République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Abderrahmane MIRA de BEJAÏA

Faculté des Sciences Exactes

Département d'Informatique

Mémoire de fin d'études en vue d'obtention du diplôme de Master II

En Informatique Option : Administration et Sécurité des Réseaux

Thème

Composition de Services Web Sémantique à base de Médiateur

Réalisé par :

M[r]. MEHENI Fares

M[r]. LALAOUI Nassim

Encadré par:

M[r] . SALHI Nadir

Devant le jury composé de :

Président: M[r]. OMAR Mawloud

Examinateur : M[r]. HAMOUMA Moumen

Examinateur : M[r]. ABBACHE Bournane

Remerciement

Nous tenons tout d'abord à remercier DIEU le tout puissant pour nous avoir donnés La force de réaliser ce travail, et aussi nos parents pour leurs soutiens et encouragements durant nos années d'études.

Nous remercions profondément notre encadreur Mr. SALHI Nadir pour son aide, ses encouragements et ses critiques constructives qui nous ont beaucoup aidés à apprécier ce travail.

Un remerciement spécial est adressés aussi à nos Co-encadreurs Mr. Mire fodil et Mr. farah pour leurs conseils et pour les nombreux éclaircissements qu'ils ont bien voulu nous apporter durant tout ce projet.

Nous remercions également les membres du jury qui ont accepté de participer à la discussion de notre travail. Nous exprimons aussi toute notre gratitude aux enseignants du département d'informatique ainsi que tous nos collègues et nos amis et à tous ceux qui ont contribué de prés ou de loin à la réalisation de notre mémoire.

Dédicace

A ma mère, et à mon père

A mes frères et sœurs

A mes amis et à mes collègues (Mouhou, Samir, Djafer,Adel…)

A tous ceux qui me sont chers, …

Chapitre. III: La médiation de services Web.

Chapitre. IV: Conception.

Introduction Générale

Introduction générale :

L'évolution d'internet et la compétitivité entre les entreprises ont été le facteur de l'explosion des services disponible sur le web et ils ont permet le développement de nouveaux Paradigmes d'interaction entre applications. Notamment, l'architecture orientée service (Service-Oriented Architecture, ou SOA) a été mise en avant afin de permettre des interactions entre applications distantes. Cette architecture est construite autour de la notion de service, qui est matérialisé par un composant logiciel assurant une fonctionnalité particulière et accessible via son interface. Un service est toujours accompagné d'une description fournissant aux applications les informations nécessaires à son utilisation.

Ces services permettent d'augmenter la vitesse du marché et de toucher aux nouveaux clients. Il est possible de nos jours, d'obtenir des devis d'un voyage, d'acheter un billet de train, de louer une voiture ou de faire ces courses, tout cela via internet. Ce sont effectivement des services que chaque entreprise concerné met à disposition de ces clients. Les services web ont été créés pour faciliter ces interactions et peuvent être définis comme des programme modulaires, ces services peuvent être découverts et invoqués via internet ou un intranet, indépendamment des plates formes et des langages utilisés.

Les tendances actuelle de la communauté des chercheures est d'exploiter les technologies du web sémantique, afin d'enrichir les services web de description sémantique. Le web sémantique est un projet qui a été créé pour la modélisation des données hétérogènes, par contre la modélisation des programmes disponible sur le web, c.à.d. les services web constitue un nouveau projet relativement récent (le web sémantique), ces derniers sont dotés de descriptions sémantique en se basant totalement sur les langages du web sémantique, en particulier les ontologies. Ils constituent la nouvelle génération pour des technologies du web. Pour l'intégration d'application distribuées et hétérogène. Ils sont réserver pour l'automatisation des différentes taches d'utilisation comme : La publication, la découverte, l'invocation, et la composition.

La composition dynamique de service consiste à combiner les composants existant afin de créer des nouvelles fonctionnalités qui ne sont pas directement dans les services de base. Il s'agit en faite, d'agréer un ensemble de fonctionnalités élémentaire dans des services de haut niveau, proche des spécifications de l'utilisateur. La composition des services est ainsi un moyen de jonction entre les fonctionnalités élémentaire et les besoins des utilisateurs. Cependant, devant la propagation du nombre de service disponible, le problème pour combiner plusieurs services

appropriés pour atteindre un certain objectif, devient une tache très délicate. A cet effet, les informations de la médiation sont considérer comme un élément important à prendre en compte lors du processus de description, découverte, composition. Plusieurs approche ont été proposé pour résoudre ces problème (a base de contexte, a base d'automates…etc.).

C'est dans cette optique que s'inscrivent les travaux de recherche présentés dans ce mémoire de master. Il s'agit de modéliser le processus métier et essayer de composer ses différents services a base d'une architecture de médiation et SOA, et dans le but de satisfaire au mieux l'utilisateur nous proposons alors un modèle de composition de services tout en résolvant le problème d'hétérogénéité.

- **Objectifs**

Notre contribution consiste en plusieurs points qui sont rédigé comme suit :

1. Modélisation d'un processus métier (exemple ifri).
2. Modélisation d'un scénario de transaction en ligne qui fait appel à toi services.
3. Adaptation du processus métier à notre architecture de médiation.
4. Utilisation d'une architecture orienté service pour résoudre le problème d'hétérogénéité et composé plusieurs services Web.
5. En s'appuyant sur l'algorithme de contextualisation du générateur de médiation, adapté à notre prototype illustratif pour générer le médiateur de composition et résoudre le problème d'hétérogénéité.
6. On a proposé un fragment d'une annotation « B_log » qui identifie son concept, son objet sémantique et son contexte.
7. On a utilisé des ontologies contextuelles et de domaine pour notre prototype illustratif.

Chapitre I: SOA et Service Web

1. Introduction

La diversité des applications d'un même système d'information, et la nécessité de manipuler des quantités de plus en plus importantes de données, posent un grand problème d'intégration d'applications. En effet, la communication entre les applications et les différents systèmes d'information devient de plus en plus difficile à cause des technologies hétérogènes utilisées qui amènent les utilisateurs à travailler dans un environnement incohérent, mal adapté et incompatible. Le manque d'interopérabilité est alors relevé comme principal problème de ces systèmes.

Dans ce contexte, et afin de résoudre ces problèmes, le concept d'EAI est apparu comme solution. Cependant, le monde d'EAI est très vaste car il englobe plusieurs approches, techniques et technologies d'intégrations qui ont évolué dans le temps depuis que le concept d'EAI a vu le jour. Actuellement, les solutions EAI de la deuxième génération sont orientées vers les processus métiers et les échanges interentreprises de sorte qu'elles prennent en compte les nouveaux modèles économiques créés et promus par Internet et ses technologies (TCP/IP, SMTP, HTTP, FTP, XML, etc.). De plus, les progiciels de gestion intégrés de la deuxième génération apportent une quantité de nouveaux modules organisés autour d'une nouvelle vision pour les systèmes d'information. Le but est de créer un nouveau modèle de collaboration, offrant une interopérabilité entre les systèmes d'information ; c'est là l'objectif de l'architecture SOA considérée, actuellement, comme la solution adéquate aux problèmes d'intégration des systèmes d'information. Cette architecture est urbanisée sous la forme de services réutilisables qu'il est possible de découvrir et composer dynamiquement, avec un couplage faible.

Pour répondre aux enjeux de la SOA, les Web services constituent une des technologies actuelles la mieux placée pour mettre en place l'architecture orientée services. Les Web services, qui sont basés sur des technologies Web dérivées du fameux standard XML, présentent de nombreux atouts pour faire communiquer des systèmes caractérisés par une hétérogénéité croissante. Ils permettent également de mettre en œuvre des services applicatifs partagés et de gérer la connectivité aux données. Ils sont devenus la technologie la plus utilisée pour l'interopérabilité et l'intégration des applications et des systèmes d'information.

Dans cette perception, la composition des Web services est devenue une solution adéquate pour implémenter les processus métiers fortement distribués et pour répondre, d'une manière efficace et rapide, aux besoins aux requêtes des internautes.

Dans ce chapitre nous présentons l'architecture orientée services et ses différentes caractéristiques. Nous exposons aussi la notion de Web services et ses concepts adjacents: technologies de base et composition.

2. Caractéristiques de l'Architecture SOA

2.1 .Historique et définition

L'architecture logicielle est une pratique relativement nouvelle dans le domaine du génie logiciel. Elle décrit les composants du système et la manière dont ces éléments interagissent à un niveau élevé. Les interactions entre les composants s'appellent les connecteurs, la configuration des composants et les connecteurs fournissent une vue structurale et comportementale du système **[1]**.

2.1.1. Histoire de la SOA

SOA n'est pas une solution, c'est une pratique. Le concept SOA a été inventé la première fois par l'analyste Gartner Yefim V. Natis en 1994 **[2]**. Selon Natis: «*SOA est une architecture de logiciel qui débute avec une définition d'interface et la construction entière d'application de topologie comme topologie des interfaces, des réalisations d'interface, et des appels d'interface…*» **[3]**.

2.1.2. Définition

SOA est un style architectural qui permet de construire des solutions d'entreprises basées sur les services. Ces derniers rassemblent les grandes applications de l'entreprise (dites applications composites) en services interopérables et réutilisables **[4], [5]**.

Le service un composant logiciel exécuté par un fournisseur (Provider) à l'attention d'un client (Requester). Cependant, l'interaction entre un client et un fournisseur est matérialisée grâce à un protocole responsable de l'échange des messages entre les deux entités.

L'objectif d'une architecture orientée services est donc de décomposer les fonctionnalités d'un système ou d'une application en un ensemble de fonctions basiques, appelées services, implémentés par des composants, et de décrire les différentes interactions possibles entre ces services. L'objectif de cette opération est de créer une architecture logicielle globale décomposée en services correspondant aux processus métiers de l'entreprise. De ce fait, les applications d'un système d'information seront vues comme une collection de services qui interagissent et communiquent entre eux. Cette communication peut consister en un simple retour de données ou en une activité plus complexe (coordination de plusieurs services).

Cependant, l'aspect le plus important de l'architecture SOA est qu'elle permet de séparer l'implémentation du service de son interface. C'est cette caractéristique qui assure le haut degré

d’interopérabilité visé par cette architecture.

2.2. Concepts de SOA

SOA est plus qu’un ensemble de technologies. Elle n'est directement liée à aucune technologie, bien qu’elle soit le plus souvent mise en application avec des Web Services qui sont considérés comme la technologie la plus appropriée pour la réalisation de SOA. Cependant, l’utilisation des Web Services n'est pas proportionnée pour construire SOA. Nous devons employer les Web services selon les concepts que SOA définit **[6]**. Les concepts de SOA les plus importants sont :

- Services
- Interfaces Self-describing (description d'individu)
- Échange des messages
- Soutien de liaison synchrone et asynchrone
- Accouplement faible
- Enregistrement de service
- Qualité du service
- Composition des services dans des processus d'affaires

2.3. Composants de la SOA

Afin de déduire les composants de bases de l’architecture SOA, il est nécessaire de présenter en premier lieu son paradigme de fonctionnement. Le paradigme "découvrir, interagir et exécuter" comme montré dans la figure 1.1 permet au consommateur du service (client) d’interroger un annuaire pour le service qui répond à ses critères. Si l’annuaire possède un tel service, alors il renvoie au client le contrat du service voulu ainsi que son adresse. SOA consiste en quatre entités configurées ensemble pour supporter le paradigme découvrir, interagir et exécuter **[7]**.

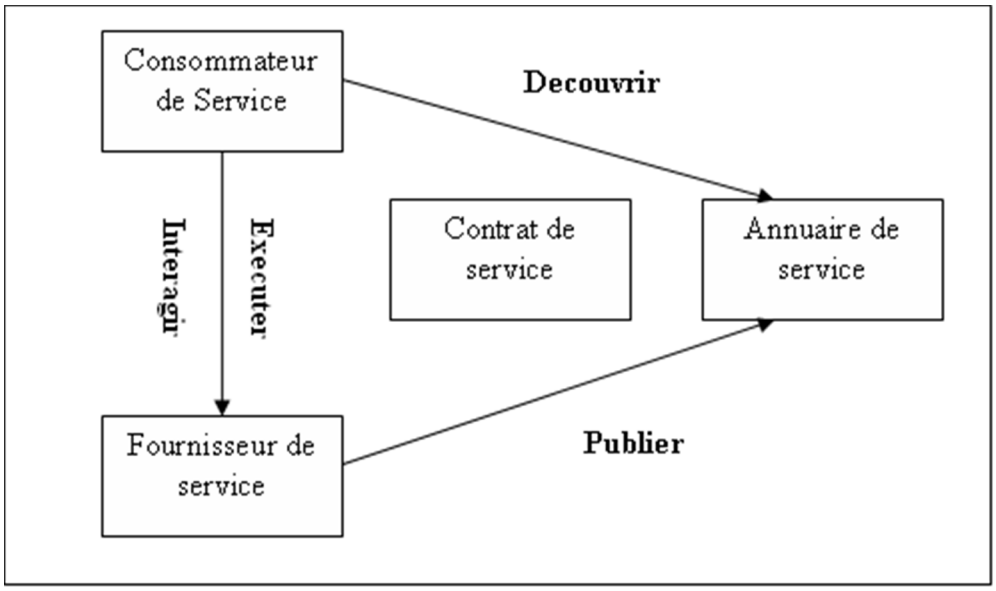


Figure 1.1. Paradigme "découvrir, interagir et exécuter".

Suivant ce protocole de fonctionnement, nous pouvons présenter les composants de l'architecture SOA comme suit **[7]**:

- **Le consommateur de service :** Le consommateur de service est une application qui requière un service. C'est l'entité qui initie la localisation du service dans l'annuaire, interagit avec le service à travers un protocole et exécute la fonction exposée par le service.
- **Le fournisseur de service :** Le fournisseur de service est une entité adressable via un réseau, il accepte et exécute les requêtes venant d'un client. Le fournisseur de service publie le contrat de service dans l'annuaire pour qu'il puisse être accédé par les clients.
- **L'annuaire de service :** L'annuaire de service est un annuaire qui contient les services disponibles. C'est une entité qui accepte et sauvegarde les contrats du fournisseur de service et présente ces contrats aux éventuels clients.
- **Le contrat de service :** Le contrat spécifie la manière dont le client de service va interagir avec le fournisseur de service. Il spécifie le format de la requête et la réponse du service.

3. Web services

D'après la définition, SOA est une approche architecturale qui ne fait aucune hypothèse sur la technologie de mise en œuvre. En particulier, l'amalgame souvent faite entre SOA et les Web services est une erreur **[8]**.

Cependant, la conception des spécifications Web services a été menée dans l'objectif de répondre au mieux aux enjeux de l'architecture SOA **[8]**. Les Web services fournissent les bases technologiques nécessaires à la réalisation de l'interopérabilité entre les applications en utilisant différentes plateformes, différents systèmes d'exploitation et différents langages de programmation **[9]**.

3.1. Définition

Un Web service est un composant logiciel identifié par une URI, qui possède une interface publiable. Cette dernière peut être découverte par d'autres systèmes, qu'ils peuvent interagir avec le Web service selon les règles prescrite par sa description, en utilisant des messages basés sur XML et portés par des protocoles standards d'internet **[13]**.

Les Web Services fournissent une couche d'abstraction entre le client et le fournisseur d'un service. Cette couche est indépendante de la plateforme et du langage d'implémentation **[10]**, grâce à un ensemble de protocoles standardisés comme SOAP (Simple Object Access Protocol), WSDL (Web Service Description Language) et UDDI (Universal Description, Discovery and Integration) **[13]**.

Cette définition n'implique que les Web services :

- sont des composants d'application ;
- communiquent en utilisant des protocoles standards ;
- sont autonomes et auto- descriptifs ;
- peuvent être découverts ;
- peuvent être utilisés par d'autres applications ;
- s'appuient sur XML (Extensible Markup Language) ; et enfin
- sont extensibles : chacun peut adjoindre ses propres données, protocoles ou mécanismes propriétaires.

3.2. Fonctionnement des Web services

Dans sa première génération, le fonctionnement des Web Services repose sur trois couches fondamentales présentées comme suit:

- **Invocation :** visant à établir la communication entre le client et le fournisseur en décrivant la structure des messages échangés.

- **Découverte :** permettant de localiser un Web service particulier dans un annuaire de services décrivant les fournisseurs ainsi les services fournis.
- **Description :** dont l'objectif est la description des interfaces des Web services (paramètres des fonctions, types de données).

Pour assurer ces fonctionnalités, trois technologies de base ont été proposées et disposées en couches pour construire l'architecture de base des Web Services, comme le montre dans le tableau suivant :

UDDI	Découverte de services
WSDL	Description de services
SOAP	Communications
XML	

Tableau 1.1 : Couches technologiques des Web Services.

Les couches XML et SOAP sont les couches de plus bas niveau, elles permettent le transport de l'information alors que WSDL permet de décrire le service aux utilisateurs externes. Enfin la couche la plus haute UDDI décrit ce que peut produire le service, c'est la couche la plus sémantique **[11]**.

Ces technologies sont présentées dans la section suivante.

3.3. Infrastructure des Web services

3.3.1. XML

XML constitue la technologie de base des architectures Web services ; c'est un facteur important pour contourner les barrières techniques. XML est un standard qui permet de décrire des documents structurés transportables sur les protocoles d'Internet. En effet, il apporte à l'architecture des Web services l'extensibilité et la neutralité vis à vis des plateformes et des langages de développement.

La technologie des Web services a été conçue pour fonctionner dans des environnements totalement hétérogènes. Cependant, l'interopérabilité entre les systèmes hétérogènes demande des mécanismes puissants de correspondance et de gestion des types de données des messages entre les différents participants (clients et fournisseurs). C'est une tâche où les schémas de type de données XML s'avèrent bien adaptés. C'est pour cette raison que la technologie des Web

services est essentiellement basée sur XML ainsi que les différentes spécifications qui tournent autour (les espaces de nom, les schémas XML, et les schémas de Type) **[12]**.

3.3.2. Communication avec SOAP

SOAP est un protocole basé sur XML qui permet l'échange léger de données structurées entre des applications exécutées sur différents systèmes d'exploitation, avec différentes technologies et différents langages de programmation. Il peut être employé dans tous les styles de communication : synchrones ou asynchrones, point à point ou multipoints. Il se contente d'offrir la possibilité de structurer des messages destinés à des objectifs particuliers. Néanmoins, il est particulièrement utile pour exécuter des dialogues
requête-réponse RPC (Remote Procedure Call) **[10] [12] [14]** bien que RPC présente un problème de compatibilité et de sécurité de sorte que les pare-feux et les serveurs proxy vont normalement bloquer ce genre de trafic. Une meilleure façon de communiquer entre les applications est en utilisant http qui est accepté par tous les navigateurs Web ainsi que tous les serveurs. SOAP a été principalement créé pour atteindre ce but.

Un message SOAP est un document XML ordinaire qui contient les éléments suivants:

- L'élément *Envelope* qui identifie le document XML comme étant un message SOAP
  L'élément *Header* qui est optionnel et qui contient des informations d'entête
- L'élément *Body* qui contient l'appel ainsi que la réponse retournée
- L'élément *Fault* qui est optionnel et qui fournit des informations sur d'éventuelles erreurs survenues lors de l'analyse du message

Tous ces éléments cité ci-dessus sont déclares dans les *namespace* de l'enveloppe SOAP :
"http://www. w3.org/2001/12/soap-envelop"
Et le *namespace* pour le SOAP encoding et les types de données :
"http://www.w3.org/2001/12/soap-encoding"

3.3.3 .Description des Web services

Le langage WSDL (Web Service Description Language **[15]**) proposé par Ariba, IBM et Microsoft auprès du W3C est un format de description des Web services fondé sur XML. Il permet de décrire de façon précise les Web services en incluant des détails tels que les protocoles, les serveurs, les ports utilisés, les opérations pouvant être effectuées, les formats des messages d'entrée et de sortie ainsi que les exceptions pouvant être renvoyées. Il permet la représentation d'un Web Service de manière plus abstraite pour la réutilisation **[10].** Il est devenu une recommandation du W3C depuis le 26 Juin 2007.

Un document WSDL contient des informations opérationnelles concernant le service. La définition du service marquée par la balise <definitions> est la racine de tout document WSDL.

Elle peut contenir les attributs précisant le nom du service et les espaces de nommage. Un document WSDL contient les entités suivantes:

- **Types:** précise les types de données complexes, pour lequel WSDL emplois XML Schéma.
- **Message:** l'abstraction décrivant les données échangées entre Web services.
- **Opération:** l'abstraction décrivant une action implémentée par un Web service.
- **Type de port:** un ensemble d'opérations implémenté par une terminaison donnée.
- **Liaison (binding):** un protocole concret d'accès à un port et un format de spécification des messages et des données pour ce port.
- **Port:** un point de terminaison identifié de manière unique par la combinaison d'une adresse Internet et d'une liaison.
- **Web Service:** une collection de ports.

Il est important de mentionner que le document WSDL est divisé en deux parties : l'interface du service et son implémentation. L'interface du service est la partie réutilisable de la définition du service, elle peut être référencée par de multiples implémentations du service. Elle est composée des balises : Types, Message, Opération et Liaison. Alors que la partie implémentation, décrite par les balises Port et Service, est unique et présente une terminaison pour invoquer le Web service. De plus, chaque document WSDL peut être documenté grâce à une balise <documentation> bien que cet élément soit facultatif.

3.3.4. Découverte des Web services

UDDI (Universal Description, Discovery and Integration) est une spécification pour la description et la découverte de Web Services en utilisant XML Schéma. Il convient alors de décrire les quatre constitutions de l'enregistrement d'un Web service: l'identité du fournisseur (pages blanches), la description du ou des services offerts (pages jaunes), l'information sur les modes d'exploitation et les différents modèles de données sous-jacents **[16]**. Ainsi, UDDI se présente comme un ensemble de bases de données utilisées par les entreprises pour enregistrer leurs Web services ou pour localiser d'autres Web services.

