Personnalisation de Services Web: Approche Fondée sur la Composition

Yacine Sam — Omar Boucelma

LSIS - CNRS, Université Aix-Marseille 3
Domaine Universitaire de Saint-Jérôme
Avenue Escadrille Normandie-Niemen
13397 Marseille, France
{yacine.sam,omar.boucelma}@lsis.org

RÉSUMÉ. *De nos jours, les exigences des clients contraignent souvent les entreprises à assurer une diversité de plus en plus accrue dans la gamme de leurs produits et services. Ceci a donné lieu, très récemment, au paradigme de personnalisation de masse. Ainsi les produits et services seront conçus de sorte à permettre un maximum de configurations différentes. La prolifération exponentielle du nombre de services offerts sur le Web et l'aspect cosmopolite de ce dernier motivent l'application de ce paradigme aux services Web. Dans cet article, nous présentons un cadre qui permet la personnalisation automatique de services Web. Il est fondé sur la transformation de la configuration de l'offre de services enregistrée par le fournisseur de services dans un annuaire vers d'autres configurations répondant aux préférences des clients.*

ABSTRACT. *In order to fulfil current customers' requirements, companies and services providers need to supply a large panel of their products and services. This situation has led recently to the Mass Customizing Paradigm, meaning that products and services should be designed in such a way that makes it possible to deliver and adapt different configurations. The increasing number of services available on the Web, together with the heterogeneity of Web audiences, are among the main reasons that motivate the adoption of this paradigm to Web services technology. In this paper we describe a solution that allows automatic customization of Web services: a supplier configuration, published in a services repository, is automatically translated into another configuration that is better suitable for fulfilling customers' needs.*

MOTS-CLÉS : *service Web, Web sémantique, composition de services Web, paradigme de personnalisation de masse*

KEYWORDS: *Web service, semantic Web, Web services composition, Mass Customizing Paradigm*

1. Introduction

De nos jours, le Web n'est plus simplement un énorme entrepôt de texte et d'images, son évolution a fait qu'il est aussi un fournisseur de services. La notion de "service Web" désigne essentiellement une application mise à disposition sur Internet par un fournisseur de services, et accessible par les clients à travers des protocoles Internet standards. Des exemples de services actuellement disponibles concernent les prévisions météorologiques, la réservation de voyages en ligne, les services bancaires ou des fonctions entières d'une entreprise comme la gestion de la chaîne logistique. Par essence, les services Web sont des composants logiciels autonomes et auto-descriptifs et constituent par ce fait un nouveau paradigme pour l'intégration d'applications.

Actuellement, les services Web sont mis en œuvre au travers de trois technologies standards : WSDL[1] , UDDI[2] et SOAP[3]. Ces technologies facilitent la description, la découverte et la communication entre services. Cependant, cette infrastructure de base ne permet pas encore aux services Web de tenir leur promesse d'une gestion largement automatisée. Cette automatisation est pourtant essentielle pour faire face aux exigences de passage à l'échelle et de la volonté de réduire les coûts de développement et de maintenance des services. Fondamentalement, elle doit s'accommoder d'un moyen pour décrire les services Web d'une manière compréhensible par une machine.

Le Web sémantique [BER 01, HOR 02a] est une vision du Web dans laquelle toute information possède une sémantique compréhensible par une machine. Appliqués aux services Web, les principes du Web sémantique doivent permettre de décrire la sémantique de leurs fonctionnalités, et les raisonnements induits constituent par conséquent une proposition d'automatisation des différentes tâches de leur cycle de vie.

En général, la description d'un service Web sémantique est construite avec des termes qui ont eux-mêmes une description formelle et qui sont regroupés dans des dictionnaires structurés appelés ontologies. DAML+OIL [HOR 02b] est l'un des langages logiques de description de services Web les plus connus. Il est utilisé directement ou au travers de DAML-S [BUR 02]. DAML-S est, quant à lui, une ontologie de concepts DAML+OIL décrivant les aspects techniques d'un service Web (entrées/sorties, paramètres, types de données, etc). Le langage OWL[4], évolution directe de DAML+OIL, a récemment été normalisé par le W3C[5], c'est désormais le principal langage standard de description d'ontologies sur le web. OWL-S est l'évolution correspondante de DAML-S.

