Prise en compte des standards de qualité et des préférences utilisateurs pour la modélisation des propriétés non fonctionnelles dans OWL-S

Stéphane Jean* — **Francisca Losavio**** — **Alfredo Matteo**** — **Nicole Levy*****

* *Laboratoire d'Informatique Scientifique et Industrielle*
LISI/ENSMA et Université de Poitiers
BP 40109, 86961 Futuroscope Cedex, France
jean@ensma.fr

** *Centro ISYS, Escuela de Computación, Facultad de Ciencias*
Universidad Central de Venezuela
francislosavio@gmail.com, almatteo@cantv.net

*** *Université de Versailles*
St-Quentin en Yvelines, France
nlevy@prism.uvsq.fr

RÉSUMÉ. *Avec la prolifération des Services Web, le problème de la découverte de services est devenu crucial. Les approches basées sur UDDI ne permettant qu'une recherche syntaxique sur l'interface des services, des approches sémantiques à base d'ontologies, comme par exemple OWL-S, ont été développées. Ces approches permettent une description plus précise des fonctionnalités des Services Web mais ne proposent que peu d'éléments pour en capturer les aspects non fonctionnels qualifiés de Qualité de Services (QoS). Pour répondre à ce besoin, des travaux ont proposé des approches de découverte de Service Web basées sur des ontologies de QoS. Cependant, ces approches ne prennent pas en compte les différents standards de qualité existants et les relations que l'on peut établir entre eux. Pourtant, ces normes fournissent un référentiel commun pour les fournisseurs et les consommateurs de services facilitant ainsi la recherche de services. Dans cet article, nous proposons une extension de OWL-S permettant de décrire la qualité d'un Service Web en fonction d'un ou plusieurs standards. Nous proposons ensuite une approche permettant d'exploiter cette extension de OWL-S pour faciliter la recherche de Services Web en fonction de critères de qualité. Cette approche repose sur une extension de*

SPARQL pour simplifier l'expression de requêtes. Elle permet de retrouver un service même si celui-ci est décrit par des critères de qualité d'un autre standard que celui utilisé pour exprimer la requête. Enfin, elle permet d'exprimer les besoins non fonctionnels comme des préférences utilisateurs servant ainsi à ordonner les services répondant aux besoins fonctionnels.

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 descriptions of Web Services (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 do not take into account existing standards about software quality and the relationships that can be established between them. Yet, these standards could be used as a shared understanding between services providers and customers and thus, they would ease the Web Services discovery process. In this article we first propose an extension of OWL-S to describe QoS according to one or many quality standards. Then, we develop an approach based on this extension of OWL-S to improve the Web Services discovery process. This approach is based on an extension of SPARQL that simplifies expression of Web Services discovery queries. Relationships between standards are used to return Web Services even if they are described with quality properties defined in an other standard that the one used to express queries. Finally, non functional requirements can be expressed as user preferences. Thus, they can be used to rank Web Services fulfilling functional requirements during the Web Service discovery process.*

MOTS-CLÉS : *Services Web, standards de qualité, propriétés non fonctionnelles, préférences utilisateurs.*

KEYWORDS: *Web Services, quality standards, non functional properties, user preferences.*

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 (Martin *et al.*, 2004) 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, par contre, que très peu d'éléments pour en décrire la qualité.

Plusieurs normes ont été définies par des institutions telles que l'ISO/IEC (ISO/IEC9126-1, 2001; ISO/IEC13236, 1999), IEEE (IEEE-Std-1061-1998, 1998) ou le W3C (Austin *et al.*, 2004) pour définir précisément des critères de qualités de produits logiciels. L'utilisation de ces standards dans le domaine des SW pourraient permettre aux fournisseurs et aux consommateurs de SW de se baser sur un référentiel commun de qualité et ainsi les aider à arriver à un accord. 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. En conséquence, il est difficile d'apprécier la qualité des SW décrits dans ces approches. De plus, même si les standards de qualité sont utilisés, 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. Il est donc nécessaire de représenter et d'utiliser les relations que l'on peut établir entre les propriétés de qualités et les métriques des différents standards.

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 (Losavio *et al.*, 2009). 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 (1) une extension du langage SPARQL (Prud'hommeaux *et al.*, 2008) pour simplifier l'expression de requêtes portant sur la qualité de SW , (2) un ensemble de règles pour exploiter les différentes relations qui peuvent être établies entre les critères de qualité provenant éventuellement de différents standards et (3) une approche pour exprimer les besoins non fonctionnels comme des préférences utilisateur et ainsi permettre à un utilisateur d'ordonner les services qui satisfont ses besoins fonctionnels selon ses préférences de qualité.

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 (Kritikos *et al.*, 2007; Tondello *et al.*, 2008; Giallonardo *et al.*, 2007; Zhou *et al.*, 2004). Dans un état de l'art récent (Tran *et al.*, 2009), 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 discutons dans cette section des principaux travaux proches de nos travaux.

OWL-Q (Kritikos *et al.*, 2007) est une extension de OWL-S basée sur un ensemble de facettes qui peuvent être développées et étendues séparément. La principale facette sur laquelle se focalise ce travail concerne la définition d'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. Cette facette permet de distinguer les attributs de qualité dont les valeurs sont statiques de celles dont les valeurs sont dynamiques (changement au cours du temps). Elle ne permet par contre pas de définir une hiérarchie entre ces attributs ni d'autres relations que l'on peut établir et qui sont définies dans les standards de qualité.

QOS-MO (Tondello *et al.*, 2008) permet de représenter la QoS de SW décrits en OWL-S. Sa particularité est de permettre de décrire l'interaction entre les fournisseurs et consommateurs 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 consommateur sur la qualité d'un SW. Les caractéristiques de qualité

sont définies par la classe `QoSCharacteristic`. Cependant cette classe ne propose que peu d'élément pour définir les types de données et les unités de mesure de ces caractéristiques. De même, elle ne permet pas de définir de relations entre ces caractéristiques, ni d'établir de lien avec les standards de qualité.

onQoS (Giallonardo *et al.*, 2007) définit des éléments permettant d'associer des concepts de QoS aux profils OWL-S de SW. Cette ontologie propose notamment un puissant système de types 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. Si cette approche fournit une spécification détaillée pour les types de données et les métriques, elle ne fournit aucun élément supplémentaire à OWL-S pour décrire les caractéristiques de qualité.

DAML-QoS (Zhou *et al.*, 2004) 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 consommateur de SW pour obtenir la qualité de service spécifiée par le fournisseur. Cependant, cette ontologie ne fournit pas de description concrète des propriétés de qualité.

