Découverte de Service : Étude d'Approche Syntaxique et Sémantique

Marie-Laetitia Denayer

Résumé Dans un projet distribué, le mécanisme de recherche de service est essentiel. En effet, ce mécanisme permet d'engager "la conversation". Différentes approches existent pour permettre des recherches sur des mots-clés, des classifications et maintenant sur la sémantique du service. Nous allons étudier ces différentes approches.

1 Introduction

Dans un projet distribué ou d'e-business, les acteurs communiquent entre eux pour atteindre leur objectif. Or, ils ne se connaissent pas. Il faut donc un mécanisme permettant à ces personnes de publier les services qu'ils offrent et de trouver les services dont ils ont besoin. Ce mécanisme doit être transparent pour l'utilisateur : les services proposés doivent correspondre à la requête de l'utilisateur.

Nous allons étudier les principes généraux de la recherche de service, et ensuite voir des applications de ces principes. Ces applications sont variées : l'accent peut être mis sur la communication (les différents messages à échanger) ou encore sur la satisfaction des acteurs (recherche du service le plus satisfaisant). Enfin, nous ferons une brève comparaison de ces applications.

2 Principes Généraux

La recherche de service confronte deux acteurs : le fournisseur ou producteur de service ; qui cherche à annoncer du mieux possible ses services et l'utilisateur ; qui ne sait pas où chercher le service de ses rêves.

Il faut donc mettre en place un agent faisant le lien entre ces deux acteurs et qui fournira les capacités suivantes :

1. Le fournisseur enregistre son service auprès de l'agent.
2. L'agent stocke cette information dans son registre, sa base de connaissance.
3. Le client envoie sa demande à l'agent.
4. L'agent répond au client par la liste des services satisfaisant sa requête.

L'information stockée par l'agent doit représenter les spécificités du service annoncé. Le **langage de description de service** donne les informations que le producteur doit fournir à l'agent.

Cette information doit être expressive autrement dit représenter toutes informations intéressantes dans le processus de recherche de service et doit être facile à utiliser : lisible mais aussi facile à écrire par le fournisseur de service.

La demande du client peut prendre deux formes, selon le choix du concepteur de l'agent :

- Soit l'agent met à disposition du client un mécanisme d'interrogation. La demande du client correspond à une interrogation du registre de l'agent.
- Soit le client exprime sa requête dans le langage de description de service et envoie cette requête à un agent qui fait correspondre la demande à l'offre via un **algorithme de matching**.

L'algorithme de matching compare toutes les annonces à la requête, pour fournir comme résultat les annonces qui sont proches de la requête. Comme ce mécanisme est commun à tous les algorithmes de matching, nous étudierons en détail ce que *proche* signifie dans les différentes applications.

Proche peut prendre plusieurs sens. En effet, nous souhaitons que notre algorithme de matching soit capable de reconnaître non seulement les services identiques mais d'autres services utilisable pour satisfaire la requête,...

Ces différents niveaux ne sont pas équivalents. On les appelle **degré de correspondance** et ils varient d'après les mécanismes.

L'algorithme permet généralement au client de fixer le degré de correspondance désiré entre sa requête et les annonces.

L'algorithme de matching doit permettre une recherche :

- précise. Les résultats renvoyés par l'algorithme doivent correspondre aux définitions des degrés de correspondance, expliqués ci-après pour chacun des mécanismes.
- efficace. Le temps de réponse doit être le plus court possible.
- efficiente. L'ensemble des résultats ne doit pas être trop grand. Mieux vaut un petit ensemble avec un degré de correspondance élevé.

L'algorithme doit réduire le nombre de faux positifs et de faux négatifs.

De plus, il serait intéressant d'encourager les acteurs à faire une description honnête de leur service. Autrement dit, si la description n'est pas honnête, le service n'est pas ou mal repris dans les résultats lui correspondant.

Nous allons maintenant examiner quelques mécanismes de recherche de service :

3 UDDI

Le but premier d'UDDI (pour Universal Description, Discovery Integration, voir [1]) est la spécification d'un framework pour décrire et découvrir des web services. En particulier, UDDI définit une structure de donnée et une API d'interrogation.