Parallèlement à ces standards du W3C, d'autres langages, certes moins utilisés, représentent des solutions élégantes aux besoins de traitement automatique des services Web. On peut citer GOLOG qui est un langage de calcul des situations, et qui a été

1. Web Services Description Language". Voir http ://www.w3.org/TR/wsdl/
2. Universal Description, Discovery and Integration". Voir http ://www.uddi.org/
3. Simple Object Access Protocol". Voir http ://www.w3.org/TR/soap12-part0/
4. Web Ontology Language voir http ://www.w3.org/TR/owl-features/
5. World Wide Web Consortium voir http ://www.w3.org/

étendu dans [MCI 02] pour la composition de services Web. LARKS[6] [SYC 02], pour sa part, est un langage qui se base sur le calcul de frames. C'est ce langage que nous allons étendre afin d'élaborer un cadre pour la personnalisation de services Web.

LARKS est un langage de description, de publication et d'interrogation de services Web. Les demandes et les offres de services sont décrites dans ce langage par des classes d'attributs, où chaque attribut représente un élément (nom, entrées/sorties, contraintes) dans la signature du service. De plus, les noms des services et leurs entrées/sorties peuvent être annotés sémantiquement par des concepts ayant une description formelle afin d'être interprétables par une machine.

Dans cet article, nous allons utiliser les concepts sémantiques entre autres pour caractériser les différentes variantes (unités de mesure) dans lesquelles sont exprimées les valeurs des entrées/sorties des services Web. Si un service de vente de voitures a par exemple comme entrée l'attribut modèle et comme sortie l'attribut *prix*, ce dernier peut être annoté par le concept EURO, qui précise l'unité de mesure de l'attribut *prix*.

Le besoin des fournisseurs de représenter dans les services Web qu'ils offrent l'information sur le type de la monnaie, la langue, ou plus généralement l'unité de mesure d'une entrée/sortie est le résultat du paradigme de personnalisation de masse [KOV 02]. Ce dernier stipule que les produits et services doivent être conçus sous différentes configurations afin de satisfaire les préférences des clients.

La prolifération exponentielle du nombre de services offerts via le Web, et l'aspect cosmopolite de ce dernier motivent l'adoption du paradigme de personnalisation de masse aux services Web. Nous proposons un cadre qui permet la personnalisation automatique de services Web dont le but est double. D'une part, les fournisseurs se voient faciliter la tâche de construction de plusieurs configurations pour un service donné. D'autre part, les clients pourront obtenir les offres de services dans les variantes qui répondent à leurs préférences, même si celles-ci ne figurent pas explicitement dans l'annuaire de services. Ainsi, les fournisseurs de services ne publient explicitement qu'une seule configuration, d'autres sont déduites dynamiquement et de façon automatique au moment de l'interrogation de l'annuaire.

Le reste de cet article est organisé comme suit : nous fournissons, dans la section 2, un exemple de motivation qui illustre le besoin des clients d'avoir des services Web personnalisables. La section 3 présente le langage LARKS et les principes de son utilisation pour la description et la recherche de services Web. Dans la section 4, nous développons notre approche pour la personnalisation de services Web. D'abord, nous proposons une nouvelle structure pour la description de services Web sémantiques qui permet, contrairement à LARKS, de tenir compte du besoin de personnalisation des services. Nous décrivons ensuite le processus d'appariement de services. Enfin, nous donnons l'algorithme de personnalisation automatique de services Web. La section 5 conclut ce travail et donne quelques perspectives.

6. Language for Advertizing and Requesting for Knowledge Sharing

2. Scénario d'illustration

Chaque année, la fête des lumières est la manifestation traditionnelle et populaire la plus importante dans la région Lyonnaise. Avant de se déplacer a Lyon, un touriste Japonnais veut obtenir des informations sur les Hôtels qui y sont disponibles et leurs tarifs. Il envoie ainsi sa requête à un annuaire qui stocke ce type d'informations sous forme de services, attendant de recevoir au retour des services qui puissent le satisfaire.