FQoS (Jeong *et al.*, 2009) définit un ensemble d'attributs fonctionnels (catégorie, nom, nom d'opération, données d'entrée/sortie, annotation) en spécifiant leur métrique ainsi que les opérations nécessaires à leur utilisation pour le processus de sélection. Elle permet également de définir des attributs de qualité (coût d'invocation, performance, disponibilité, sécurité, etc.). Ces caractéristiques fonctionnelles et non fonctionnelles sont ensuite utilisées dans le processus d'appariement entre demande et offre de services. Cependant, ces travaux ne considèrent ni la possibilité d'enrichir la description de service avec des ontologies, ni le problème de terminologie des attributs de qualité dans le processus de sélection. Par ailleurs, cette approche ne permet pas de spécifier différents niveaux de granularité pour les attributs de qualité. En effet, le terme *attribut de qualité* utilisé dans cette approche peut représenter sans distinction des caractéristiques de qualité de bas ou haut niveau. Il n'y a ainsi pas de notion d'hiérarchie entre les attributs de qualité et pas de distinction entre eux. Ainsi, l'hypothèse est faite que touts les attributs de qualité utilisés peuvent être directement mesurés y compris la disponibilité ou la sécurité. Enfin, notons que cette approche ne mentionne pas les standards de qualité existants.

SL-Ontology (Bleul *et al.*, 2005) est une ontologie construite en couches. Elle s'appuie sur l'ontologie `QoS-Ontology` pour décrire les différentes caractéristiques de qualité du service en utilisant des taxonomies afin d'identifier les caractéristiques de qualité qui correspondent sémantiquement au même concept. Chaque caractéristique de qualité est associée à une métrique définie dans l'ontologie `Metric-Ontology`. La relation entre différentes unités est définie dans l'ontologie `Unit-Transformation-Ontology`. Cette ontologie se base sur l'ontologie `Grounding-Ontology` pour définir des règles de transformation d'unité. Si cette approche utilise des taxonomies pour régler le problème de la variabilité de terminologie

pour les noms de caractéristiques de qualité, elles ne permet pas de détailler ces caractéristiques en spécifiant par exemple leur définition, leur niveau d'abstraction ou les standards de qualité qui les définissent. Ainsi l'équivalence entre caractéristiques de qualité repose uniquement sur les noms donnés.

QoSOnt (Dobson *et al.*, 2005) est un ensemble d'ontologies interconnectées permettant de spécifier les besoins de qualité de service. QoSOnt comprend une ontologie qui permet de faire le lien avec l'ontologie OWL-S, c'est à dire d'associer des attributs et des métriques aux services OWL-S. Elle utilise également une ontologie pour chaque attribut de qualité ainsi qu'une ontologie pour les métriques. Dans ces travaux, le problème principal qui est traité est celui d'avoir une même métrique associée à des unités différentes. Ils ne permettent pas de définir plusieurs niveaux d'abstractions d'attributs de qualité ni d'identifier les attributs de qualité qui représentent le même concept même si ils sont définis par des termes différents.

Tous ces travaux s'intéressent à l'amélioration du processus de découverte de SW en prenant en compte les propriétés non fonctionnelles décrites par des ontologies. Nous notons qu'aucune de ces approches ne permet d'identifier différents niveau d'abstraction entre les propriétés de qualité et de distinguer les propriétés mesurables de celles qui ne le sont pas. Ainsi, nous trouvons par exemple les notions de débit ou de latence au même niveau qu'efficacité, interopérabilité, fiabilité ou sécurité. Pourtant ces distinctions et caractérisations sont établies dans plusieurs standards de qualité. Ainsi, 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 (ISO/IEC9126-1, 2001)) du standard dans lequel elles ont été définies. Pourtant ces informations s'avèrent cruciales pour apprécier la qualité du SW décrit. 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. Par ailleurs, même si les relations d'équivalence et de subsomption sont disponibles dans OWL, elles ne s'appliquent qu'aux classes d'une ontologies et non à leurs instances. Or, les propriétés de qualité sont représentées comme des instances de classes OWL. Et donc, ces relations ne peuvent pas être utilisées pour établir de liens entre les propriétés de qualité des différents standards. Pourtant, ce type de relation permet de traiter le problème de variabilité de terminologie. Par exemple, la capacité d'un service à supporter le changement peut être appelée *changeability*, *flexibility* ou *extensibility* par différentes organisations.

Au niveau des requêtes, nous notons que ces approches se sont essentiellement intéressées au développement d'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 recherchant un SW dans un langage d'interrogation donné ni sur l'utilisation des critères de qualité pour ordonner (et non filtrer)

les résultats de ces requêtes. 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. Cette approche étend OWL-S plutôt qu'une des approches citées précédemment car la spécification et la représentation de OWL-S en OWL sont disponibles.

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 *OSQS (Ontology for Software Quality Standards)* que nous avons proposée dans (Losavio *et al.*, 2009). 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 consommateurs 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) (Austin *et al.*, 2004).

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`) qui sont ensuite raffinés en critères plus précis. Par exemple, `Security` est raffiné par `Privacy protection` et `Protection from threats`; ce dernier est à nouveau raffiné en plusieurs critères tels que `Threat of accessibility attacks`, `Authentication of the parties`, `Authentication of authorship of data` et `Authorization`. Notons que ce standard ne définit pas de métriques.

La norme ISO/IEC 9126-1 (ISO/IEC9126-1, 2001)

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 en-

suite raffinées jusqu'à atteindre des caractéristiques mesurables appelées attributs de qualité. Par exemple, `Functionality` implique l'existence d'un ensemble de fonctions qui satisfont un besoin particulier. Il est raffiné en sous-caractéristiques telles que `Suitability`, `Accuracy`, `Interoperability` et `Security`. Ces sous-caractéristiques peuvent être elles-mêmes raffinées selon le domaine d'application visé. La hiérarchie de caractéristiques de la norme ISO 9126-1 constitue le modèle de qualité d'une application dans un domaine donné.

Récemment, une nouvelle norme nommée ISO/IEC 25010 (ISO/IEC JTC1/SC7 N4522, FCD 25010, 2010) a été définie et devrait remplacer à terme la norme ISO/IEC 9126-1. Dans cette nouvelle version, `compatibility` (`co-existence`, `interoperability`) et `security` (`confidentiality`, `integrity`, `non-repudiation`, `accountability`, `authenticity`) ont été ajoutées comme caractéristiques de haut niveau. De plus, `functionality` a été renommée en `functional suitability` (`completeness`, `correctness`, et `appropriateness`). La sous-caractéristique `completeness` correspond à présent à la caractéristique `accuracy` de la norme ISO/IEC 9126-1. La caractéristique `maintainability` a également été modifiée : `modularity` et `reusability` ont été ajoutées comme nouvelles sous-caractéristiques ; `modifiability` conserve le même nom mais intègre à présent les deux sous-caractéristiques `stability` et `changeability` de la norme ISO/IEC 9126-1. Dans la suite de cet article, nous utiliserons de préférence des exemples issus de la norme ISO/IEC 9126-1 plutôt que de la norme ISO/IEC 25010, car elle nous semble à l'heure actuelle plus mature.

