Modélisation et alignement sémantique des intentions des clients avec les offres des fournisseurs

Kaouthar FAKHFAKH[*,**,***], Tarak CHAARI[***]
Said TAZI[*,**], Mohamed Jmaiel[***], Ikbel GUIDARA[***]

* CNRS ; LAAS ; 7 avenue du colonel Roche, F-31077 Toulouse, France
** Université de Toulouse ; UT1, UPS, INSA, INP, ISAE ; LAAS ; F-31077 Toulouse, France
{kchaari, tazi}@laas.fr
*** Ecole Nationale d'Ingénieur de Sfax -ReDCAD
Route de la Soukra, B.P. W, 3038 Sfax, Tunisia
tarak.chaari@redcad.org, mohamed.jmaiel@enis.rnu.tn
ikbel.guidara@gmail.com

Résumé. La mise en place de contrats de qualités de service entre les clients et les fournisseurs reste une tâche assez complexe. En effet, ces deux parties n'ont pas le même degré de connaissances et peuvent ne pas partager le même langage. Dans le but de palier à ce problème, nous avons commencé par définir des modèles sémantiques basés sur les ontologies pour exprimer les intentions des clients et les offres des fournisseurs. Ensuite, nous avons élaboré une approche sémantique qui se base sur des techniques d'inférence pour (1) détecter les correspondances linguistiques et sémantiques entre ces modèles, (2) raffiner les correspondances générées et (3) évaluer ces correspondances pour vérifier si le fournisseur pourrait répondre positivement aux attentes du client. Pour valider notre approche, nous avons implémenté un prototype qui permet de faciliter et d'automatiser la tâche de traitement des intentions des clients pour les offres des fournisseurs. Ce prototype peut être facilement étendu pour aider les clients à choisir les fournisseurs les plus proches de leurs intentions et leurs exigences.

1 Introduction

Dans les paradigmes de l'informatique orientée services, un SLA (Service Level Agreements) est un contrat de qualité de service (QdS) définissant les engagements de l'hébergeur quant à la qualité de sa prestation et les pénalités encourues en cas de manquement. Les contrats de QdS sont devenus de plus en plus complexes et détaillés avec des termes et des conditions techniques. Ces paramètres complexes rendent leur compréhension difficile par les clients. Un contrat de QdS est caractérisé par un cycle de vie (Sun et al., 2005) commençant par la préparation initiale du modèle du contrat jusqu'à arriver à la phase de surveillance de ses clauses en passant par la phase de négociation entre le client et le fournisseur. Dans cet article, nous nous intéressons à cette phase de négociation du contrat. Plus précisément, nous nous focalisons sur la première étape de cette phase qui traite la réception de la demande du client chez le fournisseur. Le traitement de ces demandes reste une tâche assez fastidieuse qui nécessite un temps d'analyse conséquent pour vérifier si le produit du fournisseur correspond aux besoins du client. Dans la procédure de négociation

habituelle, cette première étape consiste généralement à sélectionner un sous-ensemble de clauses et de valeurs parmi des choix prédéfinis par le fournisseur. Cependant, le client peut ne pas comprendre ces choix surtout s'il n'est pas un expert du domaine. Par exemple, une bibliothèque numérique de vidéos propose un service de téléchargement de ses vidéos selon différents débits de transfert. Un client qui n'est pas expert du domaine peut exprimer ses besoins en termes de " temps de téléchargement " puisqu'il ne comprend pas les aspects techniques et les significations du terme " débit ".

L'objectif de notre travail est de: (1) proposer une approche qui donne une liberté d'expression au client pour formuler ses intentions avec son propre langage et ses propres connaissances, (2) automatiser l'analyse de ces intentions afin de vérifier si elles correspondent aux offres du fournisseur et (3) générer automatiquement un contrat de qualité de service en cas de compatibilité. Dans cet article, nous nous focalisons sur les deux premiers objectifs en proposant : (1) des modèles sémantiques pour capturer les intentions des clients et les offres du fournisseur et (2) une approche automatique d'analyse sémantique de la correspondance de ces intentions et de ces offres. Cette approche se compose de trois étapes. La première étape consiste à chercher toutes les correspondances possibles entre les termes de l'intention du client et ceux des offres du fournisseur. La deuxième étape permet de raffiner les correspondances résultantes de la première étape. Enfin, la troisième étape consiste à évaluer ces correspondances pour décider si le fournisseur pourrait répondre à la demande du client.

Dans la section 2 de cet article, nous commençons par présenter quelques travaux qui s'intéressent aux contrats de qualités de service. Dans la section 3, nous présentons les modèles sémantiques que nous avons élaborés afin de faciliter l'expression des intentions du client et des offres du fournisseur. Dans la section 4, nous détaillons notre approche d'alignement sémantique entre ces intentions et ces offres. Avant de conclure, nous validons notre approche par un cas d'étude d'une bibliothèque numérique de vidéos.

2 Etat de l'art

La négociation des contrats de QdS a ciblé l'intérêt de plusieurs domaines de recherches menées par de multiples projets. On peut citer quelques protocoles de négociation, par exemple, WS-agreement (Andrieux et al, 2007), NextGrid (Dimitrakos,2007) et ICNIP (Smith, 1981). WS-Agreement, est un protocole basé sur le principe "prend le ou laisse le", c'est à dire "approuve ou pas de contrats" achevé par une phase de signature de contrat. Il se base sur une structure XML pour la représentation du contrat. Concernant NextGRID, c'est un protocole de négociation spécifique aux applications sur les grilles de calcul avec des modèles prédéfinis pour les grilles. A propos du protocole ICNIP, il est un peu avancé par rapport aux autres protocoles de négociations puisqu'il y a des échanges récursifs pour le choix des clauses du contrat. Cependant, les clauses sont prédéfinies par le fournisseur.