Après le traitement de la requête, l'annuaire de services interrogé s'avère incapable de satisfaire le besoin en information du client japonnais. En effet, les services retournés décrivent les Hôtels en Français et leurs tarifs en EURO. Ceci les rend inutiles aux yeux du client, lui qui ne comprend que le Japonais et n'utilise que la monnaie locale, en l'occurrence le YEN. Le constat fondamental à travers ce scénario est que le client n'est pas satisfait non pas parce que le service recherché n'existe pas dans l'annuaire, mais parce qu'il existe dans une configuration incompatible à ces attentes.

Le cadre que nous proposons permet la transformation du service Web publié dans l'annuaire. Le principe de cette transformation est d'engendrer deux appels dynamiques de services Web. Le premier traduit les descriptions du Français au Japonnais et le second transforme les tarifs de l'Euro en YEN. La réponse à la requête du client sera ainsi construite, d'une façon transparente, par la coordination de ces deux services et du service de l'information sur les hôtels initialement disponible dans l'annuaire.

La section suivante introduit le langage de services Web sémantiques LARKS qui sera considéré comme le langage de base de ce travail. Il sera en effet enrichi dans la section 4 pour servir de support au principe de personnalisation de services Web.

3. Le langage LARKS

LARKS est un langage de publication et d'interrogation de services Web. La spécification d'une offre ou d'une demande de services avec LARKS est une classe dont la structure est illustrée par la Figure 1. La signification de chaque attribut est ci-après :

Context
Type
Input
Output
InConstraints
OutConstraints
ConcDescriptions
TextDescription

Figure 1. *Structure d'une spécification* LARKS

– **Context** : Contexte de la spécification d'une demande ou d'une offre de services, il représente un mot-clé décrivant ce que fait le service.

– **Type** : Définition des types de données abstraits utilisés dans la spécification.

– **Input/Output** : Déclaration des variables d'entrée/sortie d'une spécification de services. *Context* et *Input/Output* sont des mots-clés décrivant le nom et les entrées/sorties des services, respectivement. Ces attributs peuvent être annotés par des concepts sémantiques stockés dans l'attribut *ConcDescriptions*.

– **InConstraints/OutConstraints** : Contraintes logiques sur les entrées/sorties. Elles peuvent être des restrictions de valeurs sur les entrées/sorties ou des contraintes logiques entre les entrées/sorties d'une demande ou d'une offre de services.

– **ConcDescripton** : Description formelle des concepts servant à l'annotation sémantique des noms et des entrées/sorties des demandes et des offres de services. Le rattachement d'un concept *C* à un mot *w* se fait sous la forme *w*C*, et qui signifie que le concept *C* est la description formelle du mot *w*.

– **TextDescription** : Description textuelle de ce que recherche un demandeur de services, ou de ce que peut offrir un fournisseur de services.

L'utilisation des ontologies dans LARKS permet de décrire sémantiquement les services Web. Les ontologies peuvent être décrites formellement dans des langages de concepts comme ITL [GUA 91], LOOM [MAC 91], CLASSIC [BOR 89], ou KIF (Knowledge Interchange Format) [GEN 91]. Nous proposons dans ce qui suit une nouvelle structure pour les services Web qui constitue la base du processus de leur personnalisation.

4. Personnalisation de services Web

Cette section introduit, successivement, une nouvelle structure pour les services Web sémantiques, le processus d'appariement qui lui est associé, et un algorithme qui supporte le processus de personnalisation de services Web.

4.1. *Une nouvelle structure pour les services Web sémantiques*

La structure proposée dans cet article est utilisée à la fois par les demandeurs et par les fournisseurs de services, Elle est composée de deux sous-systèmes : le *Système Structurel Typé*, et le *Système à Contraintes*.

4.1.1. *Le Système Structurel Typé (SST)*

Le *SST* représente la partie d'un service qui sert à sa **découverte**. Le *SST* est défini par le triplet $(\mathcal{C}, \mathcal{I}, \mathcal{O})$. $\mathcal{C}$ est le contexte de la spécification, il est défini par un mot-clé ayant trait au domaine du service spécifié. $\mathcal{I}$ est la description des variables d'entrée et de leurs types de données abstraits dans l'offre ou dans la demande de services. $\mathcal{O}$ est la description des variables de sortie et de leurs types de données abstraits dans l'offre ou la demande de services. Dans la Figure 2, l'attribut *price*, qui représente le prix d'un livre, est de type *Real* dans la sortie de la spécification de services.