Le modèle de qualité défini dans la norme ISO/IEC 9126-1 et mis à jour dans la norme ISO/IEC 25010 est un framework qui doit être adapté au domaine d'application cible. Cependant, aucun guide n'est fourni pour réaliser cette adaptation ce qui, en pratique, en rend son utilisation délicate. L'ontologie OSQS que nous proposons dans nos travaux peut aider à construire le modèle de qualité adapté en permettant de faire des requêtes sur les caractéristiques de qualité et les métriques correspondantes. Le cadre d'application de ce type de modèle de qualité inclut l'aide à la spécification et l'évaluation de logiciel pour ceux responsables de leur développement, utilisation, maintenance et audit. Ces modèles de qualité sont en particuliers très utiles pour identifier les propriétés architecturales et les besoins globaux d'un système.

Contrairement à WSA, la norme ISO/IEC 9126-1 et son évolution ne sont donc pas spécifiques 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 (ISO/IEC13236, 1999)

La norme ISO/IEC 13236 porte sur le domaine de la qualité de services. Elle définit une terminologie et des concepts sur la QoS de manière à fournir un langage commun pour les consommateurs et les fournisseurs de services distribués (par exemple, des SW mais pas obligatoirement). Par ailleurs, elle introduit un ensemble de carac-

téristiques, de métriques et de mécanismes pour permettre de caractériser, spécifier et gérer les besoins liés à la QoS. Plus précisément, il propose des catégories générales de caractéristiques de qualité. Ces caractéristiques représentent un aspect de la QoS d'une application, d'un service ou d'une ressource qui peut être identifié et quantifié. Elles concernent principalement des aspects temporels, de capacité, de fiabilité et de sécurité. Un exemple de caractéristique est `Security`. `Security` est raffinée en `Access control` (protection contre des accès non autorisés à une ressource) et `Data protection` (protection contre des accès non autorisés à des données). Ces caractéristiques sont mesurées par une valeur liée à un contrôle d'accès ou à une politique d'intégrité des données. Par exemple, la probabilité d'échec d'une politique (valeur réelle) ou la présence ou non d'une politique (valeur booléenne) sont considérées comme des métriques acceptables de ces caractéristiques.

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 (Losavio *et al.*, 2009), 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 OSQS*

Les principaux éléments de l'ontologie OSQS sont présentés sur la figure 1.

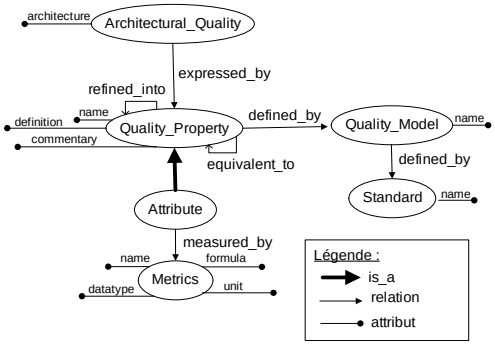

Figure 1. *Extrait de l'ontologie OSQS*

– **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 caractérisé par les propriétés de qualité `Interoperability`, `Security`, `Reliability` et `Scalability`.

– **Quality_Property** : représente une propriété de qualité. Cette propriété peut être raffinée par une ou plusieurs autres propriétés de qualité qui peuvent oui ou non être mesurées. `Security` est un exemple de propriété de qualité. Dans le standard WSA,

elle est raffinée par `Protection from threats` et `Privacy policy` ; `Protection from threats` est à son tour raffiné par la propriété de qualité mesurable (appelé attribut) `Authorization`. 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 13236, `Access control` (ISO 13236) est équivalent à `Authorization` (WSA). Pour simplifier la figure 1, 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, `Access control` défini dans la norme ISO 13236 indique si oui ou non l'application repose sur une politique de contrôle d'accès. Elle peut ainsi être mesurée par une valeur booléenne. Notons que les propriétés mesurables dont la métrique n'est pas précisée peuvent être caractérisées par la métrique d'une propriété équivalente. Par exemple, même si WSA ne fournit aucun métrique, la propriété `Authorization` de ce standard peut être associée à une métrique booléenne (montré sur la figure 2) puisqu'elle est équivalente à `Access control` de la norme ISO 13236. Notons également que certaines métriques sont implicites. Par exemple, `Privacy policy` a une valeur booléenne puisqu'un service a ou n'a pas de politique de confidentialité.

– **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. La valeur elle-même n'est pas représentée dans cette ontologie dont l'objectif est uniquement de représenter les concepts des standards de qualité. Dans la suite, nous verrons que cette valeur sera explicitement représentée en l'importante de l'ontologie OWL-S. Comme exemple de métrique nous pouvons citer `Data protection` qui est un raffinement de `Security` dans la norme ISO 13236 et qui correspond à la probabilité de défaillance de la protection. C'est une métrique de 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. La définition des unités de mesure dépend du modèle d'implémentation choisi. Par exemple, le modèle d'ontologies PLIB (Pierra, 2007) supporte nativement les unités de mesure contrairement au modèle d'ontologies OWL. Dans le second cas, nous pouvons utiliser comme extension une ontologie qui les définit comme par exemple la *Measurement Units Ontology*[1].

– **Quality_Model** : représente un ensemble de propriétés de qualité qui spécifient la qualité d'un produit logiciel. Le modèle de qualité peut être utilisé pour la spécification et l'évaluation de logiciels suivant différents points de vue (audit, maintenance, support, etc.).

– **Standard** : représente une norme établie définissant un ensemble de critères sur la qualité logicielle. ISO 13236 et WSA sont des exemples de standards qui peuvent être utilisés dans le domaine des SW.

---

1. `http://idi.fundacionctic.org/muo/muo-vocab.html`

La figure 2 reprend les exemples évoqués précédemment en montrant un exemple d'instanciation de l'ontologie OSQS. 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 `Security` définie dans le standard ISO 13236 qui est raffinée par les attributs `Data protection` et `Access control` dont les mesures sont respectivement une valeur réelle et une valeur booléenne. Cette figure montre également que l'ontologie OSQS permet d'exprimer des équivalences entre propriétés de qualité. Une équivalence est en effet indiquée sur cette figure entre les propriétés de qualité `Access control` et `Authorization`.

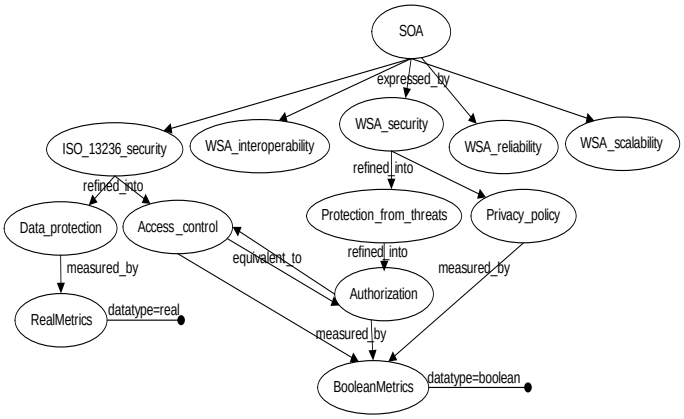

Figure 2. *Exemple d'instanciation de l'ontologie OSQS*

Nous montrons maintenant comment nous avons utiliser l'ontologie OSQS 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 (Martin *et al.*, 2004) 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 (Boer *et al.*, 2009) 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 en spécialisant `ServiceParameter` afin de supporter les propriétés de qualité. Cette spécialisation repose sur l'ontologie présentée dans la section précédente. La difficulté sous-jacente consiste à définir la sémantique de l'extension proposée comme par exemple la subsomption ou l'équivalence entre propriétés de qualité.