Tous les protocoles mentionnés auparavant s'intéressent à l'élaboration d'architectures et de protocoles avancés pour la négociation de contrats de qualité de service. Ils s'intéressent aux actions de négociation comme *send*, *recieve* et *accept*. Cependant, ils ne détaillent pas la sémantique des messages échangés. Pour automatiser cette phase de négociation, ces messages doivent être compréhensibles par la machine. Le moyen le plus approprié pour définir la sémantique de ces messages est l'utilisation des ontologies (Studer et al., 1998). En effet, les ontologies sont des formalismes qui fournissent une description formelle des

concepts, des propriétés et des relations entre les concepts modélisant la sémantique d'un domaine ou d'un système. Ainsi, il est intéressant d'utiliser des ontologies pour capturer la sémantique des besoins du client et celle des offres du fournisseur. En conséquence, cette négociation se transforme en un alignement de ces ontologies. Le processus d'alignement des ontologies consiste à chercher les similarités entre deux ontologies.

Dans la littérature, plusieurs méthodes d'alignement d'ontologies ont été proposées (Michael et al., 2004), (Chen et al.,2004). Ces méthodes exploitent des ontologies décrites dans différents langages comme RDF et DAML+OIL. Les approches ANCHOR-PROMPT (Noy, 2004) et QOM (Ehrig et al., 2004) exploitent des ontologies décrites avec le langage RDF. Etant donné que le langage OWL est un standard pour les ontologies, toute méthode d'alignement n'exploitant pas ce format présente un manque d'expression par rapport aux autres langages sémantiques. La méthode EDOLA (Zghal et al., 2007) aligne des ontologies représentées avec le langage OWL. Malgré ses fonctionnalités, OWL n'est pas un langage opérationnel en soi. En effet, il n'offre pas de mécanismes permettant d'inférer de nouvelles connaissances et actions à partir de celles exprimées par le langage lui même. Pour remédier à cette insuffisance, nous avons eu recours à utiliser le langage SWRL (Horrocks et al., 2004) qui permet de rédiger des règles exprimées en termes de concepts OWL pour fournir des capacités de raisonnement déductif sur les ontologies. L'utilisation du langage SWRL permet d'automatiser l'alignement sémantique des ontologies. Par ailleurs, la plupart des approches existantes d'alignement d'ontologies se basent sur des relations hiérarchiques (is-a) uniquement et n'exploitent pas convenablement les autres relations sémantiques exprimées entre les concepts des ontologies.

Dans les sections suivantes, nous présentons notre approche d'alignement automatique des intentions des clients avec les offres des fournisseurs en se basant sur des modèles exprimés sous la forme d'ontologies OWL et sur des inférences sémantiques avec le langage SWRL.

3 Modélisation des intentions du client et des offres du fournisseur

Dans cette section, nous présentons les structures simplifiées des ontologies *ClientOnto* et *ProviderOnto* pour modéliser respectivement les intentions du client et les offres du fournisseur.

3.1 ClientOnto : Modèle de haut niveau des intentions du client

Une intention correspond à un état d'esprit de tout acteur qui effectue une action. Pour utiliser la notion d'intention, il est indispensable d'utiliser un langage formel. Nous avons adapté le modèle intentionnel de (Kanso et al. 2007) qui présente une intention sous la forme I (A, G*, M*, R*) où : 'I' présente l'intention effectuée par une action 'A'. 'A' présente l'action que les auteurs de l'intention veulent l'effectuer. 'G' exprime le but à réaliser par l'action. 'M' présente les moyens utilisés pour accomplir l'action. 'R' explique pour quelles raisons l'auteur a choisi cette action et * indique que le nombre des actes composant l'intention peut être multiple (0 à N). Nous avons enrichi ce modèle par l'ajout d'éléments spécifiques relatifs à la négociation d'accord de qualité de service. La figure 1 présente une structure simplifiée de *ClientOnto*. Le concept racine de cette ontologie est le concept *Client*

qui permet d'identifier le client. Ce dernier veut effectuer une action et obtenir le résultat attendu. Une action est définie par un sujet pour décrire les produits cherchés par le client en fonction des contraintes qui permettent d'exprimer les exigences et les conditions. Chaque contrainte est définie par une propriété, un seuil et un opérateur pour pouvoir comparer la valeur demandée par le client avec les valeurs de l'offre du fournisseur. Par exemple, le client peut formuler une contrainte sur le temps de téléchargement des films qui l'intéressent (*Download Time* < 10mn). Dans ce cas, la propriété est le temps de téléchargement, le seuil est la valeur maximale fixée par le client (10mn) et l'opérateur est «inférieur».

