Université Claude Bernard Lyon I
UFR Informatique

École doctorale Informatique et Information pour la Société

Médiation Sémantique Orientée Contexte pour la Composition de Services Web

THÈSE

présentée et soutenue publiquement le 15 Novembre 2007

pour l'obtention du

Doctorat de l'Université Claude Bernard Lyon I

(spécialité informatique)

par

Michaël Mrissa

Composition du jury

Rapporteurs :	Nadine Cullot, Professeur à l'Université de Bourgogne Khalil Drira, Chargé de recherches CNRS, Laboratoire LAAS, Toulouse
Examinateurs :	Marie-Christine Fauvet, Professeur à l'Université Joseph Fourier, Grenoble Chirine Ghedira, Maître de conférences à l'Université Lyon 1 (Directrice de thèse) Djamal Benslimane, Professeur à l'Université Lyon 1 (Directeur de thèse)
Invité :	Zakaria Maamar, Professeur associé à l'Université de Zayed, Émirats Arabes Unis

Laboratoire d'InfoRmatique en Images et Systèmes d'information — UMR 5205

Remerciements

Ce travail a été réalisé au sein du Laboratoire d'InfoRmatique en Images et Systèmes d'information (LIRIS) de l'université Claude Bernard Lyon 1, sous la direction de Chirine Ghedira, Maître de conférences à l'université Claude Bernard Lyon 1, et de Djamal Benslimane, Professeur à l'université Claude Bernard Lyon 1, auxquels j'exprime ma profonde reconnaissance pour la confiance qu'ils m'ont accordée dans la réalisation de ce travail.

Je souhaite adresser mes sincères remerciements aux personnes qui ont accepté la tâche délicate de rapporter ce mémoire et qui ont eu la patience de juger ce travail : Nadine Cullot, professeur à l'université de Bourgogne, et Khalil Drira, chargé de recherches au Laboratoire d'Architecture et d'Analyse des Systèmes (LAAS) à Toulouse.

Je souhaite aussi remercier les examinateurs et membres du jury, Marie-Christine Fauvet, Professeur à l'université Joseph Fourier, Grenoble, ainsi que Zakaria Maamar, Professeur associé à l'université de Zayed, Émirats Arabes Unis.

Enfin, je tiens à remercier mes proches et mes ami(e)s pour les moments partagés, ainsi que ma famille, sans qui je ne serai rien.

À mes parents, à mon frère,

Table des matières

Table des figures

Résumé

L'adoption des services Web constitue une avancée majeure dans le développement des systèmes d'information interopérables. En particulier, la composition de services permet de répondre aux besoins de plus en plus complexes des utilisateurs, par la combinaison de plusieurs services Web au sein d'un même processus métier.

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 composés 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 engagés dans une composition.

Notre contribution évolue autour de la notion de contexte pour décrire la sémantique des données. Nous étudions dans quelle mesure le contexte peut enrichir l'échange des données entre services Web. Nous proposons un modèle orienté contexte pour représenter la sémantique des données, ainsi qu'une approche de médiation qui tire avantage de notre modèle pour résoudre les hétérogénéités sémantiques entre services Web composés.

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

Abstract

The adoption of Web services is a keystone in the development of interoperable systems. Particularly, Web services composition allows answering to the increasingly complex users' needs, by combining several Web services into a common business process.

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

Chapitre 1

Introduction

Les dernières décennies ont été marquées par le développement rapide des systèmes d'information distribués, et tout particulièrement par la démocratisation de l'accès à Internet. Cette évolution du monde informatique a entraîné 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[1].

Les services sont implantés par les fournisseurs, qui mettent à disposition les descriptions de services sous forme de fichiers. Ces descriptions sont centralisées et stockées dans des annuaires. La notion d'annuaire est comparable aux annuaires téléphoniques que nous utilisons pour accéder à des personnes ou des services. Les applications clientes envoient des requêtes aux annuaires pour sélectionner les services, de la même manière que nous recherchons un numéro dans un annuaire téléphonique. Elles téléchargent ensuite les descriptions des services sélectionnés, et les invoquent directement. Ces mécanismes sont illustrés par la figure 1.1.

L'objectif de la notion de service est de promouvoir un accès simple et rapide aux fonctionnalités mises à disposition par les organisations. Aussi, la vision des services en tant que composants logiciels indépendants met en exergue les possibilités de coordination, ou composition, de plusieurs services pour fournir des fonctionnalités avancées. Afin de faciliter l'utilisation et la composition des services, les applications doivent réaliser des interactions faiblement couplées, c'est-à-dire qu'elles doivent être capables de fournir et

[1]Le terme « invocation » est généralement utilisé pour évoquer l'utilisation d'un service.