Grâce à UDDI, les entreprises peuvent enregistrer des données les concernant, des renseignements sur les services qu'elles offrent et des informations techniques sur le mode d'accès à ces services. Une fois l'enregistrement terminé, les informations sont automatiquement répliquées sur l'ensemble des annuaires. Ce fonctionnement permet aux services d'être découverts par un plus grand nombre d'entreprises.

UDDI offre deux fonctionnalités au sein de l'architecture globale : la publication et la découverte. Elles permettent aux fournisseurs de publier leurs services selon un modèle de description et au client l'interrogation des services. De ce fait, la spécification UDDI constitue une norme pour les annuaires des Web services. Les fournisseurs disposent d'un schéma de description permettant de publier des données concernant leurs activités, la liste des services qu'ils offrent et les détails techniques sur chaque service. De plus, la norme UDDI offre aussi une API aux applications clientes, pour consulter et extraire des données concernant un service et/ou son fournisseur **[12]**.

3.4. Composition des Web services

Dans cette section, nous présentons la composition des Web services et ses concepts adjacents.

3.4.1. Définition

«Les composants sont destinés à être composés» **[17]**. Selon cette définition, la réutilisation est un avantage important de l'approche par composants **[18]**, **[19]**. Par définition, les Web services, tels qu'ils sont présentés, sont des composants conceptuellement limités à des fonctionnalités relativement simples qui sont modélisées par une collection d'opérations services que l'on compose jusqu'à ce que le service résultant fournisse un support entier pour les processus métiers **[12]**. De ce fait, les Web services doivent pouvoir être composés en services que l'on compose jusqu'à ce que le service résultant fournisse un support entier pour les processus métiers.

La composition des Web services a été définie par **[20]** comme étant la capacité d'offrir des services à valeur ajoutée en combinant des services existants probablement offerts par différentes organisations. Techniquement parlant, la composition permet d'assembler des Web services afin d'atteindre un objectif particulier, par l'intermédiaire de primitives de contrôles (boucles, test, traitement d'exception, *etc.*) et d'échange (envoi et réception de messages) **[21]**. Il s'agit en d'autres termes de spécifier les services qui seront invoqués, dans quel ordre et comment gérer les conditions d'exceptions **[22]**.

3.4.2 .Orchestration et Chorégraphie

La composition des Web services peut être faite en utilisant l'une des deux méthodes : orchestration ou chorégraphie.

Dans la technique de l'orchestration, un processus central (qui peut être lui-même un Web service) prend le contrôle de tous les Web services impliqués dans la composition et coordonne l'exécution des différentes opérations sur ces Web services. Ces derniers ne savent

pas (et n'ont pas à savoir) qu'ils sont invoqués dans une composition et qu'ils font partie d'un processus métier complexe. Seul le coordinateur central de l'orchestration le sait. L'orchestration est donc centralisée avec des définitions explicites des opérations et l'ordre d'invocation des Web services **[6].**

D'autre part on peut ne pas compter sur un coordinateur central. Au lieu de cela, chaque Web service impliqué dans la chorégraphie sait exactement quand exécuter ses opérations et avec qui interagir. La chorégraphie est un effort de collaboration focalisé sur l'échange de messages dans des processus métiers publics. Tous les participants à la chorégraphie doivent connaître leurs rôles dans le processus métier, les opérations à exécuter, les messages à échanger, et le temps d'échange de ces messages **[6]**.

Du point de vue de la composition des Web services, pour exécuter des processus métiers, l'orchestration est avantageuse par rapport à la chorégraphie:

- On connaît de façon exacte qui est le responsable de l'exécution du processus métier en entier.
- On peut incorporer les Web services, même ceux qui ne savent pas qui sont impliqués dans des processus métiers.
- On peut aussi fournir un scénario alternatif en cas d'erreurs.

3.4.3. BPEL pour la composition des Web services

Plusieurs initiatives ont été inventées pour automatiser la composition des Web services. Cependant le langage BPEL (Business Process Execution Langage) présente la technique la utilisée par les développeurs des processus métiers **[6]**.

a) Besoin d'un langage spécifique à la composition des Web services

La composition des Web services est définie comme étant un processus métier où les services présentent une collection d'activités qui collaborent pour implémenter ce processus.
Pour une vue extérieure, le processus métier résultant est un Web service (composite) vu comme n'importe quel autre service basique. Par conséquent, les Web services composites doivent être considérés récursivement comme des Web services et doivent garder les mêmes caractéristiques que les services basiques (services WSDL) à savoir auto-descriptifs, interopérables, et facilement intégrables **[12]**.

La mise en place d'un processus métier à travers la composition des Web services nécessite la définition de l'ensemble des services participants ainsi que l'échange des messages entre ces services. Dans ce contexte, les interactions peuvent être sans états, synchrones, asynchrones, comme elles peuvent appartenir à un état qui combine tous ces modes en même temps. De plus,

les compositions complexes de plusieurs Web services consistent souvent en des échanges de messages dans un ordre bien défini. Ainsi que les interactions sont généralement assez longues. D'autre part, un autre aspect important est la capacité de décrire la façon dont les erreurs sont traitées **[6]**.

Pour cette raison, le langage WSDL est insuffisant pour décrire les compositions complexes des Web services. WSDL fournit une description basique et une spécification des messages échangés, mais cette description ne permet que de décrire de simples interactions entre le client et le Web service. Etant donné les limitations de WSDL, il est nécessaire d'avoir un mécanisme pour décrire la composition des Web services en des processus plus complexes. L'utilisation de technologies dédiées à la composition est une chose indispensable **[6]**.

b) Présentation de BPEL

Dans ce contexte, plusieurs initiatives ont vu le jour pour standardiser l'automatisation des processus métiers. BPEL représente le langage le plus accepté dans la communauté des concepteurs de processus métiers modernes.

Microsoft, IBM, et BEA ont développé la première version de BPEL en Aout 2002 et depuis que SAP et Siebel les ont rejoints, beaucoup de modifications et d'améliorations ont été apportées. En Avril 2003, BPEL a été soumis à l'OASIS (Organisation for the Advancement of Structured Information Standards) pour standardisation.

BPEL (connu aussi sous le nom BPEL4WS ou WSBPEL) représente la convergence de deux langages, WSFL (Web Services Flow Languages) et de XLANG. Il combine les deux approches et fournit un vocabulaire riche, basé sur XML, pour la description des processus métiers.

BPEL peut décrire des processus métiers aussi bien simples que complexes. Il offre des instructions basiques comme : < **process>, <invoke>, <receive>, <reply>,** <**variable>,** <**assign>** et **<copy>** ainsi que des instructions structurées permettant de combiner les activités basiques. BPEL propose différentes activités structurées parmi lesquelles on retrouve :

- Définir un ensemble d'activités qui seront invoquées dans une séquence ordonnée en utilisant **<sequence>**
- Définir un ensemble d'activités qui seront invoquées en parallèle en utilisant **<flow>**
- Définir les structures conditionnelles en utilisant l'activité <**if> <else>**

En utilisant cette variété de constructeurs, BPEL nous permet de définir des processus métiers de manière algorithmique. Par conséquent cette définition est relativement simplifiée. De plus, BPEL facilite l'invocation des opérations des Web services, qu'elles soient synchrones ou asynchrones, et fournit un vocabulaire riche pour le traitement des erreurs.

4. Conclusion

Aujourd'hui, la faiblesse des progiciels réside principalement dans la dynamique des systèmes d'information. L'architecture orientée services est une nouvelle vision qui répond d'une manière efficace aux problèmes des systèmes d'information en terme de réutilisabilité et d'interopérabilité.

Une solution SOA est habituellement basée sur les services qui sont habituellement employés pour agir l'un avec l'autre. Le système d'information est alors vu comme une collections de services qu'il est possible de composer pour créer des services avec un plus haut niveau de granularité et permettant d'accéder de manière transparente aux applications du système d'information.

Dans ce chapitre, nous avons présenté l'architecture SOA de manière abstraite, pour ensuite discuter sa concrétisation en utilisant les Web services. Nous avons exposé les différentes technologies adjacentes aux Web services en insistant tout particulièrement sur la composition des Web services comme moyen d'implémentation des processus métiers. A ce stade, nous avons présenté le langage le plus approprié pour composer des Web services à savoir le BPEL.

Chapitre II: Les Service Web sémantique

1. Introduction

Le haut degré d'interopérabilité technique fournie par les Web services instaure le besoin d'introduire la sémantique dans cette technologie afin d'automatiser les différentes phases de leur cycle de vie, tout particulièrement la phase de découverte. La technologie du Web sémantique se présente alors comme une solution pour faciliter la découverte et la manipulation automatique (par ordinateur) des Web services et la naissance du concept des Web services sémantiques comme fruit de la convergence entre les Web services et le Web sémantique. En effet, son ultime objectif est de rendre les Web services plus accessibles à la machine en automatisant les différentes tâches qui facilitent leur utilisation.

Dans ce chapitre nous fournissons les caractéristiques du Web sémantique et la convergence entre ce dernier et les Web services. Ensuite, nous décrivons les différents langages de description sémantique des Web services.

2. Web sémantique

Avant de décrire la notion de Web services sémantiques, il est important de présenter le Web sémantique et ses concepts adjacents.

2.1. Origine

La théorie de l'information distingue fondamentalement trois niveaux pour la représentation du monde : la syntaxe, la sémantique et la pragmatique. La sémantique est généralement utilisée pour compléter la syntaxe qui constitue un élément nécessaire pour pouvoir parler de sémantique.

D'un point de vue informatique, les définitions des notions de sémantique et de syntaxe restent pratiquement similaires à celles pratiquées en linguistique : la syntaxe se réfère au respect de la grammaire formelle d'un langage, alors que la sémantique concerne plutôt l'interprétation des modèles et des symboles informatiques **[23]**. Cependant, le concept « sémantique » a émergé avec l'évolution du World Wide Web au Web sémantique **[24]** où on la considère comme "une forme formelle de représentation des connaissances humaines" **[25]**.

Au cours de la première décennie de son existence, la plupart des informations sur le Web sont conçues uniquement pour la consommation humaine. Les humains peuvent lire les pages Web et les comprendre, mais leurs significations intrinsèques ne sont pas claires pour permettre leur interprétation par des ordinateurs.

De nos jours, ce problème constitue le plus grand obstacle pour fournir un meilleur support aux utilisateurs du Web. Pour le moment, le sens du contenu du Web n'est pas "traitable

par ordinateur", même s'il existe quelques outils pour retrouver du texte et faire des traitements sur ce texte, mais quand il s'agit d'interpréter son sens ou d'essayer d'extraire des informations utiles, la capacité des applications actuelles reste toujours limitée **[26]**.

De ce fait, l'information sur le Web doit être définie de manière qu'elle puisse être utilisée par les ordinateurs, non seulement à des affichages, mais aussi pour l'interopérabilité et l'intégration entre les systèmes et les applications. Une façon de permettre l'échange machine à machine et le traitement automatisé est de fournir les informations de telle sorte que les ordinateurs peuvent comprendre. C'est précisément l'objectif du Web sémantique, pour rendre possible la transformation de l'information par les ordinateurs.

2.2. Définition du Web sémantique

Le « Web sémantique » est un terme inventé par Tim Berners-Lee qui en dit : *« Le Web sémantique est une extension du Web actuel, dans lequel l'information a un sens bien défini, et permet une meilleure coopération dans le travail entre les humains et les ordinateurs…Le Web des données qui peuvent être traitées par les ordinateurs »* **[27]**.

Le Web sémantique est donc une nouvelle approche pour l'organisation du contenu du Web instaurée dans le but d'améliorer l'interopérabilité, la découverte et la récupération de ressources.

Cependant, le Web sémantique n'est pas simplement dédié au World Wide Web. Il représente un ensemble de technologies qui fonctionneront également sur des Intranets d'une entreprise. Ainsi, le Web sémantique résoudra plusieurs problèmes principaux posés à des architectures courantes de technologie de l'information.

2.3. Architecture du Web sémantique

Le Web Sémantique n'est pas seulement une vision de l'avenir du Web. Certaines technologies sont déjà disponibles et figurent dans l'architecture illustrée dans la figure 2.1.

L'architecture repose sur des technologies de structuration de documents fondées sur XML. Sur cette base, des langages de métadonnées sémantiques de haut niveau ont été développés et permettent la description des ressources sur le Web. Afin de fournir un cadre interprétatif à ces métadonnées sémantiques le Web Sémantique utilise des ontologies. Au niveau supérieur, le raisonnement sur les données est assuré par des mécanismes d'inférence, qui permettent d'une part de construire de la connaissance, mais aussi d'en maintenir la cohérence.

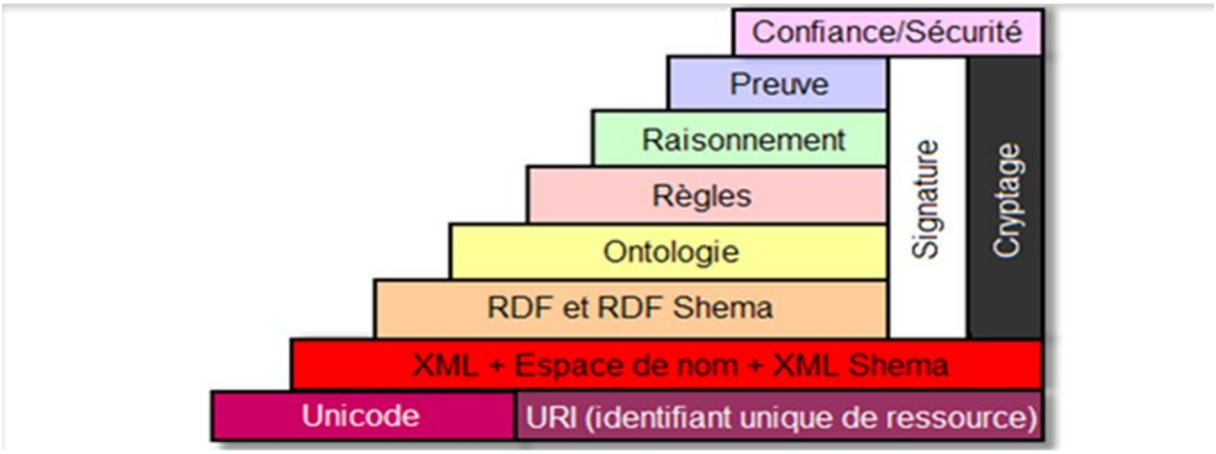


Figure 2.1 : Architecture du Web sémantique [24].

Au niveau le plus haut, nous retrouvons la notion de confiance dont la principale question est : « Sur quelles bases peut on considérer comme vraie l'information disponible sur le Web?». Les langages et outils développés dans le cadre du Web sémantique ne sont que des bases technologiques. Ils ne résolvent pas à eux seuls les problèmes de présentation et d'utilisation de ces données sémantiques **[26]**.

Dans les sections suivantes, nous présentons les concepts les plus importants liés au Web sémantique.

2.3.1. Métadonnées

En général, le terme "métadonnées" est utilisé pour désigner "des données à propos d'autres données", c'est à dire, des données qui décrivent des ressources d'information. Plus précisément, les métadonnées sont une méthode structurée pour décrire des ressources et par conséquent pour améliorer leur sens. Le terme structuré est important car des données structurées impliquent une lisibilité et une compréhensibilité par les ordinateurs **[27]**.

2.3.2. Description des ressources : RDF et RDFS

Le langage RDF (Ressource Description Framework) **[28]** a été développé par W3C comme langage basé sur les réseaux sémantiques pour décrire les ressources du Web. Les objectifs initiaux de RDF étaient la représentation de la sémantique et une meilleure exploitation

des métadonnées. Mais, de manière plus générale, RDF permet de voir le Web comme un ensemble de ressources reliées par les liens « sémantiquement » étiquetés.

Les énoncés RDF sont des triplets ressource-attribut-valeur où la valeur est une ressource ou chaîne de caractères et une ressource doit disposer d'une URI. Les triplets sont interprétables comme sujet-prédicat-objet. La simplicité du modèle, critiquable pour certains, peut être une des clés de son acceptation et de la relative simplicité de la réalisation d'outils. Afin de renforcer ce langage, RDF Schema a été construit par W3C pour comme extension de RDF comportant des primitives basées sur des frames. RDF Schema permet notamment de déclarer les propriétés des ressources ainsi que le type de ces ressources. La combinaison de RDF et RDF Schema est connue sous le nom RDFS **[23], [29]**.

2.3.3. Ontologies

Indépendamment des définitions philosophiques du terme ontologie, en informatique il existe une multitude de définitions pour ce concept. La plus fréquemment référencée dans la bibliographie et la plus synthétique est celle de Gruber : « *une ontologie est une spécification formelle explicite d'une conceptualisation partagée*» **[30]**. Le terme "conceptualisation" réfère à un modèle abstrait d'un certain phénomène de la réalité et qui permet d'identifier les concepts pertinents de ce phénomène. Le terme "explicite" signifie que le type des concepts utilisés ainsi que les contraintes sur leur emploi est réellement définies d'une manière claire et précise **[23]**.

Pratiquement, une ontologie permet de décrire formellement un domaine de discours. Elle consiste en un ensemble fini de concepts et de relations entre ces concepts où un concept est une classe du domaine. Par exemple, dans une université les étudiants, les enseignants, le personnel constitue des concepts importants. Les relations constituent en particulier les hiérarchies entres les classes, une hiérarchie spécifie qu'une classe C est une sous-classe d'une autre classe C' si chaque objet de C est inclus dans la classe C' **[26]**.

En plus des relations hiérarchiques, les ontologies peuvent inclure des informations telles que : les propriétés, les restrictions de valeurs, les déclarations de dis jointure et les spécifications de relations logiques entre les objets.

Pour que les ontologies soient cruciales pour le Web sémantique, des méthodes et des outils sont développés pour contribuer à **[29]**:

- Construire les ontologies, que ce soit à partir de sources primaires, particulièrement les corpus textuels, ou en recherchant une certaine réutilisabilité.

- Gérer l'accès aux ontologies, leur évolution, avec gestion des versions, et leur fusion. Les ontologies sont souvent riches de plusieurs milliers de concepts et ne restent alors directement appréhendables que par leur concepteur. Leur accès par des utilisateurs, mêmes professionnels, nécessite de gérer le lien entre les concepts des ontologies et les termes du langage naturel, que ce soit pour une simple compréhension ou pour l'indexation et la construction de requêtes destinées à des tâches de recherche d'information.
- Assurer l'interopérabilité des ontologies en gérant les hétérogénéités de représentations et les hétérogénéités sémantiques. Ces dernières sont les plus difficiles à gérer et nécessitent des réflexions conjointes à la problématique de l'accessibilité des ontologies.

2.3.4. OWL

Il existe plusieurs langages de spécification d'ontologies (ou langage d'ontologies) qui ont été développés pendant les dernières années, et qui deviendront sûrement des langages d'ontologie dans le contexte du Web sémantique. Cependant, ceux qui sont proposés par W3C restent les plus utilisés dans la communauté du Web sémantique.

Nous avons déjà présenté la proposition du W3C qui s'appuie au départ sur une pyramide de langages dont RDF et RDFS sont relativement stabilisées. RDF et RDFS permettent de définir, sous forme de graphes de triplets, des données ou des métadonnées. Cependant, de nombreuses limitations bornent la capacité d'expression des connaissances établies à l'aide de RDF/RDFS. On peut citer, par exemple, l'impossibilité de raisonner et de mener des raisonnements automatisés (automated reasoning) sur les modèles de connaissances établis à l'aide de RDF/RDFS ; c'est ce manque que se propose de combler OWL (Ontology Web Language).

Proposé par W3C, OWL **[31]** est un standard s'appuie sur le langage DAML+OIL, produit de la combinaison du langage américain DAML (Darpa Agent Markup Language) et OIL (Ontology Inference Layer) provenant de projets européens. Le langage OWL est actuellement construit sur RDFS, et apporte ainsi aux langages du Web sémantique, l'équivalent d'une logique de description tout en disposant aussi d'une syntaxe XML **[29]**.

Il faut savoir que dans un langage pour écrire des ontologies il faut faire l'équilibre entre la capacité d'expression et l'efficacité du raisonnement, car plus le langage est expressif, plus l'efficacité du raisonnement diminue, et parfois le raisonnement devient tellement complexe qu'il n'est plus possible de le faire par ordinateur. Le but est alors de concevoir un langage qui

permet cet équilibre et c'est ce qui a motivé le W3C à prendre la décision de définir trois sous langages d'OWL **[27]**:

a) **OWL Full :** Comme son nom l'indique OWL Full contient toutes les constructions possibles avec le langage OWL.

b) **OWL DL :** C'est un sous ensemble d'OWL Full qui suit les règles suivantes:

- Une classe ne peut être membre d'une autre classe.
- Des restrictions sont établies sur les propriétés fonctionnelles, fonctionnelles inverses, et transitives.
- Une restriction est prescrite sur OWL : imports, lorsqu'on développe avec OWL DL on ne peut importer un document écrit avec OWL Full.

c) **OWL Lite** : C'est un sous ensemble de OWL DL caractérisé par :

- owl : hasValue, owl : disjointWith, owl : unionOf, owl : complementOf et owl: oneOf ne sont pas autorisées.
- Les contraintes de cardinalités sont plus réduites.

3. Web services sémantiques

Le couplage entre les Web services et le Web sémantique s'inscrit dans le cadre d'intégration des systèmes hétérogènes, plus particulièrement l'intégration sémantique. Dans ce contexte, plusieurs approches ont été proposées comme l'intégration sémantique par les données, par les traitements et par les processus. Dans notre travail, nous nous intéressons à l'approche orientée services sémantiques comme l'une des approches les plus efficaces qui est basée à la fois sur le dynamisme et la sémantique (figure 2.2) **[32], [23]**.

Les Web services sont devenus un moyen très efficace dans l'interopérabilité des systèmes. Le besoin d'introduire la sémantique dans les Web services se fait sentir, afin d'automatiser les différentes phases de leur cycle de vie, en l'occurrence la phase de découverte. Le concept des Web services sémantiques, est le fruit de la convergence du domaine des Web services avec le Web sémantique (voir la figure 2.2, **[33]**). En effet, son ultime objectif est de rendre les Web services plus accessibles à la machine en automatisant les différentes tâches qui facilitent leur utilisation.