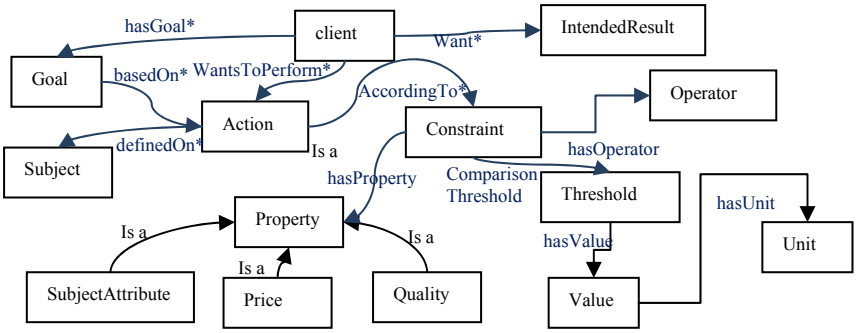

FIG. 1 - *Structure simplifiée de l'ontologie du client.*

3.2 ProviderOnto : Modèle métier du fournisseur

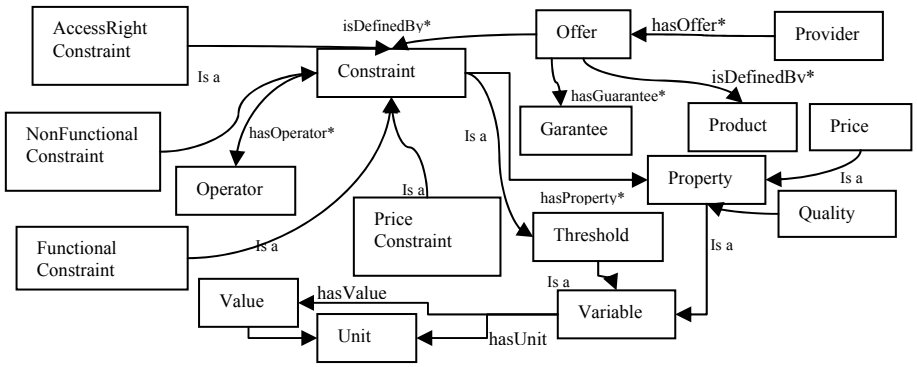

Fig. 2 – Structure simplifiée de l'ontologie du fournisseur

Dans la littérature, nous avons remarqué qu'il existe un langage de description des offres des fournisseurs nommé WSOL (Web Services Offerings Language) (Tosic et al. 2004). WSOL est un langage basé sur XML qui permet de spécifier les niveaux de qualité de service en se basant sur les descriptions du service WSDL. WSOL est adapté pour la définition des dimensions de la qualité, ses métriques et ses contraintes. Toutefois, l'extraction de l'information à partir de ce modèle est difficile en raison de sa complexité. En WSOL, les contraintes sont exprimées sous la forme de chaînes de caractères. Ceci n'aide pas à la correspondance des offres du fournisseur avec les exigences du client. La figure 2 présente une structure simplifiée de *ProviderOnto*. Chaque fournisseur propose des offres

qui sont définis par des contraintes, des garanties et des produits. Chaque contrainte est définie par une propriété qui peut être une qualité, une caractéristique du produit ou un prix, un opérateur et un seuil.

Les propriétés et les seuils sont définis comme des variables. Chaque variable admet un nom, un type, une unité de mesure et une valeur. Les contraintes définies par le fournisseur peuvent être fonctionnelles, non fonctionnelles, des contraintes de prix ou des contraintes de droit d'accès. Nous présentons un exemple détaillé d'une offre d'un fournisseur dans la section 5.

4 Alignement des intentions du client avec les offres du fournisseur

Dans cette section, nous présentons notre approche d'alignement sémantique entre les ontologies *ClientOnto* et *ProviderOnto* que nous avons présenté dans la section précédente. Notre approche se compose de trois étapes principales : la génération, le raffinement et l'évaluation des correspondances entre les deux ontologies. Dans ce qui suit, nous donnons plus de détails sur ces étapes.

4.1 Génération des correspondances

Pour pouvoir étudier l'alignement entre les besoins du client et les produits du fournisseur, il est indispensable d'étudier le degré de similarité entre leurs intentions et leurs offres. L'étape de génération des correspondances consiste à découvrir automatiquement des correspondances sémantiques entre les termes des deux ontologies (correspondants aux intentions du client et aux offres du fournisseur). Notre méthode de recherche de correspondances vise à donner une valeur quantitative à la similarité entre les concepts. Il est nécessaire alors de disposer d'une mesure de similarité capable d'évaluer les ressemblances qui existent au sein des données des deux ontologies afin de réduire l'intervention humaine et d'assurer une bonne précision. Pour ce faire, nous avons utilisé WordNet-Similarity (Pedersen et al., 2004) qui est un outil permettant d'évaluer le degré de similarité entre deux termes en exploitant les relations hiérarchiques de l'ontologie WordNet. On peut distinguer deux types de mesures de WordNet-Similarity : les mesures de proximité sémantiques et les mesures de proximité linguistique. Les mesures de proximité sémantique donnent une idée sur le degré d'équivalence des termes alors que les mesures de proximité linguistique donnent une idée sur la distance des termes dans les corpus linguistiques. Nous avons utilisé ces mesures de similarité pour élaborer automatiquement des correspondances entre les termes du client et ceux du fournisseur. Nous distinguons deux types de correspondances : correspondances sémantiques et correspondances mathématiques.

4.1.1 Correspondance sémantique entre l'ontologie du client et l'ontologie du fournisseur

Cette étape consiste à générer des correspondances entre les termes du client et ceux du fournisseur en se basant sur une mesure de proximité sémantique de WordNet ::Similarity. Ce module fournit trois méthodes de mesures de proximité sémantique nommées: hso (Hso et al), Lesk (Lesk et al. 2003), et Gloss vector (Patwardhan et al. 2006). Les auteurs de

(Patwardhan et al. 2006) ont prouvé expérimentalement que Gloss vector est la meilleure méthode dans cette catégorie de mesure puisqu'elle fournit la meilleure précision à l'aide des sens des mots.