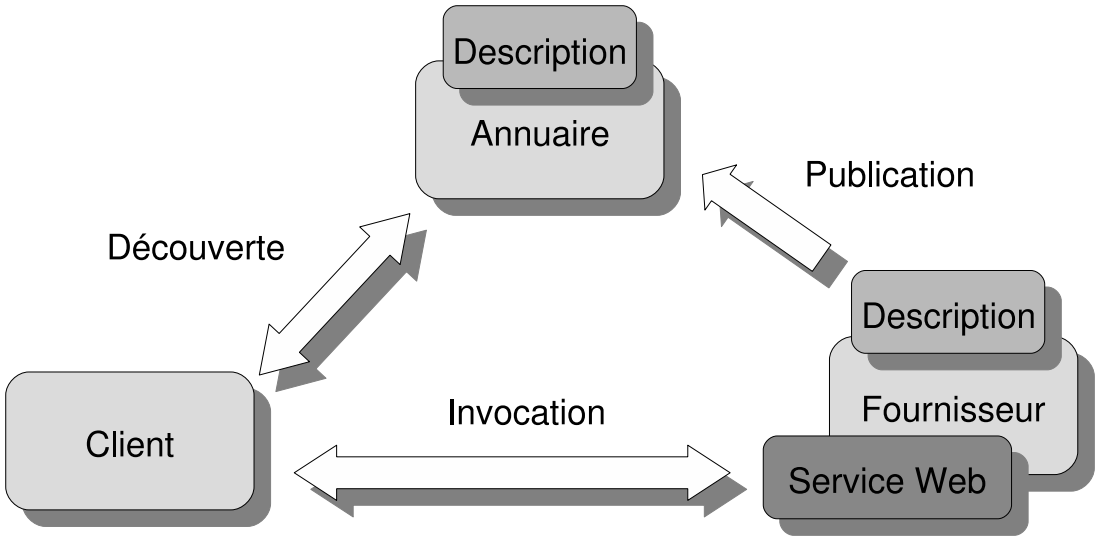

FIG. 1.1 – Interactions au sein d'une architecture orientée service

utiliser des services tout en restant indépendantes les unes des autres, et sans divulguer leur fonctionnement interne. Cette exigence est satisfaite par l'adoption de protocoles et langages standardisés qui fournissent un accès uniforme aux services et à leurs descriptions. Ainsi, les architectures orientées service se construisent autour de plusieurs protocoles et langages, selon quatre couches de fonctionnalités, à savoir :

- La **couche publication**, qui permet la centralisation, le stockage et la diffusion des descriptions de services.
- La **couche description**, qui regroupe les détails nécessaires à l'invocation des services dans un document.
- La **couche message**, qui assure la structuration et l'échange uniformes des messages.
- La **couche transport**, qui permet de véhiculer les messages à travers le réseau.

L'adoption des SOA dans le monde informatique a été renforcée par plusieurs implantations. Dans le cadre de cette thèse, nous nous intéressons aux services Web, qui représentent l'implantation la plus largement répandue. Les services Web présentent plusieurs avantages détaillés ci-dessous, et en conséquence, ils ont bénéficié d'un large support de la part des communautés informatique et scientifique.

1.1 Services Web : définition et architecture

Les quatre couches de fonctionnalités requises pour la réalisation d'une SOA sont assurées par une pile standard de protocoles construite autour de la définition des services

Web. Plusieurs définitions ont été proposées dans la littérature. Cependant, celle proposée par le World Wide Web Consortium (W3C) fait figure de référence [76] :

> « A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP-messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards. »

Cette définition met en valeur les avantages principaux d'un service Web, à savoir :

- Son interface décrite d'une manière interprétable par les machines, qui permet aux applications clientes d'accéder aux services de manière automatique.
- Son utilisation de langages et protocoles indépendants des plateformes d'implantation, qui renforcent l'interopérabilité[2] entre services.
- Son utilisation des normes actuelles du Web, qui permettent la réalisation des interactions faiblement couplées et favorisent aussi l'interopérabilité.

Autour de cette définition, la **pile standard de protocoles des services Web** fournit les fonctionnalités requises par les SOA de la manière suivante (figure 1.2) :

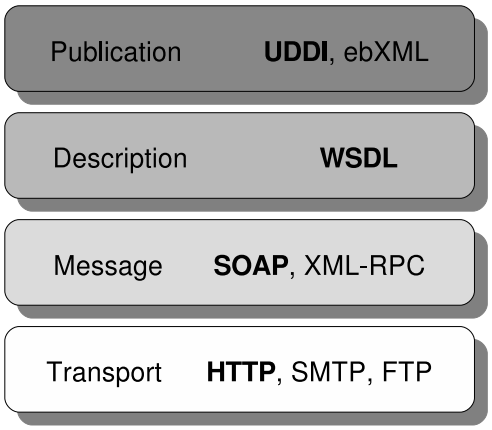

FIG. 1.2 – Couches de fonctionnalités et protocoles/langages correspondants

- La **couche publication** repose sur le protocole Universal Description, Discovery, and Integration (UDDI) [74], qui assure le regroupement, le stockage et la diffusion des descriptions de services Web. On notera aussi ebXML[3] comme une alternative permettant de fournir les mêmes fonctionnalités.

[2] L'interopérabilité est le fait que plusieurs systèmes, qu'ils soient identiques ou radicalement différents, puissent communiquer sans ambiguïté et opérer ensemble. (Wikipedia)

[3] `http://www.ebxml.org`

- La **couche description** est prise en charge par le Web Service Description Language (WSDL) [24], qui décrit les fonctionnalités fournies par le service Web, les messages reçus et envoyés pour chaque fonctionnalité, ainsi que le protocole adopté pour la communication. Les types des données contenues dans les messages sont décrits à l'aide du langage XML Schema [77].
- La **couche message** utilise des protocoles reposant sur le langage XML, car sa syntaxe unique résout les conflits syntaxiques lors de l'encodage des données. Actuellement, Simple Object Access Protocol (SOAP) [19] et XML-Remote Procedure Call (XML-RPC) sont les deux protocoles utilisés pour cette couche, SOAP étant le protocole prédominant.
- Pour la **couche transport**, le HyperText Transfert Protocol (HTTP) [32] est devenu le standard *de facto*. Ce protocole omniprésent sur Internet est généralement toléré des pare-feu. Il est donc particulièrement adapté aux communications entre organisations. Cependant, d'autres protocoles peuvent être utilisés, tels que le Simple Mail Transfer Protocol (SMTP) ou le File Transfer Protocol (FTP), permettant ainsi aux services Web de rester indépendants du mode de transport utilisé.

