR/rdf-sparql-query/`. En particulier, la syntaxe complète permet l'utilisation des opérateurs `FILTER` et `OPTIONNAL` dans la clause `QUALITY`.

5. voir `https://forge.morfeo-project.org/wiki_en/index.php/How_to_use_MUO` pour une explication détaillée.

entre une unité de mesure et un type de données se fait sur le namespace (`xsd` pour les types de données, `mymw` pour les unités).

Exemple. Voici l'interprétation de l'attribut `maxTimeByTransaction` caractérisé par l'unité `second`.

(?profile maxTimeByTransaction "1"^^mymw : second)
⇒ (?profile maxTimeByTransaction "1") .
(maxTimeByTransaction measured_by ?m) .
(?m unit "mymw : second")

Nous venons de présenter notre extension de SPARQL pour faciliter la recherche de services selon des besoins fonctionnels et non fonctionnels. Nous montrons maintenant une approche permettant d'exploiter les relations que l'on peut établir entre propriétés de qualité.

5.3 Prise en compte des relations entre propriétés de qualité

De nombreuses relations peuvent être établies entre propriétés de qualité. Nous nous basons sur les relations établies par Kruchten [10] et plus précisément sur l'interprétation qui en est donnée dans [2]. Cependant, contrairement à ces travaux qui les utilisent pour les audits de qualité, nous montrons comment ces relations peuvent être interprétées pour faciliter la recherche de services.

5.3.1 Relations entre propriétés de qualité

Dans la section 3.2, nous avons présenté la relation `equivalent_to` qui permet d'indiquer que deux propriétés de qualité sont équivalentes. Cette relation est particulièrement utile pour établir des correspondances entre standards de qualité. Nous considérons en plus de cette relation quatre autres relations : `constrains`, `subsumes`, `forbids` et `conflicts`. Contrairement à la relation `equivalent_to`, ces relations s'appliquent uniquement à des propriétés de qualité évaluées par une valeur booléenne. Ces relations sont les suivantes :

- `constrains`. Si `X constraints Y`, la propriété de qualité `Y` ne peut être satisfaite que si `X` est satisfaite. Par exemple, on peut définir que `authentification constrains authorization` si on considère que lorsqu'une application ne réalise pas d'authentification de l'utilisateur, elle ne peut pas réaliser de contrôle d'accès.
- `subsumes`. Si `X subsumes Y` et que la propriété de qualité `X` est satisfaite alors `Y` est également satisfaite. Par exemple, on peut définir que `dataEncryption subsumes digitalSignature` si on considère que lorsqu'une application réalise du chiffrement de données, elle utilise des signatures numériques.
- `forbids`. Si `X forbids Y` et que la propriété de qualité `X` est satisfaite alors `Y` ne peut pas être également satisfaite. Par exemple, on peut définir que `portability forbids codedInC` si on considère que lorsqu'une application est portable elle ne peut pas avoir été codée en `C`.
- `conflicts`. Si `X conflicts Y` alors `X forbids Y` et `Y forbids X`. Par exemple, on peut définir que `portability conflicts codedInC` si on considère également qu'une application codée en `C` ne peut pas être portable.

Ces relations peuvent être utilisées pour faciliter la recherche de service.

5.3.2 Interprétation des relations entre propriétés de qualité

Les relations entre propriétés de qualité décrites précédemment peuvent être utilisées pour effectuer du raisonnement, c'est à dire déduire de nouvelles informations à partir des informations existantes. Ces nouvelles informations peuvent être exploitées au moment du traitement des requêtes pour retourner des résultats pertinents qui, sans les raisonnements, n'auraient pas été retournés.

Dans cette section, nous présentons les déductions permises par chacune des relations entre propriétés de qualité. Nous les exprimons sous forme de règles de déduction et montrons un exemple d'application de cette règle.