4.2. *Extension de OWL-S proposée*

L'extension de OWL-S que nous proposons est présentée sur la figure 3. En OWL-S, le `ServiceProfile` caractérise les fonctionnalités du service (`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 inchangé 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 donnée par l'attribut `sParameter` hérité de `ServiceParameter`. Pour les instances de `Quality_Property`, la valeur de cette propriété est une valeur anonyme (un noeud blanc en RDF).

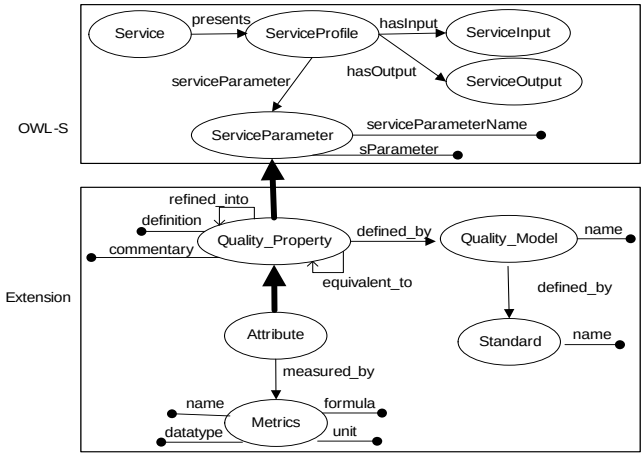

Figure 3. *Extension de OWL-S avec OSQS*

Dans cette section, nous avons d'abord identifié les limitations de OWL-S pour la description de la qualité de SW puis montré comment nous pouvons l'étendre avec l'ontologie OSQS pour prendre en compte les standards de qualité logicielle. Nous montrons maintenant l'intérêt de cette extension pour la 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 offertes par notre extension de OWL-S*

OWL-S étant une ontologie OWL, sa représentation sous forme de triplets RDF est disponible et peut être interrogée avec le langage SPARQL (Prud'hommeaux *et al.*, 2008). La difficulté sous-jacente consiste à écrire des requêtes dont la syntaxe et la sémantique soient pertinentes par rapport à l'extension de OWL-S que nous proposons pour la qualité de service. Notons que nous aurions pu utiliser d'autres langages d'interrogation d'ontologies pour illustrer notre approche. Nous l'avons d'ailleurs mise en oeuvre non seulement pour SPARQL mais également pour le langage OntoQL (Jean *et al.*, 2006) comme nous le verrons dans la section 6.

Exemple. La requête SPARQL[2] suivante permet de rechercher les services qui prennent en entrée une carte de crédit et qui utilise un `Access control` tel qu'il est défini dans dans la norme ISO 13236.

```
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/ISO13236>
  ?parameter serviceParameterName "Access_control"
  ?parameter sParameter ?value FILTER (?value = true) }
```

Explication. Le premier triplet de la clause `WHERE` introduit la variable `?service` pour parcourir l'ensemble des services. Ces services peuvent être instance de la classe `Service` ou d'une de ses sous-classes (parcours de la relation de subsumption). La variable `?service` 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. Cette propriété peut être une instance de `ServiceParameter` ou d'une de ses sous-classes (`Quality_Property` et `Attribute`). Les deux triplets suivants permettent de vérifier que cette propriété de qualité fait partie d'un modèle de qualité défini par la norme ISO 13236. Les trois suivants vérifient que cette propriété de qualité a pour nom `Acces_control` 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

2. Par mesure de concision, les espaces de noms sont omis dans les 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 trois principales limitations suivantes.

– 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` du standard WSA même si celle-ci est déclarée comme étant équivalente à la propriété `Access_control` de la norme ISO 13236. Notons qu'un utilisateur sachant que ces propriétés sont équivalentes pourrait écrire une requête SPARQL utilisant l'opérateur `UNION` pour obtenir le résultat attendu. Cependant, nous souhaitons qu'automatiquement la requête précédente retourne le bon résultat et donc que son interprétation prenne en compte la relation d'équivalence ainsi que les autres relations que l'on peut établir.

– Les contraintes que l'on peut exprimer sur les propriétés de qualité dans les requêtes sont des contraintes strictes (un service ne les respectant pas ne sera pas retourné). Or, généralement, un utilisateur recherchera avant tout un service qui réalise les fonctionnalités dont il a besoin (contraintes strictes sur les besoins fonctionnels) et préférera ensuite ceux qui satisfont ses critères de qualité (contraintes lâches sur les besoins non fonctionnels).

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 en 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. Cette extension vise ainsi à simplifier l'écriture de requête utilisant des propriétés de qualité tout en assurant une dépendance faible vis-à-vis de l'extension de l'ontologie OWL-S proposée.

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[3] 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 ce qui peut être utile pour retrouver sa description. Le 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 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
```

3. 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`.

