Un Modèle de composition des services web sémantiques

N. TEMGLIT[1], H.ALIANE[2], M.AHMED NACER[3]

Division Théorie et Ingénierie des systèmes Informatiques(DTISI)
Centre de Recherche sur l'Information Scientifique et Technique (CERIST)
ALGERIE

[1]t_nacera@yahoo.fr, [2] haliane@cerist.dz

[3]Laboratoire des Systèmes Informatiques(LSI)
Université des Sciences et de la Technologie Houari Boumediene (USTHB)
ALGERIE

anacer@cerist.dz

RÉSUMÉ. Le travail présenté ici vise à proposer un modèle pour la composition des services web sémantiques. Ce modèle est basé sur une représentation sémantique de l'ensemble des concepts manipulés par les services web d'un domaine d'application, à savoir, les opérations et les concepts statiques utilisés pour décrire les propriétés des services web. Différents niveaux d'abstraction sont donnés au concept opération pour permettre un accès progressif aux services concrets. Ainsi, deux plans de composition à granularités différentes (abstrait et concrets) sont générés. Ceci permettra de réutiliser des plans déjà construits pour répondre à des besoins similaires et même avec des préférences modifiées.

ABSTRACT. The work presented here aims to provide a composition model of semantic web services. This model is based on a semantic representation of domain concepts handled by web services, namely, operations and the static concepts used to describe static properties of Web services. Different levels of abstraction are given to the concept of operation to allow gradual access to concret services. Thus, two different levels of the composition plan are generated (abstract and concret). This will reuse plans already constructed to meet similar needs even with modified preferences.

MOTS-CLÉS : composition de services web, matching sémantique, template de composition.

KEYWORDS: Web Service composition, semantic matching, composition template.

1. Introduction

L'évolution d'Internet et la compétitivité entre les entreprises ont été les facteurs de l'explosion des services Web. La notion de service web désigne essentiellement une application mise à disposition sur Internet par un fournisseur de service, et accessible par des clients à travers des protocoles Internet standards. Le modèle de base des services fait intervenir trois catégories d'acteurs: les fournisseurs de services, les clients et les annuaires La dynamique entre ces trois acteurs est normalisée à travers 3 standards: un protocole abstrait de description et de structuration des messages, SOAP **[14]**, une spécification de publication et localisation des services, UDDI **[15]** et un format de description des services Web, WSDL **[17]**. Néanmoins, cette infrastructure s'est avérée insuffisante pour rendre automatique et efficace plusieurs tâches liées au cycle de vie des services web, telles que: la composition et aussi la découverte des services requis. En particulier, WSDL fournit une description concrète mais de bas niveau d'un service Web qui reste insuffisante pour qu'un agent logiciel puisse interpréter la signification réelle des opérations WSDL. Par conséquent, la communauté web sémantique a développé un langage ontologique pour les services web s'inspirant de OWL**[8]** appelé OWL-S**[9]** (Ontology Web Language for Web Services).

2. Problématique de la composition des services web

La composition des services web se réfère au processus de création d'un service composite offrant une nouvelle fonctionnalité, à partir de services web existants plus simples, par le processus de découverte dynamique, d'intégration et d'exécution de ces services dans un ordre bien défini **[6].** A titre d'exemple, nous pouvons modéliser le service de planification de voyage comme la composition de plusieurs Services web plus simples : réservation de vol, réservation d'hôtel et allocation de véhicule.
Selon les travaux effectués dans le champ des services web, on peut classifier la composition selon différents points de vue : manuelle, semi-automatique et automatique ou bien statique et dynamique. Dans le cas de la composition statique, le demandeur doit définir à priori le modèle abstrait de processus décrivant le plan de composition, par exemple, sous forme d'un graphe de tâches. Dans ce type d'approche, uniquement la sélection et la liaison aux services basiques se font de manière automatique, citons en l'occurrence le système **eFlow[3]**. Par contre, la composition dynamique vise à créer le modèle de processus abstrait, sélectionner et lier les services atomiques aux tâches abstraites de manière automatique et à la demande **[13].** Ce type de composition soulève un défi pour la communauté web sémantique qui essaye d'appliquer des techniques de planification de l'IA pour générer le plan de composition de façon automatique. Nous citons, en l'occurrence, les travaux suivants **[5] [4] [11] [6] [16] [12]**.

Cependant, les méthodes citées auparavant ne prennent pas suffisamment en compte le problème d'interaction avec un utilisateur voulant exécuter un service composite, sachant que les services web concernés sont à priori inconnus et dispersés à l'intérieur d'un vaste entrepôt de services.