`equivalent_to` :

(prop1 value v) ∧ (prop1 equivalent_to prop2)
⇒ (prop2 value v)

Explication. Cette règle de déduction indique que si une propriété de qualité `prop1` à pour valeur `v` et est équivalente à une propriété `prop2`, alors on peut en déduire que `prop2` a également pour valeur `v`.

Exemple. L'équivalence des propriétés `authentification` et `confidentiality` permet d'écrire la règle suivante.

(authentification value true) ∧
(authentification equivalent_to confidentiality)
⇒ (confidentiality value true)

`constrains` :

(prop1 value false) ∧ (prop1 constrains prop2)
⇒ (prop2 value false)

Explication. Cette règle de déduction indique que si une propriété de qualité `prop1` n'est pas satisfaite et contraint une propriété `prop2`, alors on peut en déduire que `prop2` n'est également pas satisfaite.

Exemple. Si la propriété `authentification` contraint la propriété `authorization`, on peut écrire la règle suivante.

(authentification value false) ∧
(authentification constrains authorization)
⇒ (authorization value false)

`subsumes` :

(prop1 value true) ∧ (prop1 subsumes prop2)
⇒ (prop2 value true)

Explication. Cette règle de déduction indique que si une propriété de qualité `prop1` est satisfaite et subsume une propriété `prop2`, alors on peut en déduire que `prop2` est également satisfaite.

Exemple. Si la propriété `dataEncryption` inclut la propriété `digitalSignature`, on peut écrire la règle suivante.

(`dataEncryption value true`) ∧
(`dataEncryption subsumes digitalSignature`)
⇒ (`digitalSignature value true`)

`forbids` :

(`prop1 value true`) ∧ (`prop1 forbids prop2`)
⇒ (`prop2 value false`)

Explication. Cette règle de déduction indique que si une propriété de qualité `prop1` est satisfaite et interdit une propriété `prop2`, alors on peut en déduire que `prop2` n'est pas satisfaite.

Exemple. Si la propriété `portability` interdit la propriété `codedInC`, on peut écrire la règle suivante.

(`portability value true`) ∧
(`portability forbids codedInC`)
⇒ (`codedInC value false`)

`conflicts` :

(`prop1 conflicts prop2`)
⇒ (`prop1 forbids prop2`) ∧ (`prop2 forbids prop1`)

Explication. Cette règle de déduction indique que si une propriété de qualité `prop1` est en conflit avec une propriété `prop2`, alors `prop1` interdit `prop2` (règle précédente) et vice-versa.

Exemple. Si les propriétés `portability` et `codedInC` sont en conflit, on peut écrire la règle suivante.

(`portability conflicts codedInC`)
⇒ (`portability forbids codedInC`) ∧
(`codedInC forbids portability`)

Dans cette section, nous avons montré comment les relations entre propriétés de qualité peuvent être utilisées pour réaliser des déductions et ainsi améliorer le résultat des requêtes. Notons que ces relations peuvent s'appliquer entre propriétés de qualité appartenant à différents standards. Ceci permet de retrouver un service avec des critères portant sur un standard particulier même si ce service a été décrit dans un autre standard.

6. MISE EN OEUVRE ET ÉVALUATION

Dans les deux sections précédentes nous avons présenté l'extension de OWL-S que nous proposons et son exploitation dans les requêtes. Dans cette section, nous montrons comment nous avons validé cette approche en l'implémentant, puis discutons de ses avantages et de ses limitations.

6.1 Implémentation de notre approche sur la Base de Données OntoDB

L'approche présentée dans cet article pourrait être implémentée sur tout système permettant la gestion d'ontologies et de leurs instances et proposant une implémentation du langage SPARQL. Les Bases de Données à Base Ontologique (BDBO) [14] sont des exemples de tels systèmes qui s'appuient sur la technologie des bases de données et ainsi bénéficient de leurs avantages (par exemple, la scalabilité ou la gestion de la concurrence). Aussi, pour valider notre approche, nous avons choisi d'implémenter notre approche sur la BDBO OntoDB [4]. Nous décrivons dans cette section les différentes étapes que nous avons suivies pour réaliser cette implémentation.