L'atout principal des protocoles SOAP et UDDI ainsi que du langage WSDL est de reposer sur l'eXtensible Markup Language (XML) [20]. XML est un langage de description des données qui possède les avantages suivants :

- une syntaxe unique qui permet son adoption par divers systèmes d'exploitation,
- une structure arborescente qui facilite sa lisibilité,
- une grande extensibilité, car il n'impose aucune restriction d'utilisation en dehors de sa syntaxe.

Ainsi, l'utilisation exclusive de langages et protocoles reposant sur le langage XML, ainsi que des normes actuelles d'Internet comme le protocole HTTP, favorise l'interopérabilité entre systèmes d'information distribués. Les services Web restent indépendants des systèmes d'exploitation et des plateformes de développement, ce qui facilite les interactions avec les applications clientes, mais aussi entre plusieurs services Web participant à une composition.

1.2 Composition et médiation de services Web

La composition consiste à combiner les fonctionnalités de plusieurs services Web au sein d'un même processus métier (ou business process) dans le but de répondre à des demandes complexes qu'un seul service ne pourrait pas satisfaire [13]. Le processus métier est une représentation concrète des tâches à accomplir dans une composition. Le déroulement

d'une composition nécessite la réalisation des étapes suivantes :

1. **La découverte des services Web** pouvant répondre aux besoins de la composition se fait généralement de manière manuelle par envoi de requêtes aux annuaires UDDI. De nombreux travaux visent à automatiser cette étape [11, 58, 59, 8, 10, 14].
2. **L'organisation des interactions** entre les services Web composés est distinguée par les termes choréographie et orchestration [23]. La choréographie gère les échanges de messages avec un unique service Web. L'orchestration, quant à elle, organise les interactions entre plusieurs services Web. Une composition est associée à une spécification pour gérer les échanges de messages et mettre en place les structures de contrôle nécessaires (boucles itératives, opérateurs conditionnels et gestion des exceptions). Plusieurs langages de spécification ont été proposés dans la littérature : WSCI [5], WSFL [41], XLANG [72], et plus récemment WS-CDL [38], YAWL [75], XPDL [80], BPMN [56] et WS-BPEL [26]. Actuellement, WS-BPEL s'est révélé comme le standard *de facto* pour la composition de services Web.
3. **L'exécution de la composition** est l'étape d'invocation effective des services Web participants à une composition. Une spécification de composition est hébergée par un serveur et son exécution est prise en charge par un moteur de composition, tel que Apache Ode™. La surveillance de l'exécution par le moteur de composition permet de gérer certaines erreurs dynamiquement.

Malgré son organisation en trois étapes facilitant une mise en place effective, la composition des services Web reste confrontée à de nombreux problèmes d'hétérogénéités. En effet, les services composés n'ont pas initialement été conçus pour interagir les uns avec les autres. Ainsi, des hétérogénéités peuvent survenir à chaque étape de la composition :

1. **Lors de la découverte**, les services sélectionnés ne fournissent pas toujours la fonctionnalité correspondant aux besoins de la composition. Par exemple, une requête de fonctionnalité demandant un service Web de « vente d'avions » peut renvoyer comme résultat aussi bien des services proposant la vente d'avions grandeur nature que d'avions miniatures pour enfants.
2. **Lors de l'organisation des interactions**, de nombreuses hétérogénéités peuvent apparaître entre les fonctionnements des services Web. Ces derniers peuvent :
   (a) suivre des types d'interactions différents (requête simple, requête avec réponse, réponse simple, réponse avec accusé de réception),
   (b) échanger les données selon différentes séquences d'échanges de messages,
   (c) utiliser des protocoles de transport incompatibles,

   (d) prendre en charge à des degrés différents la gestion des transactions, de la sécurité, etc.

3. **Lors de l'exécution**, les données échangées entre les services peuvent présenter des différences d'encodage, de structure et de signification.

La résolution de ces hétérogénéités, appelée médiation, est nécessaire pour réaliser une composition de services. L'utilisation de la pile standard de protocoles des services Web prend en charge une partie des hétérogénéités, principalement les hétérogénéités d'ordre syntaxique. Cependant, des mécanismes supplémentaires sont nécessaires pour obtenir des interactions réussies entre services Web. Ces mécanismes sont implantés par les médiateurs. La notion de médiateur a été introduite en 1992 avec les travaux de Wiederhold, qui définit un médiateur comme suit [78] :

> « A mediator is a software module that exploits encoded knowledge about some sets or subsets of data to create information for a higher layer of applications. »

Le travail de Wiederhold est lié au domaine des bases de données : le rôle du médiateur est de résoudre les hétérogénéités entre des données provenant de sources différentes, afin de les adapter à la représentation de l'utilisateur. Dans le cadre de la composition de services Web, le rôle du médiateur est différent : il comprend non seulement l'adaptation des données échangées entre les services, mais aussi la résolution des différences de fonctionnement des services Web, comme les différences de gestion des interactions ou de la sécurité. De nombreux travaux, présentés dans le chapitre 2, proposent des solutions de médiation pour la composition de services Web.

Généralement, les médiateurs exploitent l'information contenue dans les descriptions des services pour obtenir des informations sur leurs fonctionnements et les données qu'ils échangent. Ainsi, la médiation s'effectue aussi au niveau sémantique[4], car des services identiques peuvent utiliser des vocabulaires différents dans leurs descriptions, et inversement des services Web différents peuvent utiliser le même vocabulaire dans des sens différents. La médiation sémantique a pour but de résoudre ces hétérogénéités de signification entre les vocabulaires utilisés pour la description des services Web.