3. Un Modèle de composition des services web

Le but de notre travail est de définir une approche progressive de composition des services web en partant d'une requête déclarative simple spécifiée par un utilisateur non-spécialiste. Ce dernier aura simplement à spécifier son besoin en terme de tâches à effectuer. La découverte des services concrets et leur composition seront effectuées de façon automatique et transparente vis-à-vis de l'utilisateur. Pour ce faire, nous avons basé notre modèle sur un support sémantique et un schéma de mise en correspondance que nous décrivons tout de suite. Ajoutant à cela, le modèle proposé permettra de générer des plans à deux niveaux de granularités (abstrait et concret) et cela pour répondre aux éventuels problèmes de disponibilité des services et permettre la réutilisation des plans déjà construits pour répondre à des demandes similaires même avec des préférences modifiées.

3.1 Le support sémantique

Les services web tel qu'ils sont présentés sur le web ne permettent pas à leurs utilisateurs de découvrir les fonctionnalités complexes qu'ils peuvent offrir. Pour cela, nous avons basé notre modèle sur une description ontologique qui distingue, d'un coté, trois niveaux d'abstraction des opérations du domaine d'application. D'un autre coté, elle permet de décrire de manière hiérarchique les concepts statiques des services web.

3.1.1 Les opérations : Nous représentons les opérations du domaine selon 3 abstractions différentes : Opération abstraite générique, opération abstraite simple et opération concrète. Cette distinction a pour but d'élever le niveau d'abstraction de telle manière que l'utilisateur pourra spécifier sa requête en termes de tâches à accomplir et non pas en termes de services qu'il doit invoquer (Fig.1).

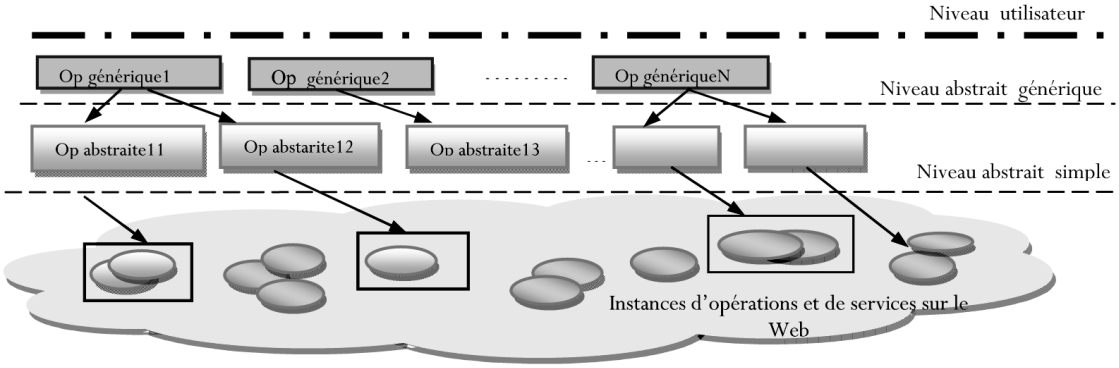


Figure 1. *Les trois niveaux d'abstraction des opérations du domaine*

• Opération générique : modélise une tâche abstraite qui peut être décomposée en opérations plus simples. Les opérations génériques représentent les traitements métiers les plus fréquemment sollicités dans un domaine d'application donné. Elle est décrite par les attributs: Catégorie[domaine, synonymes], Fonction[nom-fonctionnalité, synonymes], E/Sorties[nom, synonymes, type] et une Description textuelle de l'opération. Par exemple, l'opération de réservation de vol peut être décrite ainsi :

Cat: *[flight-opération]*, Fonc: *[flight-booking, flight-reservation],*In: *[dep-date], [return-date], [from, dep-city], [to, arr-city],* Out: *[airline], [flight-Nr], [Seat-Nr].*

• Opération abstraite : décrit de façon abstraite une fonctionnalité qui peut être implémentée par plusieurs opérations concrètes appartenant à des services différents. Contrairement à une opération générique, une opération abstraite n'est pas décomposable mais elle est directement instanciée en opération(s) concrète(s). Par exemple : la combinaison des deux opérations décrites ci-dessous produit l'opération générique citée auparavant (flight-reservation):

1ère opération: Cat: *[flight-operation],* Fonc: *[flight-lookup, serachFor-flight],* In: *[dep-date], [arr-date], [from, origin, dep-city], [to, dest, arr-city],* Out: *[airline], [flight-Nr]..*

2ème opération: Cat: *[flight-opération],* Fonc: *[buy-flightTicket],* In: *[price], [credit-card-Nb], [flight-Nb],* Out: *[buyTicket-confir], [..].*

• Opération concrète est une opération appartenant à un service web disponible sur le web, donc elle possède une implémentation réelle de sa fonctionnalité. Sa description hérite celle de l'opération abstraite avec quelques extensions décrivant les détails d'implémentation (type de *Binding tel que* SOAP/HTTP) et d'autres informations décrivant sa qualité. En effet, la qualité d'une opération recouvre trois aspects: exécution [Temps de réponse, disponibilité,..], sécurité [authentification, confidentialité,..] et la qualité métier [Coût, réputation..].