```
WHERE {
 ?service rdf:type Service . ?service presents ?profile
 ?profile hasInput <http://cordio.bs/CreditCard> }
QUALITY {
 ?profile Access_control "true"
 Access_control defined_by ?qualityModel
 ?qualityModel defined_by <http://www.iso.org/ISO13236> }
```

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 :

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

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 triplet indique que l'objet du triplet (`o`) correspond à la valeur de la propriété `sParameter`. 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 `Access_control` comme une propriété usuelle en SPARQL est interprétée de la manière suivante :

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

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.

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

Explication. Si dans une requête une valeur `o` d'une propriété de qualité `p` est associée à un type de données, on réalise une réécriture de requête afin de vérifier que `p` 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 `Access_control` caractérisé par le type de données `xsd:boolean`.

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

Pour le traitement des unités de mesure, nous utilisons une approche fréquemment utilisée [4] 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.

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

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 requêtes, la distinction 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).

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

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

$$(?\texttt{profile}\ \texttt{maxTimeByTransaction}\ "\texttt{1}"\hat{}\texttt{mymw} : \texttt{second})$$
$$\Rightarrow (?\texttt{profile}\ \texttt{maxTimeByTransaction}\ "\texttt{1}")\ .$$
$$(\texttt{maxTimeByTransaction}\ \texttt{measured\_by}\ ?\texttt{m})\ .$$
$$(?\texttt{m}\ \texttt{unit}\ "\texttt{mymw} : \texttt{second}")$$

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 (Krutchen, 2004) et plus précisément sur l'interprétation qui en est donnée dans (Boer *et al.*, 2009). 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, trois autres relations : `constrains`, `subsumes` et `forbids`. 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`. Notons que cette propriété est symétrique : si `X`

`forbids Y` alors `Y forbids X`. En conséquence cette relation correspond également à la relation `conflicts` définie dans (Krutchen, 2004).

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` :

$$(\texttt{prop1 value v}) \wedge (\texttt{prop1 equivalent\_to prop2}) \Rightarrow (\texttt{prop2 value v})$$

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 `Access_control` et `authorization` permet d'écrire la règle suivante.

$$(\texttt{Access\_control value true}) \wedge$$
$$(\texttt{Access\_control equivalent\_to authorization})$$
$$\Rightarrow (\texttt{authorization value true})$$

`constrains` :

$$(\texttt{prop1 value false}) \wedge (\texttt{prop1 constrains prop2}) \Rightarrow (\texttt{prop2 value false})$$

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.

$$(\texttt{authentification value false}) \wedge$$
$$(\texttt{authentification constrains authorization})$$
$$\Rightarrow (\texttt{authorization value false})$$

subsumes :

$$(\texttt{prop1 value true}) \wedge (\texttt{prop1 subsumes prop2}) \Rightarrow (\texttt{prop2 value true})$$

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.

$$(\texttt{dataEncryption value true}) \wedge$$
$$(\texttt{dataEncryption subsumes digitalSignature})$$
$$\Rightarrow (\texttt{digitalSignature value true})$$

forbids :

$$(\texttt{prop1 value true}) \wedge (\texttt{prop1 forbids prop2}) \Rightarrow (\texttt{prop2 value false})$$

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.

$$(\texttt{portability value true}) \wedge (\texttt{portabilitiy forbids codedInC})$$
$$\Rightarrow (\texttt{codedInC value false})$$

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.Dans la section suivante, nous montrons que la recherche de service peut également être facilitée en exprimant les critères de qualité comme des préférences utilisateurs.

5.4. *Expression des besoins non fonctionnels comme des préférences utilisateurs*

Si les besoins fonctionnels sont généralement des critères qui doivent être satisfaits par le service recherché, les besoins non fonctionnels sont souvent des critères moins

rigides utilisés pour ordonner les services répondant aux fonctionnalités voulues. Or, la plupart des approches actuelles traitent ces deux types de critère de la même façon et donc ne retournent pas les services qui satisfont les besoins fonctionnels et partiellement les besoins non fonctionnels. Ceci rend difficile la recherche d'un service par un utilisateur, le laissant souvent sans résultat s'il exprime ses besoins non fonctionnels.

Ces dernières années, plusieurs travaux se sont intéressés à l'expression des préférences utilisateurs dans le contexte des ontologies (Siberski *et al.*, 2006; Gurský *et al.*, 2008; Toninelli *et al.*, 2008; Tapucu *et al.*, 2009). Ces approches permettent d'associer des préférences à des ontologies qui pourront ensuite être traitées comme des contraintes lâches dans les requêtes. Ainsi, ces approches sont bien adaptées à la recherche de service par les besoins non fonctionnels. Cependant, elles doivent être étendues pour permettre d'exprimer des besoins non fonctionnels comme des préférences utilisateur. Nous avons choisi d'étendre l'approche de Tapucu et al. (Tapucu *et al.*, 2009) car elle propose un modèle complet et extensible pour exprimer des préférences utilisateurs et une extension de SPARQL et OntoQL pour les exploiter. Nous commençons d'abord par présenter succinctement cette approche.

5.4.1. *Modèle de préférence utilisé*

Les principaux éléments du modèle de préférence sur lequel nous nous appuyons sont présentés sur la figure 4. Dans ce modèle, chaque préférence (`Preference`) est identifiée par une URI (`Preference_URI`). Les différents types de préférences que nous utilisons pour l'expression des besoins non fonctionnels sont les suivants.

– `Numeric_Pref` : elles correspondent aux préférences définies par une valeur numérique (`numberValue`). Par exemple, ce type de préférence permet d'exprimer une préférence pour les hôtels ayant 4 étoiles.

– `Fuzzy_Pref` : elles correspondent à des préférences exprimées en terme de valeurs de probabilité (`probValue`). Par exemple, ce type de préférence permet d'exprimer que les probabilités de préférer les hôtels 2, 3 et 4 étoiles sont respectivement 0.1, 0.2, et 0.7.

– `Boolean_Pref` : elles correspondent à une liste de propriétés booléennes dont on préfère que la valeur soit à vrai. Par exemple, ce type de préférence permet d'exprimer une préférence pour les chambres d'hôtel disposant d'une TV, de l'air conditionné et du wifi.

– `Interval_Pref` : elles correspondent aux préférences que l'on exprime via une valeur minimum et maximum. Par exemple, le coût d'un matériel peut être associé à une préférence `cheap` ou `expensive`. Dans ce cas, ces préférences permettent d'indiquer que `cheap` correspond à l'interval [45..60] et `expensive` à [90..100].

– `Enumerated_Pref` : elles correspondent à l'énumération d'instances d'une ontologie qui sont préférées. Par exemple, ce type de préférence permet de définir une préférence pour les hôtels `économiques` comme étant `{Hotel(Formule1), Hotel(Premiere)}`.

La partie gauche de la figure 4 montre comment les préférences sont liées au modèle d'ontologies. Ce lien associe les préférences du modèle de préférence et les classes et propriétés du modèle d'ontologies. Les classes et propriétés sont modélisées comme une entité nommée `Property_Or_Class`. Le lien est réalisé par une entité nommée `Pref_Link`.

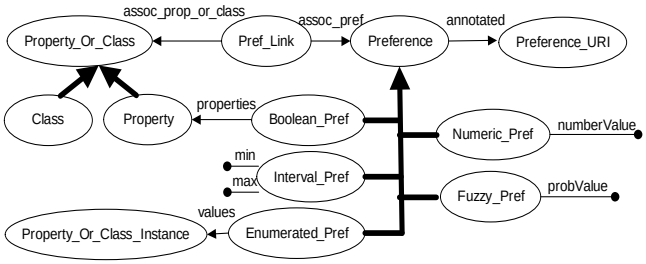

Figure 4. *Extrait du modèle de préférence utilisé*

Le modèle utilisé permet de définir des préférences qui peuvent ensuite être exploitées dans les requêtes. En effet, l'approche de Tapucu et al. (Tapucu *et al.*, 2009) propose l'extension des langages SPARQL et OntoQL avec une clause `PREFERRING` pour prendre en compte les préférences. Par exemple, la requête suivante permet à un client préférant les hôtels `cheap` de rechercher un hôtel pertinent.