Importation de l'ontologie OWL-S étendue dans OntoDB

OntoDB a été définie à l'origine pour le modèle d'ontologies PLIB. Cependant, cette BDBO dispose également d'un outil d'import d'ontologies OWL. Aussi, nous avons choisi d'utiliser l'éditeur Protégé [6] pour réaliser l'extension de OWL-S présentée dans la section 4 puis utilisé l'outil d'import pour la charger dans OntoDB. Plus précisément, l'ontologie OWL-S est décomposée en quatre fichiers OWL [7] : `Service.owl`, `Profile.owl`, `Process.owl` et `Grounding.owl`. Notre extension portant sur la partie Profile de OWL-S, nous avons chargé puis édité le fichier `Profile.owl`.

Extension de SPARQL avec la clause QUALITY

Le langage d'exploitation associé à OntoDB est le langage OntoQL. Ce langage a été implémenté sur OntoDB en le traduisant en requête SQL. OntoDB dispose également d'une implémentation du langage SPARQL qui consiste à traduire une requête SPARQL en une requête OntoQL. Aussi, nous avons choisi de réaliser l'interprétation de la clause `QUALITY` telle que nous l'avons spécifiée dans la section 5.2, lors de la traduction en OntoQL. Cette approche est illustrée sur la figure 4.

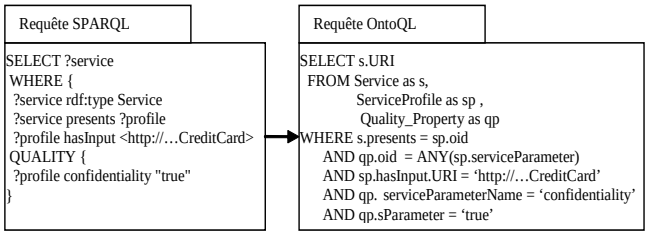

Figure 4: Interprétation de la clause `QUALITY` de SPARQL par traduction en OntoQL

Explication. La requête SPARQL initiale est présentée sur la

6. `http://protege.stanford.edu/`
7. disponibles à l'adresse : `http://www.ai.sri.com/daml/services/owl-s/1.2/`

gauche de la figure et sa traduction en OntoQL à droite. La clause `WHERE` de la requête SPARQL est traduite en OntoQL par une jointure entre les classes `Service` et `ServiceProfile` et par un prédicat dans la clause `WHERE` portant sur la propriété `hasInput`. La clause `QUALITY` est interprétée par une nouvelle jointure avec la classe `Quality_Property` et de nouveaux prédicats dans la clause `WHERE` pour rechercher la propriété de qualité portant le nom `confidentiality` et ayant la valeur `true`. Cette interprétation code les règles de réécriture que nous avons exprimées dans la section 5.2.

Implémentation des règles permettant d'exploiter les relations entre propriétés de qualité

OntoDB ne dispose pas d'un moteur de règles (base de données non déductive). Aussi, pour implémenter les règles permettant d'exploiter les relations entre propriétés de qualité spécifiées dans la section 5.3, nous avons utilisé le mécanisme de déclencheurs (trigger) des bases de données. Par exemple, pour la règle portant sur la relation `equivalent_to`, nous avons défini un déclencheur sur la table stockant les valeurs des propriétés de qualité dont le pseudo-code est le suivant :

```
Lors de l'insertion ou de la mise à jour d'une valeur v
d'une propriété de qualité p1
  Si p1 est équivalente à une propriété p2 alors
     mettre à jour la valeur de p2 à v
```

Nous avons fait de même pour les autres règles.

6.2 Avantages et limitations de notre approche

Comparée aux autres ontologies proposées par la QoS (voir section 2), l'originalité de notre proposition d'extension de OWL-S est de permettre la représentation des standards de qualité et des liens que l'on peut définir entre ces standards. L'exploitation de cette particularité au niveau des recherches de SW repose sur une extension de SPARQL pour séparer clairement l'expression des besoins fonctionnels des besoins non fonctionnels. Elle permet d'exploiter les relations que l'on peut établir entre critères de qualité grâce à des règles de déduction. Notons tout de même qu'établir des équivalences entre propriétés des différents standards nécessite une importante expertise sur les standards de qualité.