Listing 1. Algorithme de correspondances entre les termes du client et les termes du fournisseur

```
generateCorrespondences(rootTerm1, rootTerm2)
{
For each individual1 instanceOf rootTerm1Do
For each individual2 instanceOf rootTerm2Do
   semanticSimilarity := wordnet::similarity(individual1, individual2);
   createCorrespondence(individual1, individual2, semanticSimilarity);
}
```

L'algorithme présenté dans le Listing 1 présente les étapes que nous avons définies pour la génération des correspondances sémantiques entre les termes du client et du fournisseur. Le principe de notre algorithme consiste à chercher toutes les correspondances possibles entre les termes des deux ontologies en utilisant la méthode Gloss Vector. Pour chaque couple de termes (*individual1*, *individual2*), une nouvelle correspondance est générée ayant comme un terme source *individual1*de *ClientOnto*, comme terme destination *individual2* de *ProviderOnto* et une valeur de degré de similarité *semanticSimilarity* donnée par la méthode Gloss Vector.

4.1.2 Correspondance mathématique en utilisant une ontologie de qualité de service

L'étape précédente (présentée dans la section 4.1.1) permet de détecter les correspondances directes entre les termes du client et ceux du fournisseur en se basant sur les relations sémantiques qui existent entre ces termes. Dans de nombreux cas, surtout lorsque nous nous intéressons à la qualité de service, il peut y avoir des relations mathématiques entre les termes. Par exemple, si le client définit une contrainte sur le temps de téléchargement d'un film, celui-ci peut être calculé mathématiquement à partir de la taille du film et du débit du transfert offert par le fournisseur. Pour détecter ce genre de correspondances, nous avons besoin d'un modèle qui permet d'exprimer les relations mathématiques qui existent entre les QdS. Dans la littérature, nous trouvons des ontologies existantes de QdS telles que OWL-QoS (Chia, 2005), QoSOnt (Dobson et al., 2005), SL-ontologie (Bleul et al., 2006). Cependant, elles n'expriment pas les dépendances mathématiques entre les métriques de QdS quand elles sont composées. En conséquence, nous ne pouvons pas calculer les valeurs de ces métriques composées. Pour cela, nous avons développé une nouvelle ontologie de qualité de service nommée *QoSOnto* en s'inspirant de l'ontologie OWL-QoS. Cette ontologie modélise toutes les relations mathématiques de base entre les métriques de qualité de service. Elle sera utilisée au cours de l'algorithme de correspondance par l'expert du fournisseur. En effet, nous avons défini une interface java basée sur une fonction abstraite pour calculer la valeur d'une métrique composée. Nous avons aussi défini des rangs pour les opérandes de chaque fonction afin de spécifier l'ordre qui doit être respecté pour calculer les valeurs composées. Pour définir une nouvelle fonction de dépendance de qualité de service, il suffit d'implémenter l'interface java que nous avons élaborée. Pour détecter les correspondances mathématiques, nous avons développé un algorithme qui réutilise celui de Listing 1 pour rechercher toutes les correspondances entre les termes du client avec les termes de l'ontologie *QoSOnto*. Après cela, nous vérifions si les

termes identifiés de cette ontologie ont des correspondances avec les termes de l'ontologie *ProviderOnto*. Si oui, nous conservons ces correspondances dans la nouvelle ontologie d'alignement. La figure 3 montre un exemple d'une correspondance mathématique à l'aide de l'ontologie *QoSOnto*. Le terme *temps de téléchargement* a une correspondance avec le terme *temps de transfert* de l'ontologie de QdS. La valeur correspondante de ce terme dépend de la taille du fichier à télécharger et du débit de communication.

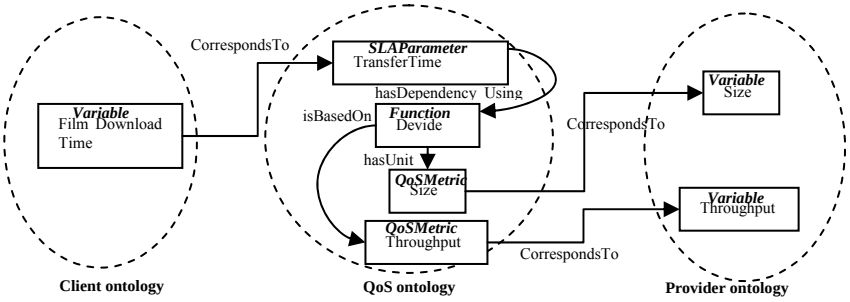

FIG. 3 - *Exemple d'une correspondance indirecte en utilisant l'ontologie QoSOnto*

4.2 Raffinement des correspondances

Dans cette étape, nous procédons au raffinement des correspondances générées dans l'étape précédente afin d'obtenir des correspondances plus correctes entre les termes du client et les termes du fournisseur. En effet, un même terme pourrait avoir plusieurs significations. Ainsi, certaines erreurs peuvent se produire dans les mesures de similarité entre ces termes. Pour corriger ces erreurs, nous avons défini un algorithme de raffinement dans notre approche d'alignement. Dans cet algorithme, nous commençons par générer les adjacences entre les correspondances créées en fonction de leur proximité linguistique et nous terminons par appliquer un processus de stabilisation pour corriger les erreurs isolées.