3.1 Langage de Description

L'information stockée dans les registres UDDI correspond à la figure 1 et comprends les quatre types d'informations suivantes :

businessEntity : donne les informations sur la compagnie offrant le service ;

businessService : donne les informations sur le service offert ;

bindingTemplate : donne l'information permettant d'utiliser un web service particulier ;

tModel : (technical model) représente toute sorte d'informations. Cela peut être une interface de service, une classification, la sémantique d'une opération,...

La relation entre les éléments est one-to-many. Ainsi, une compagnie peut proposer plusieurs services, un service peut avoir plusieurs "implémentations".

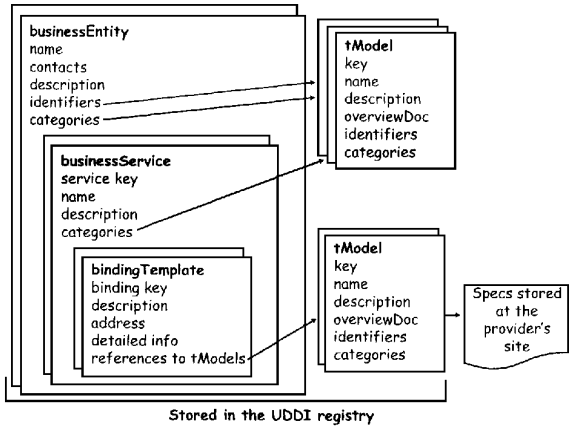

Fig. 1. structure de données d'UDDI

Le tModel, grâce à ses liens avec les autres éléments, est réellement le cœur des spécifications UDDI. Le contenu du tModel se trouve dans overviewDoc : il peut représenter tout type d'information et être écrit dans n'importe quel langage. Le tModel est référencé par sa clé unique

3.2 Mécanisme d'Interrogation

UDDI fournit une API d'interrogation de son registre. Elle permet une recherche par mot-clé, par tModel et donc, par voie de conséquence, une recherche par spécification (ex : classification, service utilisant WSDL,...).

On peut comparer l'information disponible à celle fournie par un annuaire :

les pages blanches : liste alphabétique des différentes compagnies, d'un numéro de contact ainsi que des différents services proposés

les pages jaunes : classification des services et des compagnies suivant une taxonomie standardisée ou définie par l'utilisateur.

les pages vertes : indications techniques pour faire appel au service, joindre le fournisseur.

4 LARKS

Une simple recherche par mot-clé ou via une classification ne répond pas à tous les besoins. Des recherches ont donc été réalisées afin d'utiliser des mécanismes plus intelligents, incluant la connaissance du monde (le but est de détecter que l'annonce : journal financier correspond à la requête : cours récent de bourse).

Un premier outil est LARKS (voir [2]), celui se base non seulement sur la syntaxe du service mais aussi sur sa sémantique. Pour cela, les auteurs de LARKS ont utilisé des techniques de IR, software engineering et de description logique.

4.1 Langage de Modélisation

Les services sont modélisés suivant le schéma suivant (voir figure 2).

Context	context of specification
Types	Declaration of used variable types
Input	Declaration of input variables
Output	Declaration of output variables
InConstraints	Constraints on input variables
OutConstraints	Constraints on output variables
ConcDescriptions	Ontological description of used words
TextDescriptions	Textual description of specification

Fig. 2. specification de LARKS

Cette modélisation définit une définition du service en terme de la transformation effectuée par le service et non par la fonction remplie par le service.

Cette modélisation implique la définition par chaque service des termes qu'ils emploient[1]. Ainsi, on trouvera dans le champs ConcDescription, la définition d'une mission aérienne comme étant une mission disposant d'avions.

L'agent n'a donc qu'un vue partielle du dictionnaire. Celle-ci dépend des services dont il connaît l'existence.