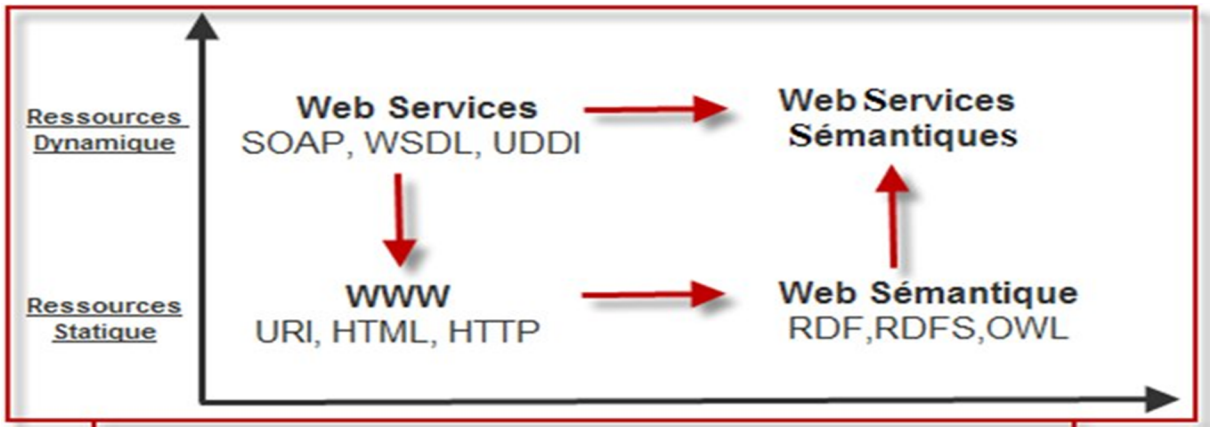


Figure 2.2 : Origine des Web services sémantiques [32].

Dans cette section, nous présentons les approches les plus représentatives des Web services sémantiques, spécialement en ce qui est, à notre sens, le plus efface et le plus utilisé dans la communauté du Web sémantique à avoir OWL-S.

3.1. OWL-S

Sachant que différents Web services peuvent offrir différentes fonctionnalités, prendre différents paramètres d'entrées et différents paramètres de sorties, chacun de ces Web services peut être décrit en répondant à des questions telles que **[27]**:

- Quelle est la fonction que remplit ce Web service ?
- Comment fonctionne-t-il ?
- Comment peut-il être invoqué?

Si on crée une ontologie qui nous permettra de répondre à ces questions générales, cela constituera aussi une ontologie générale sachant que les classes et les propriétés de cette ontologie ne doivent être liées à aucun domaine **[27]**.

Prédécesseur de DAML-S (Darpa Agent Markup language for Services), OWL-S **[34]** est une ontologie appelée *upper Ontology* qu'on appellera ontologie supérieure OWL-S pour (Web Ontology Language-Services), elle est écrite en utilisant OWL et a pour objectif la description des Web services **[27]**.

OWL-S : est une ontologie dédiée à la description des capacités et des propriétés des Web services. Le but d'OWL-S est de permettre l'automatisation de *la recherche*, de *la découverte*, de *l'invocation* et de *l'interconnexion* des Web services. Il fournit des éléments de description et spécifie les relations entre ceux-ci ainsi qu'un modèle permettant de décrire les Web services en introduisant des informations sémantique et en séparant les fonctionnalités du service de son fonctionnement interne et de la façon d'y accéder.

L'objectif d'OWL-S est de répondre aux trois questions précédemment citées. Pour cela, une ontologie OWL-S est constituée de trois sous ontologies : *Profile.owl*, *Process.owl* et *grounding.owl* qui ont pour but de décrire un Web service. Pour relier ces trois sous-ontologies ensemble, OWL-S présente une ontologie de plus haut niveau final appelé l'ontologie de Service : *service.owl* **[27]**. Toutes ces sous-ontologies sont écrites en utilisant OWL et la figure 2.3 présente la description des Web services dans la mesure du OWL-S.

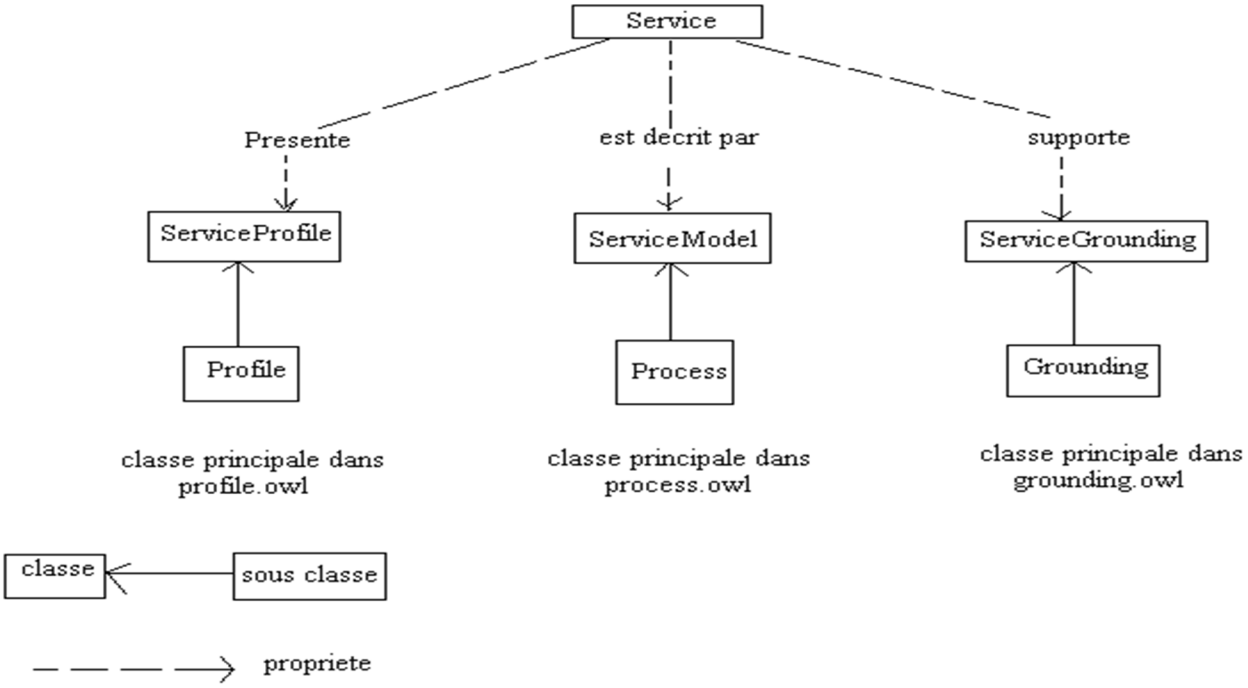


Figure 2.3 : Ontologie de haut niveau d'OWL-S [34].

Le rôle de chaque sous ontologies apparaissant dans la figure précédente est décrit comme suit **[27]**:

- **Fonctions d'un service Web**

L'ontologie supérieure OWL-S contient une sous ontologie appelée *Profile (profile.owl)* qui permet de définir des classes et des propriétés répondant à la question relative à la fonction du Web service. Cette ontologie a pour objectif la découverte du Web service. Elle sera donc utilisée lors de la recherche de ce dernier et c'est celle qui nous intéresse tout particulièrement.

- **Fonctionnement du Web service**

Une autre sous ontologie utilisée est *Process (process.owl)*. Elle définit les classes et les propriétés pour pouvoir décrire comment fonctionne le Web service. Plus précisément, elle décrit les procédures nécessaires pour que le client puisse interagir avec le Web service.

- **Invocation du Web service**

Une autre sous ontologie est *Grounding (grounding.owl),* elle permet de savoir comment un demandeur peut techniquement accéder à un service (par exemple quel est le protocole utilisé lors de l'échange).

3.2 .WSMO

WSMO (*Web Service Modeling Ontology*) **[35]** est un langage formel et une ontologie pour la description de divers aspects liés aux Web services sémantiques. Dans l'approche WSMO, on distingue deux composants principaux qui sont WSML (Web service Modeling Language) **[36]** et WSMX (Web Service Execution Environment) **[35]**.

WSML fournit un langage formel pour la description des éléments définis dans l'architecture WSMO. Il est basé sur différents langages logiques qui sont la logique de description, la logique de premier ordre et la logique de programmation. WSMX est un environnement d'exécution qui permet d'offrir un certain nombre de modules utilisables en temps d'exécution permettant de prendre en charge la découverte, la sélection, la médiation et l'invocation des services sémantiques **[23]**.

WSMO définit les éléments de modélisation pour la description de plusieurs aspects des Web services sémantique fondés sur l'échouement conceptuel mis en place dans le cadre de la modélisation des Web services **[36]**. Quatre composantes principales sont définies (figure 2.4):

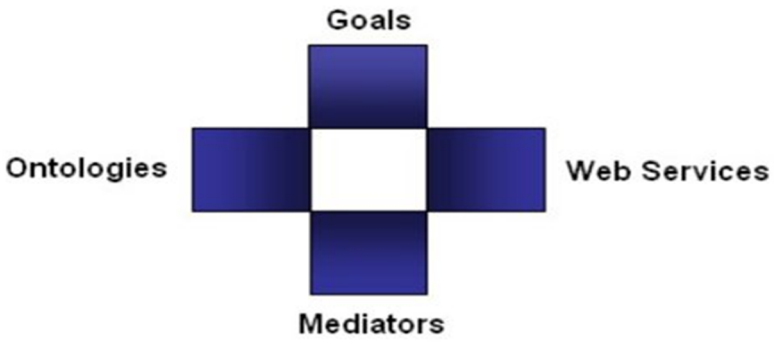


Figure 2.4: Composants de WSMO.

- ***Ontologies :*** elles représentent un élément clé dans le WSMO parce qu'ils fournissent des terminologies pour décrire les autres éléments. Son objectif est de définir de la sémantique formelle de l'information et de relier la machine et la terminologie de l'homme.
- ***Web Services :*** ils représentent un élément atomique des fonctionnalités qui peuvent être réutilisées pour en construire de plus complexes. Afin de permettre leur découverte, l'invocation de composition, d'exécution, de suivi, de médiation et d'indemnisation, les Web services sont décrits dans WSMO à partir de trois points de vue différents: les propriétés non fonctionnelles, les fonctionnalités et le comportement. Ainsi, un Web service est défini par *ses propriétés non fonctionnelles*, *ses ontologies importées*, *ses médiateurs utilisés*, *sa capacité* et *ses interfaces*.
- ***Objectifs:*** ils décrivent les aspects liés aux désirs des utilisateurs à l'égard de la fonctionnalité demandée par opposition à la fonctionnalité fournie décrite dans la capacité du Web service.
- ***Médiateurs :*** ils décrivent les éléments qui visent à surmonter l'inadéquation structurelle, sémantique ou conceptuelle qui apparaissent entre les différentes composantes qui construisent une description WSMO. Actuellement, le cahier des charges porte sur quatre différents types de médiateurs:
    - ***OOMediators*** : L'importation de l'ontologie cible dans l'ontologie source pour résoudre tous les décalages de représentation entre la source et la cible;
    - ***GGMediators*** : la connexion des buts retrouvés dans une relation de raffinement et résolution des inadéquations existants entre eux;
    - ***WGMediators*** : Lien de Web services à des objectifs et résolution des inadéquations;
    - ***WWMediators*** : Connexion de plusieurs Web services pour la collaboration.

3.3 .METEOR-S

METEOR (Managing End-To-End OpeRations) est un projet réalisé dans le laboratoire LSDIS dans l'université Georgia. METEOR-S (METEOR for Semantic Web Services) a pour objectif de fournir des Web services sémantiques dans le cadre du projet METEOR **[38]**. Il propose un système basé sur l'annotation sémantique de descriptions WSDL et fournit un canevas représentatif pour intégrer la sémantique des données, la sémantique fonctionnelle, la qualité de service et l'exécution de Web services.

METEOR-S permet de décrire une approche automatique pour l'annotation sémantique de la description WSDL d'un Web service. Il définit les ontologies basées sur le langage OWL pour représenter des services et permet de définir une composition de services par un processus abstrait qui spécifie le flot d'opérations. METEOR-S a choisi BPEL4WS comme langage de spécification de ces processus abstraits.

Pour accomplir ces tâches, METEOR-S définit quatre niveaux de sémantiques : de données, fonctionnelle, de qualité de service et d'exécution.

- **Sémantique des données** : est l'annotation (description sémantique) des entrées/sorties et des messages d'erreurs d'un Web service. Dans METEOR-S cette sémantique est représentée en utilisant l'ontologie RosettaNet.
- **Sémantique fonctionnelle** d'un Web service est décrite via les opérations offertes par un Web service. Chaque opération est décrite par l'ensemble des données échangées par l'opération, la fonctionnalité de l'opération ainsi que ses pré-conditions et post-conditions.
- **Sémantique de qualité de service** caractérise des aspects tels que la performance d'un Web service. Cette spécification doit être comprise par tous afin que les clients puissent choisir au mieux les services qui répondent à certaines de leurs exigences.
- **Sémantique d'exécution** qui permettra d'effectuer des tests de validation du service.

Le projet METEOR-S s'est développé selon trois phases principales qui ont permis d'introduire les trois concepts importants de cette initiative **[23]** :

- MWSDI (*METEOR-S Web Service Discovery Infrastructure*) qui consiste à installer une infrastructure de découverte sémantique définie au dessus d'un registre UDDI **[39]**.
- MWSAF (METEOR-S Web Service Annotation Framework) qui permet d'enrichir sémantiquement les Web services en utilisant une extension améliorée de WSDL appelée WSDL-S (WSDL Semantics) **[40]**. Le rôle de ce dernier est d'enrichir sémantiquement les fichiers WSDL et également le registre UDDI.
- MWSCF (METEOR-S Web Service Composition Framework) pour la composition et l'exécution des services pour lesquelles METEOR-S a choisi BPEL4WS.

4. Conclusion :

Dans ce chapitre avons présenté les limites des Web services et de leurs compositions en termes de découverte des Web services en particulier au niveau de leur interopérabilité sémantique. Dans ce cadre, nous avons étudié l'apport de la technologie du Web sémantique aux Web services et avons expliqué le concept de Web services sémantiques ainsi que les différents langages proposés dans ce domain.

Chapitre III: la médiation de services Web

1. Introduction

La résolution des hétérogénéités entre services Web est primordiale pour la réalisation de leur composition. En effet, les compositions conduiraient la plupart du temps à des échecs sans une médiation entre les fonctionnements des services et entre les données échangées par ces derniers. Dans ce chapitre, nous montrons les différents niveaux de médiation entre services Web, suivie d'une présentation des travaux traitant de la médiation de services Web.

2. Définition

D'une manière générale, la médiation consiste à résoudre les confits entre deux acteurs. Cette tache est effectuée par un élément spécifique appelé médiateur. Dans le domaine informatique, la notion de médiateur a été initialement utilisée pour les bases de données **[42]**, puis adaptée aux services Web. La médiation de services Web a pour objectif de résoudre les hétérogénéités présentes entre services Web afin de permettre des interactions réussies. Il est possible de classifier selon différentes perspectives ces hétérogénéités et les taches de médiation qui permettent leur résolution. Athanasopoulos et coll. classifient les hétérogénéités entre services Web selon trois dimensions **[43]**:

- **La dimension domaine métier :** concerne le niveau conceptuel des processus métiers. C'est la dimension la plus abstraite. Deux processus métier sont considères comme compatibles pour la dimension domaine métier s'ils distinguent les mêmes étapes pour leur exécution, et s'ils utilisent le même vocabulaire pour designer ces étapes.
- **La dimension application :** concerne les instances d'exécution des processus métiers. Elle englobe les structures des données échangées pendant l'exécution, l'orchestration des services Web dans le processus métier, et la description des fonctionnalité´es fournies par les services Web. Des processus métiers sont considères comme compatibles pour la dimension application si les services Web participants :

(a) suivent des représentations de données similaires,

(b) décrivent de manière identique les fonctionnalités qu'ils fournissent,

(c) et adoptent des séquences d'échanges de messages compatibles

- **La dimension plateforme :** correspond aux protocoles utilises pour l''´echange des messages, aux langages utilisés pour la description des types de

données, au langage de définition des interfaces et aux types de communication au niveau applicatif.

En outre, les auteurs identifient plusieurs niveaux d'interopérabilité, qui sont orthogonaux aux dimensions susmentionnées, car abordes selon une perspective différente :

- **Le niveau signature :** comprend les hétérogénéités entre les interfaces des services Web, incluant les opérations décrites ainsi que leurs paramètres d'entrée/sortie. Il appartient au domaine plateforme.
- **Le niveau protocole :** concerne la façon dont sont effectues les échanges entre services Web. Il se situe dans les domaines processus métier et application.
- **Le niveau sémantique :** est lie aux problèmes d'interprétation des données lors des interactions entre différents services ou avec le client. Il se situe dans le domaine processus métier.
- **Le niveau qualité :** concerne l'adaptation aux exigences des services Web en termes de qualité de service. Il se situe dans les domaines application et plateforme.
- **Le niveau contexte :** concerne les différences de représentation de contexte pour les systèmes embarques. Il se situe dans les domaines application et processus métier.

Cette classification combine deux approches orthogonales, ce qui lui donne beaucoup de richesse. Cependant, elle se révèle trop générale pour le cas particulier des services Web. En effet, elle concerne toutes les implantations du paradigme des SOA, et pour cette raison, elle prend en considération des hétérogénéités déjà résolues et des niveaux d'interopérabilité qui ne sont pas primordiaux dans le domaine des services Web. Les hétérogénéités de la dimension plateforme sont résolues par l'utilisation de la pile standard de protocoles des services Web. De plus, les niveaux d'interopérabilité relatifs à la qualité et au contexte ne sont pas indispensables dans le cadre des services Web, et ne sont pas pertinents pour tous les services.

Bussler propose une classification des méthodes de médiation pour les services Web **[44].** Il désigne les interactions entre services Web sous le nom d'´ev`enements métiers, et il distingue deux niveaux de médiation pour les services Web :

- La médiation au niveau des interactions est supposée résoudre les hétérogénéités qui apparaissent entre les messages en termes de différences de structure, de vocabulaire et de format de représentation des données.
- La médiation au niveau des processus métiers concerne les différences entre les séquences d'échanges de messages et les différences de gestion du processus de composition.

A l'inverse de la classification précédente, Bussler se concentre sur les différents types de médiation plutôt que sur les types d'hétérogénéités rencontrées. Sa classification des hétérogénéités se limite à deux catégories, qui ne distinguent pas assez les différentes hétérogénéités auxquelles nous pouvons être confrontes pendant les étapes de composition, à savoir, la découverte, l'organisation des interactions, et l'exécution.

Cabral et Domingue, quant à eux, différencient la médiation de données, la médiation de fonctionnalité, et la médiation de processus métier **[45]**.

- La médiation de données concerne l'adaptation des données d'entrée/sortie échangées par les services Web.
- La médiation de fonctionnalités établit une correspondance entre la fonctionnalité fournie par le service Web et celle demandée par le client.
- La médiation de processus métier résout les problèmes d'interaction entre services Web au sein de la composition. Elle comprend les différences de séquences d'échanges de messages, de protocoles et d'ordre des opérations à exécuter.

Cette classification repose sur les éléments constitutifs des services Web pour distinguer clairement les différentes facettes de la médiation : la médiation des entrées/sorties, des fonctionnalités fournies par les services et de leurs interactions dans la composition. Cependant, elle ignore les aspects de médiation non spécifiques aux services Web, tels que le niveau de support des transactions, ou bien la prise en charge de la sécurité. Il est nécessaire de prendre en compte ces aspects pour obtenir une classification exhaustive. De plus, cette classification ne distingue pas les propriétés des services Web d´écrites dans le document WSDL de celles qui ne le sont pas. Or, cette distinction est nécessaire pour la médiation, car le médiateur exploite les informations contenues dans les descriptions des services pour leur réconciliation.

3. Intégration de services Web

3.1. Hétérogénéités entre propriétés non fonctionnelles

Les propriétés non fonctionnelles des services Web sont généralement évaluées et comparées pendant l'étape de découverte des services. Elles participent en tant que critères à la procédure de sélection des services **[46]**. Les hétérogénéités entre les propriétés non fonctionnelles sont généralement résolues par l'utilisation d'une ontologie commune pour leur représentation et par la sélection de services compatibles. Cependant, certains travaux proposent des solutions de médiation pour résoudre ces hétérogénéités. Ces travaux se concentrent sur des propriétés non fonctionnelles particulières, à savoir :

Les séquences d'échanges de messages :

Les hétérogénéités entre les séquences d'échanges de messages peuvent empêcher l'exécution correcte d'une composition. En effet, si un service nécessite un message de confirmation de réception du résultat envoyé pour terminer sa tâche, et que les services qui interagissent avec lui n'en envoient pas, alors ce service restera dans un état instable, ce qui compromettrait la composition dans laquelle il est engagé. En nous appuyant sur plusieurs travaux **[47, 48, 49]**, nous distinguons les types de disparates suivants pour les séquences d'échanges de messages :

- ordre des messages différents,
- présence de messages supplémentaires,
- absence de messages requis,
- séparation de messages nécessaire,
- fusion de messages nécessaire.

Les propriétés transactionnelles :

Ce sont les quatre propriétés essentielles d'un système de traitement de transactions. Elles sont représentées par l'acronyme ACID, référant aux propriétés suivantes, adaptées aux services Web :

- Atomicité : une transaction doit être complètement validée ou complètement annulée.
- Cohérence : aucune transaction ne peut terminer dans un état incohérent.
- Isolation : une transaction ne peut voir aucune autre transaction en cours d'exécution.
- Durabilité : une probabilité suffisante que le service Web ne perdra aucune transaction validée.

Ces propriétés sont plus ou moins supportées selon les capacités des services Web.

La qualité de service (QoS) :

Le terme regroupe les propriétés non fonctionnelles relatives aux performances des services Web, telles que la disponibilité, la rapidité, le cout, la fiabilité, etc.