1.3 Sémantique et services Web

La dimension sémantique reste encore ignorée par la pile standard de protocoles des services Web. Lors de la réalisation d'une composition, des conflits sémantiques apparaissent

[4]Le terme « sémantique » est utilisé pour décrire ce qui a trait à la signification véhiculée par un objet, généralement un mot.

quand les services sont décrits selon des vocabulaires différents, ou lorsque la signification du vocabulaire utilisé diffère d'un service Web à un autre. Ces conflits peuvent générer des dysfonctionnements :

- **Lors de la découverte des services**, les fonctionnalités offertes sont décrites de manière hétérogène, et les services découverts peuvent ne pas correspondre aux attentes des clients.
- **Lors de l'organisation des interactions**, les compositions peuvent atteindre des états instables, car les interactions des services Web sont décrites selon des vocabulaires différents.
- **Lors de l'exécution de la composition**, les problèmes de sémantique entraînent l'inconsistance des échanges de données. En effet, les données doivent être correctement interprétées par les services Web pour donner une signification globale à la composition.

De nombreux langages [44, 6, 60, 39] ou extensions de langages [58, 46, 65], étudiés dans le chapitre 3, ont pour objectif de résoudre les problèmes de sémantique rencontrés par les services Web. Ces langages intègrent la sémantique des données dans les descriptions des services Web, qui deviennent des services Web sémantiques. Une description de service Web sémantique repose sur un vocabulaire commun décrit dans une **ontologie**, qui est définie comme « une conceptualisation partagée des connaissances d'un domaine » [35].

Une ontologie décrit le vocabulaire d'un domaine de connaissance, incluant les concepts utilisés dans le domaine et les relations entre ces concepts. Elle repose sur un langage de description, qui fournit les structures pour décrire les concepts et leurs relations. L'adoption d'une ontologie suppose un accord de tous les utilisateurs sur une représentation commune des connaissances d'un domaine, garantissant ainsi une interprétation identique des données.

1.4 Problématique

Avant son exécution, une composition est spécifiée par un processus métier, qui orchestre les échanges de données entre les services Web composés. Ces données sont représentées par les paramètres d'entrée/sortie détaillés dans les descriptions des services. Ce sont respectivement les données transmises au service Web et nécessaires à son exécution, et les données émises par le service comme résultat de cette exécution. Du point de vue de la composition, un échange de données entre deux services consiste à recevoir les données envoyées par un service Web émetteur, et à les transmettre à un service Web destinataire.

Pendant l'étape d'exécution d'une composition, il est possible que des hétérogénéités

sémantiques provoquent l'inconsistance des échanges de données entre services Web. En effet, la sémantique associée aux données par le service émetteur peut être différente de celle attendue par le service destinataire. Ainsi, notre problématique a pour but de fournir une solution de médiation qui permette la réalisation d'échanges consistants, au niveau sémantique, entre des services Web participant à une composition.

Malgré des avantages indéniables, nous constatons que dans le cadre des services Web, l'utilisation des ontologies pour réaliser un agrément sur un vocabulaire commun atteint ses limites. En effet, les services Web ont été mis en avant avec de fortes attentes en termes de dissémination mondiale et d'adoption par l'ensemble des communautés informatique, scientifique et industrielle. La composition de services Web se réalise donc dans un contexte très ouvert et soumis à de nombreux changements. Chaque réalisation d'une composition peut sélectionner des services Web différents, et à chaque exécution, les services peuvent présenter de nouvelles hétérogénéités sémantiques. En effet, chaque service Web possède une sémantique locale qui lui est propre, et qui peut différer de la sémantique proposée par une ontologie commune.

Ainsi, notre travail est construit autour des constats suivants :

1. Lors d'un échange de données entre services composés, la sémantique des données doit être adaptée à l'attente du service Web destinataire.
2. Cette adaptation nécessite une description explicite de la sémantique attachée aux données. Cependant, la pile standard de protocoles des services Web, qui est utilisée par la majorité des services, ne prend pas en charge la description des données au niveau sémantique.
3. La construction d'un agrément commun autour d'une ontologie est problématique dans le cadre des services Web, car nous sommes placés dans un contexte ouvert et dynamique, en l'occurrence celui d'Internet, dans lequel les services évoluent rapidement, et adoptent des représentations sémantiques hétérogènes.

Les recherches que nous menons dans ce mémoire de thèse visent à remédier aux lacunes et problèmes liés aux hétérogénéités sémantiques dans une composition. Notre objectif est de proposer une approche générique de médiation sémantique entre services Web permettant leur composition consistante. À partir de cet objectif et des constats énoncés précédemment, nous identifions plusieurs conditions à satisfaire pour résoudre les hétérogénéités sémantiques entre services, à savoir :

1. Fournir les mécanismes de médiation sémantique nécessaires pour intercepter les données échangées durant l'exécution de la composition et les adapter au service Web destinataire. Ces mécanismes doivent être en mesure d'évaluer les possibilités de conversion des données entre les représentations sémantiques des services composés.

2. Décrire la sémantique des données échangées entre services Web d'une manière interprétable par les machines, tout en respectant dans la mesure du possible les langages de description les plus utilisés par la communauté, et ce, afin de minimiser les modifications à effectuer pour les services existants.
3. Apporter une solution de représentation sémantique qui facilite l'adhésion des services Web à une ontologie commune, tout en considérant les différences locales entre les représentations sémantiques adoptées par les fournisseurs.

1.5 Contribution

Actuellement, dans le domaine des services Web, les ontologies sont utilisées pour représenter la sémantique des données et former un agrément sur un vocabulaire commun. Cependant, elles ignorent la notion de **contexte**. Dey [27] donne une définition générale du contexte comme « toute information qui caractérise la situation d'une entité », une entité étant « une personne, un lieu ou un objet considéré comme pertinent relativement à une interaction entre un utilisateur et une application, incluant l'utilisateur et l'application eux-mêmes ». Le contexte peut être associé à l'utilisateur(trice), à une machine, ou à un composant logiciel tel qu'un service Web. Dans les travaux de Sciore et coll. [67] et de Sheth et Kashyap [37], le contexte est utilisé pour décrire les différents aspects sémantiques d'un objet, sous la forme d'un ensemble de métadonnées. Dans la suite de ce document, nous adoptons cette dernière définition de la notion de contexte, que nous présentons plus en détail dans le chapitre 4.