4.2.1 Génération des adjacences entre les correspondances

Dans cette étape, nous considérons que deux correspondances sont adjacentes s'il existe une relation entre leurs sources ou entre leurs destinations (c'est-à-dire une relation entre deux concepts de la première ontologie ou une relation entre deux concepts de la deuxième ontologie). Une adjacence est caractérisée par une correspondance source, une correspondance destination et une probabilité (flottant compris entre 0 et 1). Cette probabilité exprime le degré de proximité linguistique entre les deux termes sources ou les deux termes destinations qui sont en relation. L'algorithme de cette étape présenté dans le Listing 2 consiste à parcourir toutes les correspondances créées dans l'étape précédente et à créer une nouvelle adjacence entre elles lorsqu'elles ont des termes liés directement dans les ontologies du client, du fournisseur ou dans l'ontologie de QoS. Pour chaque adjacence créée, on calcule la valeur de la proximité linguistique entre les termes sources ou les termes destinations de ces correspondances. Nous nous intéressons alors dans cette étape aux distances des termes d'un point de vue hiérarchique dans les ontologies et pas à leurs sens tel qu'il est le cas dans l'étape de génération des correspondances. Pour cette raison, nous allons utiliser les mesures de proximité linguistiques de WordNet::Similarity. Après avoir comparé les mesures des proximités linguistiques offertes par WordNet::Similarity, nous avons choisi

la mesure Wup pour attribuer des pondérations aux adjacences puisqu'elle se base sur la mesure des distances des mots dans l'ontologie WordNet (Wup et al. 1994).

Listing 2. Algorithme de génération d'adjacences lingustiques entre les correspondances sémantiques

```
createCorrespondenceAdjacencies()
{
For each correspondence1 instanceOf Correspondence Do
 sourceTerm1:= getSourceTerm(correspondence1);
 destinationTerm1:= get DestinationTerm(correspondence1);
 For each correspondence2≠correspondence1 instanceOf Correspondence Do
   sourceTerm2:= getSourceTerm(correspondence2);
   destinationTerm2:= get DestinationTerm(correspondence2);
   linguisticRelatedness:=0;
   adjacencyNumber:=0;
   If ∃ R ∈ Rclient OR R ∈ Rqos /R=(sourceTerm1, sourceTerm2)
     linguisticRelatedness+= wordnet::relatedness(sourceTerm1, sourceTerm2);
     adjacencyNumber++;
   If ∃ R ∈ Rqos OR R ∈ Rprovider /R=( destinationTerm1, destinationTerm2)
     linguisticRelatedness+= wordnet::relatedness(destinationTerm1, destinationTerm2);
     adjacencyNumber++;
   If adjacencyNumber≠0 then
     linguisticRelatedness:= linguisticRelatedness/adjacencyNumber;
   createAdjacency(correspondence1, correspondence2, linguisticRelatedness);
}
```

4.2.2 Processus de stabilisation des correspondances

Pour raffiner les mesures de similarité calculées entre les termes du client et les termes du fournisseur, nous avons défini un algorithme de stabilisation basé sur les probabilités des correspondances et les poids des adjacences. Initialement, nous considérons que les correspondances qui ont des valeurs de probabilités égales à 1 sont stables et elles ne seront pas incluses dans l'étape de stabilisation. Chaque nœud (sauf celui qui est stable) sera influé par les nœuds qui lui sont adjacents en utilisant une formule de moyenne pondérée (1) pour calculer sa nouvelle probabilité *newCertainty*. Cette dernière est comparée à chaque fois avec l'ancienne valeur de probabilité *oldCertainty*. La correspondance est considérée comme stable lorsque ces deux valeurs de probabilités sont proches (par exemple la différence entre les deux valeurs est inférieurs à 0,001). Cette valeur est fixée par l'expert du fournisseur.

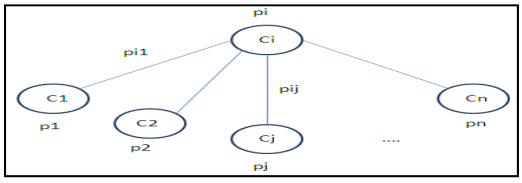

FIG. 4 - *Propagation des probabilités des correspondances*

Le principe de l'algorithme de stabilisation (voir le Listing 3) est : pour chaque correspondance C_i (voir figure 4) qui a une probabilité p_i, nous récupérons toutes les adjacences A_{ij} ayant comme probabilité p_{ij} dont la correspondance C_i est l'une de ses extrémités (source ou destination). Pour chaque adjacence trouvée, nous récupérons la probabilité p_j de sa deuxième extrémité qui est la correspondance C_j. La fonction de

moyenne pondérée (1) est ainsi appliquée pour calculer la nouvelle probabilité p_i' de la correspondance C_i où n est le nombre d'adjacences de C_i. Ce calcul sera répété tant qu'il existe des nœuds qui n'ont pas atténué l'état de stabilité.

$$pi' = \frac{pi+\sum_{j=1}^{n}(\frac{pij}{n}*pj)}{1+\sum_{j=1}^{n}(\frac{pij}{n})} \quad (1)$$