4.2 Algorithme de Matching

Le mécanisme de recherche est basé sur cinq filtres. Ces filtres se basent sur différents aspects de correspondance entre requête et annonce :

1. le **contexte**. La requête et l'annonce doivent partager le même domaine Le filtre calcule les distances entre les différents mots-clés d'annonce et de requête. Cette distance est basée sur les liens entre les mots tels que la généralisation, la spécialisation et l'association positive.
2. le **profil**. La requête et l'annonce doivent partager un certain nombre de mots dans leurs spécifications. Ce filtre compare les occurrences des mots de l'annonce et de la requête.
3. la **similarité**. La structure de la requête et la structure de l'annonce doivent être similaires, c.à.d. les concepts utilisés, par exemple dans la partie input, correspondent dans les deux cas. Au lieu de faire une comparaison de l'ensemble du document, on compare les différentes sections (input, output, pré condition, ...) avec le filtre de contexte.
4. la **signature**. La signature de l'annonce doit ressembler à la signature de la requête. On compare les différents termes (chaque paramètre individuellement) grâce au filtre de similarité. De plus, pour les paramètres input et output, on vérifie également que les types sont compatibles (équivalent, sous-classe, sur-classe).
5. les **contraintes**. L'utilisateur voulant utiliser un service doit vérifier les pré conditions de ce service. De même, le service doit répondre aux attentes de l'utilisateur. On peut exprimer ces conditions par : $(Pre_r \Rightarrow Pre_a) \wedge (Post_a \Rightarrow Post_r)$[2]. L'implication logique n'est pas décidable pour la logique du premier ordre. La vérification se fait donc sur une relation plus faible que l'implication logique.

La composition de ces filtres permet d'établir différents degrés de correspondance :

– **exact match.** Le service correspond parfaitement à la requête.
On compare la requête avec l'annonce au moyen des cinq filtres.

[1] Cela permet de prendre en compte le même mot mais dans un sens différents et deux mots avec le même sens.

[2] Pre_r = pré conditions de l'utilisateur
Pre_a = pré conditions de l'annonceur
$Post_r$ = post conditions de l'utilisateur
$Post_a$ = post conditions de l'annonceur

– **plug in match.** Le service annoncé peut correspondre (être branché) à la requête.
  Par exemple, l'utilisateur désire un service triant une liste d'entiers. Le médiateur propose un service de tri de liste d'entiers et de strings.
  On compare la requête avec l'annonce au moyen des deux derniers filtres : signature et contraintes.
– **relaxed match.** La relation relaxed match ne signifie pas que l'annonce correspond à la requête mais détermine que leur distance sémantique est inférieure à un seuil déterminé.
  Par exemple, un utilisateur cherche l'adresse d'un magasin vendant des ordinateurs HP et un service fournissant le prix et le numéro de téléphone d'un tel magasin.
  On compare la requête avec l'annonce au moyen des trois premiers filtres. Il est donc le moins consommateur de ressource.

En toute logique, les résultats du relaxed match devraient inclure les exact et plug in match.

5 DAML-S

DAML-S est une ontologie de service dont les instances représentent des web services, l'idée est également de pouvoir faire des recherches sur le contenu du service plutôt que simplement sur base de mot-clé.

5.1 Langage de Description de Service

L'intérêt de DAML-S (voir [3], [4]) est d'exprimer les caractéristiques d'un service et de permettre le partage. Ce langage a été conçu dans le but de permettre : la découverte automatique de web services, l'invocation automatique de web services, la composition et l'interopérabilité automatique des web services et la surveillance d'exécution.

Comme toute ontologie, elle n'a un sens que si les différents concepts employés ont un sens clairement défini et accepté par tous. Il est donc nécessaire de définir pour chaque domaine une ontologie. Cela permet d'ajouter des liens vers le monde mais surtout une bonne connaissance sémantique de ce monde.

Un service est décrit par :

service profile : description du service en terme d'offre à la communauté, constituée principalement : des inputs, outputs, pré conditions, post conditions,... Nous pouvons ainsi décrire un service : vente de livre.

Cette information est très utile pour la recherche de service.

service model : description du service en terme d'algorithme. Cette information décompose le service en terme de services plus simples. Par exemple, le service model correspondant à l'exemple de profile explicité ci-dessus serait

l'exécution séquentielle des étapes suivantes : d'abord naviguer pour trouver le livre désiré, ensuite vente du livre choisi à l'étape précédente.