```
SELECT ?name ?price
WHERE { ?h type Hotel . ?h name ?name .?h price ?price }
PREFERRING { cheap }
```

Explication. La clause PREFERRING permet de spécifier les préférences qui doivent être prises en compte dans la requête. Ces préférences sont indiquées par des labels qui sont ensuite interprétés en fonction du type de préférence considéré. Dans notre exemple, `cheap` est une préférence de type `Interval_Pref` qui porte sur la propriété `price` et dont l'intervalle est [45..60]. Ainsi, les hôtels dont le prix est dans cet interval seront retournés en premier comme résultats. Puis ce seront ceux dont le prix est le plus proche de cet interval.

Ayant présenté le modèle de préférence sur lequel nous nous appuyons, nous montrons maintenant comment nous pouvons l'adapter à notre extension de OWL-S.

5.4.2. *Préférences sur les propriétés de qualité*

Le modèle de préférence présenté dans la section précédente permet d'exprimer des préférences sur des classes et des propriétés. Il peut donc être utilisé pour associer des préférences à des SW (instances de la classe `Service`). Par contre, il ne permet pas d'associer des préférences à des propriétés de qualité de la même manière qu'aux

autres propriétés. Par exemple, les `Boolean_Pref` peuvent être associées à des propriétés usuelles mais ne peuvent pas être associées à des propriétés de qualité puisque ces dernières sont des instances de `Quality_Property`. En conséquence, nous proposons d'étendre ce modèle pour pouvoir exprimer des préférences utilisateurs sur des propriétés de qualité. La figure 5 présente en italique et gras les nouveaux éléments liés à cette extension ainsi que les liens qui unissent ces éléments au reste du modèle initial. Les trois extensions notoires sont les suivantes.

– Pour associer une préférence à une propriété de qualité, `Pref_Link` est liée à `Quality_Property` (élément de notre extension de OWL-S) via la propriété `assoc_qualityProp`.

– Pour permettre de définir une préférence booléenne sur des propriétés de qualité, `Boolean_Pref` est liée à `Quality_Property` via la propriété `qualProperties`.

– Pour permettre de définir une préférence énumérée sur des valeurs de propriétés de qualité, `Enumerated_Pref` est liée à `Quality_Property_Value` (ensemble des valeurs de propriétés de qualité) via la propriété `qualProp_values`.

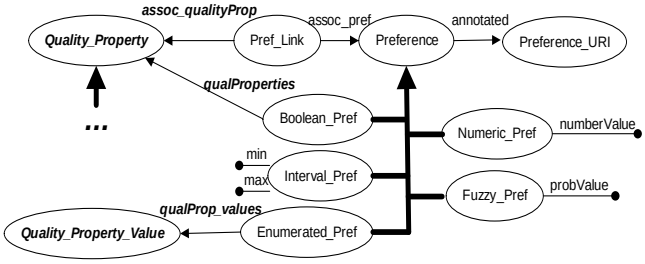

Figure 5. *Extension du modèle de préférence*

La figure 6 illustre l'extension du modèle de préférence que nous proposons. Elle présente l'instanciation de cette extension pour définir deux préférences sur des propriétés de qualité. La première préférence définie est nommée `Fast`. C'est une préférence de type intervalle (`Interval_Pref`) liée par le `Pref_Link` `L1` à la propriété de qualité `avgTimeByTransaction`. Elle indique que l'on considère comme `Fast`, les services dont le temps moyen par transaction est compris entre 10 et 20 millisecondes. La seconde préférence nommée `Secured` porte cette fois-ci sur les services via le lien `L2`. C'est une préférence booléenne qui indique que l'on considère comme `Secured` les services satisfaisant les propriétés de qualité `authorization` et `dataEncryption`.

Les préférences sur les propriétés de qualité pourraient être utilisées dans les requêtes en utilisant la clause `PREFERRING` définie dans (Tapucu *et al.*, 2009). Cependant, notre but étant de proposer une approche modulaire pour gérer les besoins non fonctionnels, les préférences portant sur des propriétés de qualité doivent être clairement distinguées des préférences usuelles. En conséquence, nous avons modifié la syntaxe de la clause `PREFERRING` de la manière suivante :

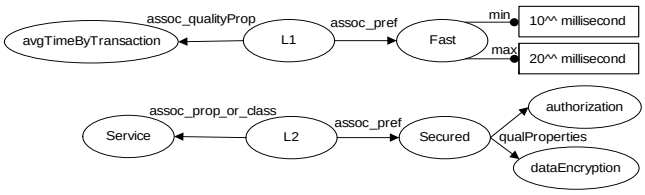

Figure 6. *Exemple de préférences sur des propriétés de qualité*

```
preferring-clause ::= PREFERRING '{' boolean-expr '}'
boolean-expr ::= boolean-term | boolean-expr OR boolean-term
boolean-term ::= boolean-factor | boolean-term AND boolean-factor
boolean-factor ::= [ NOT ] preference-identifier
                 | [ NOT ] QUALITY preference-quality-identifier
```

Explication. La syntaxe de la clause `PREFERRING` permet d'exprimer des expressions booléennes (avec les opérateurs `AND` et `OR`) sur des identifiants de préférences (`preference-identifier`). Nous avons ajouté (partie en italique) la possibilité d'utiliser des identifiants de préférences portant sur des propriétés de qualité (`preference-quality-identifier`). Ces identifiants sont introduits après le mot clé `QUALITY` pour bien les distinguer des préférences portant sur une propriété classique. De plus, ceci permet de faciliter l'interprétation de la requête.

Prenons un exemple pour illustrer cette syntaxe et illustrer l'intérêt d'utiliser des préférences. Considérons le tableau suivant décrivant deux services (`ServicePaiement` et `ServiceRemboursement`) sous la forme de triplets. Pour simplifier cet exemple, nous avons seulement indiqué les principaux triplets nécessaires.

ServicePaiement	rdf:type	Service
ServicePaiement	hasInput	CreditCard
ServiceRemboursement	rdf:type	Service
ServiceRemboursement	hasInput	CreditCard
ServiceRemboursement	serviceParameter	Authorization
Authorization	value	True
ServiceRemboursement	serviceParameter	DataEncryption
DataEncryption	value	True
Service	serviceParameter	avgTimeByTransaction
avgTimeByTransaction	value	15 ^^ milliseconds

Considérons à présent la requête suivante qui permet de rechercher les services prenant en entrée une carte de crédit sachant que l'on préfère les services `Secured` et `Fast` (représentés sur la figure 6)