3.1.2. Les concepts statiques

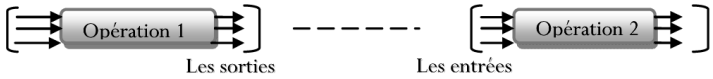
Les concepts statiques sont utilisés pour décrire les propriétés des services web, par exemple, les E/Ss d'une opération doivent avoir une représentation sémantique qui explicite leurs liens de subsomption afin de pouvoir vérifier leur compatibilité de composition. Pour cela, nous les organisons sous forme hiérarchique explicitant les liens de subsomption entre les concepts les plus spécifiques à leurs concepts génériques.

3.2 Schéma de mise en correspondance

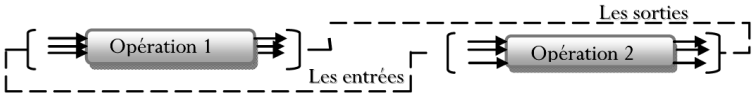
La mise en correspondance sémantique est un mécanisme qui vise à trouver les similarités sémantiques entre deux types d'informations: l'information recherchée et l'information publiée. Notre schéma de correspondance se base sur l'algorithme de Paolucci **[10]**. Cependant nous l'avons modifié pour qu'il devienne plus flexible dans le

sens où il n'est pas nécessaire que toutes les E/Ss du service recherché soient mises en correspondance avec toutes les E/Ss du service publié. Dans le cadre de notre travail, nous distinguons trois modes de correspondance :

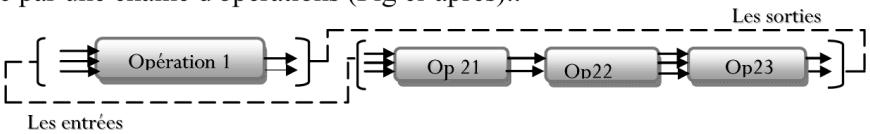
a) H**orizontale :** elle est appliquée lorsqu'on veut vérifier si deux opérations OP1 et OP2 peuvent être enchaînées ensemble. (Fig ci-après).



b) **Verticale 1:1:** est nécessaire lorsqu'on veut vérifier si deux opérations OP1 et OP2 ont des fonctionnalités similaires (Fig ci-après)..



c) **Verticale 1:N :** est appliquée lorsqu'on veut vérifier si une opération OP1 peut être substituée par une chaine d'opérations (Fig ci-après)..



Nous définissons les règles de correspondance entre les paramètres d'E/Ss comme suit :

∗ **Les sorties:** Pour les sorties, nous avons besoin de faire correspondre toutes les sorties du service recherché à toutes ou quelques sorties du service publié :

a) Exact: $Out(OP_R) == Out(OP_P)$ si :

- $Out(OP_R)$ et $Out(OP_P)$ ont le même nombre de variables et
- $\forall x \in Out(OP_R)$ (resp. $Out(OP_P)$),$\exists y \in Out(OP_P)$ | *match(x,y)= "exact "* .

b) Plug-In: $Out(OP_R) =\approx Out(OP_P)$ si :

- $Out(OP_R)$ et $Out(OP_P)$ ont le même nombre de variables de sortie et
- $\forall x \in Out(OP_R)$,$\exists y \in Out(OP_P)$ | *match(x,y)= "exact "* ou *match(x,y)= "plug-in"* et $\exists z \in Out(OP_R)$,$\exists t \in Out(OP_P)$ | *match(z,t)= "plug-in"*

c) Inclusion exacte: $Out(OP_R) \subset Out(OP_P)$ si:

- $Out(OP_P)$ contient plus de variables que $Out(OP_R)$
- $\forall x \in Out(OP_R)$,$\exists y \in Out(OP_p)$ | *match(x,y)= "exact "* .

d) Inclusion Plug-In: $Out(OP_R) \subset\approx Out(OP_P)$ si :

- $Out(OP_P)$ contient plus de variables que $Out(OP_R)$.
- $\forall x \in Out(OP_R)$,$\exists y \in Out(OP_P)$ | *match(x,y)= "exact "* ou *match(x,y)= "plug-in"* et $\exists z \in Out(OP_R)$,$\exists t \in Out(OP_P)$ | *match(z,t)= "plug-in"*

D'après **[10]**: *match(x,y)="exact"* signifie que x, y font référence à la même classe dans l'ontologie et *match(x,y)="plug-in"* signifie que classe de x dans l'ontologie est une sous classe de y (le concept publié y *subsume* le concept recherché x).