Notre contribution repose sur l'utilisation de la notion de contexte pour faciliter la médiation des données échangées entre services Web composés. Nous proposons un enrichissement sémantique de la description des entrées/sorties des services avec le contexte. Afin de favoriser l'interopérabilité sémantique entre services Web, nous proposons une architecture de médiation qui adapte les données aux différents contextes des services, et permet leur interprétation correcte. Notre solution se résume en trois contributions, à savoir :

1. Un modèle de description sémantique des données reposant sur la notion de contexte, qui a pour objectif de faciliter la médiation des données. Cette contribution est rattachée au domaine de recherche sur le contexte.
2. Un enrichissement des données échangées dans une composition avec l'information sémantique nécessaire au bon déroulement du processus de médiation, par l'annotation des descriptions WSDL des services. Cette contribution est liée au domaine de recherche sur la description sémantique des services Web.

3. Une architecture de médiation exploitant le modèle et l'annotation précédemment cités, afin d'effectuer la médiation sémantique des données. Cette contribution est relative au domaine de la médiation de services Web.

1.6 Organisation

Ce document est structuré de la manière suivante :

La partie I dresse un état de l'art des travaux qui s'apparentent à notre problématique. Nous présentons dans le chapitre 2 les approches de médiation de services Web. Les travaux sont classifiés et évalués selon les différents aspects des services pris en considération par le processus de médiation. Nous distinguons la médiation des propriétés non fonctionnelles, des propriétés fonctionnelles et des entrée/sorties des services. Dans le chapitre 3 nous détaillons les propositions de description sémantique pour les services Web. Nous distinguons les langages de description inspirés du Web sémantique, et les annotations sémantiques de langages existants. Nous évaluons les limites des propositions actuelles, et concluons chaque chapitre par une analyse des travaux présentés.

La partie II se compose de trois chapitres qui constituent notre contribution. Le chapitre 4 présente notre approche de description sémantique orientée contexte, qui a pour objectif de rendre explicite la sémantique des données échangées entre services Web composés afin de dépasser les limites rencontrées par les travaux existants. Cette approche aboutit à notre modèle construit autour des notions de *contexte* et d'*objet sémantique*. Nous illustrons notre proposition avec un exemple de planification de voyage en ligne. Le chapitre 5 détaille notre architecture de médiation sémantique orientée contexte pour les services Web. Nous proposons une annotation sémantique du langage de description WSDL ainsi que l'utilisation d'ontologies contextuelles pour intégrer l'information contextuelle dans l'architecture des services Web. Une solution pour effectuer la médiation sémantique entre les services Web engagés dans une composition est proposée. Le chapitre 6 présente la mise en œuvre de notre architecture de médiation dans un environnement de composition de services Web. Ce chapitre est illustré par l'exemple développé tout au long du document. Les fonctionnements des composants de notre architecture de médiation sont détaillés.

Le chapitre 7 termine ce mémoire par une conclusion qui discute les apports de notre travail, ainsi que les perspectives de recherche envisagées.

Première partie

État de l'art

Chapitre 2

État de l'art de la médiation de services Web

Sommaire

2.1 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 proposons une classification des différents niveaux de médiation entre services Web, suivie d'une présentation selon cette classification des travaux traitant de la médiation de services Web.

2.2 Définition et classification

D'une manière générale, la médiation consiste à résoudre les conflits entre deux acteurs. Cette tâche 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 [78], 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 tâches de médiation qui permettent leur résolution. Athanasopoulos et coll. classifient les hétérogénéités entre services Web selon trois dimensions [7] :

1. La dimension **domaine métier** concerne le niveau conceptuel des processus métiers. C'est la dimension la plus abstraite. Deux processus métier sont considérés comme compatibles pour la dimension domaine métier s'ils distinguent les mêmes étapes pour leur exécution, et s'ils utilisent le même vocabulaire pour désigner ces étapes.
2. 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 fonctionnalités fournies par les services Web. Des processus métiers sont considérés 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.
3. La dimension **plateforme** correspond aux protocoles utilisés pour l'échange 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 abordés selon une perspective différente :

1. 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.

2. Le niveau **protocole** concerne la façon dont sont effectués les échanges entre services Web. Il se situe dans les domaines processus métier et application.
3. Le niveau **sémantique** est lié aux problèmes d'interprétation des données lors des interactions entre différents services ou avec le client. Il se situe dans le domaine processus métier.
4. Le niveau **qualité** concerne l'adaptation aux exigences des services Web en terme de qualité de service. Il se situe dans les domaines application et plateforme.
5. Le niveau **contexte** concerne les différences de représentation de contexte pour les systèmes embarqués. 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 [22]. Il désigne les interactions entre services Web sous le nom d'évènements métiers, et il distingue deux niveaux de médiation pour les services Web :

1. 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.
2. 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.

À 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 confrontés 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 [23].

1. La médiation de données concerne l'adaptation des données d'entrée/sortie échan-

gées par les services Web.

2. La médiation de fonctionnalités établit une correspondance entre la fonctionnalité fournie par le service Web et celle demandée par le client.
3. 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 décrites 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.