Cette information permettra une analyse plus en profondeur du service, le suivi de l'exécution du service,...

service grounding : description du service en terme des messages échangés.

Cette information décrit comment accéder au service en faisant le lien avec un protocole de communication tel que SOAP, CORBA,... Cette partie spécifie également pour chaque type une manière non ambiguë d'envoyer les données (référence à un format).

Contrairement à LARKS, cette modélisation prend en compte les liaisons avec d'autres protocoles. De plus, le service model et le service grounding donnent les informations nécessaires pour utiliser le service.

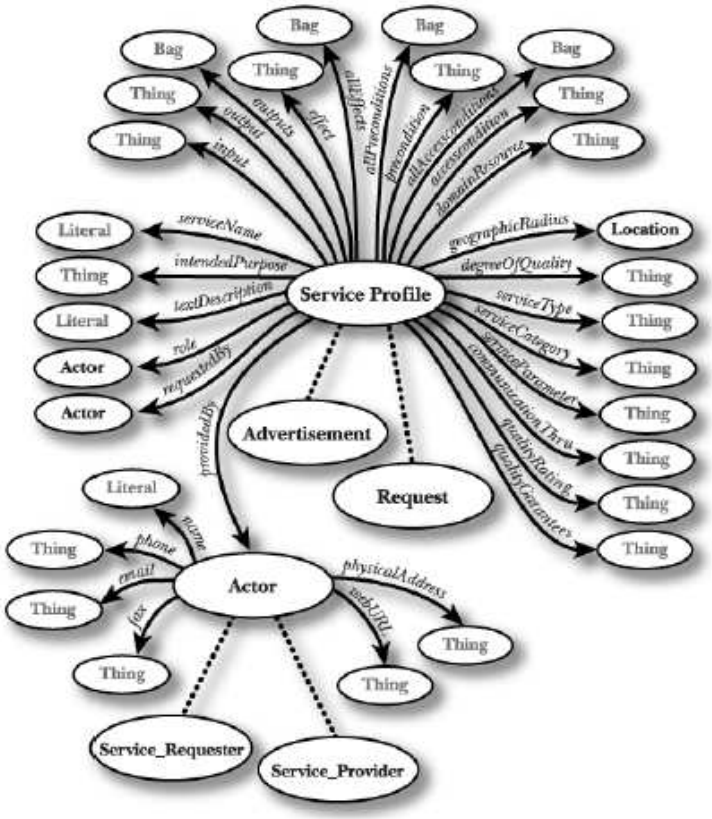

Fig. 3. service profile

Dans le cadre de la recherche automatique de service, le service profile est la partie la plus informative. Nous allons donc nous concentrer sur cette partie. Si nous détaillons le service profile (voir figure 3), nous obtenons les éléments suivants :

- une description du service lisible par l'homme.
- une spécification de la fonctionnalité ou description fonctionnelle. Cette partie décrit le service en terme d'aptitude : input/output, pré conditions et effets.
  Un input est ce qui est requis par le service pour produire l'output désiré. Les pré conditions sont des conditions dans le monde pour que l'execution soit réussie. L'exécution peut aussi avoir des actions sur le monde. C'est ce que décrivent les effets.
- les attributs fonctionnels : les attributs qui vont permettre de choisir entre deux services comme le degré de qualité (la plus rapide, le moins cher,...), le serviceParameter (une liste de propriété telle que le coût d'invocation, le temps de réponse),...
- information sur l'acteur : soit sur le fournisseur de service, soit sur le demandeur de service.

5.2 Algorithme de Matching

La technique de matching (voir [5]) se base sur les propositions de LARKS, en particulier le filtre sur la signature (qui est le plus contraignant), et sur le langage DAML-S.

Ce langage est très intéressant, en effet, il est en forte connexion avec DAML+OIL qui permet de raisonner sur le concept subsume (englobe).