Listing 3. Algorithme de stabilisation des correspondances entre les termes du client et les termes du fournisseur

```
correspondenceStabilization()
{
For each correspondence1 instanceOf Correspondence / correspondence1.isMarked("unStabilized") Do
 oldCertainty:= correspondence1.hasCertainty;
 ExtraCertainties:=0; AdjacenciesRelatedness:=0;
 For each adjacency instanceOf Adjacency / adjacency.hasExtremity(correspondence1) Do
   correspondence2 := adjacency.hasExtremity /correspondence2≠correspondence1;
   AdjacenciesRelatedness +=adjacency.hasRelatedness;
ExtraCertainties +=adjacency.hasRelatedness*correspondence2.hasCertainty;
newCertainty:=(oldCertainty+ ExtraCertainties)/(1+ AdjacenciesRelatedness);
correspondence1.hasCertainty:= newCertainty;
If newCertainty-oldCertainty<CERTAINTY_PRECISION mark(correspondence1,"stabilized");
   correspondenceStabilization();
}
```

4.3 Evaluation des correspondances

La troisième étape de notre approche est l'évaluation des correspondances. Cette étape vise à vérifier la compatibilité entre les intentions du client et les offres du fournisseur en se basant sur les résultats des étapes précédentes. Cette étape est composée de deux types d'évaluation: évaluation structurelle et évaluation des contraintes. L'évaluation structurelle sert à vérifier si un nombre suffisant de termes définis dans l'intention du client ont des correspondances avec les termes des offres du fournisseur. Elle consiste à calculer la moyenne globale des correspondances. Ce calcul est une moyenne optimiste de mesure de similarités des correspondances attachées aux termes du client. Nous prenons le maximum des mesures de similarité des correspondances de chaque terme du client dans le cas où ce terme a plusieurs correspondances avec les termes du fournisseur. Ensuite, nous calculons la moyenne globale de ces mesures de similarités maximales. Si elle est inférieure à un seuil fixé par l'expert commercial du fournisseur, nous considérons qu'il n'y a pas de compatibilité entre l'intention du client et l'offre du fournisseur. Dans ce cas, il n'est pas nécessaire de faire une analyse plus approfondie pour tester l'évaluation des contraintes. La deuxième sous étape consiste à vérifier que toutes les contraintes imposées par le client sont satisfaites par les offres du fournisseur. Pour chaque contrainte du client, nous considérons sa propriété, la valeur de son seuil et son opérateur et nous essayons d'obtenir sa valeur correspondante (ou l'intervalle de valeurs) chez le fournisseur. Dans cette étape nous ne considérons que les correspondances qui ont une valeur de similarité assez élevée. S'il y a des contraintes du client qui n'ont pas une liaison directe ou indirecte avec l'ontologie du fournisseur, alors nous affirmons qu'il y a une incompatibilité entre les deux ontologies. Dans le cas contraire, nous enregistrons les valeurs du fournisseur appropriées à chaque propriété du client. L'étape suivante sera alors la comparaison de la valeur du seuil de chaque propriété avec toutes ses valeurs qui lui correspondent chez le fournisseur selon l'opérateur défini par le client. Pour

tester toutes les combinaisons de valeurs possibles, nous nous sommes basés sur une règle SWRL *checkConstraintsRule* (voir Listing 4) qui permet de vérifier toutes les contraintes imposées par le client. Ainsi, la règle *checkConstraintsRule* permet de tester toutes les combinaisons de valeurs des propriétés possibles en comparant à chaque fois la valeur du seuil du client et sa valeur candidate chez fournisseur. S'il existe au moins une combinaison de valeurs chez le fournisseur pour laquelle toutes les contraintes du client sont respectées, nous procédons à une génération automatique d'un contrat de qualité de service entre le client et le fournisseur. Cette génération ne sera pas traité dans cet article par manque de place.

Listing 4. Partie simplifiée de la Règle SWRL « CheckConstraintRule »

client:hasProperty(?client:downloadTimeConstraint, client:downloadTime) ∧
client:comparisonThreadshold(?client:downloadTimeConstraint, ?client:downloadTimeThreshold) ∧
client:hasValue(?client:downloadTimeThreshold, ?client:downloadTimeValue) ∧
client:hasOperator(?client:downloadTimeConstraint, ?client:downloadTimeOperator) ∧
client:downloadTimeProviderValues(?client:downloadTimeInstance) ∧
matchingTools:checkConstraint(?client:downloadTimeValue,?client:downloadTimeOperator,?client:
downloadTimeInstance)∧.... →
matchingTools:testCompatibility(client:downloadTime, client:filmType, client:ClientPrice)

5 Implémentation et expérimentation

Dans cette section, nous présentons la mise en œuvre de notre approche pour vérifier la correspondance entre les intentions du client avec les offres du fournisseur. Pour cela, nous considérons un fournisseur d'une bibliothèque numérique de vidéos. Le fournisseur offre deux services : un service de téléchargement de films et un service de recherche de films. Le fournisseur impose deux contraintes sur ses offres : une contrainte fonctionnelle sur les types de ses films qui peuvent être de type comédie ou d'action et une contrainte non fonctionnelle concernant le débit qui est fixé à 8 Mb/s. Les prix des films sont fixés par ce fournisseur à 2 Euros. Ces films peuvent avoir deux tailles différentes : 600 Mo ou 900 Mo. Nous supposons que le client (qui n'est pas un expert dans le domaine des technologies informatiques) veut *télécharger* (instance du concept *action* dans le modèle *ClientOnto*) des *films* (instance du concept *subject* dans l'ontologie C*lientOnto*) de type *comédie* avec un *temps* de *téléchargement inférieur* ou égal à *dix minutes* et un *prix* maximum *égal* à *trois euros* par film (instances du concept *constraint* dans l'ontologie C*lientOnto*). Pour se faire, nous avons créé une instance de l'ontologie *ClientOnto* et une instance de l'ontologie du fournisseur *ProviderOnto* en utilisant l'outil « protégé[1]».