Pour notre part, nous proposons trois niveaux de médiation, organisés de la manière suivante :

1. **Le niveau intégration de services Web** a pour but de résoudre toutes les hétérogénéités entre les services Web, notamment les hétérogénéités entre les propriétés non fonctionnelles. Ces propriétés non fonctionnelles ne sont pas décrites dans les documents WSDL. Toutes les propriétés qui caractérisent un service et qui ne sont pas directement liées à la fonctionnalité délivrée par le service, sont considérées comme des propriétés non fonctionnelles. Par exemple, le support de la sécurité, l'organisation des séquences d'échanges de messages, et le support des transactions sont des propriétés non fonctionnelles.
2. **Le niveau adaptation d'interfaces** concerne toutes les propriétés des services décrites dans un document WSDL classique et non annoté[5], que nous identifions par le terme « propriétés fonctionnelles ». Les paramètres d'entrée/sortie, la fonctionnalité du service, le protocole d'échange utilisé, et l'encodage des données sont des propriétés fonctionnelles.
3. **Le niveau médiation de données** concerne les hétérogénéités des données échangées entre services Web composés. Ces données sont décrites via les paramètres

[5]Pires et coll. [61] proposent une solution d'adaptation d'interfaces étendues, en ajoutant les aspects de sécurité dans les descriptions de services Web. Ces travaux comprennent des aspects relatifs au niveau intégration de services. Par conséquent, ils sont traités dans cette catégorie.

d'entrées/sortie contenus dans les descriptions de services. Le niveau médiation de données est un aspect particulier de l'adaptation d'interfaces qui se distingue des autres par sa complexité.

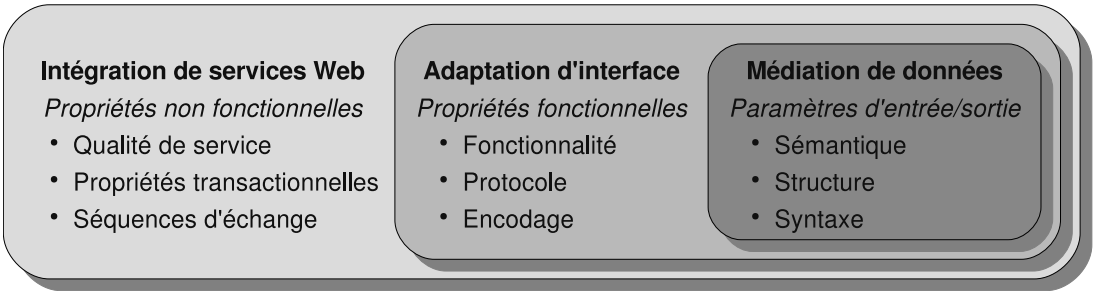

Fig. 2.1 – Classification des niveaux de médiation entre services Web

Comme illustré sur la figure 2.1, les niveaux de médiation que nous avons définis sont des sous-ensembles les uns des autres. La médiation de données est un sous-ensemble de l'adaptation d'interfaces, qui elle-même est un sous-ensemble de l'intégration de services Web. Dans la suite, nous considérons les aspects de la médiation spécifiques à chaque niveau :

- Nous étudions tout d'abord la médiation des propriétés non fonctionnelles dans la section « intégration de services Web ».
- Ensuite, nous examinons la médiation des propriétés fonctionnelles dans la section « adaptation d'interfaces de services Web ».
- Finalement, nous explorons les propositions de médiation des données échangées durant la composition dans la section « médiation de données pour les services Web ».

Dans chaque section, nous présentons les hétérogénéités spécifiques au niveau de médiation concerné (figure 2.1), puis nous proposons une classification, une présentation et une analyse des approches de médiation relatives à ces hétérogénéités.

2.3 Intégration de services Web

2.3.1 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 [64]. 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 [9, 42, 30], nous distinguons les types de disparités 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).** Ce terme regroupe les propriétés non fonctionnelles relatives aux performances des services Web, telles que la disponibilité, la rapidité, le coût, la fiabilité, etc.

2.3.2 Classification et présentation des approches

Nous classifions les travaux cherchant à résoudre par la médiation les hétérogénéités entre les propriétés non fonctionnelles citées précédemment en différentes catégories correspondant aux types d'approches proposées. Nous distinguons les types d'approches suivants :

- Approches à base d'adaptateurs,
- Approches à base de communautés,
- Approches à base de services Web,
- Approches à base de descriptions étendues.

Approches à base d'adaptateurs

Ces approches reposent sur une couche d'adaptateurs qui encapsulent les services Web. La couche d'adaptateurs est intégrée à une architecture homogène qui supporte leur découverte, leur composition et leur administration. Les fournisseurs établissent manuellement les correspondances entre leurs services et les adaptateurs lors de l'intégration des services dans l'architecture, ce qui permet de résoudre les hétérogénéités entre les services.

Plusieurs architectures de composition de services reposent sur une couche d'adaptateurs, comme Medjahed et coll., avec WebBIS [45]. Pour plus de détails sur ces architectures, Dustdar et Schreiner [31] proposent un état de l'art exhaustif comprenant une étude comparative des approches de composition de services Web.

Nguyen et coll. [52] proposent une solution de médiation de qualité de service. Ils présentent un algorithme qui évalue les différentes combinaisons de contraintes de qualité que les services Web peuvent supporter afin de satisfaire les exigences d'une composition. Si aucune combinaison ne satisfait les contraintes de la composition, de nouveaux services sont sélectionnés et évalués. Les adaptateurs qui encapsulent les services Web sont implantés comme des agents, ce qui leur permet de communiquer les contraintes de qualité des services qu'ils encapsulent et d'interagir entre eux pour répondre aux exigences des compositions.