L'idée est de chercher l'ensemble des annonces utilisables par le client. Pour ce faire, la signature de l'annonce doit correspondre à la signature de la requête de telle sorte que :

$input_{fournisseur}$ = {paramètres du service offert}
$input_{utilisateur}$ = {paramètres du service demandé}
$output_{fournisseur}$ = {paramètres du service offert}
$output_{utilisateur}$ = {paramètres du service demandé}
$input_{fournisseur} \subseteq input_{utilisateur} \wedge output_{utilisateur} \subseteq output_{fournisseur}$

Autrement dit, l'utilisateur sait contenter le service et le service renvoie ce que l'utilisateur demande.

Quand peux-t'on dire que cette contrainte est satisfaite. En effet, la notion d'équivalence entre paramètre n'est pas définie. $param_A$ est équivalent à $param_B$ si le type de $param_A$ est le même que celui de $param_B$. Cette définition ne permet de trouver que les annonces relativement identiques à la requête, la notion d'équivalence a donc été élargie et d'après cette définition le degré de correspondance de l'annonce avec la requête sera différent.

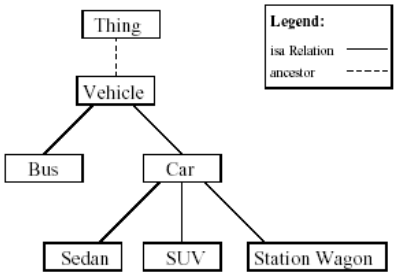

Fig. 4. fragment d'une ontologie sur les véhicules

Afin d'expliquer les différents degrés de correspondance, nous allons nous référer à un service de vente de voiture. La figure 4 représente l'ontologie du domaine voiture. Supposons un utilisateur cherchant un service dont l'output est une *Sedan*, le degré de correspondance sera :

– **exact :** le service proposé est celui attendu.

paramA = paramR [3]
outR subclassOf outA
inA subclassOf inR

Fig. 5. conditions pour avoir un degré de correspondance exact

Si l'annonce renvoie un paramètre de type *Sedan* ou un paramètre de type *Voiture*.
La deuxième assertion de la figure 5 signifie qu'un service proposant en output le type *Voiture* est un service capable d'offrir tous les sous-types de voiture : *Sedan, SUV, StationWagon*. S'il n'en est pas capable, il faut redéfinir l'annonce.

– **plug in :** le service annoncé peut être branché sur le service requis. Il remplit les fonctionnalités attendues dans la plupart des cas.
Si l'annonce renvoie un paramètre de type *Véhicule*. En effet, le type *Véhicule* englobe le type *Sedan*.

[3] paramA = paramètres du service Annoncé,
outA = correspond aux paramètres de l'annonce en sortie,
inA = correspond aux paramètres de l'annonce en entrée,
paramR = paramètres du service Requis, outR = correspond aux paramètres de la requête en sortie,
inR = correspond aux paramètres de la requête en entrée

[4] subsume = englobe, ispartOf.
subclassOf est un cas particulier.

outA subsumes[4]outR $\Rightarrow outR \subseteq outA$
inR subsumes inA $\Rightarrow inR \supseteq inA$

Fig. 6. conditions pour avoir un degré de correspondance plug in

La relation est plus faible car la relation entre outR et outA (voir figure 6) est plus faible que dans le cas ci-dessus. On peut s'attendre à ce que ce service puisse fournir des *Voitures* mais pas tous les types de *Sedan*.

– **subsume :** le service annoncé est plus général que le service requis. Le service annoncé ne sera satisfaisant que dans quelques rares cas particuliers.

outR subsumes outA $\Rightarrow outR \supseteq outA$
inA subsumes inR $\Rightarrow inR \subseteq inA$

Fig. 7. conditions pour avoir un degré de correspondance subsume

L'utilisateur devra changer ses plans ou utiliser des services complémentaires pour arriver à son but.

– **fail :** si aucune correspondance.

Le degré de correspondance du service dépend du plus mauvais score des paramètres du service.

5.3 Une Approche Plus Complète

Cet algorithme de matching ne prend en compte que la signature. Or la signature ne nous semble pas un élément suffisant pour identifier un service.

Tout d'abord deux services différents peuvent avoir la même signature. Des éléments tels que le contexte, les pré conditions, les effets permettent de souligner les différences dans les services.

Enfin, aucune indication n'est donnée sur la manière de filtrer les résultats avec les *attributs fonctionnels*.