Les mots-clés du triplet $(\mathcal{C}, \mathcal{I}, \mathcal{O})$ peuvent être annotés par des concepts formels définis dans une ontologie partagées entre les utilisateurs de services du domaine.

Exemple 1 *La Figure 2 illustre le SST d'un service de vente de livres. Il est décrit par son nom "Book" et ses entrées/sorties "your-book"/("Price","presentation"). Les paramètres de sortie "Price" et "Presentation" sont annotés par les concepts* ***Price*** *et* ***Description****, respectivement. Tous les deux sont des concepts définis dans l'ontologie du domaine (que nous considéront unique, voir Figure 3).*

$\mathcal{C}$	Book
$\mathcal{I}$	Your-Book : String
$\mathcal{O}$	Price***Price** : Real, Presentation***Description** : String

Figure 2. *SST d'une spécification de services dans le domaine du livre.*

Les concepts sémantiques peuvent être utilisés pour la désignation des unités de mesure des entrées/sorties des servies Web. Par les types des entrées/sorties nous ne signifions pas les types de données abstraits (Integer, Real, etc), mais les unités de mesure utilisées pour exprimer les valeurs des entées/sorties dans l'ontologie du domaine. Dans l'exemple de la Figure 2, l'unité de mesure de la sortie *Price* est définie par le concept **Price** dans l'ontologie de la Figure 3. Ce dernier correspond à un ensemble de monnaies, en l'occurrence le Dollar(USD), le Yen(YEN) et l'Euro(EUR).

Le fait que le concept **Price** contient plusieurs unités de mesure peut sembler inconsistant, et cela par rapport au fait qu'une valeur ne peut avoir qu'une seule unité de mesure à la fois. Cependant, l'annotation avec ce type de concepts (ensembles d'unités de mesure) sert seulement à un appariement partiel qui détermine les services *susceptibles* de satisfaire la requête. Il y'a une deuxième phase d'appariement où seuls les services pouvant *certainement* répondre seront sélectionnés. Dans la cas où ces derniers n'existent pas dans l'annuaire, c'est un des services sélectionnés pendant la première phase qui sera transformé afin de pouvoir s'apparier avec la requête.

Price = **Money**
Money = (and Real (all in-currency aset(USD, EUR, YEN)))
Euro = (and Real (all in-currency aset(EUR)))
Yen = (and Real (all in-currency aset(YEN)))
Dollar = (and Real (all in-currency aset(USD)))
Description = **Language**
Language = (and String (all in-currency aset(English,French, Japanese)))
Japanese = (and String (all in-currency aset(japanese)
French = (and String (all in-currency aset(French)

Figure 3. *Définition terminologique de Concepts dans le langage ITL*

4.1.2. *Le Système à Contraintes (SC)*

Le *SC* permet de vérifier la consistance des services découverts pendant l'appariement des *SST*. En d'autre termes, la sélection des services qui puissent certainement satisfaire la requête du client. En effet, la compatibilité des *SST* des spécifications d'une offre et d'une demande de services ne suffit pas pour conclure qu'il ya similarité de services. Il faut aussi que leurs contraintes ne soient pas contradictoires. Le *SC* permet la représentation de deux sortes de contraintes : les contraintes sur les valeurs des entrées/sorties et les contraintes sur leurs types dans l'ontologie du domaine.

Le SC est défini par le quadruplet $(\mathcal{I}_{ct}, \mathcal{O}_{ct}, \mathcal{I}_{cv}, \mathcal{O}_{cv})$ où les éléments sont des ensembles de contraintes sur les types des entrées, types des sorties, valeurs des entrées, valeurs des sorties respectivement.

Notre intérêt principal étant la personnalisation de services par transformation des offres de services en fonction des préférences des clients, nous mettons l'accent sur les contraintes de typage qui permettent de préciser l'unité de mesure de la valeur d'une entrée/sortie dans la spécification d'une demande ou d'une offre de services.