Notre extension de OWL-S se focalisant sur l'intégration des standards de qualité, nous notons que, sur d'autres aspects, cette extension est moins expressive que d'autres propositions d'ontologies pour la QoS. Par exemple, OWL-Q [9] propose un système de métriques et d'unités de mesure beaucoup plus élaboré que celui sur lequel repose notre extension. Il permet en effet d'établir des équivalences entre métriques, de réaliser des conversions entre unités, d'en définir de nouvelles, etc. De même, WSMO-QoS [18] permet de prendre en compte le fait que certaines propriétés de qualité nécessitent de mettre à jour leur valeur périodiquement et donc, qu'en conséquence, leur valeur n'est valide que dans une période de temps donné. Cet aspect qui a un impact pour la recherche de SW n'est pas supporté nativement par notre approche. Intégrer les capacités des différentes ontologies proposées pour la QoS dans notre extension fait partie des perspectives de ces travaux.

7. CONCLUSION

Dans cet article nous avons proposé une extension de OWL-S pour faciliter la recherche de SW. L'originalité de cette extension est qu'elle permet de représenter des critères de qualité tels qu'ils sont définis dans les principaux standards liés à la qualité logicielle. Cette ontologie permet non seulement de conserver l'origine de chaque critère de qualité défini mais également de représenter les liens que l'on peut établir (par exemple, l'équivalence ou la subsomption.) entre ces critères. Nous avons ensuite étudié l'intérêt de cette extension pour la recherche de SW. Nous avons d'abord montré que cette extension permet de rechercher un service selon des critères définis dans un standard donné mais que la syntaxe de SPARQL pour exprimer ces requêtes n'est pas adaptée. En conséquence, nous avons proposé une extension de ce langage qui permet clairement de distinguer l'expression des besoins fonctionnels des besoins non fonctionnels. Constatant ensuite que les relations que l'on peut établir entre des critères de qualité n'étaient pas utilisées pour améliorer le résultat des requêtes, nous avons défini un ensemble de règles de déduction pour les exploiter. Ces règles permettent à un utilisateur de retrouver un service même si celui-ci est décrit par des critères de qualité définis dans un autre standard que ceux utilisés pour exprimer la requête.

Ce travail ouvre plusieurs perspectives. Les besoins non fonctionnels n'étant généralement pas cruciaux pour un utilisateur mais servant plutôt à ordonner les services répondant aux besoins fonctionnels, nous développons une approche permettant d'exprimer ces besoins comme des préférences utilisateurs. A plus long terme, nous souhaitons étudier l'intérêt de notre approche pour permettre de rechercher des compositions de SW permettant de répondre, autant que possible, aux besoins non fonctionnels. Nous envisageons également d'étendre notre ontologie sur la QoS pour intégrer les capacités des ontologies de QoS existantes. Enfin, il serait nécessaire de développer des outils permettant aux différents acteurs impliqués dans l'utilisation de SW de bénéficier de notre approche.

8. REFERENCES

[1] D. Austin, A. Barbir, C. Ferris, and S. Garg. Web Services Architecture Requirements. *W3C Working Group Note 11 February 2004*, 2004. `http://www.w3.org/TR/wsa-reqs/`.

[2] R. C. Boer, P. Lago, A. Telea, and H. V. Vliet. Ontology-Driven Visualization of Architectural Design Decisions. In *Proceedings of the 8th Working IEEE/IFIP Conference on Software Architecture (WICSA 2009)*.

[3] M. Dean and G. Schreiber. *OWL Web Ontology Language Reference*. World Wide Web Consortium, 2004. `http://www.w3.org/TR/owl-ref`.