Nous constatons que ce fournisseur peut satisfaire la contrainte du prix proposée par le client, qui est égal à deux euros. En ce qui concerne la contrainte du temps de téléchargement, notre approche utilise l'ontologie *QoSOnto* pour calculer le temps de téléchargement du client qui est égal à la division de la valeur de la taille du film transféré par la valeur du débit offert au client ce qui donne une valeur égale à 600 secondes (qui est le résultat de la division de 600 Mo par 8 Mb/s). Par conséquent, nous remarquons qu'il y a une compatibilité entre les deux ontologies étudiées. Nous avons développé un prototype en langage Java qui implémente notre approche d'alignement. Ce prototype permet de charger

[1] http://protege.stanford.edu/

deux fichiers OWL contenant les instances de *ClientOnto* et *ProviderOnto*. Nous pouvons appliquer avec ce prototype les trois étapes principales de notre approche d'alignement ainsi qu'une étape supplémentaire qui garantie la génération complète d'un contrat de qualité de service (SLA) en cas de compatibilité.

Term 1	Term 2	Correspondance initiale	Correspondance raffinée
downloadTime	TransfertTime	0,83	0,83
downloadTime	SearchFilm	0,40	0,33
FilmType	FilmType	1	1
Film	FilmSize	0,58	0,60
TransfertSize	FilmType	0,46	0,28

TAB. 1 – *Quelques valeurs de correspondances sémantiques calculées avant et après le processus de raffinement*

Le tableau 1 présente quelques valeurs de mesures de similarité entre les termes du client et du fournisseur. Nous remarquons une amélioration dans ces valeurs après l'étape de raffinement selon la perception humaine. Ceci nous permet de faire une évaluation plus précise des contraintes du client par rapport aux offres du fournisseur. En effet, pour chaque propriété d'une contrainte exprimée par le client nous récupérons ses valeurs correspondantes directement de l'ontologie du fournisseur ou bien indirectement à travers l'ontologie de QdS. Cette récupération se base sur les valeurs de proximité sémantique et linguistiques issues des étapes de génération et de raffinement des correspondances qui sont supérieurs à un seuil fixé par un expert commercial du fournisseur (0,6 pour notre cas d'étude).

6 Conclusion générale et perspectives

Les contrats de qualités de services sont devenus de plus en plus complexes avec l'augmentation importante du nombre de fournisseurs et avec l'utilisation croissante de la sous-traitance dans les architectures orientées services. Ces paramètres rendent la compréhension de ces contrats très difficile pour les clients qui n'ont pas les outils pour exprimer librement leurs exigences par leurs simples connaissances et leurs propres langages. Dans cet article, nous avons présenté une approche d'alignement automatique entre les intentions du client avec les offres du fournisseur pour réduire les écarts de connaissances entre eux. Dans cette approche, nous avons commencé par modéliser les intentions du client et les offres du fournisseur. Nous avons choisit les ontologies comme moyen de représentation de nos modèles dans le but de faciliter le développement d'une approche d'alignement automatique d'exigences de qualités de service. Notre approche est basée sur trois étapes principales. La première étape consiste à générer des correspondances entre les termes du client et les termes du fournisseur en leurs attribuant des valeurs de mesures de similarités selon leur équivalence sémantique. La deuxième étape raffine et corrige ces valeurs. Dans la troisième étape, nous évaluons les contraintes du client avec les offres du fournisseur dans l'ontologie d'alignement générée.

Notre approche exploite la puissance d'inférences sémantiques liées aux structures des ontologies. Ceci nous a permis d'obtenir un haut degré d'automatisation de cette première phase de négociation entre les intentions du client et les offres du fournisseur. Nous croyons

que notre approche est bénéfique, d'une part, au fournisseur puisqu'elle assure l'automatisation de l'analyse des exigences des clients. D'autre part, elle peut être bénéfique au client en l'aidant à sélectionner et à comparer des services qui lui conviennent pour une même intention et de générer automatiquement une version préliminaire d'un SLA. Dans ce cas, un outil compréhensif et complet qui aide les clients à exprimer leurs intentions est nécessaire. Actuellement, nous sommes entrain de travailler sur l'optimisation du temps d'exécution de la recherche des correspondances entre les termes du client et les termes du fournisseur. En effet, cette génération de correspondances demande beaucoup de ressources de calculs puisqu'elle évalue toutes les similarités sémantiques qui existent entre tous les termes du client et ceux du fournisseur avec une complexité actuelle de O (nombre de termes du client * nombre de termes du fournisseur). Nous pensons alors à déployer notre prototype sur une grille de calcul pour améliorer le temps d'exécution de notre approche. Nous comptons aussi donner au client plus de liberté d'expression en définissant ses contraintes requises et celles qui sont facultatives. De plus, nous envisageons à tester notre approche sur d'autres cas d'études dans d'autres domaines d'applications.