4. Adaptation d'interfaces de services Web

L'adaptation d'interfaces concerne les propriétés fonctionnelles des services Web, qui sont décrites dans un document WSDL classique. Ces propriétés comprennent les paramètres d'entrée/sortie, les fonctionnalités fournies, ainsi que les protocoles et l'encodage des données utilisé.

4.1. Hétérogénéités entre propriétés fonctionnelles

Afin de détailler les hétérogénéités liées aux interfaces, il est nécessaire de rappeler les éléments constitutifs d'un document WSDL, document descriptif de l'interface d'un service Web. Un document WSDL doit respecter le metamodele présente par la figure 3.1. L'élément <portType> fournit une description abstraite du service Web. Il décrit les fonctionnalités offertes par le service Web, grâce à des éléments <opération>, dont chacun symbolise une fonctionnalité particulière.

Chaque opération contient un unique élément <input>, un <output>, et un <fault>. Chacun d'entre eux contient un unique attribut <message> qui décrit les données reçues (pour l'élément <input>) et envoyées, lorsque l'exécution du service Web échoue (élément <fault>) ou réussit (élément <output>). Les éléments <message> décrivent les données échangées pour une opération.

Chaque message comprend une ou plusieurs parties, symbolisées par des éléments <part>, généralement appelés < paramétrés >. Chaque élément <part> possède un sous élément <name>, un sous-élément <type>, et peut contenir des sous-éléments additionnels. Ces derniers définissent respectivement les noms et types des paramétrés.

Pour conserver son indépendance par rapport à la plateforme d'accueil, WSDL utilise la syntaxe XML pour définir les types de données. Certains messages peuvent être complexes, et former des structures de plusieurs éléments XML Schéma. L'élément <type> définit la représentation des données. Finalement, un élément <binding> définit les formats de message et protocoles utilisés pour chaque élément <portType>. Des hétérogénéités peuvent apparaitre pour tous les éléments d'une interface. Nous étudions dans cette section les solutions de médiation correspondantes.

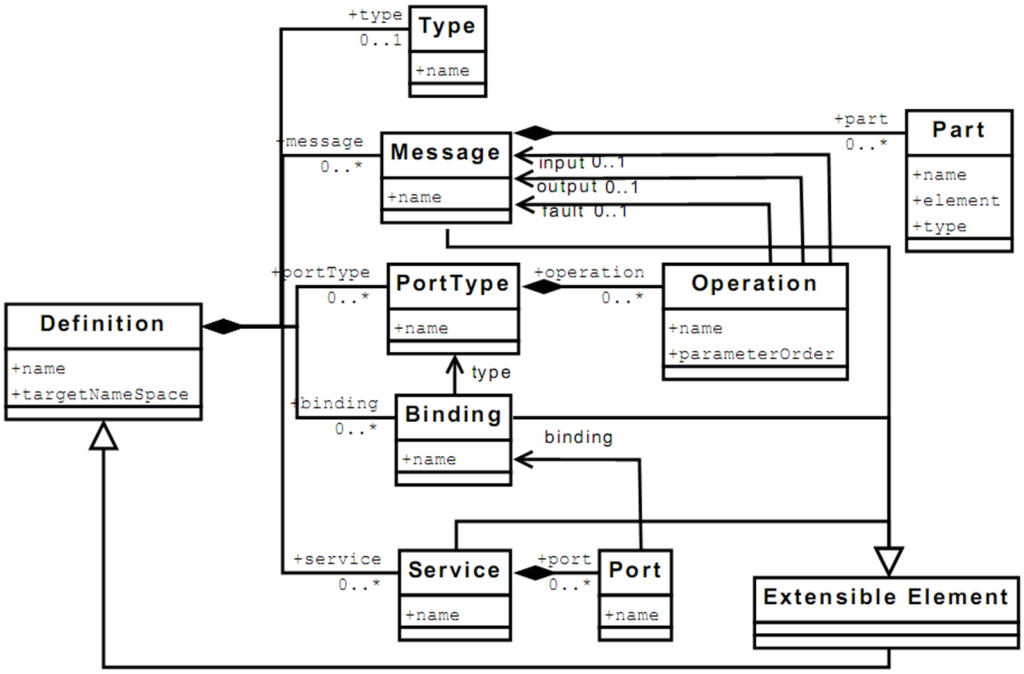


Figure.3.1 : Metamodele du langage WSDL [51].

5. Médiation de données pour les services Web

La médiation des données échangées entre services Web est un cas particulier des niveaux de médiation étudies précédemment. En effet, la médiation de données n'est effective que pendant l'exécution de la composition, lorsque les données transitent entre les services Web. Aussi, ce type de médiation est similaire à la médiation de données dans d'autres domaines, comme celui des bases de données, mais il est place dans le contexte spécifique de la composition de services Web. Ce contexte particulier pose des contraintes supplémentaires à la médiation. En effet, cette dernière doit être intégrée dans la composition, et les services Web ont eux aussi des contraintes de qualité de service.

5.1. Hétérogénéités des données échangées entre services Web

Comme mentionne précédemment, les services Web présentent des hétérogénéités au niveau des données qu'ils échangent. Nous distinguons trois niveaux d'hétérogénéités distincts: syntaxique, structurel et sémantique **[50, 52]** :

- **Le niveau syntaxique :** concerne l'encodage des données,
- **Le niveau structurel**
- **:** est relatif aux différentes représentations des données au niveau schéma,
- **Le niveau sémantique :** englobe la signification véhiculée par les données.

Nagara jan et coll. proposent une classification différente dans **[53]**. Ils identifient un niveau supplémentaire appelé < modèle/représentationnel > qui englobe les différences des modèles sous-jacents de représentation des données. Cependant, ce niveau qui était important dans le domaine des bases de données peut être ignore dans le domaine des services Web, car les hétérogénéités de ce type sont résolues par l'utilisation des langages de description li´es aux services Web, c'st-à-dire XML, XML Schema, RDF et OWL.

Une donnée possède trois attributs essentiels : son nom, son type, et sa valeur. Nous observons que les niveaux d'hétérogénéités présentes ci-dessus sont orthogonaux par rapport aux attributs (figure 3.1). Cela signifie que les trois niveaux d'hétérogénéité s'appliquent à la fois au nom, au type et à la valeur de la donnée que l'on souhaite décrire. Certaines hétérogénéités sont résolues par l'utilisation des langages de description liés aux services Web, de la manière suivante :

- **Au niveau syntaxique :** les hétérogénéités d'encodage sont résolues par l'utilisation de XML. Ce langage impose une syntaxe unique pour toutes les plateformes existantes, pour décrire le nom, le type ou la valeur de la donnée.
- **Au niveau structurel :** le nom d'une donnée est stocke dans une ontologie, sa structure est donc représentée sous la forme de triplets RDF. Le type d'une donnée est représenté via son schéma XML, et il peut être de type complexe. La valeur d'une donnée est une instanciation du schéma XML, et sa syntaxe est unique, seules les valeurs de l'instance changent.
- **Au niveau sémantique**, l'utilisation d'ontologies partagées permet de fixer la signification des vocabulaires utilises **[54]**. Les conflits de niveau sémantique lies au type des données sont résolus par la spécification XML Schéma, qui décrit la signification des différents types de données possibles de manière explicite. Concernant la sémantique de la valeur, les travaux existants considèrent que l'information sémantique liée au nom de la donnée est suffisante pour interpréter la valeur de manière appropriée.

5.2. Représentation des données échangées entre services Web : un modèle oriente contexte

5.2.1. Définition du contexte dans le cadre des services Web

Dans notre travail, nous associons toujours le contexte à une donnée, car nous nous intéressons uniquement au contexte des données échangées entre services Web composés.

Lors d'un échange de données entre deux services Web, deux contextes entrent en jeu:

- le contexte des données lors de leur émission, qui est lié au service émetteur des donnés.
- le contexte des donnés lors de leur interprétation, qui est lié au service destinataire des données.

Ces contextes trouvent leurs origines dans les différents environnements des services émetteur et récepteur. Par conséquent, le processus de médiation consiste à transformer la donnée transmise du contexte dans lequel elle a été modélisée vers le contexte dans lequel elle doit être interprétée, tout en conservant la signification souhaitée par l'émetteur. Pour ce faire, nous introduisons la notion d'objet sémantique pour les services Web, qui a pour but de décrire d'une manière explicite la sémantique des données échangées entre services en utilisant le contexte.

5.2.2. Notion d'objet sémantique

En 1999, Bornhovd et Goh et coll. présentent deux extensions du modèle initial qui sont très similaires. Ils s'intéressent tous deux à l'intégration de données semi-structurées, telles qu'on les trouve de plus en plus sur Internet, et introduisent la notion d'objet sémantique dans **[55, 56]** et **[57]**. L'idée dans ces travaux est de reprendre les fonctionnalités du modèle initial tout en utilisant un formalisme logique oriente objet adaptable aux données semi-structurées pour représenter l'information et son contexte.

5.2.3. Définitions de l'objet sémantique

La séparation des vues abstraites et concrètes pour la description des données est un état de fait dans le domaine des services Web sémantiques. Cette séparation permet l'utilisation de types de représentations différents pour les instances d'un même concept, et laisse libre choix aux fournisseurs de services Web quant à la représentation concrète de l'information. Ainsi, une donnée dont la sémantique est explicitement décrite possède :

- **Un concept**, qui définit la famille ontologique à laquelle cette donnée appartient, en d'autres termes la classe abstraite dont cette donnée est une instance.
- **Un type**, qui définit le genre de contenu de la donnée. Pour les services Web, le type d'une donnée est décrit avec le langage XML Schéma, et peut être simple ou complexe.
- **Une valeur**, qui est la donnée elle-même. Cette valeur est une instance du type de la donnée.

- **Un contexte**, qui apporte des précisions sur l'interprétation de la donnée.

Par conséquent, nous définissons un *objet sémantique S* comme un quadruplet, représenté de la manière suivante :

S = (*c; v; t ; C*), où

– *c* est le concept auquel l'objet sémantique se référence,

– *v* est la valeur de l'objet sémantique,

– *t* est le type dans lequel cette valeur est décrite,

– *C* est le contexte de l'objet sémantique.

Ce contexte est lui-même constitué d'objets sémantiques que nous appelons *modifieurs*, car ils appartiennent au contexte. Les modifieurs ont la capacité de modifier la signification de l'unique objet sémantique auquel ils sont associés. Cette représentation d'un objet sémantique initial avec d'autres objets sémantiques rend la définition d'objet sémantique auto descriptive. Une définition formelle d'un contexte *C* est :

$C = \{(c1; v1; t1; C1) :::::::: (cn; vn; tn; Cn)\}; n \in IN$

Où (*ci ; vi ; ti ; Ci*); 1 <= i <= *n*, sont les modifieurs qui décrivent les différentes propriétés sémantiques de *S*. La définition de l'objet sémantique est résumée par la figure 3.2.

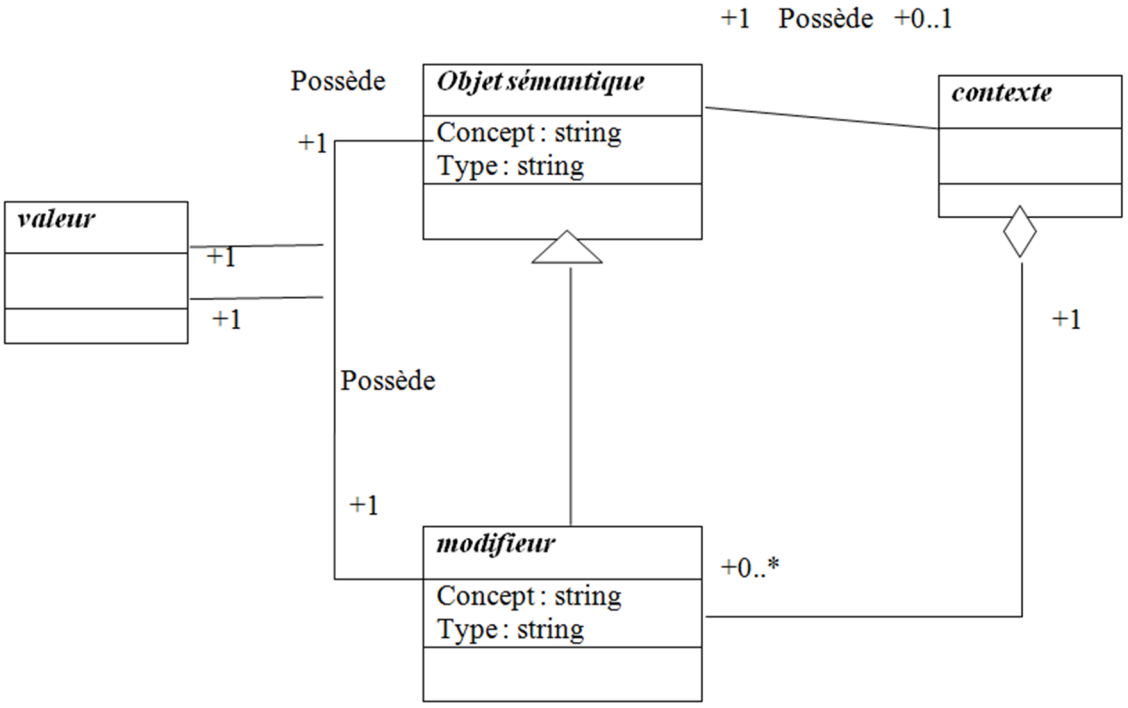


Figure 3.2 : représentation UML de l'objet sémantique

5.2.4. Modifieurs statiques et dynamiques

A partir de la définition présentée ci-dessus, nous définissons les notions de modifieur statique et dynamique. Ces notions sont utiles pour identifier les dépendances entre les modifieurs d'un contexte, en effet :

- **Un modifieur statique :** est indépendant des autres modifieurs. Il possède une valeur qui doit être explicitement décrite afin d'apporter une information quant à l'interprétation de l'objet sémantique, et qui ne peut pas être déduite des valeurs prises par les autres modifieurs du contexte.
- **Un modifieur dynamique:** est dépendant d'un ou plusieurs autres modifieurs. Il possède une valeur qui peut être déduite, par une fonction ou un ensemble de règles logiques, des valeurs prises par un ensemble d'autres modifieurs appartenant au même contexte. Par exemple, la valeur du modifieur < *Format de date* > peut être déduite lorsque l'on connait la valeur du modifieur < *Pays* >, qui appartient au même contexte. La règle logique qui permet cette déduction peut être décrite comme suit :

< Si *pays* = *Algérie*, alors *Format de date* = *dd.mm.yyyy* >.

Par contre, aucune information contenue dans ce contexte ne nous permet de déduire la valeur du modifieur < TVA Incluse >. La figure 3.3 donne un exemple d'objet sémantique **S** qui p eut être envoyé au service Web < Addition >.

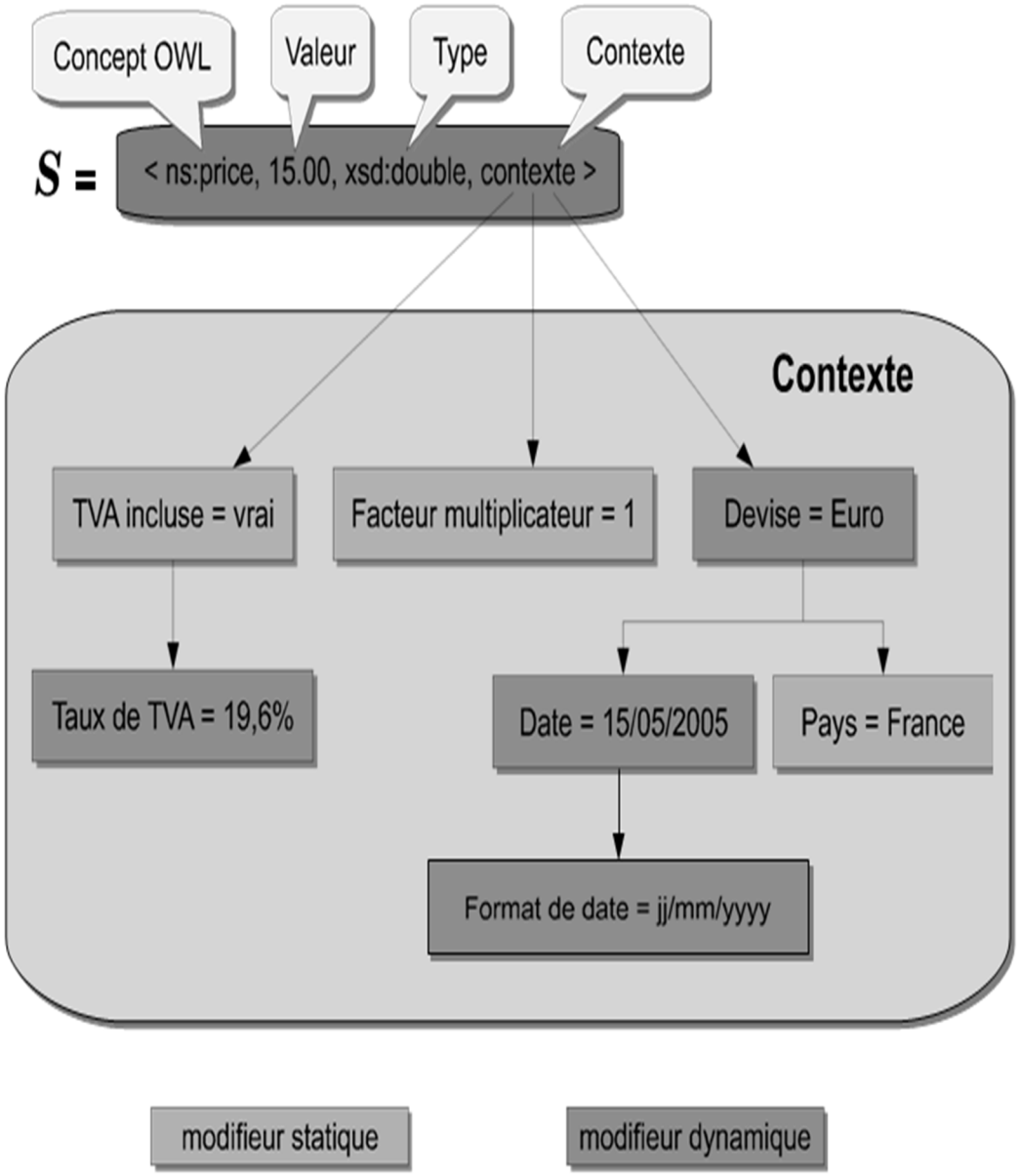


Figure.3.3 : Représentation de l'objet sémantique S avec son contexte

- l'attribut ns : Price renvoie au concept < Price > de l'ontologie symbolisée par l'espace de nommage < ns >.
- l'attribut 15: 00 est la valeur de la donnée transmise entre les services Web, son type est décrit par l'attribut xsd :double , ce qui signifie que la valeur est de type < double > , suivant le langage décrit dans l'espace de nommage < xsd > pour XML Schema Description.

- l'attribut contexte est une liste de modifieurs qui permet d'effectuer une interprétation correcte de l'objet sémantique. Ici, les modifieurs indiquent que l'objet est une devise en Euro, possède un facteur multiplicateur de 1, et inclut une TVA de 19,6 %. Le modifieur devise est lui-même décrit par des modifieurs additionnels. Certains modifieurs sont dynamiques et d'autres sont statiques.

5.2.5. Conversion entre objets sémantiques

La description du contexte sous la forme de modifieurs statiques et dynamiques, en plus du concept attaché aux données, fournit l'information nécessaire à une interprétation correcte des données. Ainsi, il devient possible de convertir les données d'un contexte à un autre, en utilisant des fonctions de conversion.

Ces fonctions de conversion sont spécifiques aux objets sémantiques. Elles peuvent être appliquées sur les modifieurs qui forment le contexte ou sur les attributs de l'objet sémantique (type et concept). Par conséquent, ces fonctions peuvent modifier la valeur de l'objet sémantique.

Propriétés des fonctions de conversion : Les fonctions de conversion entre objets sémantiques présentent des propriétés qui offrent des caractéristiques intéressantes. Nous distinguons ci-après les fonctions totales et non totales, avec et sans pertes et monotones croissantes.

Catégories des fonctions de conversion : Nous distinguons trois catégories de fonctions de conversions, selon l'attribut de l'objet sémantique auquel elles s'appliquent. Nous les désignons par les qualificatifs suivants : fonctions de conversion contextuelles, de type et de concept.

6. Annotation des langages existants

Contrairement aux langages précédents, plusieurs travaux proposent d'étendre les normes existantes par une annotation, soit en utilisant les éléments d'extensibilités prévus à cet effet [46, 65], soit en modifiant les fonctionnalités initiales des normes [58, 60].

6.1. Annotation du processus métier

SEmantic Service MArkup (SESMA) : Le langage SESMA **[58]** est un autre format de description pour services Web sémantiques, qui a été conçu pour fournir un langage simple à utiliser, avec une syntaxe compacte et une forte intégration avec les langages existants WSDL, SOAP et WS-BPEL. SESMA a été construit avec les objectifs suivants :

- une syntaxe reposant sur le langage XML, pour une distribution du langage à large échelle,
- une sémantique précise qui clarifie la signification des descriptions,
- une forte complémentarité avec les langages existants,
- des possibilités d'extension permettant une évolution vers d'autres langages éventuels.

Son principal avantage est d'être un langage léger, et sa sémantique n'est pas construite à partir de OWL. De plus, il fournit un support pour la description de services composites, sous la forme d'annotations de processus métiers WS-BPEL.

6.2. Annotation du langage de description

Exploitation des éléments d'extensibilité de WSDL : Avec WSDL-S, Miller et al. Annotent WSDL avec plusieurs extensions relatives aux opérations et messages d'entrée/sortie des services Web **[59]**. Ces extensions contiennent des références aux concepts décrits dans les ontologies de description du domaine de connaissance associe au service Web, afin de spécifier la sémantique des messages, mais aussi les pré-conditions et effets des opérations. WSDL-S vise à fournir une approche < allegee > d'annotation sémantique de services Web.