```
SELECT ?service
```

```
WHERE {
 ?service rdf:type Service . ?service presents ?profile
 ?profile hasInput <http://cordio.bs/CreditCard>
PREFERRING { QUALITY Secured AND QUALITY Fast }
```

Explication. Notre extension permet maintenant d'exprimer les besoins non fonctionnels comme des préférences utilisateur dans la clause `PREFERRING`. Les préférences `Secured` et `Fast` portant sur des propriétés de qualité, elles sont précédées du mot clé `QUALITY`. Sans cette clause, le résultat de cette requête retournerait les deux services `ServicePaiement` et `ServiceRemboursement` sans ordre particulier. Ayant spécifié nos préférences, cette requête retournera le service `ServiceRemboursement` avant `ServicePaiement` puisque sa qualité correspond aux préférences spécifiées.

L'interprétation des préférences portant sur des propriétés de qualité est similaire à celle portant sur des propriétés classiques (détaillée dans (Tapucu *et al.*, 2009)). Par exemple, la préférence `Fast` est interprétée par le prédicat `avgTimeByTransaction >= 10 ^^ millisecond AND avgTimeByTransaction <= 20^^millisecond`.
De plus, nous avons détaillé dans la section 5.2 comment une propriété de qualité pouvait être utilisée comme une propriété classique. Ainsi, tous les éléments permettant d'interpréter des préférences portant sur des propriétés de qualité ont été définis.

6. Mise en oeuvre et évaluation

Dans les deux sections précédentes, nous avons présenté l'extension de OWL-S que nous proposons et son utilisation pour faciliter la recherche de SW. Dans cette section, nous présentons l'implémentation de cette approche que nous avons réalisée pour la valider 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) (Pierra *et al.*, 2005) 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 de l'implémenter sur la BDBO OntoDB (Dehainsala *et al.*, 2007) qui, de plus, dispose d'une implémentation du modèle de préférence sur lequel nous nous basons. 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[5] 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[6] : `Service.owl`, `Profile.owl`, `Process.owl` et `Grounding.owl`. Notre extension portant sur la partie Profile de OWL-S, nous avons utilisé 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 traduction repose sur l'introduction d'une jointure avec l'élément `Quality_Property` ajouté dans notre extension. Ce mécanisme est illustré sur la figure 7.

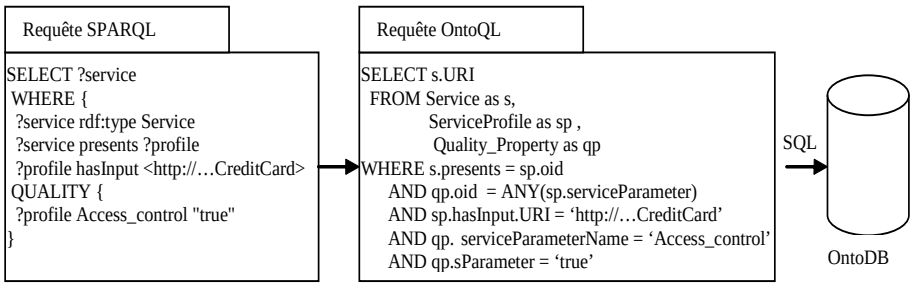

Figure 7. *Interprétation de la clause* `QUALITY` *de SPARQL par traduction en OntoQL*

Explication. La requête SPARQL initiale est présentée sur la 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` nécessite d'introduire une nouvelle jointure dans la requête OntoQL résultante. Cette jointure concerne la classe `Quality_Property` et est réalisée en introduisant de nouveaux prédicats dans la clause `WHERE`. Dans cet exemple, cette jointure permet de rechercher la propriété de qualité portant le nom `Access_control` et ayant la valeur `true`. Cette interprétation code ainsi les règles

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

de réécriture que nous avons définies dans la section 5.2. Elle est codée en JAVA dans l'interpréteur de requêtes SPARQL de la BDBO OntoDB. Une fois la requête traduite en OntoQL, elle sera alors elle-même traduite en SQL pour être exécutée sur OntoDB qui repose sur le serveur de gestion de base de données PostgreSQL.

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). Ainsi les règles permettant d'exploiter les relations entre propriétés de qualité spécifiées dans la section 5.3 ne peuvent pas être implémentées directement sur OntoDB. En conséquence, nous avons utilisé le mécanisme de déclencheurs (trigger) des bases de données pour les implémenter. Par exemple, la règle portant sur la relation `equivalent_to` est implantée par un déclencheur sur la table stockant les valeurs des propriétés de qualité. Le pseudo-code de ce déclencheur 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
```

OntoDB étant implémenté sur le serveur de gestion de bases de données PostgreSQL, le langage PL/pgSQL a été utilisé pour coder ce déclencheur. Les autres règles sont implantées de la même façon.

Extension du modèle de préférence

Le modèle de préférence présenté à la section 5.4.1 a été implémenté sur OntoDB en utilisant le langage OntoQL. Plus précisément, un script OntoQL permet de générer l'ensemble des éléments de ce modèle dans OntoDB. Nous avons donc modifié ce script pour prendre en compte l'extension que nous avons définie dans la section 5.4.2. Par exemple, l'instruction OntoQL suivante permet de modifier les préférences booléennes `Boolean_Pref` pour qu'elles puissent s'appliquer à des propriétés de qualité :

```
ALTER ENTITY #Boolean_Pref
  ADD qualProperties REF(Quality_Property) ARRAY
```

Un script OntoQL permet ainsi d'étendre le modèle de préférence initial de OntoDB. Cependant, il est également nécessaire de modifier l'interprétation des requêtes portant sur l'extension de ce modèle. Aussi, nous avons à nouveau modifié le moteur SPARQL de OntoDB pour qu'il interprète des préférences portant sur des propriétés de qualité. Cette interprétation étant proche de celles portant sur des propriétés classiques, nous avons pu nous appuyer sur le code JAVA existant. Pour tester l'implémentation réalisée nous sommes appuyés sur des tests similaires à l'exemple présenté section 5.4.2 où les entrées de la requête et le résultat attendu sont spécifiés.

6.2. *Avantages et limitations de notre approche*

Comparée aux autres ontologies proposées pour la QoS (détaillées dans la section 2), l'originalité de notre proposition est de permettre la représentation des standards de qualité et des liens que l'on peut définir entre ces standards dans OWL-S. Notre approche utilise cette particularité pour faciliter la recherche de SW en proposant une extension du langage SPARQL qui sépare clairement l'expression des besoins fonctionnels des besoins non fonctionnels. Elle permet, de plus, d'exploiter les relations que l'on peut établir entre critères de qualité grâce à des règles de déduction[7]. Enfin, les besoins non fonctionnels peuvent être exprimés comme des contraintes lâches (préférences utilisateurs) dans les requêtes. Ainsi les utilisateurs peuvent rechercher les services répondant à leur besoin en les triant selon des critères 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 (Kritikos *et al.*, 2007) 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 (Wang *et al.*, 2006) 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 dans 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. Cette extension consiste en un ensemble de concepts permettant 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. Comparée aux autres ontologies proposées pour la QoS, la prise en compte des standards de qualité est la principale originalité de notre approche.

Pour aller plus loin, nous avons ensuite étudié l'intérêt de l'extension de OWL-S proposée 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 consé-

7. Notons tout de même qu'établir des équivalences entre propriétés des différents standards nécessite une importante expertise sur les standards de qualité. L'ontologie OSQS contribue à organiser cette connaissance.

quence, 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. Enfin, les besoins non fonctionnels n'étant généralement pas cruciaux pour un utilisateur mais servant plutôt à ordonner les services répondant aux besoins fonctionnels, nous avons développé une approche permettant d'exprimer ces besoins comme des préférences utilisateurs. Cette approche s'appuie sur un modèle de préférence qui permet d'exprimer des préférences variées. Ces préférences sur la qualité de SW peuvent ensuite être exploitées dans les requêtes et sont clairement distinguées des préférences usuelles.