Une évolution de cet algorithme serait d'ajouter des filtres comme dans LARKS, afin de prendre en compte toute l'information fournie par DAML-S. Une proposition (voir [6]) donne la figure 8.

6 Comparaison

Les langages expriment la même information mais des différences existent.

DAML-S et LARKS focalisent leur description sur la signature et les contraintes. Mais DAML-S prend en compte d'autres éléments pour différencier le service comme les attributs fonctionnels.

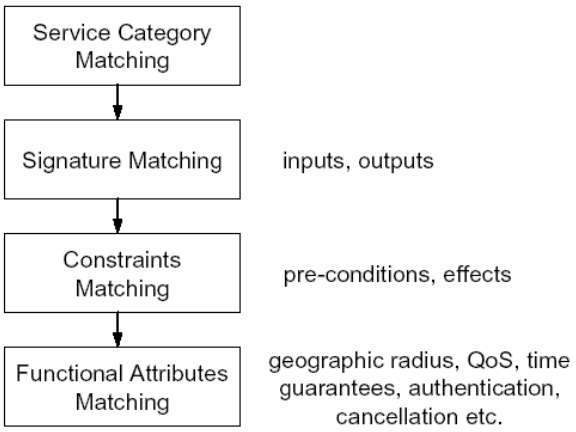

Fig. 8. algorithme de matching de DAML-S au moyen de filtre

UDDI et DAML-S prévoient dans leur description l'information nécessaire pour contacter le service.

Les algorithmes de matching proposent les résultats d'après différentes requêtes.

UDDI ne permet qu'une recherche par mot-clé ou par classification mais ces types de recherche sont suffisants dans la majorité des cas.

Mais la recherche par sémantique offre de nouveaux horizons. Toutefois l'utilisateur n'est pas toujours capable de rédiger sa requête dans le langage de description de service (sans outils).

L'approche par filtre est très intéressante. Elle permet d'examiner un grand nombre d'annonce tout en utilisant des techniques relativement précises et donc coûteuses pour les derniers filtres tout en conservant un temps de réponse raisonnable.

Pour conclure, nous allons examiner la réussite de ces applications en terme d'utilisation. UDDI est largement adopté, cela même dans l'industrie. DAML-S devient utilisé dans d'autres projets de recherche (comme ce projet [7]).DAML-S peut s'appuyer sur UDDI comme couche inférieure.

Enfin DAML-S, comme langage de description de services, a un avenir dans la localisation de services mais également dans d'autres domaines comme la composition automatique de web service.

Références

1. Alonso, G., Casati, F., Kuno, H., Machiraju, V. In : Web Services. (2003)
2. Sycara, K., Widoff, S., Klusch, M., Lu, J. : Larks : Dynamic matchmaking among heterogeneous software agent in cyberspace. Autonomous Agents and Multi-Agent Systems 5 (2002) 173–203
3. Ankolekar, A., Burstein, M., Hobbs, J.R., Lassila, O., Martin, D.L., McIlraith, S.A., Narayanan, S., Paolucci, M., Payne, T., Sycara, K., Zeng, H. : Daml-s : Semantic markup for web services. In : Proceedings of the International Semantic Web Working Symposium (SWWS). (2001)
4. Ankolekar, A., Burstein, M., Hobbs, J.R., Lassila, O., McDermott, D., Martin, D., McIlraith, S.A., Narayanan, S., Paolucci, M., Payne, T., Sycara, K. : Daml-s : Web service description for the semantic web. In : Proc. 1st Int'l Semantic Web Conf. (ISWC 02). (2002)
5. Paolucci, M., Kawamura, T., Payne, T.R., Sycara, K. : Semantic matching of web services capabilities. In : First International Semantic Web Conference. (2002)
6. Mohsin, W. : Semantic discovery of web services. http ://www.stanford.edu/w̃mohsin/semantic_discovery_of_web_services.pdf (2002)
7. Wroe, C., Stevens, R., Goble, C., Roberts, A., Greenwood, M. : a suite of daml+oil ontologies to describe bioinformatics web services and data. International Journal of Cooperative Information Systems **12** (2003) 197–224