Extension de WSDL avec SAWSDL : Le World Wide Web Consortium propose avec les Semantic Annotations for Web Services Description Language (SAWSDL) **[60]** un moyen d'annoter les descriptions WSDL 2.0 tout en supportant WSDL 1.1. SAWSDL est un ensemble d'attributs d'extension permettant de décrire la sémantique des éléments contenus dans les documents WSDL. L'objectif de SAWSDL est de définir comment une annotation doit être réalisée, tout en laissant le choix du langage utilise pour la description sémantique. SAWSDL fournit les mécanismes permettant d'attacher des concepts décrits dans des ontologies aux annotations des descriptions WSDL décrits dans des ontologies aux annotations des descriptions WSDL. Cette proposition étend WSDL-S, elle p eut être considérée comme une continuité de ce langage. Elle vise à apporter la valeur ajoutée de la sémantique non seulement lors de l'invocation des services Web, mais aussi durant la phase de découverte. Trois attributs d'extensibilités sont définis par défaut : l'attribut

<modelReference> permet l'association entre un comp osant WSDL et un concept d'une ontologie, et les attributs < liftingSchemaMapping > et < loweringSchemaMapping > sont ajoutes aux definitions de types pour spécifier les correspondances entre les éléments du schéma des données et l'information sémantique de l'ontologie.

6.3. Annotation des registres

Extension sémantique d'UDDI : Paolucci et coll. **[61]** présentent une approche visant à améliorer la publication et la découverte de services Web dans les registres UDDI. Ils étendent les descriptions UDDI avec l'information sémantique nécessaire pour décrire les capacités des services Web, en utilisant le langage DAML-S **[62]**, prédécesseur de OWL-S. Ils soutiennent que la découverte des fonctionnalités de services Web ne peut être effectuée qu'au niveau sémantique. En effet, sans sémantique explicite, deux descriptions équivalentes peuvent être interprétées différemment, selon les circonstances de leur utilisation. Ainsi, l'automatisation de la découverte des services Web passe par la description explicite de leurs capacités. Les auteurs décrivent une méthode pour effectuer la traduction de la représentation DAML-S vers la représentation UDDI. De plus, ils proposent un algorithme de correspondance sémantique des fonctionnalités offertes par les services Web, sur la base de leur représentation UDDI modifiée. Ainsi, la découverte de services Web tire avantage des descriptions sémantiques qui ont été insérées dans le standard UDDI.

Extension sémantique d'ebXML : Dogac et coll. **[63]** proposent aussi un enrichissement des registres avec de la sémantique, mais leur proposition concerne le standard ebXML. Leur motivation est similaire à celle de Paolucci et coll., cependant les mécanismes développés sont différents en raison de la structure d'ebXML qui possède de nombreuses différences par rapport à UDDI. En effet, ebXML permet de stocker la description sémantique d'un service Web directement dans le registre, alors qu'UDDI doit faire référence à un document extérieur. Aussi, ebXML fournit la possibilité de définir des correspondances vers des concepts sémantiques directement dans le fonctionnement du registre. Ainsi, les auteurs utilisent les possibilités d'ebXML pour insérer les ontologies directement dans les registres. Ils proposent une table de correspondance entre le vocabulaire de OWL et les structures de description offertes par ebXML, lesquelles sont étendues pour atteindre la richesse de description offerte par OWL. Grâce à cette solution, les registres ebXML sont capables de contenir des descriptions sémantiques de services Web, qui peuvent être découvertes par les applications clientes à l'aide de modèles de requêtes spécifiques aux descriptions sémantiques proposées par les auteurs.

7. Conclusion

Un point important, qui se dégage de la plupart des travaux, concerne l'utilisation de l'information sémantique pour faciliter la médiation. Que ce soit au niveau de l'intégration de services, de l'adaptation d'interfaces, ou bien de la médiation des données, les descriptions sont généralement enrichies avec de l'information sémantique, ce qui permet d'automatiser au moins partiellement le traitement des données, en résolvant les conflits de signification grâce aux ontologies.

En effet, l'utilisation de langages de description sémantique apporte des possibilités de raisonnement qui permettent de définir des règles de conversion d'une représentation de données à une autre. Notons que de nombreuses approches omettent les hétérogénéités structurelles des données sous prétexte que le niveau sémantique est suffisant pour résoudre ces dernières. Cependant, il est tout aussi nécessaire de résoudre les hétérogénéités au niveau structurel, ou bien il faut supposer que le niveau structurel est pris en charge manuellement lors de la conception de la composition.

Chapitre IV: Conception

1. Introduction

En effet, rappelons qu'une composition peut être confrontée à de nombreuses hétérogénéités, notamment, des hétérogénéités sémantiques peuvent se présenter entre les donnes échangées par les services Web. Une médiation des donnes au niveau sémantique est nécessaire pour obtenir une interprétation correcte de ces dernières, de la part des services mis en jeu dans la composition. L'architecture de médiation que nous proposons ci-après a pour but de résoudre ce type d'hétérogénéités sémantiques, tout en facilitant la tache des fournisseurs de services et des concepteurs de composition.

Nous fournissons tout d'abord une vue conceptuelle de l'ensemble de l'architecture, avant de détailler les différentes étapes permettant la réalisation de notre architecture, à savoir :

– l'intégration du contexte dans la pile standard de protocoles des services Web par l'annotation de leurs descriptions et l'utilisation d'ontologies contextuelles.

– l'intégration de la médiation au sein de la composition de services, supportée par un algorithme permettant la détection des hétérogénéités sémantiques et l'insertion de services Web médiateurs dans un processus métier.

2. Exemple de motivation

Dans cette section, nous nous basons sur l'exemple de composition de la transaction en ligne « cas ifri » afin d'illustrer concrètement notre problématique.

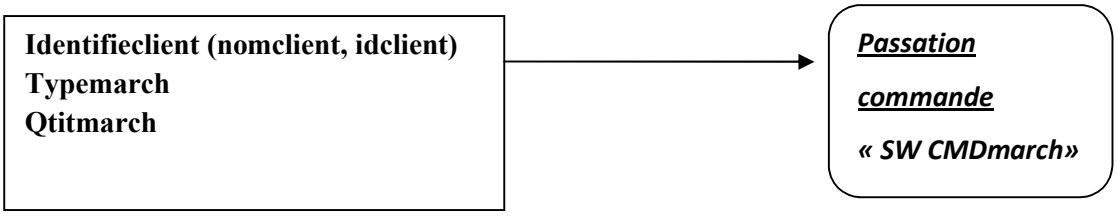
2.1. Scénario de la transaction en ligne « cas ifri »

Considérons une transaction en ligne d'un client situé en France, qui comprend la passation d'une commande et la location d'un véhicule (camion) pour la transmission de sa marchandise au port afin de l'exporter en France dans les plus brefs délais de transmission. Cette planification nécessite la coordination de plusieurs services Web au sein d'une même composition, dans notre cas :

- Un service web de passation de commande de marchandise « CMDmarch ».
- Un service web de location de véhicule de poids lourd « B_log » (dans notre cas en prend comme exemple service web **Bejaïa-logistique** qui est une filiale de SARL IBRAHIM ET FILS « IFRI »)
- un service Web pour additionner les prix des services précédant.

Les étapes de composition de service web sont représentées par le processus métier de la figure 4.1. Plusieurs hétérogénéités peuvent survenir lors de la création d'un processus métier comme celui de notre exemple. Dans le cadre de notre travail, nous nous intéressons

spécifiquement aux hétérogénéités sémantiques des données échangées par les services composés et la composition elle-même.



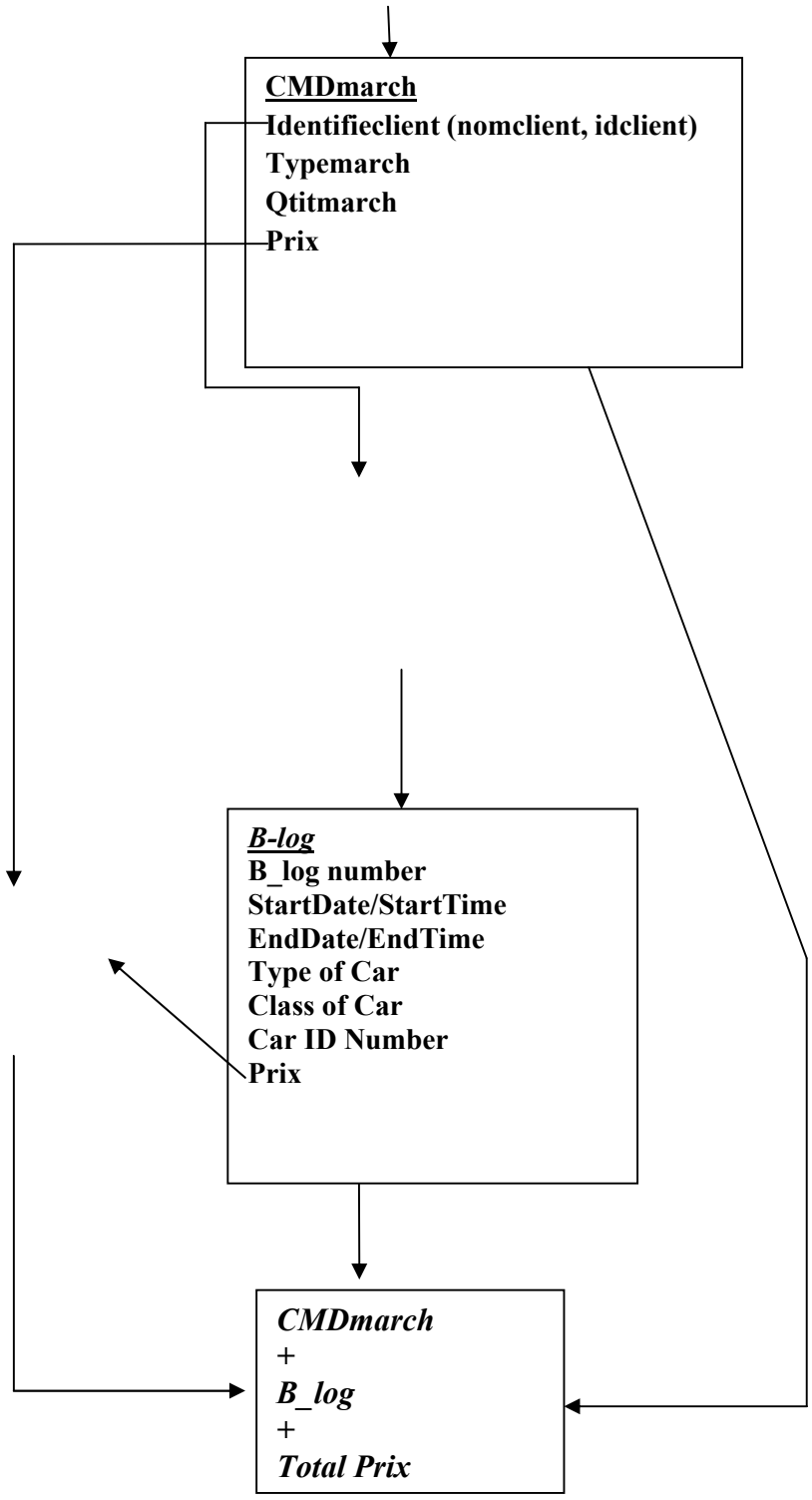


Figure 4.1 : processus métier de la transaction de marchandise «CMDmarch » et location de véhicule poids lourd « Bejaia-logistique »

2.2. Hétérogénéités lié au contexte

Le processus de sélection des services Web, même lorsqu'il est réalisé manuellement, ne permet pas toujours la composition de services compatibles au niveau sémantique. Ainsi, notre composition utilise les services suivants, qui ont été choisis au cours de l'étape de Sélection :

– le service < CMDmarch > assure la passation de commande du client.

– le service < B_log > prend en charge la location d'un véhicule de poids lourd et/ou léger pour la transmission de la marchandise du client vers le port pour l'exporter en France

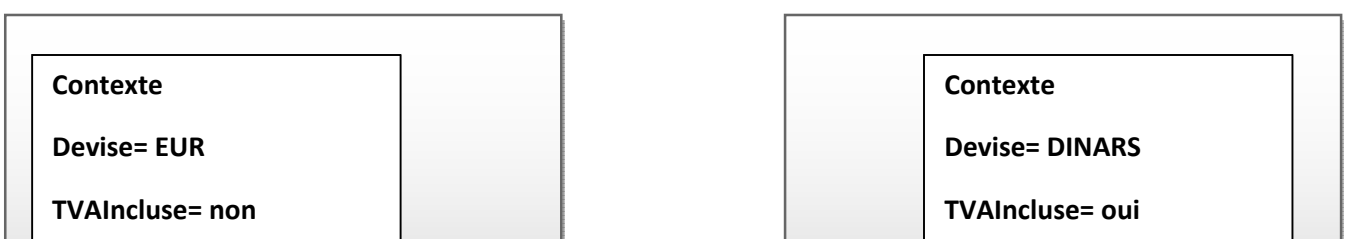
– le service < Addition > calcule le prix total généré par les deux services.

Ces services présentent les hétérogénéités de représentation des données suivantes :

– le service Web de passation de commande « CMDmarch » est un service qui traite les prix en EURO et avec un facteur multiplicateur de 1 et le service Web de location de véhicule est un service, qui traite les prix en DINAR algérien, ainsi les valeurs retourné par ce service doivent être multiplié par 140 afin d'obtenir la valeur émise.

Dans cette composition, les prix utilisés doivent être convertis dans la même devise et dans le même facteur multiplicateur avant d'être envoyés au service Web <Addition >.

Cet exemple montre qu'une description des données établie par le langage WSDL ne remplit pas les exigences de l'échange sémantique. Malgré une compatibilité des types de données (type < double > dans la figure 4.2) et des concepts sémantiques utilisés (Concept <prix > dans la figure 4.2), les données doivent être interprétées différemment, car elles sont placées dans des contextes différents.



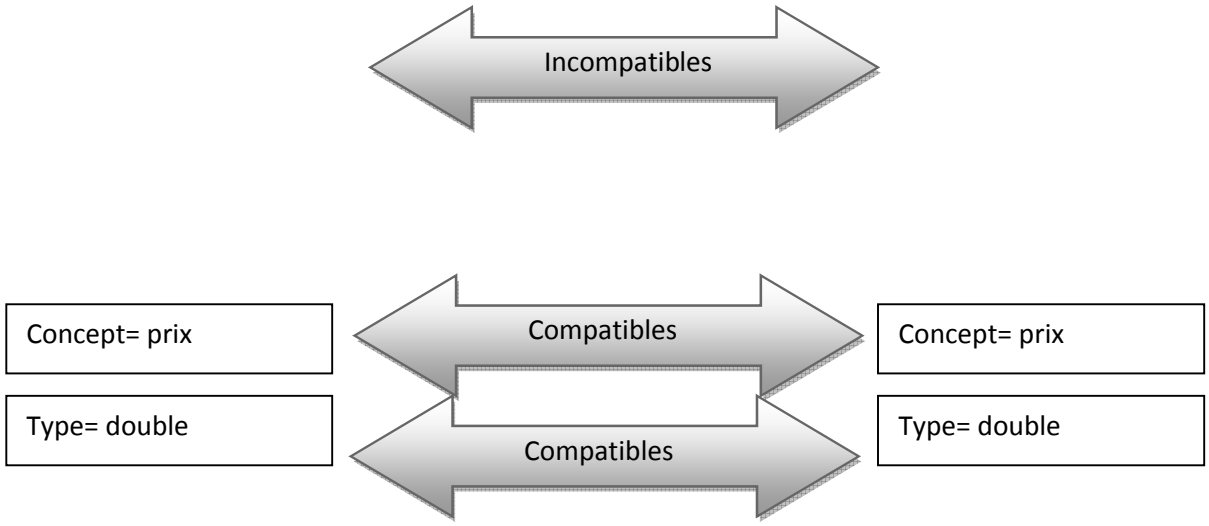


Figure 4.2 : Conflits entre les contextes des services < CMDmarch > et < B_log >

3. Architecture conceptuelle

L'architecture conceptuelle soutenant notre architecture se compose de trois couches principales, présentées par la figure 4.3.

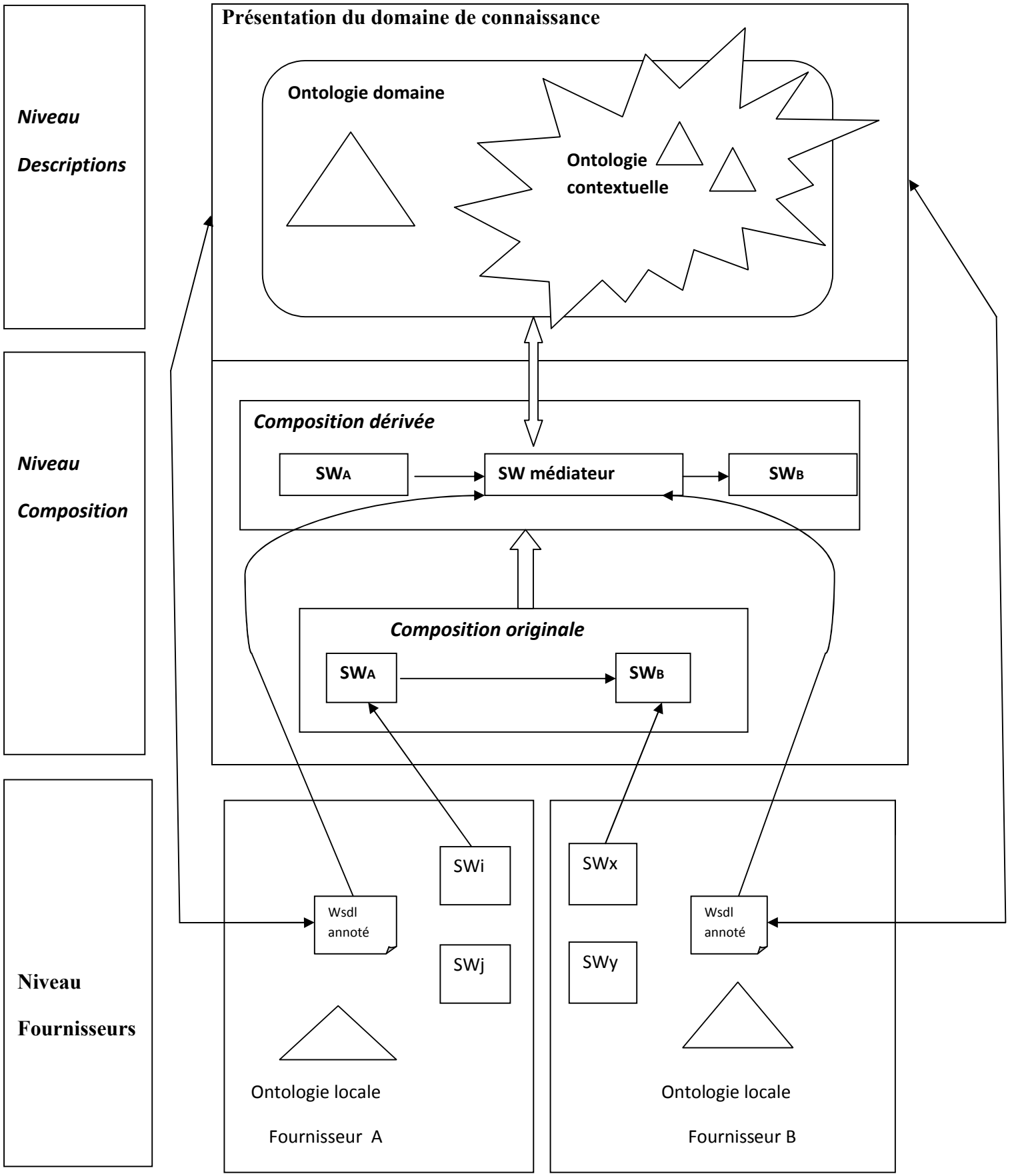


Figure 4.3 : Architecture conceptuelle de l'approche de médiation proposée

3.1. La couche fournisseurs

Comprend les services disponibles, regroupés par fournisseur. Chaque fournisseur possède une sémantique locale, symbolisée dans notre architecture conceptuelle par une ontologie. Les données échangées par les services Web sont représentées suivant les sémantiques locales de leurs fournisseurs.

- **Fournisseur et la sémantique des données :** Les fournisseurs doivent décrire de manière explicite la sémantique utilisée par leurs services Web. Cette tache reste à leur charge, car ils sont les plus aptes à décrire la sémantique de leurs services.
- **Fournisseur et annotation WSDL :** une annotation de la description d'un service par un acteur autre que le fournisseur du service contiendra une probabilité d'erreur qui ne garantirait pas l'optimisation de l'architecture.
- **Fournisseur et langage description :** adopté un langage de description standard de services WSDL, nous oriente vers une annotation des descriptions de services avec l'information nécessaire pour décrire la sémantique qui leur est associée.
- **Les fournisseurs et l'interopérabilité :** les fournisseurs de service doivent trouver un accord commun minimum concernant les noms des concepts sémantiques utilisés par leurs services pour permettre l'interopérabilité, mais sans modifier leurs interprétations locales.
- **Les fournisseurs et l'ontologie :** les fournisseurs doivent décrire explicitement les particularités locales d'interprétation des concepts. Pour ce faire, nous précisons le rôle des ontologies de domaine, et nous les associons à des ontologies contextuelles. Les fournisseurs doivent tout d'abord adhérer à une ontologie de domaine commune, puis mettre à jour, avec leur sémantique locale.
- **L'avantage de l'adisions des fournisseurs à une ontologie de domaine commune :**

- Limiter le rôle de l'ontologie de domaine à une description minimum des concepts communs.
- Facilitation de l'adhésion des fournisseurs de services.
- la mise à jour d'une ontologie contextuelle est seulement nécessaire lors de la première adhésion d'un service web.

3.2. La couche composition

Contient les compositions de services Web qui sont décrites dans des processus métiers. Les compositions originales de services sont transformées en compositions dérivées

qui prennent en charge les hétérogénéités sémantiques des données grâce à des services Web médiateurs.