∗ **Les entrées:** nous avons besoin de faire correspondre toutes ou quelques entrées du service recherché à toutes les entrées du service publié. Par conséquent, nous définissons $In(OP_R)==In(OP_P)$ (exact), $In(OP_R)=\approx In(OP_P)$ (plug-in) de façon similaire que $Out(OP_R)==Out(OP_P)$, $Out(OP_R)=\approx Out(OP_P)$ et les autres cas comme suit :

c) Inclusion exacte: $In(OP_R)\supset In(OP_P)$ si:

- $In(OP_R)$ contient plus de variables que $In(OP_P)$
- $\forall x \in In(OP_P), \exists y \in In(OP_R) \mid match(y,x)= "exact"$.

d) Inclusion Plug-In: $In(OP_R) \approx\supset In(OP_P)$ si :

- $In(OP_R)$ contient plus de variables que $In(OP_P)$.
- $\forall x \in In(OP_P), \exists y \in In(OP_R) \mid match(y,x)= "exact"$ ou $match(y,x)= "plug\text{-}in"$ et $\exists z \in Out(OP_R), \exists t \in In(OP_R) \mid match(z,t)= "plug\text{-}in"$

Ainsi pour chaque niveau et selon la combinaison de l'ensemble des entrées avec l'ensemble des sorties, nous distinguons 4 niveaux de correspondance pour chaque mode de correspondance: exact, plug-in, inclusion exacte et inclusion plug-in.

3.3 Cycle de vie de la requête de composition

Afin de permettre à un utilisateur d'accéder graduellement, à travers une requête simple, aux fonctionnalités complexes offertes par une combinaison de services web, nous proposons différents niveaux de modélisation du plan de composition. Ces derniers constitueront les différentes phases de résolution de la requête de composition.

- **Requête utilisateur :** L'utilisateur peut ne pas avoir une idée complète et assez précise concernant le service qu'il cherche. Dans le cas général, nous considérons la requête comme étant composée de: Nom du domaine sur lequel il veut faire sa recherche, une ou plusieurs sous-tâches et leurs paramètres d'E/Ss, une valeur entière entre 0 et 1 représentant le seuil de correspondance qu'il précise lui même, un ensemble de conditions sur les paramètres apparaissant dans la requête et un profile dans lequel l'utilisateur spécifie ses préférences en terme de qualité (sous forme de PQ= val).

Requête :- nomDomaine, $\cup_{i=1}^{n}$ *Sous-tâche*$_i$ *(Nom,In,Out) , seuil,* $\cup_{j=1}^{k}$ *Condition*$_j$*(In, Out), profile.*

- **Template générique :** Elles sont considérées comme des processus métiers connus dans un domaine d'application particulier. Dans notre cas, une template générique est la combinaison de plusieurs opérations génériques suivant un modèle d'orchestration (ex : diagramme d'activité). Elles sont, donc, utilisées pour définir un plan initial de composition et permettent d'offrir une vue générale d'un processus métier complexe du domaine d'application avec un haut niveau de réutilisabilité. Citons des exemples de processus très souvent sollicités qui peuvent être modélisés grâce à des templates génériques: organisation d'un voyage, achat de véhicule… et.

- **Le plan abstrait de composition :** Représente le deuxième niveau de traitement de la requête de composition. Un plan abstrait de composition est une agrégation d'un ensemble d'opérations abstraites suivant un modèle d'orchestration. Le passage d'une Template générique au plan abstrait se fait grâce à une mise en correspondance de chaque opération générique avec une opération abstraite ou une combinaison d'opérations abstraites suivant les règles de correspondance (verticale 1:1 ou 1:N et horizontale) définies auparavant.
- **Le plan concret de composition (le plan exécutable):** C'est une agrégation d'un ensemble d'opérations concrètes suivant un modèle d'orchestration définissant le flux de contrôle et de données entre les opérations participantes. En effet, chaque opération abstraite du plan abstrait sera instanciée en une opération concrète grâce aux règles de correspondance verticale1:1. Plusieurs plans concrets peuvent être générés car plusieurs opérations concrètes peuvent implémenter la même fonctionnalité décrite par une opération abstraite. Pour cela, nous considérons à ce niveau de correspondance d'autres informations concernant la qualité d'une opération (ex: le coût, le temps de réponse..).
- **Spécification détaillée et exécution du service composite:** Cette dernière phase consiste à générer une description détaillée du service composite en utilisant un langage de spécification de flux tel que BPEL4WS**[2]** ou autres. Cette description comprend une définition des services participants, les messages échangés, leurs paramètres et le flux de contrôle et de données entre eux. Cette spécification sera soumise directement à un moteur d'exécution de workflow tel que le BPWS4J ou le BPEL Orchestration Server **[1]**.