Les contraintes sur les types des entrées/sorties des services peuvent tout simplement être considérées comme des **spécialisations** des concepts utilisés lors de l'annotation sémantique des *SST*. Il s'agit de préciser par une seule unité de mesure chaque entrée/sortie annotée par un ensemble d'unités – concept extensionnel dans le *SST*. L'exemple ci-dessous illustre cette situation.

D'après la Figure 3, le concept **price** est équivalent au concept **Money**. Ce dernier peut renfermer plusieurs types de monnaies et il est utilisé pour l'annotation sémantique de l'attribut *price* dans le *SST* d'une offre ou d'une demande de services. Si dans la Figure 2 le concept **price** est utilisé pour l'annotation sémantique de l'attribut *price* dans les *SST* d'une demande et d'une offre de services, une précision doit être apportée dans les *SC* si un fournisseur veut offrir (un demandeur exige) son service dans un type de monnaie particulier. Ainsi, un concept moins général que le concept **Money** peut être assigné au concept **Price**, il s'agit par exemple du concept **EURO** si un utilisateur veut spécifier le prix de son service en Euro.

Dans le même exemple, le concept **French** peut être assigné au concept **Description** si un fournisseur veut offrir la description de son service en Français. La Figure 4 montre deux contraintes de typage qu'un client ou un fournisseur peuvent spécifier dans les *SC* correspondants au *SST* de la Figure 2. Avec de telles contraintes, les clients et les fournisseurs peuvent offrir les précisions nécessaires sur les valeurs des entrées/sorties des services dans une seule unité de mesure.

4.2. *Le processus d'appariement de services Web*

Dans le système précédemment présenté, l'appariement entre une requête et un service se déroule en deux étapes :

price = **EURO**
Description = **French**

Figure 4. *Contraintes de typage*

Durant la première phase, l'annuaire compare les spécifications des demandes et des offres de services. Il compare syntaxiquement les mots-clés décrivant, respectivement, le triplet ($\mathcal{C}, \mathcal{I}, \mathcal{O}$), et les triplets (C', I', O')$_s$ de chaque offre de services disponible dans l'annuaire, et sémantiquement les concepts qui leur sont éventuellement rattachés. Le résultat de cette phase est un ensemble de services susceptibles de satisfaire la requête du client.

La deuxième phase concerne la comparaison du *SC* de la demande de services avec les *SC* des offres de services sélectionnées dans l'étape précédente. Cette phase comporte également deux étapes : d'abord la comparaison des contraintes de typage des entrées/sorties, ensuite celle des contraintes sur leurs valeurs. Nous illustrons dans ce qui suit le processus d'appariement des contraintes de typage des entrées/sorties de services. Pour l'appariement des contraintes sur les valeurs des entrées/sorties, la résolution peut s'effectuer par des algorithmes de satisfaction de contraintes [LIU 04].

Le processus d'appariement des contraintes de typage des entrées/sorties de services concerne la demande de services et l'ensemble des services sélectionnés lors de l'appariement des *SST*. Si toutes les entrées/sorties d'une demande et d'une offre de services sont de types similaires, alors il n'y a pas de conflit et le processus d'appariement continue avec leurs contraintes sur les valeurs. Si au contraire, au moins une entrée/sortie possède deux unités de mesures différentes dans la demande et dans l'offre de services, alors un conflit de typage apparaît. Nous définissons dans ce qui suit la notion de conflit après l'introduction de la notion d'axiome de couverture.

Définition 1 ***(Axiome de couverture)***
Soientt $A_1, A_2, ..., A_n$ un ensemble de concepts formels.
Un axiome de couverture est une assertion de la forme A :=$A_1 \vee A_2 \vee ... \vee A_n$, et qui signifie que les concepts $A_1, A_2, ..., A_n$ sont tous subsumés par le concept A.