Dans cette couche, nous détaillons la solution développée afin d'intégrer l'architecture de médiation contextuelle au sein du processus métier qui décrit la composition. Des médiateurs sont insérés dans la composition en tant que services Web, et interagissent avec les ontologies de domaine et contextuelles ainsi, des mécanismes de médiation doivent être déployés afin de prendre en charge les hétérogénéités sémantiques des données au niveau de la composition.

Pour ce faire, il est nécessaire de :

- ✓ détecter les potentielles hétérogénéités sémantiques des données dans un processus métier.
- ✓ déployer des médiateurs pour résoudre les hétérogénéités potentielles.
- ✓ modifier le processus métier pour qu'il intègre ces médiateurs.

3.2.1. Contextualisation d'un processus métier

C'est la modification du processus métier en vue de l'intégration des médiateurs nécessaires à la médiation sémantique des données par la prise en charge du contexte. Notre méthode de contextualisation repose sur les éléments suivants :

– un algorithme de détection des potentielles hétérogénéités sémantiques des données dans le processus métier.

– une méthode de génération et de déploiement de services Web médiateurs.

– une méthode de mise à jour de processus métier qui permet l'intégration des services Web médiateurs.

3.2.2. Avantages d'une solution orientée service

Dans le but d'intégrer les médiateurs nécessaires à la résolution des hétérogénéités contextuelles dans la composition de services, nous proposons une solution orientée service. En effet, nos médiateurs sont implantés comme des services Web et sont nommés *services Web médiateurs*. Cette solution présente les avantages suivants :

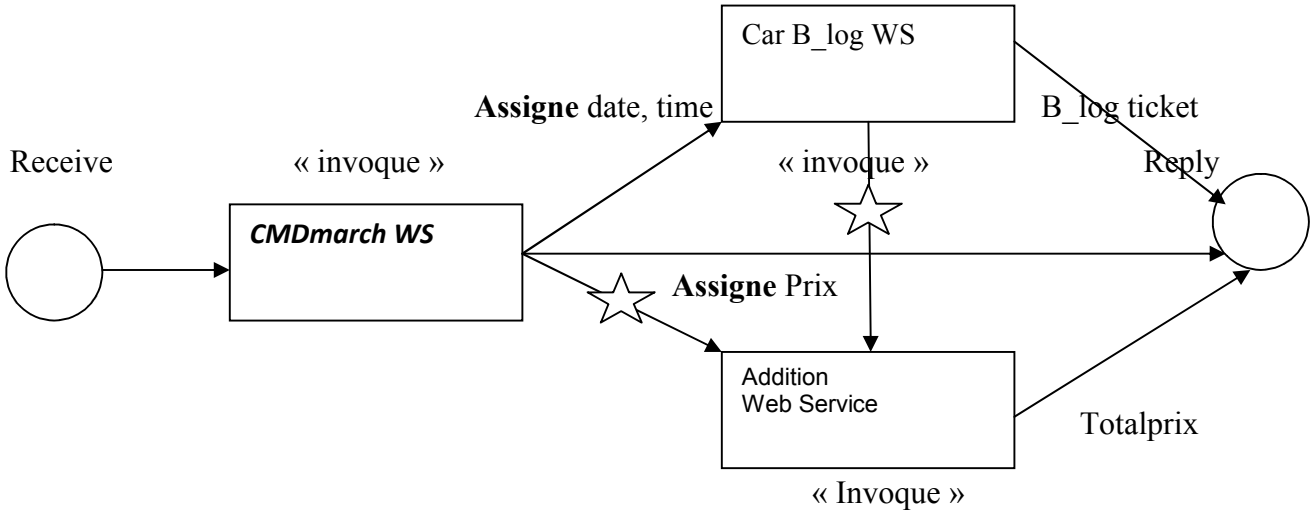
1. L'accès standardisé aux services Web à travers leurs descriptions WSDL permet une meilleure indépendance par rapport aux langages et moteurs de composition.

2. La gestion orientée service du processus de médiation facilite la mise à l'échelle, car elle ne nécessite l'extension d'aucun langage, ni la modification des architectures de composition existantes. En outre, elle permet la réutilisation des composants logiciels déjà déployés.

3. L'aspect faiblement couplé des architectures orientées service permet de conserver l'indépendance entre les aspects liés à la médiation et les fonctionnalités originales des services Web.

3.2.3. Etapes de Contextualisation des processus métiers

Pour présenter les étapes de contextualisation d'un processus métier, nous utilisons l'exemple développé précédemment cas CMDmarch ifri. Le processus métier décrit par la figure 4.4 décrit la logique de composition. Il a été démontré précédemment que plusieurs hétérogénéités liées au contexte gênent L'exécution correcte de cette composition. L'hétérogénéité dans notre cas est due à l'utilisation de différents formats (devise) de représentation pour les monnaies.



☆ Flux de données représentant des hétérogénéités sémantiques potentielles.

Figure 4.4: Représentation du processus métier original

Dans le but de réaliser les étapes de contextualisation d'un processus métier, nous considérons que les étapes pré-requises pour la mise en place de l'architecture de médiation ont été réalisées, c'est-à-dire que les fichiers WSDL des services Web sont correctement annotés avec l'information contextuelle, et que les ontologies de domaine et contextuelles requises sont disponibles. La contextualisation d'un processus Métier se constitue des étapes suivantes, résumées dans la figure 4.5.

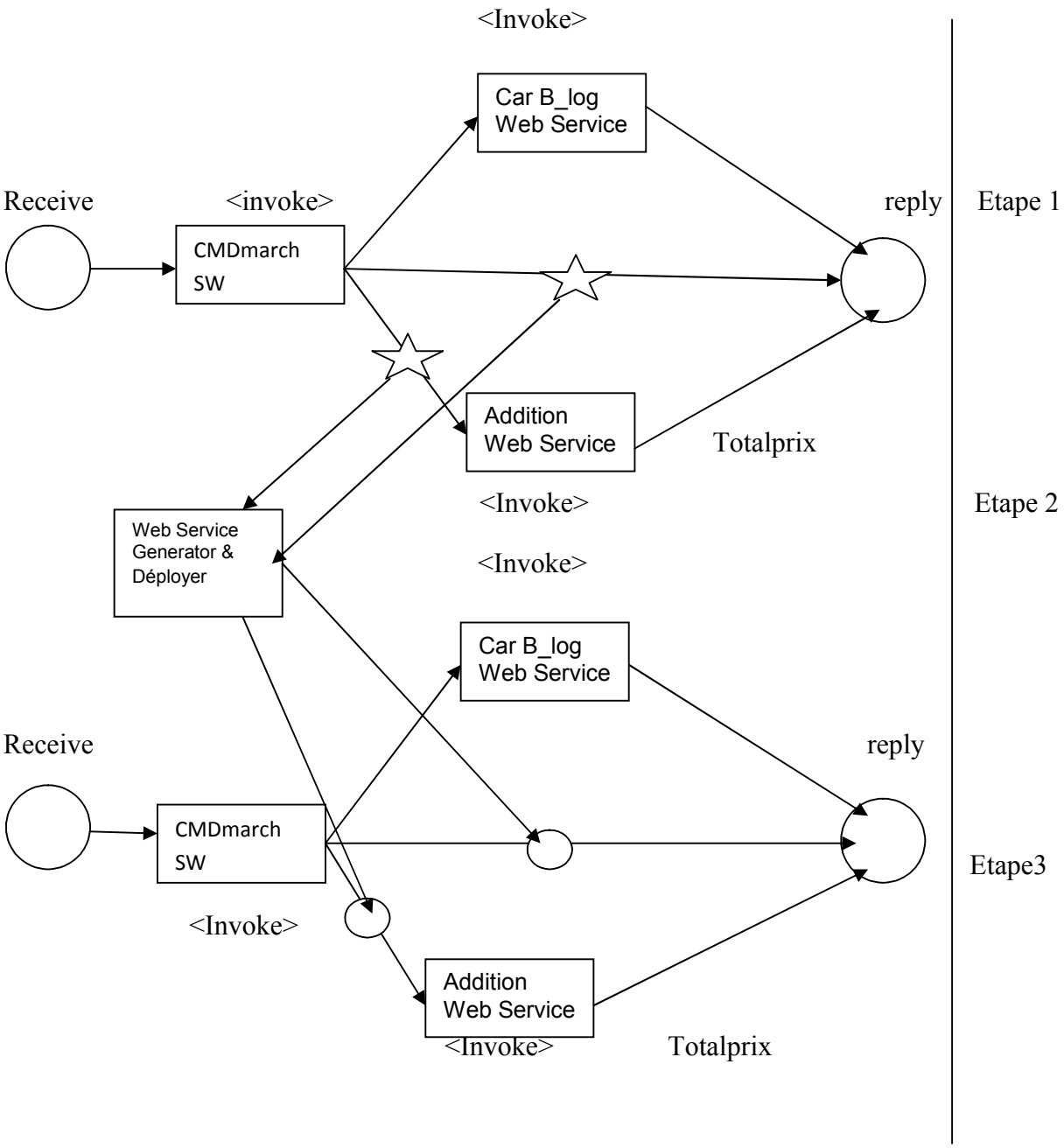


Figure. 4.5 : Etapes de génération du service Web médiateur.

Service web médiateurs.

Flux de données représentant des hétérogénéités sémantiques potentielles.

Étape 1 : Détection des hétérogénéités sémantiques des données.

Étape 2 : Génération des services Web médiateurs.

Étape 3 : Insertion des services Web médiateurs dans le processus métier et exécution.

Localisation des hétérogénéités potentielles :

Un algorithme, présenté en section 5.4.2.4, analyse le processus métier pour localiser les flux de données explicites et implicites. Dans notre exemple, les flux de données intéressants sont lorsque les prix sont envoyés au service Web d'addition.

Génération automatique des services Web médiateurs :

Dans cette étape, un service Web médiateur est automatiquement généré et déployé pour chaque flux de données où l'algorithme de contextualisation a détecté des hétérogénéités sémantiques potentielles. La génération des services Web médiateurs est prise en charge par notre Web Service Code Generator (WS-CG). Chaque service Web médiateur généré propose une opération appelée mediateX2Y, où *X* et *Y* sont remplacés par les noms des messages d'entrée et sortie des opérations. Ces dernières sont spécifiées dans les fichiers WSDL des services mis en relation via le service Web médiateur. Les entrées de l'opération de médiation sont les valeurs qui doivent être transformées dans la représentation requise, lesquelles sont spécifiées par les annotations des fichiers WSDL. Les sorties de l'opération sont les valeurs transformées, qui ont la signification désirée par l'opération cible dans la composition.

Mise à jour de la composition originale :

Les invocations des services Web médiateurs générées lors de la deuxième étape sont insérées dans le code BPEL original en suivant l'algorithme de Contextualisation BPEL .Le code BPEL inséré utilise les URI de référence (endpoint) des services Web médiateurs générés dynamiquement, en combinaison avec les éléments <invoke> nécessaires. Un exemple générique de code pour l'invocation de services Web médiateurs est décrit dans le listing 4.1. Après Avoir remplacé tous les flux de données implicites et explicites, le processus métier contextualisé est prêt à être exécuté par n'importe quel moteur d'exécution BPEL classique. Pendant l'exécution du processus métier contextualisé, les services Web médiateurs sont Invoqués afin de résoudre les hétérogénéités sémantiques des données à l'aide du contexte.

3.2.4. Génération dynamique du processus contextualisé

Le choix de WS-BPEL :

WS-BPEL a été conçu pour la < programmation à large échelle > pour décrire les flux de données de manière explicite. Les flux de données dans WS-BPEL sont encapsulés dans des éléments <variable>. Nous distinguons les flux de données décrits dans le processus

métier et partagés entre les services Web de manière implicite de ceux copiés explicitement via des éléments <assign>. Notre approche consiste à localiser les deux types de flux et à les remplacer par des invocations de services Web médiateurs.

Intégration des services web médiateur dans WS-BPEL :

Considérons les flux de données explicites décrits avec des éléments <assign>. De tels éléments contiennent un ou plusieurs éléments <copy>, eux-mêmes contenant un élément <from> et un élément <to>, qui décrivent respectivement d'ou les données viennent et ou elles vont. Le travail de médiation concerne les éléments qui sont assignés d'une variable à une autre seulement. En effet, les données entrées manuellement par le concepteur de la composition (expression ou valeurs littérales) respectent la sémantique du processus métier. Afin d'intégrer les services Web médiateurs dans WS-BPEL, nous remplaçons les éléments <assign> sélectionnés par une invocation au service Web Médiateur généré, à l'aide de la séquence décrite dans le listing suivant :

```
<sequence>
<assign>
<copy>
<from variable =‘‘source ncname” part= ‘‘ncname”/>
<to variable =‘‘mediation_input_ncname” part=‘‘ncname”/>
</ copy>
</ assign>
<! --- Call to the mediator service Web here--->
<invoke name=‘‘mediation” partnerLink=‘‘mediator”
PortType =‘‘cm:ContextMediator” operation =‘‘mediate”
InputVariable =‘‘mediation input ncname”
OutputVariable =‘‘mediation output ncname”/>
<assign>
<copy>
<from variable =‘‘mediation output ncname” part=‘‘ncname”/>
<to variable =‘‘destination_ncname” part=‘‘ncname”/>
</ copy>
</ assign>
<Sequence>
```

Listing 4.1 : Invocation de médiateur dans WS-BPEL

Explication du listing :

L’élément <sequence> :

L’élément <sequence> permet une exécution consécutive des éléments contenus, qui construisent d’abord le message d’entrée du service Web médiateur, à l’aide d’une activité <assign>.

Extraction des données :

Le médiateur est invoqué avec l'élément <invoke> suivi d'une activité <assign> qui extrait les données transformées du message de sortie vers la variable de destination. En remplaçant l'élément <assign> original avec le code BPEL présenté dans le listing 4.1.

Interception et adaptation des flux de données :

Le service Web médiateur est inséré dans la composition BPEL afin d'intercepter et d'adapter le flux de données. Afin de gérer les flux de données implicites, nous avons besoin de localiser les variables partagées, c'est-à-dire les variables qui sont d'abord utilisées comme sortie d'un élément <invoke>, et ensuite directement en entrée d'un prochain élément <invoke> consécutivement appelé. Dans le langage BPEL, cette situation peut arriver dans les cas suivants :

1. un élément <sequence> contient plusieurs éléments <invoke>,
2. un élément <flow> contient plusieurs éléments <invoke> qui sont liés par un élément <Link>.

Localisation des éléments et vérification des liens parenté entre les éléments :

Dans les deux cas précédemment cité, nous localisons les éléments <invoke> et vérifions que leurs attributs inputVariable et outputVariable sont égaux. Pour identifier le premier cas seulement, nous vérifions aussi que les éléments <invoke> sélectionnés sont enfants du même élément <sequence>. Pour identifier le second cas, nous vérifions aussi que les éléments <invoke> sélectionnés sont enfants du même élément <flow>, et ont respectivement un élément fils <source> et un élément fils <target> avec le même attribut linkName.

L'algorithme de contextualisation :

L'algorithme décrit ci-dessous montre la détection décrite précédemment des flux de données implicites et explicites dans un processus métier écrit dans le langage WS-BPEL, ainsi que les modifications apportées pour insérer le code de médiation décrit dans le listing 4.1.

Algorithme 1 Algorithme de Contextualisation BPEL

1: **for all** assign ∈ findElements (< assign >) **do**

2: in ← getFromElement (assign)

3: out ← getToElement (assign)

4: **if** in ∈ <variable> **and** out ∈ <variable> then

5: newAssign ← createMediationSeq (in; out)

6: replace (assign; newAssign)

7: **end if**

8: **end for**

9: **for all** seq ∈ findElements (< sequence >) **do**

10: **for all** (a, b) ∈ (getInvokeChildren($s\ e\ q^2$))**do**

11: **if** getOutputVar (a) = getInputVar (b) **and** isBefore (a, b) **then**

12: mediationCode ← createMediationSeq (getOutputVar (a); getInputVar (b))

13: insertBefore (b; mediationCode)

14: **end if**

15: **end for**

16: **end for**

17: **for all** flow ∈ findElements (< flow >) **do**

18: **for all** (a, b) ∈ (getInvokeChildren($f\ l\ o\ \hat{w}$)) **do**

19: **if** getOutputVar (a) = getInputVar (b) **then**

20: **if** a.hasChild (< source >) **and** b.hasChild (< target >) **then**

21: src ← a.getChild (< source >)

22: *target* ← *b.getChild* (< *target* >)

23: **if** *getLinkName* (*src*) = *getLinkName* (*target*) **then**

24: *mediationCode* ← *createMediationSeq* (*getOutputVar* (*a*)*; getInputVar* (*b*))

25: *mediationCode.getChild* (< *sequence* >)*.append* (*b*)

26: *replace* (*b; mediationCode*)

27: **end if**

28: **end if**

29: **end if**

30**: end for**

31**: end for**

Explication de l'algorithme de contextualisation :

La première partie de l'algorithme (lignes 1 to 8), détecte les flux de données explicites décrits avec les éléments <assign>. La fonction *findElements* est utilisée pour localiser les éléments <assign>. Ensuite, les éléments fils <from> et <to> sont extraits de chaque élément <assign> (lignes 2-3) et si les deux sont des variables (ligne 4) ils sont utilisés par la fonction *createMediationSeq* pour créer le code de médiation (ligne 5) qui remplace l'élément <assign> précédent (ligne 6).

Les lignes 9 à16 montrent la détection des éléments <invoke> consécutifs dans une séquence. La fonction *getInvokeChildren* (*sequence*) prend les éléments fils <invoke> qui appartiennent à la même séquence. Les fonctions *getInputVar* (*invoke*) et *getOutputVar* (*invoke*) sont utilisées pour extraire l'information contenue dans les attributs respectifs inputVariable et outputVariable de l'élément <invoke> sélectionné.

La fonction *isBefore* (*a; b*) vérifie que l'élément *a* est exécuté avant l'élément *b* dans le code BPEL. Donc, si deux éléments <invoke> d'une séquence (ligne 9-10) ont des variables d'entrée et de sortie correspondantes et suivent l'ordre d'exécution attendu (ligne11), le code de médiation (ligne 12) est inséré juste avant la deuxième opération <invoke> à l'aide de la fonction *insertBefore* (ligne 13), pour que la médiation soit seulement effectuée si nécessaire. En effet, d'autres éléments tels que <Switch> pourraient changer l'exécution du processus métier.

Les lignes 17 à 31 montrent la détection des éléments <invoke> en correspondance dans un *flow*. L'algorithme identifie les éléments <invoke> qui ont des attributs *inputVariable* et *outputVariable* identiques, ce qui caractérise la présence possible d'un flux de données implicite (lignes 17-19). Ces éléments doivent être mis en correspondance à l'aide d'éléments fils <source> et <target> qui ont le même attribut *linkName* (lignes 20-23). Dans ce cas, le code de médiation généré (ligne 24) inclut le second élément <invoke> (ligne 25) qui est ajouté par la fonction *append*. Ainsi, il remplace l'élément <invoke> original avec une séquence qui inclut à la fois le code de médiation et l'invocation originale.

L'algorithme présenté ci-dessus est essentiel afin de générer le BPEL contextualisé, il permet d'entrelacer les opérations de médiation dans le processus métier original, en incluant des appels aux services Web médiateurs.

3.3. La couche description

Comprend une représentation commune du domaine de connaissance. Cette représentation est constituée d'une ontologie de domaine et d'un ensemble d'ontologies contextuelles associées aux concepts des ontologies de domaines.

3.3.1 Intégration du contexte dans la description des Services Web

Le concept sémantique attaché aux données est décrit dans une ontologie de domaine le concept contexte est explicitement détaillé en utilisant des méta-attributs additionnels appelés modifieurs, dont la sémantique est décrite dans une ontologie contextuelle. Cependant, pour pouvoir exploiter cette information supplémentaire, il est nécessaire de fournir une méthode pour l'intégrer dans la pile de protocoles des services Web. Cette section décrit notre proposition pour réaliser cette intégration, par l'utilisation d'ontologies contextuelles et d'une annotation du langage de description des services Web WSDL.

Ontologies de domaine et contextuelles :

Une ontologie contextuelle a pour but de décrire le contexte d'un concept de l'ontologie de domaine. Ainsi, à chaque concept d'une ontologie de domaine, on associe une ontologie contextuelle. La mise en place d'une ontologie contextuelle est simple. Il s'agit d'une ontologie classique, qui modélise les différentes propriétés sémantiques d'un concept de l'ontologie de domaine, ainsi que leurs relations.

Inconvénients liés à l'utilisation des ontologies de domaine :

1. certaines parties des ontologies restent implicites.
2. les ontologies imposent un contexte unique d'interprétation, qu'il soit décrit explicitement ou pas dans l'ontologie.

Objectifs liés à l'utilisation des ontologies :

1. limiter le rôle des ontologies de domaine sur la description des connaissances qui peuvent être décrites en suivant une approche < top-down >.
2. adopter une approche < bottom-up > pour réconcilier les contextes des services Web.
3. laisser aux fournisseurs de services la responsabilité de décrire leurs contextes locaux, et d'expliciter les correspondances avec les contextes des autres fournisseurs lorsqu'ils adhérent à l'ontologie de domaine.

Exemple :

Une ontologie contextuelle associée au concept < prix > précise les diverses propriétés sémantiques utilisées par les fournisseurs, comme la devise qui lui est associée, ou l'inclusion de la TVA dans le prix. Ainsi, les difficultés liées à l'obtention d'un accord sur une représentation partagée d'un domaine de connaissances sont explicitement décrites dans les ontologies contextuelles.

La séparation entre ontologies de domaine et ontologies contextuelles :

Cette séparation permet de résoudre les conflits liés aux contextes des utilisateurs lors de la mise à jour des ontologies contextuelles. En effet, lors de l'adhésion à une ontologie de domaine, les fournisseurs doivent mettre à jour les ontologies contextuelles associées aux concepts de l'ontologie de domaine, avec les propriétés sémantiques qu'ils jugent nécessaires à l'interprétation correcte des concepts.