Lin et coll. [42] proposent une architecture appelée Requester-based Mediation Framework (RMF), qui résout les hétérogénéités liées aux séquences d'échanges de messages. Cette architecture repose sur un « proxy » qui intercepte les appels aux services Web et adapte les flux de messages. Leur architecture n'utilise pas le terme « adaptateur », mais le proxy qui effectue la médiation à chaque appel de service est assimilable à un adaptateur.

Approches à base de communautés

D'autres travaux regroupent plusieurs services fournissant une fonctionnalité équivalente derrière une interface unique. Ces services forment alors une communauté. Plusieurs médiateurs établissent alors les correspondances entre l'interface de la communauté et les réalisations concrètes de la fonctionnalité. Benatallah et coll. proposent une telle solution

avec l'architecture SELF-SERV [12].

Taher et coll. [71] se servent de la notion de communauté pour apporter une réponse au problème de substitution de services. La substitution de services consiste à remplacer un service Web par un autre fonctionnellement équivalent. Elle a pour but de satisfaire les besoins d'une composition lorsqu'un service est défaillant ou ne répond plus aux exigences de qualité de service requises par la composition. La substitution d'un service par un autre nécessite la médiation des communications entre le client du service original et le service remplaçant. La notion de communauté permet de regrouper les services derrière une interface unique pour standardiser l'accès à ces derniers. L'interface de la communauté décrit la fonctionnalité fournie uniquement de manière abstraite. Des médiateurs déployés comme des services Web sont ensuite utilisés pour accéder aux réalisations concrètes de la fonctionnalité. Chaque médiateur est le point de contact d'un service Web spécifique. Les médiateurs communiquent avec les services Web qu'ils représentent et effectuent les opérations de médiation nécessaires à des interactions réussies. Le médiateur correspondant le mieux aux exigences de la composition est sélectionné selon des critères de qualité de service (vitesse, fiabilité, réputation, etc.). Un composant générique appelé Open Service Connectivity (OSC) fournit les fonctions requises pour interagir avec les interfaces abstraites, par exemple pour la sélection d'un service Web et son invocation. La communauté prend aussi en charge l'ajout et la suppression de services.

Approches à base de services Web

Ces approches insèrent les médiateurs directement au sein de la composition, comme des services Web ordinaires. Dans cette catégorie, Benatallah et coll. [9] assistent les concepteurs de composition dans le développement semi-automatique de services Web médiateurs. Ces derniers sont accédés à la place des services Web, de manière à être compatible avec le composant (un autre service Web ou un client) qui invoque le service Web original. Cette approche est une autre application du principe de substitution. Les auteurs ont développé des modèles de disparités, qui regroupent les différences entre services Web en catégories. Ils ont identifié et classifié les types communs de disparités, tout en laissant la possibilité aux développeurs d'ajouter des types spécifiques. Chaque catégorie de service Web médiateur possède les caractéristiques suivantes :

- un nom qui identifie le service Web médiateur,
- un type qui décrit le type de disparités pris en charge,
- un ensemble de paramètres qui doivent être fournis par le développeur lors de la création du service Web médiateur, afin de pouvoir générer son code,
- un modèle qui contient le code ou pseudo-code décrivant l'implantation du service

Web médiateur,
- et des exemples d'utilisation qui contiennent les informations nécessaires pour guider les développeurs dans le processus de personnalisation du service Web médiateur à des situations particulières.

La personnalisation et l'intégration des services Web médiateurs au sein d'une composition sont faites manuellement, et le déploiement automatique des services Web médiateurs est une des perspectives de travail des auteurs.

Haller et coll. [36] utilisent un moteur de choréographie reposant sur des médiateurs afin de réconcilier les séquences d'interactions des services Web et de leurs clients par l'intermédiaire de l'architecture WSMX. Ces médiateurs sont chargés des tâches telles que le groupement de plusieurs messages en un seul, le changement de l'ordre des messages, ou la suppression de messages inutiles. Les auteurs précisent que ces médiateurs fonctionnent grâce à un travail manuel effectué durant la phase de conception du processus métier. Ce travail consiste à créer des règles, qui permettent d'identifier les équivalences entre les différentes choréographies. Ces règles, une fois stockées, sont utilisées durant l'exécution par le moteur de choréographie afin de réconcilier les différentes gestions des séquences de messages. En pratique, l'architecture WSMX elle-même est implantée comme un service Web, qui fournit les fonctionnalités proposées par l'intermédiaire de ces opérations.

Avec l'architecture IRS-III [23], Cabral et Domingue modélisent les contraintes de communication par un ensemble de règles logiques. Un médiateur de processus s'occupe de résoudre les conflits de communication en utilisant ces règles, tout en se positionnant entre le client et le service Web afin d'intercepter les communications et de les adapter. Il utilise un composant appelé interpréteur de choréographie, qui permet d'exécuter un service Web en créant les messages requis par ce dernier sur la base de sa description. Il garde en mémoire l'état d'avancement des communications entamées avec le service Web, en mémorisant les appels exécutés par un composant appelé invocateur.

Approches à base de descriptions étendues

Ces approches ajoutent de l'information supplémentaire dans les descriptions des services Web. Les solutions de médiation utilisent ces descriptions améliorées pour réconcilier les services Web. Dans cette catégorie, citons Dumas et coll. [30], Williams et coll. [79] et Pires et coll. [61].

Dumas et coll. [30] étendent la notion d'interface d'un service en y ajoutant les contraintes d'ordonnancement des messages. Pour ce faire, ils modélisent une interface comme une suite ordonnée d'actions, et ils développent une algèbre de transformation contenant six opérateurs permettant d'établir des correspondances, à partir d'une « in-

terface source » vers une « interface cible ». Ces six opérateurs permettent d'adapter une action d'une interface à l'autre, de regrouper plusieurs actions en une seule, de séparer une action en plusieurs, ou bien de regrouper plusieurs messages provenant d'une même action en un seul, et inversement. Enfin, un opérateur permet d'ignorer un message non requis.