[4] H. Dehainsala, G. Pierra, and L. Bellatreche. OntoDB : An Ontology-Based Database for Data Intensive Applications. In *Proceedings of the 12th International Conference on Database Systems for Advanced Applications (DASFAA'07)*, volume 4443 of *Lecture Notes in Computer Science*, pages 497–508. Springer, 2007.

[5] E. Giallonardo and E. Zimeo. More Semantics in QoS Matching. In *Proceedings of the IEEE International Conference on Service-Oriented Computing and*

Applications (SOCA 2007), pages 163–171, Washington, DC, USA, 2007. IEEE Computer Society.

[6] ISO/IEC13236. Quality of service : Framework. Technical report, International Organisation for Standardisation / International Electrotechnical Commission, 1999.

[7] ISO/IEC9126-1. Quality characteristics and guidelines for their use (1, 2). Technical report, International Organisation for Standardisation / International Electrotechnical Commission, 2001.

[8] S. Jean, Y. Aït-Ameur, and G. Pierra. Querying Ontology Based Database Using OntoQL (an Ontology Query Language). In R. Meersman and Z. T. et al., editors, *Proceedings of On the Move to Meaningful Internet Systems 2006 : CoopIS, DOA, GADA, and ODBASE, OTM Confederated International Conferences (ODBASE 2006)*, volume 4275 of *Lecture Notes in Computer Science*, pages 704–721. Springer, 2006.

[9] K. Kritikos and D. Plexousakis. OWL-Q for Semantic QoS-based Web Service Description and Discovery. In *Proceedings of the 1st Workshop on Service Matchmaking and Resource Retrieval in the Semantic Web (SMRR 2007)*, 2007.

[10] P. Krutchen. An ontology of architectural design decisions in software intensive systems. In *Proceedings of the 2nd Groningen Workshop on Software Variability Management*, pages 54–61, October 2004.

[11] F. Losavio, A. Matteo, and N. Levy. Web Services Domain Knowledge with an Ontology on Software Quality Standards. In *Proceedings of the Third International Conferences on Internet Technologies and Applications (ITA 2009)*, pages 74–85, 2009.

[12] D. Martin, M. Burstein, E. Hobbs, O. Lassila, D. Mcdermott, S. Mcilraith, S. Narayanan, B. Parsia, T. Payne, E. Sirin, N. Srinivasan, and K. Sycara. *OWL-S : Semantic Markup for Web Services*. World Wide Web Consortium, 2004. `http://www.w3.org/Submission/OWL-S/`.

[13] G. Pierra. Context Representation in Domain Ontologies and its Use for Semantic Integration of Data. *Journal Of Data Semantics (JODS)*, X :34–43, 2007.

[14] G. Pierra, H. Dehainsala, Y. Aït-Ameur, and L. Bellatreche. Base de Données à Base Ontologique : principes et mise en œuvre. *Ingénierie des Systèmes d'Information*, 10(2) :91–115, 2005.

[15] E. Prud'hommeaux and A. Seaborne. SPARQL Query Language for RDF. *W3C Recommendation 15 January 2008*, 2008. `http://www.w3.org/TR/rdf-sparql-query/`.

[16] G. F. Tondello and F. Siqueira. The QoS-MO ontology for semantic QoS modeling. In *Proceedings of the 2008 ACM symposium on Applied computing (SAC 2008)*, pages 2336–2340, New York, NY, USA, 2008. ACM.

[17] V. X. Tran and H. Tsuji. A Survey and Analysis on Semantics in QoS for Web Services. *Proceeding of the 23rd International Conference on Advanced Information Networking and Applications (AINA 2009)*, 0 :379–385, 2009.

[18] 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)*, pages 390–401, 2006.

[19] C. Zhou, L.-T. Chia, and B.-S. Lee. DAML-QoS Ontology for Web Services. In *Proceedings of the IEEE International Conference on Web Services (ICWS 2004)*, page 472, Washington, DC, USA, 2004. IEEE Computer Society.