Exemple :

Supposons qu'un fournisseur anglophone utilise le mot <currency> pour décrire la devise dans laquelle le prix est exprimé, et qu'un fournisseur francophone utilise le mot < devise >. Lors de la mise à jour de l'ontologie contextuelle, une relation d'équivalence est ajoutée par le dernier fournisseur à l'ontologie contextuelle, qui identifie la propriété sémantique existante comme équivalente à celle qu'il vient d'ajouter.

Les langages de description tels que OWL permettent d'établir des relations sémantiques complexes entre les différents contextes, et ainsi apportent les capacités de raisonnement nécessaires pour résoudre les hétérogénéités entre les différents contextes des Fournisseurs. La figure 4.6 montre la séparation entre ontologies contextuelles et de domaine, et clarifie la terminologie utilisée pour la description du contexte.

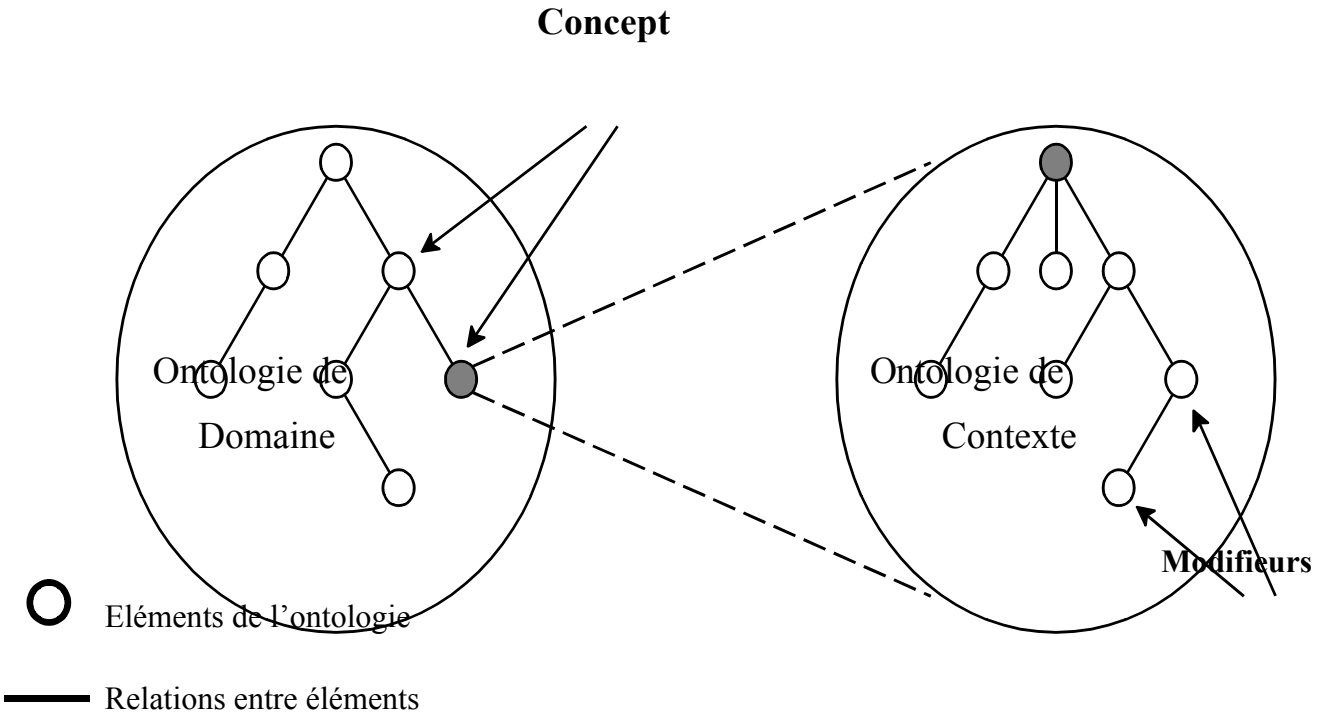


Figure 4.6 : Séparation des ontologies contextuelles et de domaine

3.3.2. Intégration des modifieurs statique et dynamique

Les ontologies contextuelles fournissent les vocabulaires qui permettent de spécifier les différentes représentations structurelles et sémantiques des modifieurs. Cependant, il est aussi nécessaire de spécifier les valeurs de ces modifieurs. Nous proposons deux solutions différentes, une pour les modifieurs dynamiques et une pour les modifieurs statiques. En effet, nous avons établi précédemment que les modifieurs statiques doivent être explicitement décrits pour clarifier l'interprétation d'un objet sémantique, alors que les valeurs des modifieurs dynamiques peuvent être calculées, via des mécanismes de raisonnement, à partir des valeurs d'autres modifieurs du contexte.

La solution que nous proposons consiste à insérer les noms et valeurs des modifieurs statiques nécessaires à la construction de l'objet sémantique dans la description WSDL du service Web, de telle sorte que notre approche reste compatible avec la pile de protocole standard des services Web. Les valeurs des modifieurs dynamiques nécessaires sont calculées par la suite à partir des valeurs des modifieurs statiques, en utilisant les règles appropriées.

Nous présentons ci-dessous notre méthode pour l'insertion des noms et valeurs des modifieurs statiques dans une description WSDL.

3.3.3 .Annotation contextuelle de WSDL

La mise en place de notre architecture de médiation nécessite l'enrichissement des descriptions de services Web avec l'information contextuelle. Il est nécessaire d'associer aux données qui vont être échangées entre les services Web le contexte qui permettra leur interprétation correcte.

Pour ce faire, nous proposons une solution reposant sur l'annotation des parties de messages décrites dans WSDL. Cette annotation a pour but de rendre explicite le contexte des données, de manière à ce que ces dernières puissent être traitées comme des objets sémantiques. La transformation des données en objets sémantiques par l'ajout du contexte permet d'effectuer la médiation au niveau sémantique.

Notre annotation doit respecter les éléments d'extensibilité fournis par la spécification WSDL et on doit définir à quelle endroit il est nécessaire de placer cette dernière.

Comme expliqué dans le chapitre 3,Chaque opération proposée par un service Web possède un message d'entrée représenté par un élément <input> et un message de sortie représenté par un élément <output>, chacun étant composé de plusieurs parties symbolisées par des éléments <part>, que nous appelons <paramètres>. Chaque paramètre d'une description WSDL possède un attribut <name> et un attribut <type> qui représentent respectivement le nom du paramètre et le type dans lequel la valeur du paramètre est représentée.

La spécification WSDL autorise l'ajout d'attributs supplémentaires à la suite de ces deux attributs. La figure 4.7 détaille le metamodele WSDL, en mettant en valeur notre annotation contextuelle, encadrée en pointilles.

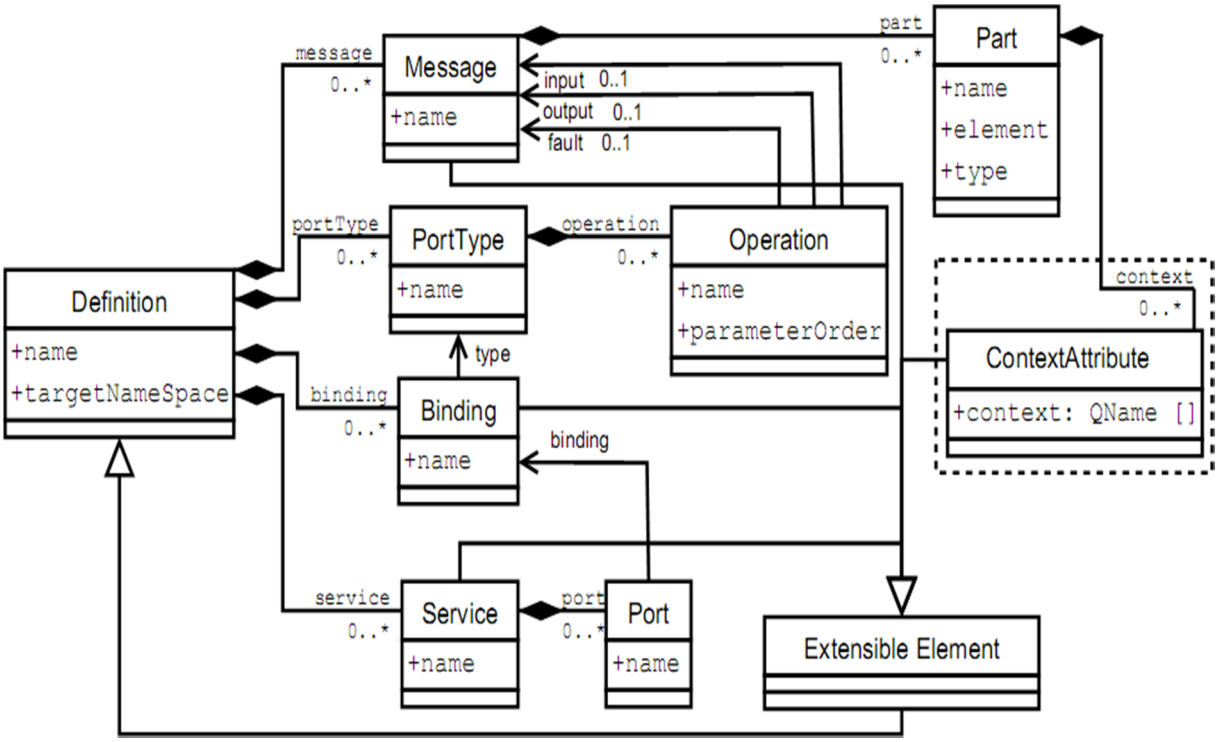


Figure 4.7 : Représentation du contexte dans le metamodele WSDL

Nous annotons les éléments <part> d'une description WSDL avec un attribut contexte qui décrit les noms et valeurs des modifieurs, en utilisant une liste de noms qualifiés. Nous associons le premier nom qualifié de la liste au concept de l'ontologie de domaine auquel le paramètre est rattaché, *C* dans la définition de l'objet sémantique. Cette information nous permet d'identifier le concept de domaine concerné par l'annotation.

```
<? Xml version= ‘‘1.0” encoding=‘‘UTF-8”?>
<wsdl: definitions . . .>. . .
<wsdl: message name=‘‘B_log”>
<wsdl: part name=‘‘inputPrix”    type=‘‘xsd: double”
ctxt: context =‘‘dom1: Prix   ctxt1 : Algérie
ctxt1: TVAIncluded ctxt1: ScaleFactorOne”/>
</wsdl: message>. . .
</ wsdl: definitions>
```

Listing 4.2: B_log Annotation (fragment)

Illustration de l'annotation B_log :

Le listing 4.2 illustre l'extension proposée avec le service Web < B_log >. Le premier élément de notre annotation identifie le concept < Prix > qui identifie l'objet sémantique tel qu'il est décrit dans l'ontologie de domaine représentée par l'espace de nommage < dom1 >, puis les éléments suivants identifient les valeurs des modifieurs statiques du contexte. Dans notre exemple, ils indiquent que le pays d'origine du prix est l'Algérie, que la TVA est incluse dans le prix, et que le facteur multiplicateur utilisé est 1.

Grâce à cette annotation, une valeur *v* et son type *t* décrits dans un document WSDL sont enrichis avec le concept *c* et les modifieurs nécessaires pour définir le contexte *C*, formant ainsi un objet sémantique $\boldsymbol{S} = (c; v; t ; C)$. Des règles logiques permettront d'inférer les valeurs des modifieurs dynamiques nécessaires pour compléter le contexte *C*, pendant l'étape d'exécution de la composition.

Les règles logiques offrent, par leur utilisation, de nombreux avantages :

- Elles sont facilement modifiables, ce qui améliore leur adaptabilité à de fréquents changements.
- Elles permettent une compréhension et une transmission plus évidente des connaissances que le code d'un programme.
- Elles permettent d'obtenir des valeurs de modifieurs qui varient dans le temps et ne peuvent pas être stockées dans la description d'un service, comme les valeurs des taux de conversion de devises ou de TVA.
- Elles conservent l'indépendance entre la logique applicative et le reste du système. Ainsi, leur modification ne requiert pas la réécriture ni de recompilation du code de l'application.

3.3.4. Intégration du contexte : avantages

L'utilisation d'ontologies contextuelles et d'annotations de descriptions WSDL présente de nombreux avantages :

- Elle aide les fournisseurs à décrire le contexte des données échangées entre les services Web de manière explicite et compréhensible par les machines.

- Elle facilite l'intégration du contexte dans la pile standard de protocoles des services Web.
- Elle facilite la mise à l'échelle par le découplage des ontologies contextuelles et de domaine.
- Elle facilite la médiation sémantique des données échangées durant l'exécution de la composition.

4. Exemple d'application

Dans cette section, nous voulons présenter l'implémentation de quelques composants du Processus métier « commande marchandise ifri ». Cet exemple présente notre scénario de composition dite « CMDmarch» permettant de passer une commande en ligne et « b_log» qui permet de louer un moyen de transport. L'utilisateur peut utiliser en entrée plusieurs arguments comme :

Identifiant client, Référence marchandise, type marchandise, quantité de marchandise …etc.

Pour implémenter cette application, nous voulons composer plusieurs Web services

. Le scénario suivant présente la composition des trois Web service (notre cas illustratif).

- CMDmarch : introduire la requête du client.
- B_log : retourne le prix d'une course de transport en Dinard algérien.
- Addition : convertie et retourne un prix TOTALE en Dinard algérien

.

Web service	entré	sortie
CMDmarch	Idclient ou type marchandise ou référence marchandise	Prix (euro) Idclient
B_log	idclient	B_lognumber, startdate/startTime, prix(dinard)
Addition	Prix (dinar),prix (euro)	TOTALE prix (euro)

Tableau 4.1 : Entrées/sorties des services Web candidats.

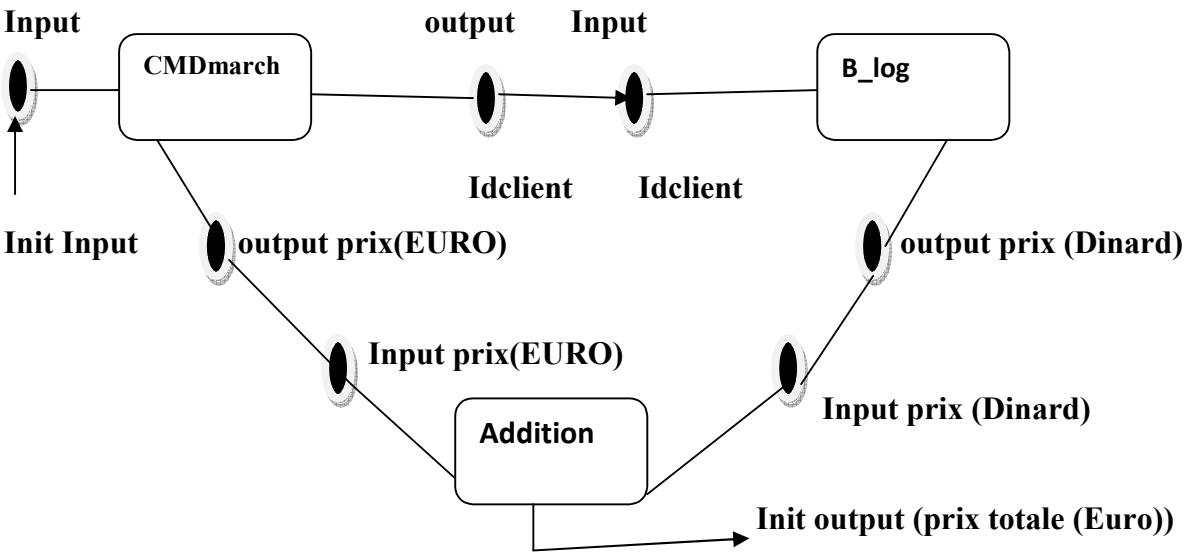


Figure 4.8 : Exemple d'un Web service composé par les 3 web service.

4.1. Présentation des outils technologiques utilisés

Afin de développer l'application permettant aux utilisateurs de générer leur requête et d'obtenir les résultats de celle-ci nous avons opté pour les langages suivants :

a) Langages

- Le langage de développement à utilisé est le langage Java dans sa version d'entreprise "Java EE 5 SDK". Le kit Java EE 5 SDK peut être téléchargé à partir du lien :
  http://java.sun.com/javaee/
- L'autre langage important à utilisé pour développer notre application est le langage WS-BPEL2.0 (ou simplement BPEL), ce dernier a été utilisé pour réaliser la composition des Web services.
- Les autres langages à utilisés pour développer notre application sont le langage OWLDL et OWL-S.

b) Serveur

Apache axis2 est un projet open source généralement utilisé pour faciliter le déploiement du service web sur un serveur tomcat, en effet Axis2 offre certaine fonctionnalité qui permet de générer les fichiers de description essentielle pour la création et le déploiement d'un service web.

c) Système de gestion de base de données

SGBD Java DB version 10.4 inclus dans la Java SE Development Kit pour implémenter notre base de données qui stockera les services web, il est disponible sur le site : http://developers.sun.com/javadb/

d) algorithme de contextualisation

notre algorithme de contextualisation à implémenté est sous WS-BPEL2.0.

e) Environnement de développement

Pour développer quelques composants de notre architecture, nous opteront pour Netbeans comme environnement de développement dans sa version 6.5 téléchargeable à partir de : http://www.netbeans.org.

f) Bibliothèque Jena

Jena est une bibliothèque java pour développer des applications du Web sémantique, elle fournit un environnement de programmation pour RDF, RDF Schéma, et OWL. Elle peut être téléchargée à partir du lien : http://jena.sourceforge.net/

g) Editeur d'Ontologies OWL

Néanmoins, *Protégé* n'est pas un outil spécialement dédié à OWL, mais un éditeur hautement extensible, capable de manipuler des formats très divers. Disponible à : http://protege.stanford.edu/

4.2. Résumé sur le fonctionnement des services web médiateur

Dans cette section, nous détaillons les fonctionnalités et mécanismes internes des services Web médiateurs. Ces derniers sont insérés dans le processus de composition entre les services Web qui participent à la composition originale et qui pourraient présenter des hétérogénéités de contexte. Considérons le flux de données entre le service< B_log > et le service < Addition >, qui est intercepté par le service Web médiateur inséré entre les deux. Le service Web médiateur prend comme entrée la Partie de message < Prix > envoyée par le service Web < B_log > ensuite il effectue les étapes décrites par la figure 4.8, avant d'envoyer le résultat de ses calculs dans une partie de message < Prix >, au service Web < Addition>.

Processus de composition

SW médiateur

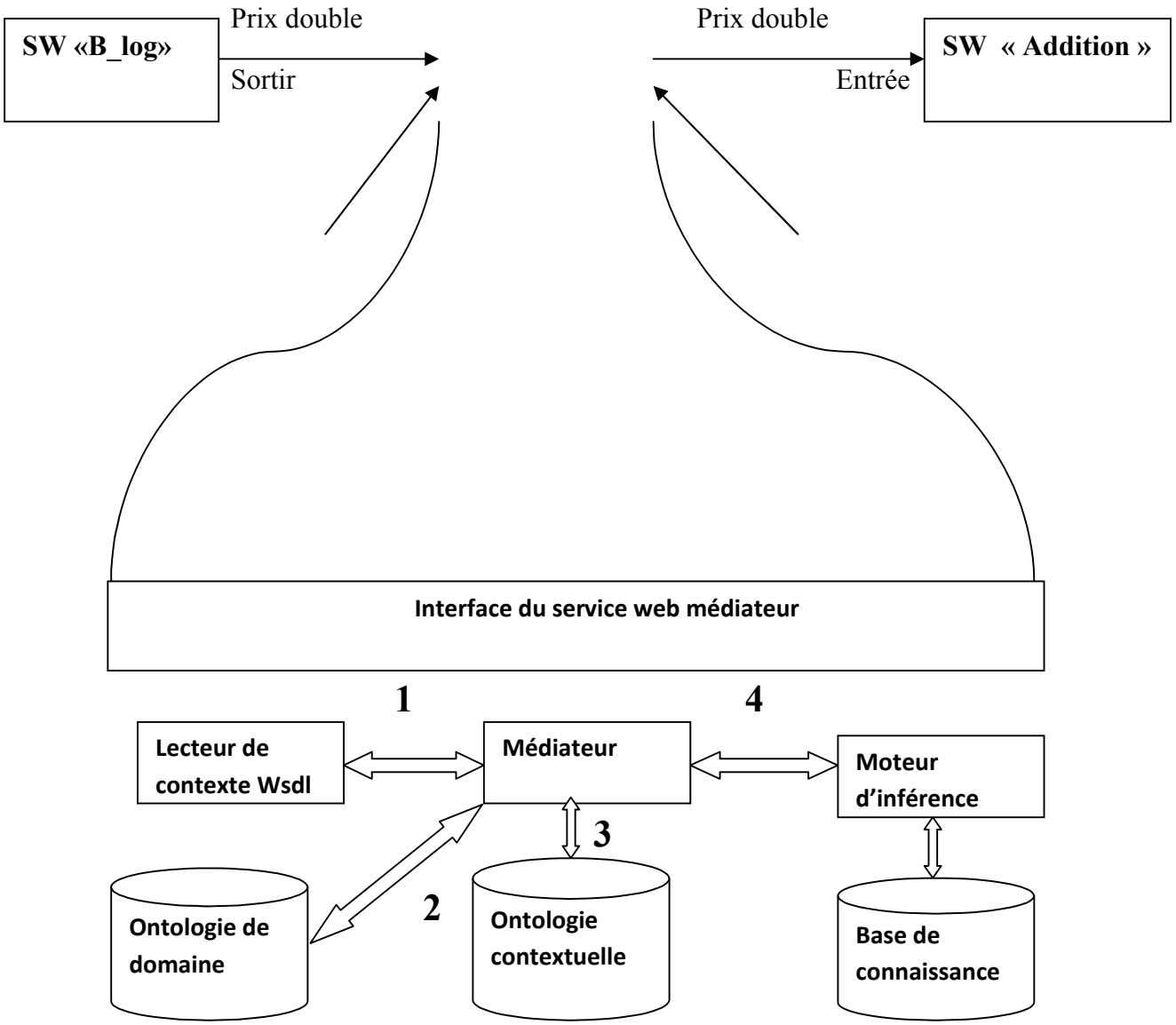


Figure 4.9 : Vue détaillé du service Web médiateur.