Les axiomes de couverture représentent des connaissances qui permettent à l'annuaire de faire la différence entre les concepts pouvant causer des conflits de typage et les autres concepts. Un axiome est associé à chaque concept défini en extension et pouvant provoquer des conflits dans l'ontologie du domaine. Dans l'exemple de la Figure 2, le concept **Price** (équivalent au concept **Money**) peut constituer la tête d'un axiome de couverture, car le prix d'un produit peut être spécifié dans plusieurs monnaies différentes. Ainsi, nous aurons l'axiome suivant :
Money := EURO $\vee$ YEN$\vee$ ... $\vee$ DOLLARS

Définition 2 ***(Conflit de typage)***
Soient $\mathcal{C}$ un ensemble de concepts décrits dans l'ontologie d'un domaine d'application, x, y, z trois concepts définis en extension dans l'ensemble $\mathcal{C}$, et les deux contraintes :
$x = y$ (Contrainte de typage d'une demande de services)
$x = z$ (Contrainte de typage d'une offre de services)
Si $y \neq z \wedge (\exists\, c \in \mathcal{C} \mid y \sqsubseteq c \wedge z \sqsubseteq c)$, et s'il existe un axiome de couverture $c := y \vee ... \vee z$, alors il y a un conflit dû à la différence entre les unités de mesure de l'entrée/sortie x dans la demande et dans l'offre de services.

La sémantique de la résolution de conflits de typage que nous proposons est la transformation de services Web : de la configuration disponible dans l'annuaire vers celle exigée par le client. C'est ce que nous présentons dans la section suivante.

4.3. *Processus de personnalisation*

L'appariement entre une demande et une offre de services peut révéler plusieurs conflits de typage entre leurs entrées/sorties. Nous utilisons **la composition de services Web** comme un moyen de résolution de conflits. Les mot-clés des services à appeler pour résoudre un conflit entre les demandes et les offres de services sont récupérés à partir d'un ensemble de règles appelées *règles d'association de contexte*.

Définition 3 ***(Règle d'association de contexte)*** *Une règle d'association de contexte est un symbole de prédicat à deux arguments* ***ConflictResolution(****Concept,Context****)****. "Concept" est une variable qui représente des concepts définis en extension dans une ontologie du domaine, et appartenant à la tête d'un axiome de couverture. "Context" est une variable destinée à recevoir le contexte du service de résolution de conflits.*

Un conflit de typage est induit par la différence entre deux concepts subsumés tous les deux par le concept figurant dans le premier argument du prédicat "***ConflictResolution***". Les deux concepts en conflit sont récupérés à partir des concepts d'annotation d'une même entrée/sortie dans une demande et dans une offre de services. Ainsi, dans chaque ontologie, les concepts représentant un ensemble d'unités de mesure pouvant provoquer des conflits doivent être munis de règles d'association de contexte.

Exemple 2 *Les règles de la Figure 5 sont des règles d'association de contexte rattachées à l'ontologie de la Figure 3. La première règle relie le concept* ***Money*** *au mot-clé (context) du service de résolution du conflit sur la monnaie : $ConversionMoney$. La seconde relie le concept* ***Language*** *au mot-clé du service de résolution du conflit sur la langue de présentation de la description du livre : Translation.*

Le contexte du service à appeler pour résoudre un conflit de typage d'une entrée/sortie s'extrait en explorant les règles d'association de contexte. En effet, le prédicat *"ConflictResolution"* ayant comme premier argument le concept posant le conflit

ConflictResolution(Money, ConversionMoney**)**
ConflictResolution(Language, Translation**)**

Figure 5. *Règles d'association de contexte*

et une variable désignant le nom du service de résolution de conflits comme deuxième argument, sera émis à l'ensemble des règles d'association de contexte.

Le contexte du service de résolution du conflit est déterminé par la substitution de la variable *Context* par un mot-clé (context) figurant dans une des règles d'association de contexte, et cela à travers le principe de l'unification des termes [ROB 65]. Quand à ses entrées/sorties, deux cas sont à distinguer. Si le conflit est sur une entrée, son entrée est l'entrée du services, et sa sortie est l'entrée de la requête. Si le conflit est sur une sortie, l'entrée du service de résolution de conflits est la sortie de l'offre de services, et sa sortie est la sortie de la demande de services.