Ce travail ouvre plusieurs perspectives. Nous souhaitons d'abord compléter notre approche, d'une part, en intégrant les capacités des ontologies de QoS existantes et, d'autre part, en développant des outils permettant aux différents acteurs impliqués dans l'utilisation de SW de bénéficier de notre approche. Ensuite, nous souhaitons poursuivre la formalisation de notre approche pour, par exemple, montrer la complétude des règles d'inférences que nous utilisons pour établir des relations entre propriétés de qualité. Enfin, nous envisageons de développer divers algorithmes basés sur notre extension de OWL-S. Par exemple, nous souhaitons développer un algorithme permettant de trouver un SW qui puisse en remplacer un autre en cas de panne, tout en évaluant la différence de qualité engendrée.

Remerciements. Ces travaux ont été partiellement supportés par la Faculté de Sciences et le Conseil Scientifique de l'Université Centrale du Venezuela, le projet CDCH PG ADIRE, le projet UCV No 2009000624 de Fonacit et enfin le projet O2M.

8. Bibliographie

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

Bleul S., Weise T., « An Ontology for Quality-Aware Service Discovery », *Proceedings of the First International Workshop on Engineering Service Compositions (WESC'05)*, p. 35-42, 2005.

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

Dehainsala H., Pierra G., Bellatreche L., « OntoDB : An Ontology-Based Database for Data Intensive Applications », *Proceedings of the 12th International Conference on Database Systems for Advanced Applications (DASFAA'07)*, vol. 4443 of *Lecture Notes in Computer Science*, Springer, p. 497-508, 2007.

Dobson G., Lock R., Sommerville I., « Quality of Service Requirements Specification Using an Ontology », *In Proceedings of the SOCCER Workshop, Requirements Engineering*, 2005.

Giallonardo E., Zimeo E., « More Semantics in QoS Matching », *Proceedings of the IEEE International Conference on Service-Oriented Computing and Applications (SOCA 2007)*, IEEE Computer Society, Washington, DC, USA, p. 163-171, 2007.

Gurský P., T.Horváth, Jirásek J., Vojtás P., « User Preference Web Search – Experiments with a system connecting web and user », *To appear in the Computing and Informatics Journal*, p. 25-32, 2008.

IEEE-Std-1061-1998, IEEE Standards for a Software Quality Methodology, Technical report, IEEE Computer Society, 1998.

ISO/IEC JTC1/SC7 N4522, FCD 25010, Software engineering-Software product Quality Requirements and Evaluation (SQuaRE) Quality model, Technical report, FCD Ballot, 2010.

ISO/IEC13236, Quality of Service : Framework, Technical report, International Organisation for Standardisation / International Electrotechnical Commission, 1999.

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

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

Jeong B., Cho H., Lee C., « On the functional quality of service (FQoS) to discover and compose interoperable web services », *Expert Syst. Appl.*, vol. 36, p. 5411-5418, April, 2009.

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

Krutchen P., « An Ontology of Architectural Design Decisions in Software Intensive Systems », *Proceedings of the 2nd Groningen Workshop on Software Variability Management*, p. 54-61, October, 2004.

Losavio F., Matteo A., Levy N., « Web Services Domain Knowledge with an Ontology on Software Quality Standards », *Proceedings of the Third International Conferences on Internet Technologies and Applications (ITA 2009)*, p. 74-85, 2009.

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

Pierra G., « Context Representation in Domain Ontologies and its Use for Semantic Integration of Data », *Journal Of Data Semantics (JODS)*, vol. X, p. 34-43, 2007.

Pierra G., Dehainsala H., Aït-Ameur Y., Bellatreche L., « Base de Données à Base Ontologique : principes et mise en œuvre », *Ingénierie des Systèmes d'Information*, vol. 10, n° 2, p. 91-115, 2005.

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

Siberski W., Pan J. Z., Thaden U., « Querying the semantic web with preferences », *In Proceedings of the 5th International Semantic Web Conference (ISWC 2006)*, p. 612-624, 2006.

Tapucu D., Jean S., Aït-Ameur Y., Ünalir M. O., « An Extention of Ontology Based Databases to Handle Preferences », *Proceedings of 11th International Conference on Enterprise Information Systems (ICEIS 2009)*, p. 208-214, 2009.

Tondello G. F., Siqueira F., « The QoS-MO ontology for semantic QoS modeling », *Proceedings of the 2008 ACM symposium on Applied computing (SAC 2008)*, ACM, New York, NY, USA, p. 2336-2340, 2008.

Toninelli A., Corradi A., Montanari R., « Semantic-based discovery to support mobile context-aware service access. », *Computer Communications*, vol. 31, n° 5, p. 935-949, 2008.

Tran V. X., Tsuji H., « 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)*, vol. 0, p. 379-385, 2009.

Wang X., Vitvar T., Kerrigan M., Toma I., « A QoS-Aware Selection Model for Semantic Web Services », *Proceedings of the 4th International Conference on Service-Oriented Computing (ICSOC 2006)*, p. 390-401, 2006.

Zhou C., Chia L.-T., Lee B.-S., « DAML-QoS Ontology for Web Services », *Proceedings of the IEEE International Conference on Web Services (ICWS 2004)*, IEEE Computer Society, Washington, DC, USA, p. 472, 2004.

Stéphane Jean *est Maître de Conférences à l'Université de Poitiers. Il est membre du Laboratoire d'Informatique Scientifique et Industrielle (LISI) situé à l'École Nationale Supérieure de Mécanique et d'Aérotechnique (ENSMA). Ses travaux portent sur la modélisation à base ontologique ainsi que sur la persistance et la manipulation de modèles et d'ontologies au sein de bases de données.*

Francisca Losavio *est Professeur à l'École d'Informatique de la Faculté des Sciences de l'Université Centrale du Venezuela. Elle est responsable du laboratoire Modelos, Software y Tecnologías (MoST) et de la filière Génie Logiciel du Comité Académico de Postgrado en Science de l'Informatique. Elle a été membre fondateur du premier centre de recherche en génie logiciel du Venezuela, le centre de Ingeniería de Software Y Sistemas (ISYS) en 1995 et a participé à plusieurs projets de recherche nationaux et internationaux. Ses axes de recherche sont l'architecture logicielle, la qualité logicielle et les modèles et méthodologies de développement logiciels.*

Alfredo Matteo *est Professeur à l'École d'Informatique de la Faculté des Science de l'Université Centrale du Venezuela. Il est directeur du Comité Académico de Postgrado en Science de l'Informatique et membre du laboratoire Modelos, Software y Tecnologías (MoST). Il a été membre fondateur du premier centre de recherche en génie logiciel du Venezuela, le centre de Ingeniería de Software Y Sistemas (ISYS) en 1995 et a participé à plusieurs projets de recherche nationaux. Ses axes de recherche sont l'ingénierie dirigée par les modèles, l'ingénierie du domaine et les modèles et standards pour le développement logiciels.*

Nicole Levy *est professeur en Informatique au CNAM, Paris et membre du laboratoire Cedric depuis Septembre 2010. Ses travaux portent sur l'utilisation de méthodes formelles pour spécifier des systèmes complexes et des architectures logicielles prenant en compte aussi bien les propriétés fonctionnelles que non fonctionnelles recherchées.*