Références

Andrieux et al, 2007A. Andrieux et. al : Web Services Agreement Specification (WS-Agreement). Recommended Standard, Open Grid Forum,March 2007

Bleul et al. 2006 Steffen Bleul, Thomas Weise, Kurt Geihss.: An Ontology for Quality-Aware Service Discovery. Special Edition Editorial: Engineering Design and Composition of Service-Oriented Applications, Computer Systems Science & Engineering, Volume 5, Number 21 – 2006

Chen Zhou, Liang-Tien Chia, and Bu-Sung Lee. Daml-qos ontology for web services. In IEEE International Conference on Web Services, pages 472Ã 479, 2004.

Chia 2005 Chia, Bu-Sung Lee.: QoS Measurement Issues with DAML-QoS. IEEE InterChen Zhou, Likang-Tien national Conference on e-Business Engineering (ICEBE'05) pp. 395 403.

Dobson et al. 2005 G. Dobson, R. Lock, I. Sommerville. :QoSOnt: a QoS Ontology for Service-Centric System, EUROMICRO Conference on Software Engineering and Advanced Applications, Porto, Portugal, Aug. 2005

Dimitrakos, 2007 a T. Dimitrakos (Editor).:Design patterns for SOA and Grid. BEinGRID Meta-Deliverable AC1 (Integrating D1.1.2, D1.3.2, D1.4.2, D1.5.2, D1.6.2) v.1.0, July 2007

Ehrig et al., 2004 Ehrig M., Staab S., « QOM : Quick Ontology Mapping », Proceedings of The 3rd ISWC, GI Jahrestagung (1), Hiroshima, Japon, p. 356-361, November, 2004

Hso et al. 1998 Hso: Hirst, G., and St-Onge, D. 1998.: Lexical chains as representations of context for the detection and correction of malapropisms. In Fellbaum, C., ed., WordNet: An electronic lexical database. MIT Press. 305–332

Horrocks et al., 2004 I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean. SWRL: A semantic web rule language combining OWL and RuleML. W3C Member Submission, 21 May 2004

Kanso et al. 2007 Hassan Kanso, Chantal Soulé-Dupuy, Saïd Tazi.: Representing Author's Intentions of Scientific Documents. Dans : International Conference on Enterprise Information Systems (ICEIS 2007), Funchal, Protugal, 12/06/07-16/06/07, Vol. 3, INSTICC Press, p. 489-492, juin 2007

Lesk et al. 2003 Lesk: Banerjee, S., and Pedersen, T. 2003.: Extended gloss overlaps as a measure of semantic relatedness. In Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, 805–810

Michael E. Maximilien and Munindar P. Singh. Toward autonomic web services trust and selection. In ICSOC 04: Proceedings of the 2nd international conference on Service oriented computing, pages 221, New York, NY, USA, 2004. ACM Press

Noy, 2004 Noy N.: Semantic Integration: A Survey of Ontology-based Approaches, SIGMOD, 2004

Patwardhan et al. 2006 Patwardhan S, Pedersen T.: Using WordNet-based context vectors to estimate the semantic relatedness of concepts. In: Proceedings of the EACL 2006 workshop, making sense of sense: Bringing computational linguistics and psycholinguistics together. Trento, Italy; 2006. p. 1-8.

Pedersen et al., 2004 Overview: Pedersen, Ted and Patwardhan, Siddharth, and Michelizzi, Jason :WordNet::Similarity - Measuring the Relatedness of Concepts In: Proceedings of Fifth Annual Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL-04), pp. 38-41, Boston, May 2004.

Smith, 1981 R.G. Smith.: The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver. IEEE Transactions on Computers, 29(12):1104–1113, 1981

Studer et al., 1998 Studer R., Benjamins V.R. et Fensel D.: Knowledge engineering: principles and methods, in IEEE Transactions on Data and Knowledge Engineering, 25(1&2), 1998, pp.161-197

Sun et al., 2005 W. Sun et al. : The Role of XML in Service Level Agreements Management. Network Management Research Center, Beijing Jiaotong University, IEEE, 2005

Tosic et al. 2004 Tosic, V., Ma, W., Pagurek, B., Esfandiari, B.: Web Service Offerings Infrastructure (WSOI) - a management infrastructure for XML Web services. In: Network Operations and Management Symposium, 2004. NOMS 2004. IEEE/IFIP 1, 817–830

Wup et al. 1994 Wup: Wu, Z., and Palmer, M. 1994. :Verb semantics and lexical selection. In 32nd Annual Meeting of the Association for Computational Linguistics, 133–138

Zghal et al., 2007b Zghal S., Kamoun K., Yahia S. B., Nguifo E. M., Slimani Y., :EDOLA : Une nouvelle méthode d'alignement d'ontologies OWL-Lite , Proceedings of 4`eme conférence francophone en Recherche d'information et Applications CORIA'2007, Saint-Étienne, France, p. 351-366, 2007b

Alignement sémantique des intentions des clients avec les offres des fournisseurs

Abstract

The establishment of Service Level Agreement (SLA) between clients and providers remains a complex task. Indeed, providers and clients don't share the same knowledge degree. To address this problem, we started by modeling client intentions and provider offers using ontologies. These semantic models helped us to develop algorithms verifying the matching process between the client ontology and the provider ontology. The developed algorithms are based on inference techniques and semantic interpretation to (1) detect semantic matching between the demands and the offers (2) evaluate the resulting matching and (3) automatically generate a Service Level Agreement in case of compatibility. To validate our matching approach, we have implemented a prototype based on semantic similarity measures between the terms of the client ontology and the provider ontology.