4. Conclusion et perspectives

Le modèle proposé permet une résolution progressive de la requête de composition. En effet, l'utilisateur spécifie simplement ce qui doit être fait en terme de tâches qui doivent être effectuées et non pas en termes de services qu'il doit invoquer. La découverte des services participants et leur composition sont effectuées de façon automatique et transparente vis-à-vis de l'utilisateur grâce à un schéma de mise en correspondance flexible. Ce schéma définit différents niveaux de correspondance (exact, plug-in, inclusion, …) permettant de composer des services qui propose les mêmes ou une partie des fonctionnalités qui sont suffisamment pertinents pour satisfaire la requête de l'utilisateur.

Vu la complexité du système de composition, plusieurs extensions peuvent être envisagées, notamment: Enrichir la description sémantique par de nouvelles informations telles que les pré/post-conditions et rendre ainsi plus intelligent le schéma de matching. Nous envisageons aussi de développer un processus de résolution de la requête et d'optimisation du plan de composition généré en terme de qualité.

5. Références Bibliographiques

[1]: *"Business Process Modeling Language".* URL: http://www.bpmi.org/

[2]: BEA, IBM, Microsoft (2003) *"Business Process Execution Language for Web Services (BPEL4WS)"*. http://xml.coverpages.org/bpel4ws.html.

[3]: F. Casati, S. Ilnicki, and L. Jin. *"Adaptive and dynamic service composition in Eflow"*. In Proc of 12th International Conference on Advanced Information Systems Engineering (CAiSE), Stockholm, June 2000.

[4]: Ghallab, M., Howe, A., Knoblock, C., McDermott, D., Ram, A., Veloso, M., Weld, D., *"PDDL: the planning domain definition language".* In AIPS-98 Planning Committee (1998).

[5]: S. McIlraith and T. C. Son. *"Adapting Golog for composition of Semantic Web services".* In proc of the 8th International Conference on Knowledge Representation and Reasoning(KR), Toulouse, April 2002.

[6]: B . Medjahed, A. Bouguettaya, and A. K. Elmagarmid. *"Composing Web services on the Semantic Web"*. The VLDB Journal, 12 (4), November 2003.

[7]: Tarek MELLITI, *"Interopérabilité des services Web complexes, Application aux systèmes multi-agents"*, thèse de Doctorat de l'Université Paris IX Dauphine, décembre 2004.

[8]: http://www.w3.org/TR/owl-features/

[9]: http://www.ai.sri.com/daml/services/owl-s/1.2/

[10]: Paolucci, M., Kawamura, T., Payne, T., Sycara, K., "*Semantic Matching of Web Services Capabilities*", in Proc. of Intl. Semantic Web Conference, Sardinia, Italy, June 2002.

[11]: S. R. Ponnekanti and A. Fox. *"SWORD: A developer toolkit for Web service composition"*. In Proceedings of the 11th World Wide Web Conference, Honolulu, HI, USA, 2002.

[12]: Rao, P. K¨ungas, M. Matskin. *"Application of Linear Logic to Web service composition"*. In Proceedings of the 1st International Conference on Web Services, Las Vegas, USA, June 2003.

[13]: Jinghai Rao and Xiaomeng Su, *"A Survey of Automated Web Service Composition Methods" Norwegian University of Science and Technology Department of Computer and Information Science* N-7491*,* Trondheim, Norway, 2004.

[14]: W3C (2003), *"Simple Object Access Protocol (SOAP)"*. http://www.w3.org/TR/soap.

[15]: W3C (2003), *"Universal description, discovery, and integration (UDDI)".* http://www.uddi.org

[16]: D. Wu, E. Sirin, J. Hendler, D. Nau, and B. Parsia. *"Automatic Web services composition using SHOP2"*. In *Workshop on Planning for Web Services*, Trento, Italy, June 2003.

[17]: W3C (2003), "Web Services Description Language (WSDL)". http://www.w3.org/TR/wsdl.