Les étapes de médiation effectuées par les services Web médiateurs

(suivant la figure 5.2) :

Première étape : (1)

Téléchargement des fichiers WSDL des services Web < B_log> et <Addition> par service Web médiateur. Ces fichiers sont analyses pour extraire les éléments requis. En particulier, nous sommes intéressés par les annotations des parties de messages que le service Web médiateur reçoit et envoie, dans notre exemple, les parties de message nommées *Prix* de sortie et d'entrée des services Web <B_log> et <Addition> respectivement. Les annotations extraites font référence au concept de l'ontologie de domaine et aux éléments du contexte nécessaires à une interprétation correcte des données.

Deuxième étape : (2)

Identification des concepts échangés dans les ontologies de domaine. L'annotation est une liste de méta-attributs, dont le premier attribut renvoie au concept de référence de l'ontologie de domaine. Dans notre exemple, les parties de message annotées font référence au concept *Prix* de l'ontologie de domaine identifiée par le namespace dom1 du fichier WSDL. Le service Web médiateur vérifie que les concepts utilises par les services Web <B_log> et <Addition> correspondent, c'est-à-dire qu'ils vérifient une relation de subsomption où d'équivalence, laquelle peut être vue comme un cas particulier de subsomption réciproque entre deux concepts. Cette approche de correspondance sémantique est simpliste, cependant des capacités additionnelles peuvent être intégrées au médiateur.

Troisième étape : (3)

Pour chaque service, une représentation en mémoire sous forme de structure arborescente du contexte de l'objet sémantique *Prix* est construite à partir de l'ontologie contextuelle. Les annotations contextuelles du concept *Prix* sont identifiées dans les ontologies contextuelles et leurs valeurs sont ajoutées pour instancier la structure. En effet, les annotations contextuelles font référence à des instances OWL, donc elles décrivent non seulement la sémantique et la structure des modifieurs, mais aussi les valeurs qu'ils prennent dans le contexte du service Web.

Nous considérons que les fournisseurs de services Web insèrent correctement l'information relative à leur contexte dans l'ontologie contextuelle avant d'annoter les fichiers

WSDL. Ainsi, les modifieurs statiques du contexte sont identifiables et utilisables par les services Web médiateurs. Si nécessaire, l'interprétation de ces modifieurs peut être encore précisée avec des annotations contextuelles supplémentaires.

Quatrième étape : (4)

Le service Web médiateur communique avec un moteur d'inférence pour effectuer deux opérations. La première consiste à déduire les valeurs des modifieurs dynamiques appartenant au contexte, en utilisant les règles logiques stockées dans la base de connaissances du moteur d'inférence. Dans notre exemple, sachant que le modifieur statique country du service <Addition> possède la valeur Algérie donnée par la description WSDL, le moteur d'inférence en déduit que le modifieur dynamique currency du service possède la valeur Dinar en questionnant sa base de connaissances, qui contient une règle associant la devise Dinar au pays Algérie. Ainsi, l'attribut contextuel currency se voit affecte la valeur Dinar. De cette manière, les services Web médiateurs déduisent les valeurs des modifieurs dynamiques nécessaires pour la conversion des données.

La deuxième opération consiste à effectuer cette conversion des données dans la représentation contextuelle requise. A partir des étapes précédentes, nous obtenons deux arbres de représentation du contexte en mémoire, qui ont des feuilles. Ces feuilles sont des modifieurs contenant des valeurs, formant ainsi le contexte.

Le service Web médiateur compare chaque élément du contexte l'un à l'autre et tente d'établir des correspondances d'équivalence entre les modifieurs des deux contextes grâce à l'information contenue dans l'ontologie contextuelle. Ensuite, il demande au moteur d'inférence de vérifier la convertibilité des valeurs des modifieurs correspondants et d'effectuer leur conversion, si possible. La conversion des valeurs est effectuée à l'aide de fonctions de conversions telles que décrites dans la section précédente. Ces fonctions de conversion sont enregistrées et stockées sous forme de règles logiques dans la base de connaissances consultée par le service Web médiateur. Si une valeur de modifieur manque, le moteur d'inférence cherche dans sa base de connaissances une règle définissant une valeur par défaut. Si une telle règle n'existe pas, la conversion est annulée et une exception est levée.

Considérons notre exemple « cas ifri », avec les valeurs des prix V et leurs facteurs multiplicateurs SF. La règle logique permettant de gérer le modifieur *scaleFactor* est stocké dans la base de connaissances comme suit :

$$V_{target} = \frac{V_{source} * SF_{source}}{SF_{target}}$$

Où ***source*** est le contexte d'origine des données.

Target est le contexte vers lequel les données doivent être converties.

5. Conclusion

Dans ce chapitre, nous avons présente notre architecture de médiation contextuelle, pour résoudre les hétérogénéités sémantiques entre services Web. Cette architecture est construite autour de deux objectifs qui sont partie intégrante de notre problématique : l'intégration du contexte dans la description des services, et l'intégration de la médiation dans la composition de services.

Afin de remplir notre premier objectif, nous avons proposé :

– l'utilisation d'ontologies contextuelles couplées à des ontologies de domaine, afin de simplifier l'adhésion aux ontologies de domaine en limitant les contraintes de représentation sémantique imposées aux fournisseurs. La sémantique locale de chaque fournisseur et les correspondances avec celles des autres fournisseurs sont spécifiées par le biais des ontologies contextuelles.

– l'annotation des descriptions de services, afin de fournir l'information contextuelle pouvant répondre aux besoins de la médiation sémantique tout en conservant une compatibilité des documents avec la spécification WSDL.

Pour atteindre notre deuxième objectif, nous avons adopte une approche orientée service. Nous avons propose une méthode de contextualisation de composition, qui permet d'insérer des services Web médiateurs au sein de la composition. Cette méthode comprend :

– une étape de détection des hétérogénéités sémantiques potentielles dans la composition,

– une étape de génération des services Web médiateurs nécessaires,

– une étape de mise à jour du processus métier décrivant la composition initiale par l'ajout des appels aux services Web médiateurs.

Conclusion Générale & Perspectives

Au cours de ce mémoire de recherche nous nous sommes intéressés à la modélisation et la composition des services web en se basant sur la médiation. C'est un domaine de recherche très actif qui se situe à la convergence du web et l'intelligence artificielle. Il vise à offrir des services adapté au contexte de l'utilisateur, et qui respecte l'interopérabilité des formats. Cependant, beaucoup de travaille reste à faire dans ce domaine sur le plan d'automatisation de la composition au niveau des services web. C'est ainsi que les travaux présentés dans ce mémoire ont fait l'objet d'une composition et une résolution du problème d'hétérogénéités d'un prototype illustratif du cas ifri entre un service de passation de commande et un service de transport à base d'une architecture de médiation qui s'appuient sur l'architecture orienté service et le web sémantique.

Nous avons adapté une approche de médiation construite selon un processus métier (cas ifri). Cette approche orientée service favorise l'indépendance entre les fonctionnalités originelles des services Web et les besoins de la médiation, tout en restant homogène avec son environnement d'intégration, qui est lui même oriente service.

Nous avons facilité le travail des fournisseurs, grâce à l'utilisation d'ontologies contextuelles, entre la représentation unique imposée par l'utilisation d'ontologies de domaine et la multiplicité des représentations locales adoptées par les services Web et une annotation des descriptions de services qui reste compatible avec le format de description WSDL.

La notion de contexte, autour de laquelle nous avons adapté notre solution de médiation sémantique, apporte de nombreux avantages, et nous pensons que les possibilités d'évolutions envisageables sur la base de notre travail sont multiples. Des perspectives restent ouvertes, non seulement dans le domaine des services Web, mais d'une manière plus générale dans les divers domaines concernes par l'interopérabilité des données.

Perspectives envisagées

Comme perspective, nous envisageons les points suivants :

- Prise en compte de la sécurité et de la confidentialité lors du processus de composition de service web.
- Prise en compte de l'authentification de l'utilisateur lors de l'établissement de la communication entre le service basique et l'utilisateur.
- Construire un model complet de gestion de la médiation (avec sa description de contexte, son objet sémantique son annotation, son intégration…)

- Construire un vrai processus métier du cas ifri (puisque dans notre cas on a juste supposé un prototype illustratif afin de soutenir notre solution de médiation pour la composition tout en résolvant le problème d'hétérogénéités).
- introduire le paramètre de qualité de service (QoS) en termes de couts, de temps d'exécution, des services pertinents…etc.
- Enfin implémenter un vrai prototype pour l'architecture proposé dans ce mémoire.

[1]: M. Stevens, "*SOA*" November 2009,

[2]: Y. V. Natis, R. W. Schulte "Introduction to Service Oriented Architectue », Gartner, 2003.

[3]: B. **A.** Christudas, M. Barai, V. Caselli, "*Service Oriented Architecture with Java, Using SOA and web services to build powerful Java applications* ", Edition Packet Publishing, 2008.

[4] : Antoine CROCHET-DAMAIS, JDN Solution, "*Comprendre les Architecture Orientée Services (SOA)* ", 2005. http://www.journaldunet.com/solutions/dossiers/pratique/soa.

[5]: M. Rosen, B. Lublinsky, K. T. Smith, M. J. Balcer, "Applied *SOA ServiceOriented Architecture and Design Strategies"* Edition Wiley Publishing, 2008.

[6]: M. B. Juric, B. Mathew, P. Sarang, *"Business Process Execution Language for Web Services",* 2eme Edition Packet Publishing, 2006.

[7]: M. Stevens **"***Service-Oriented Architecture Introduction*", 2002. http://www.developer.com/services/article.php/1010451.

[8] : X. Fournier-Morel, P. Grojean, G. Plouin, C. Rognon, "SOA le guide de l'architecte ", Edition Dunod, 2006.

[9]: M. Paolucci, K. Sycara, T. Nishimura and N. Srinivasan**,** "Using DAML-S for P2P Discovery", *In the Proc of International Conference on Web Services (ICWS).* Las Vegas, Nevada, USA, 2003, pp. 203-207.

[10] : S. Hammoudi et D. Lopes, "*Introduction aux Services Web*", 2003 http://www.dee.ufma.br/~dlopes/course/WebServices/presentation-WS-3.3_economic.pdf.

[11] : A. VEZAIN, "*LES SERVICES WEB – Présentation générale*", Version 0.0.1, Association HE.R.M.E.S, Château du Montet, 2005.

[12] : T. MELLITI *" Interopérabilité des services Web complexes. Application aux systèmes multi-agents* ", thèse de doctorat, Université Paris IX Dauphine 2004.

[13] : Y. BELAID, "*SERVICE WEB – SOAP*", Centre d'enseignements de Grenoble, 2009.

[14]: Simple Object Access Protocol (SOAP) 1.1, spécification W3C, 2000.

[15]: E. Christensen, F. Curbera, G. Meredith, S. Weerawarana « *Web Service Description Language (WSDL) 1.1* », spécification W3C, 2001.

[16] : J-M Chauvet «*Services Web avec SOAP, WSDL, UDDI, ebXML…*» Edition EYROLLES, 2002.

[17]: Alan W Brown. *Component-based software engineering: selected papers from the Software Engineering Institute.* Los Alamitos, CA: IEEE Computer Society Press, 1996.

[18]: Clemens Szyperski "*Component software: beyond object-oriented programming"*. NewYork: ACM Press Harlow, England Reading, Mass: Addison-Wesley, 1997.

[19]: C. Pfister et C. Szyperski. Why objects are not enough. In *Proceedings, International Component Users Conference* , Munich, Germany, 1996.

[20]: F. Casati, et M.-C Shan, "*Event-Based Interaction Management for Composite Eservices in eFlow",* Information Systems Frontiers*,* 4(1), pp. 19-31, 2002.

[21] : P. Kellert, et F. Toumani, "*Les services web sémantiques*", Revue I3 (Information-Interaction-Intelligence), 2006.

[22] : K. Boukadi, "*Coopération interentreprises à la demande:Une approche flexible à base de services adaptables* ", thèse de doctorat, l'École Nationale Supérieure des Mines de Saint-Étienne, 2009.

[23] : Saïd IZZA, " *Intégration des Systèmes d'Information* : *Une approche flexible basée sur les services sémantiques* ", Thèse de doctorat, l'École Nationale Supérieure des Mines de Saint-Étienne, 2009.

[24]: T. Berners-Lee, J. Hendler, et O. Lassila, "*The Semantic Web*".Scientific Amercian, 2001.

[25]: W. A. Woods, "*What's in a link: foundations of semantic networks"*.In D.G. Bobrow & A. M. Collins, 1975.

[26] G. Antoniou et F. van Harmelen "*A Semantic Web Primer*", The MIT Press Cambridge, Massachusetts London, England, Cooperative information systems, 2008.

[27]: L. Yu "*Introduction to semantic web and semantic web services* ", Chapman & Hall/CRC, 2007.

[28] : Ressource Description Framework, Standard W3C, http://www.w3.org/RDF/

[29] : P. Laublet, C. Reynaud, J. Charlet "*Sur quelques aspects du Web sémantique*", Actes des deuxièmes assises nationales du GdR I3, 2003.

[30]: T. R. Gruber "A translation approach to portable ontologies", *Knowledge Acquisition* , 5(2):199-220, 1993.

[31]: Web Ontology Language, Standard W3C, http://www.w3.org/TR/owl-features/

[32]: J. Cardoso, "*Approaches to Developing Semantic Web Services*", International Journal of Computer Science, 1 (1), 2006.

[33]: D. Fensel, C. Bussler, A. Maedche: Semantic Web Enabled Web Services. International Semantic Web Conference, 2002.

[34]: OWL-S: "Semantic Markup for Web Services", spécification W3C, http://www.w3.org/Submission/OWL-S/

[35] : C. Bussler, et al., "*WSMO"*. The Eleventh International Conference on Artificial Intelligence: Methodology, Systems, 2004.

[36]: J. Bruijn, H. Lausen, R. Krummenacher, A. Polleres, M. Kifer et D. Fensel, "*Deliverable D16.1v0.2, The Web Service Modeling Language WSML*". Final Draft, *WSMO project,* http://www.wsmo.org/TR/d16/d16.1/v0.2/20050320/, 2005.

[37]: D. Fensel et C. Bussler, "*The Web Service Modeling Framework WSMF*". In Electronic Commerce Research and Applications, 1(2), *Elsevier, Scinces B. V.*, 2002.

[38]: Semantic Web services and Processes, disponible à http://lsdis.cs.uga.edu/projects/meteor-s/

[39] : K. Verma, K. Sivashanmugam, A. Sheth, A. Patil, S. Oundhakar, and J. Miller, *"METEOR-S WSDI: A Scalable Infrastructure of Registries for Semantic Publication and Discovery of Web Services",* Journal of Information Technology and Management, Special Issue on Universal Global Integration, 17–39, 2005.

[40]: WSDL-S, "*Web Service Semantics - WSDL-S, Version 1.0*". W3C Member Submission, http://www.w3.org/Submission/WSDL-S/, 2005.

[41]: M. Hall and L. Brown, “Core Servlets and JavaServer Pages“ Prentice Hall PTR, 2003.

[42]: G. Wiederhold. Mediators in the architecture of future information systems. IEEE Computer, 25(3) :38–49, 2008.

[43]: G. Athanasop oulos, A. Tsalgatidou, and M. Pantazoglou. Interop erability among heterogeneous services. In So ciety [69], pages 174–181.

[44]: C. Bussler. Semantic web services : Reflections on web service mediation and comp osition. In WISE , pages 253–260. IEEE Computer So ciety, 2003.

[45]: L. Cabral and J. Domingue. Mediation of semantic web services in irs-iii. In First International Workshop on Mediation in Semantic Web Services (MEDIATE 2005) held in conjunction with the 3rd International Conference on Service Oriented Computing (ICSOC 2005), Amsterdam, The Netherlands. , Decemb er 12th 2005.

[46]: S. Ran. A mo del for web services discovery with qos. SIGecom Exch. , 4(1) :1–10, 2008.

[47]: B. Benatallah, F. Casati, D. Grigori, H. R. M. Nezhad, and F. Toumani. Developing adapters for web services integration. In O. Pastor and J. F. e Cunha, editors, CAiSE, volume 3520 of Lecture Notes in Computer Science , pages 415–429. Springer, 2005.

[48]: B. Lin, N. Gu, and Q. Li. A requester-based mediation framework for dynamic

invo cation of web services. In So ciety [69], pages 445–454.

[49]: M. Dumas, M. Sp ork, and K. W. 0002. Adapt or p erish : Algebra and visual notation for service interface adaptation. In S. Dustdar, J. L. Fiadeiro, and A. P. Sheth, editors, Business Process Management, volume 4102 of Lecture Notes in Computer Science, pages 65–80. Springer, 2006.

[50]: M. Mrissa, D. Benslimane, C. Ghedira, and Z. Maamar. A mediation framework for web services in a distributed healthcare information system. In IDEAS-DH ’04 : Proceedings of the IDEAS Workshop on Medical Information Systems : The Digital Hospital (IDEAS-DH’04) , pages 15–22, Washington, DC, USA, 2004. IEEE Computer So ciety.

[52]: M. Mrissa, C. Ghedira, D. Benslimane, Z. Maamar, and J. Fayolle. A mediation framework for web services in a p eer-to-p eer environment. In In Proceedings of The 3rd

ACS/IEEE International Conference on Computer Systems and Applications (AICCSA'2005), pages 133–, Cairo, Egypt., 2005. IEEE Computer So ciety.

[53]: M. Nagara jan, K. Verma, A. P. Sheth, J. Miller, and J. Lathem. Semantic interop erability of web services - challenges and exp eriences. In ICWS, pages 373–382. IEEE Computer So ciety, 2006.

[54]: T. Grub er. What is an ontology ? http://www- ksl.stanford.edu/kst/ what- is- an-ontology.html , 2007.

[55]: C. Bornh ¨ ovd. Semantic metadata for the integration of web-based data for electronic commerce. In Int'l Workshop on E-Commerce and Web-based Information Systems (WECWIS), Santa Clara, CA , pages 137–145, 1999.

[56]: C. Bornh ¨ ovd. Mix - a representation mo del for the integration of web-based data, tech. rep. dvs98-1. Technical rep ort, DVS1, Dep. CS, Darmstadt University of Technology, Germany, Nov. 1998.

[57]: C. H. Goh, S. Bressan, S. E. Madnick, and M. Siegel. Context interchange : New features and formalisms for the intelligent integration of information. ACM Trans. Inf. Syst. , 17(3) :270–293, 1999.

[58]: J. Peer. Semantic Service Markup with SESMA. In Web Service Semantics Workshop (WSS'05) at the Fourteenth International World Wide Web Conference (WWW'05) , 2005.

[59]: J. Miller, K. Verma, P. Ra jasekaran, A. Sheth, R. Aggarwal, and K. Sivashanmugam. WSDL-S : Adding Semantics to WSDL - White Pap er. Technical rep ort, Large Scale Distributed Information Systems, 2004. http://lsdis.cs.uga.edu/library/ download/wsdl- s.pdf.

[60]: SAWSDL Working Group. Semantic Annotations for WSDL. W3c working draft, The World Wide Web Consortium (W3C), Sept. 2006. http://www.w3.org/TR/ 2006/WD- sawsdl-20060928/.

[61]: M. Paolucci, T. Kawamura, T. Payne, and K. Sycara. Imp orting the Semantic Web

in UDDI. In Proceedings of E-Services and the Semantic Web Workshop , 2009.

[62]: A. Ankolekar, M. Burstein, J. Hobbs, O. Lassila, D. Martin, D. McDermott, S. McIlraith, S. Narayanan, M. Paolucci, T. Payne, and K. Sycara. DAML-S : Web Service Description for the Semantic Web, 2002.

[63]: A. Dogac, Y. Kabak, and G. Laleci. Enriching ebxml registries with owl ontologies for efficient service discovery. In RIDE , pages 69–76. IEEE Computer So ciety, 2004.

Résumé :

Les services Web constituent la nouvelle génération des technologies du web pour l'intégration d'application. La notion de services Web désigne une application mise à disposition par un fournisseur et invocable sur internet par des clients. L'ambition portée par les services Web st de permettre une plus grande interopérabilité entre application sur internet. On envisage ainsi des services Web capables, automatiquement, de se découvrir et d'être découverts, de négocier entre eux ou de se composer en des services plus complexes.

Cependant, malgré cette large adoption des services Web, de nombreux obstacles empêchent leur réconciliation sémantique lors de la composition. L'interprétation consistante des données échangées entre services Web comp os´es est gênée par des différences de représentation et d'interprétation sémantiques. Dans ce mémoire, nous nous intéressons aux hétérogénéités sémantiques des données échangées entre les services Web engages dans une composition.

Notre travail évolue autour de la notion de contexte pour décrire la sémantique des données. Nous étudions dans quelle mesure le contexte p eut enrichir l'´echange des données entre services Web. Nous proposons un modèle oriente contexte pour représentera sémantique des données, ainsi qu'une approche de médiation qui tire avantage de notre modèle pour résoudre les hétérogénéités sémantiques entre services Web comp os´es.

Mots-clés: services Web, composition, médiation, sémantique, contexte

Abstract :

Web services represent the next generation of the technologies for integration application. The concept of services means a web application made available an Internet provider and invoked by clients. Ambition driven by Web services is to enable greater interoperability between applications on the Internet. There are plans and Web services can, automatically, to discover and be discovered, to negotiate with each other or to deal with more complex services.

However, despite this widespread adoption of Web services, several obstacles still hinder their semantic reconciliation when being composed. Consistent understanding of data exchanged between composed Web services is hampered by different semantic interpretations and representations. In this report, we focus on the semantic heterogeneities of data exchanged between Web services engaged in a composition.

Our contribution revolves around the notion of context in order to describe data semantics. We evaluate to which extent context enriches data exchange between Web services. We propose a context-based model to describe the semantics of data, and a mediation approach that takes advantage of our model to solve semantic heterogeneities between composed Web services.

Keywords: Web services, composition, mediation, semantics, context