Après la résolution des conflits, les valeurs des entrées/sorties des demandes et des offres de services seront dans une même unité de mesure. Ainsi, le processus d'appariement passe à la vérification de la non contradiction des contraintes sur les valeurs des entrées/sorties de l'offre et de la demande de services initialement en conflit. L'appariement des contraintes sur les valeurs permet la sélection des services pouvant, effectivement, répondre à la requête du client (voir l'algorithme, un peu plus loin).

5. Conclusion et perspectives

Pour pouvoir offrir un service dans un marché de plus en plus globalisant, les fournisseurs de services doivent varier leurs offres de services en fonction des besoins des clients. Ils doivent passer de services destinés pour la grande masse de clients [UME 00], à des services personnalisables. Ce nouveau paradigme, appelé *Mass Customizing Paradigm*, assure une offre flexible de services, assurant ainsi l'accessibilité à un nombre plus important de demandeurs de services [DEW 00].

Nous avons adopté ce paradigme aux services offerts via le Web dont l'intérêt et le nombre ne cesse d'augmenter. Nous avons proposé une structure de spécification de services Web sémantiques proche du langage LARKS. Cette nouvelle structure permet la tenue en compte de l'expression des unités de mesure des entrées/sorties des demandes et des offres de services. L'appariement de services dans cette structure s'effectue à deux niveaux : d'abord, la découverte d'un ensemble de services susceptibles de répondre à la requête du client, puis la sélection du service qui puisse, effectivement, y répondre. Dans le cas où ce dernier n'existe pas, un mécanisme est offert pour la transformation d'un des services existants dans l'annuaire (sélectionné pendant le premier niveau) vers le service exigé par le client de services.

La prochaine étape dans de ce travail est l'implémentation du système de transformation de services et sa mise en œuvre dans un annuaire de services pour servir de

Algorithm Personnalisation de services Web

Phase 1 : *Appariement des Systèmes Structurels Typés*

1) Chercher un ensemble $\mathcal{S}$ d'offres de services dont le contexte et les entrées/sorties sont similaires à ceux de la demande de services $\mathcal{Q}$.

2) Si l'ensemble $\mathcal{S}$ est vide alors le processus d'appariement échoue, et aucun résultat ne peut être retourné pour la requête du client, sinon passer à la phase 2.

Phase 2 : *Appariement des Systèmes à Contraintes*

Pour chaque offre de services s de l'ensemble $\mathcal{S}$ faire :

A. Détecter les entrées/sorties dont les types sont en conflit avec les entrées/sorties de la demande de services. Si aucun conflit n'est détecté alors passer à l'étape (c), sinon passer à l'étape (b).

B. Chercher un service de résolution de conflits pour chaque incompatibilité de typage d'une entrée/sortie dans l'offre et dans la demande de services. Deux cas sont à distinguer, selon que le conflit concerne une entrée ou une sortie.

α.**SI(** ConceptAnnotate(I_Q)[7] = C1
$\wedge$ $ConceptAnnotate(I_S))$[8] $= C2$ // C1 $\neq$ C2
$\wedge$ $(\exists\, C \mid C1 \sqsubseteq C \wedge C2 \sqsubseteq C$ // C : subsumant immédiat de C1 et C2
$\wedge$ $C := C_1 \vee C_2 \vee ...$ // Axiome de couverture)
Alors Trouver le service dont la structure est :

Context	ConflictResolution(C,context)
$\mathcal{I}$	$\mathcal{I}_s$
$\mathcal{O}$	$\mathcal{I}_Q$
$\mathcal{I}_{ct}$	$\mathcal{I}_{ct(s)}$
$\mathcal{O}_{ct}$	$\mathcal{I}_{ct(Q)}$

β.**SI(** ConceptAnnotate(O_Q)[9]= C1
$\wedge$ $ConceptAnnotate(O_S))$[10]$= C2$ // C1 $\neq$ C2
$\wedge$ $(\exists\, C \mid C1 \sqsubseteq C \wedge C2 \sqsubseteq C$ // C : subsumant immédiat de C1 et C2
$\wedge$ $C := C_1 \vee C_2 \vee ...$ // Axiome de couverture)
Alors Trouver le service dont la structure est :

Context	ConflictResolution(C,context)
$\mathcal{I}$	$\mathcal{O}_s$
$\mathcal{O}$	$\mathcal{O}_Q$
$\mathcal{I}_{ct}$	$\mathcal{O}_{ct(s)}$
$\mathcal{O}_{ct}$	$\mathcal{O}_{ct(Q)}$

C. Évaluer les contraintes sur les valeurs des entrées/sorties. Si elles ne sont pas en contradiction, alors le service peut répondre à la requête, sinon le service n'est pas convenable comme réponse à la requête.

système de personnalisation. Aussi, il peut jouer le rôle de système de substitution de services, si au sein d'une combinaison de services, les sorties d'un services i ne sont pas compatibles avec les entrées d'un service $i + 1$. La combinaison de services ne

va pas ainsi échouer à cause d'une simple incompatibilité entre deux services qui se suivent dans la composition.

6. Bibliographie

[BER 01] BERNERS-LEE T., HENDLER J., LASSILA O., Eds., *The Semantic Web*, May, 2001.

[BOR 89] BORGIDA A., BRACHMAN R. J., MCGUINNESS D. L., RESNICK L. A., « CLASSIC : A Structural Data Model for Objects. », *SIGMOD Conference*, 1989, p. 58-67.

[BUR 02] BURSTEIN M. H., HOBBS J. R., LASSILA O., MARTIN D. L., MCDERMOTT D. V., MCILRAITH S. A., NARAYANAN S., PAOLUCCI M., PAYNE T. R., SYCARA K. P., « DAML-S : Web Service Description for the Semantic Web. », Horrocks, Hendler [HOR 02a], p. 348-363.

[DEW 00] DEWAN R. M., JING B., SEIDMANN A., « Achieving first-mover advantage through product customization on the Internet. », *ICIS*, 2000, p. 272-285.

[GEN 91] GENESERETH M. R., « Knowledge Interchange Format. », *KR*, 1991, p. 599-600.

[GUA 91] GUARINO N., « A Concise Presentation of ITL. », *PDK*, 1991, p. 141-160.

[HOR 02a] HORROCKS I., HENDLER J. A., Eds., *The Semantic Web - ISWC 2002, First International Semantic Web Conference, Sardinia, Italy, June 9-12, 2002, Proceedings*, vol. 2342 de *Lecture Notes in Computer Science*, Springer, 2002.

[HOR 02b] HORROCKS I., PATEL-SCHNEIDER P. F., VAN HARMELEN F., « Reviewing the Design of DAML+OIL : An Ontology Language for the Semantic Web. », *AAAI/IAAI*, 2002, p. 792-797.

[KOV 02] KOVSE J., HARDER T., RITTER N., « Supporting Mass Customomization by Generating Adjusted Repositories for Product Configuration. », *CAD*, 2002, p. 17-26.

[LIU 04] LIU C., FOSTER I. T., « A Constraint Language Approach to Matchmaking. », *RIDE*, 2004, p. 7-14.

[MAC 91] MACGREGOR R. M., « Inside the LOOM Description Classifier. », *SIGART Bulletin*, vol. 2, n° 3, 1991, p. 88-92.

[MCI 02] MCILRAITH S. A., SON T. C., « Adapting Golog for Composition of Semantic Web Services. », *KR*, 2002, p. 482-496.

[ROB 65] ROBINSON J. A., *A machine oriented logic based on the resolution principle*, J.ACM, 12(1) :23-41, 1965.

[SYC 02] SYCARA K. P., WIDOFF S., KLUSCH M., LU J., « Larks : Dynamic Matchmaking Among Heterogeneous Software Agents in Cyberspace. », *Autonomous Agents and Multi-Agent Systems*, vol. 5, n° 2, 2002, p. 173-203.

[UME 00] UMEDA Y., NONOMURA A., TOMIYAMA T., « Study on life-cycle design for the post mass production paradigm. », *AI EDAM*, vol. 14, n° 2, 2000, p. 149-161.

7. Concept de description sémantique de l'entrée de la demande de services
8. Concept de description sémantique de l'entrée de l'offre de services
9. Concept de description sémantique de la sortie de la demande de services
10. Concept de description sémantique de la sortie de l'offre de services