Un langage visuel de représentation des transformations assiste les concepteurs de composition dans le processus d'adaptation des interfaces de services Web. L'implantation de leur proposition comprend un outil visuel de transformation d'interfaces, ainsi qu'un moteur de composition de services supportant une médiation qui repose sur l'algèbre et la notation visuelle proposées. Le déploiement d'une transformation d'interface dans le moteur de composition a pour résultat un nouveau point d'accès de service qui correspond aux exigences requises par l'interface avec laquelle il communique. Ce point d'accès transmet les messages au service adapté, tout en effectuant les transformations requises à la volée.

Williams et coll. [79] adaptent les séquences d'échanges de messages, appelées « protocoles », entre services Web composés. Afin de modéliser les « protocoles » guidant les interactions entre services Web, ils utilisent la notion de rôle, qui définit les comportements des différents partenaires et la notion de conversation, qui est formée d'une séquence d'actes communicatifs[6]. Dans le travail de Williams et coll., un acte communicatif est défini comme une séquence d'échanges de messages réalisant une intention de communication, les émetteurs et récepteurs étant des services Web. Par défaut, une séquence est composée de quatre primitives :

1. l'envoi du message,
2. la réception du message,
3. l'envoi d'un accusé de réception,
4. la réception d'un accusé de réception.

Afin de décrire les échanges de messages entre services Web et de résoudre leurs hétérogénéités, les auteurs définissent un langage appelé Very Simple Choreography Language (VSCL) qui permet de décrire les interfaces de services Web à partir des éléments suivants :

- Une ontologie de domaine qui décrit les concepts associés au domaine de connaissances et leurs relations ;
- un catalogue des rôles adoptés par les participants et des actes communicatifs qui leur sont associés ;

[6]Un acte communicatif est un concept originaire du domaine de la théorie du langage, qui signifie « *l'expression d'une intention de communication* », et qui est exprimé par un émetteur et perçu par un récepteur comme décrit dans les travaux de Searle [68].

- pour chaque rôle, une description du protocole abstrait qui dirige les séquences de primitives modélisant les actes communicatifs ;
- pour chaque fournisseur de service, une expression du protocole concret associé avec l'envoi et la réception des actes communicatifs ;
- pour chaque protocole concret, la description des transformations de données nécessaires pour une compréhension entre les différents acteurs.

Pires et coll. proposent une solution pour gérer les propriétés transactionnelles des services Web avec WebTransact [61]. Leur approche multiniveaux repose sur l'utilisation de « services médiateurs », qui sont des services virtuels fournissant une interface homogène, qui permet d'accéder à des services Web fournissant la même fonctionnalité ; mais adoptant des interfaces qui peuvent être différentes[7]. Les services Web médiateurs sont virtuels, c'est-à-dire qu'ils n'implantent pas eux-mêmes les fonctionnalités fournies, mais qu'ils délèguent l'exécution à un des services Web qu'ils représentent.

Les quatre types d'opérations suivantes sont définis, du moins restrictif au plus restrictif :

- Les **opérations compensables** possèdent une opération d'annulation qui annule les effets de l'opération initiale.
- Les **opérations virtuellement compensables** sont en réalité des opérations annulables. Elles attendent que la composition ait atteint un état dans lequel il est sans danger de confirmer définitivement le succès de leur exécution avant de le faire.
- Les **opérations réessayables** réussissent au moins une fois leur exécution dans un nombre fixé d'essais.
- Les **opérations pivots** ne sont ni réessayables, ni compensables.

Afin de décrire les propriétés transactionnelles des services Web, les auteurs proposent le langage WSTL (Web Service Transaction Language), qui est une extension de WSDL. Une description WSTL contient une section `transactionDefinitions` ajoutée au WSDL classique, qui décrit les propriétés transactionnelles supportées par chaque opération fournie. Pour les opérations compensables, l'opération à invoquer en cas d'annulation est spécifiée. Ainsi, les propriétés transactionnelles des services Web sont prises en compte pendant le déroulement de la composition. Les services pivots, par exemple, ne sont invoqués que lorsque les autres services ont confirmé des exécutions réussies.

Les services médiateurs sélectionnent les services Web de la manière la moins restrictive possible, selon les propriétés transactionnelles disponibles. En effet, si au moins un des services est compensable, alors le service Web médiateur peut participer à n'importe quelle composition, car il est toujours possible d'annuler son exécution sans condition. À

[7] Nous étudions la résolution des différences entre interfaces dans la section 2.4.

l'inverse, si tous les services Web pris en charge par le service médiateur sont pivots, alors le service Web médiateur ne peut être appelé que par des compositions qui ont atteint un état dans lequel elles sont prêtes à se terminer avec succès.

2.3.3 Conclusion

Nous avons distingué plusieurs types d'approches, développées pour réconcilier les propriétés non fonctionnelles des services Web : les approches à base d'adaptateurs, de communautés, de services Web et d'interfaces étendues.

Notons que les approches reposant sur une couche d'adaptateurs qui encapsulent les services Web pour les homogénéiser sont les plus simples. De plus, elles sont améliorées par l'utilisation de communautés de services. Les communautés permettent de grouper les services Web par fonctionnalité, et de sélectionner les services offrant une QoS plus adaptée aux exigences de la composition, via une interface commune. Cependant, ces approches présentent des inconvénients importants. Leurs implantations sont dépendantes des architectures dans lesquelles elles sont intégrées, et le travail d'adaptation des services Web à l'interface commune reste le plus souvent à la charge du fournisseur.

Les approches à base de services Web présentent comparativement plusieurs avantages. L'implantation de services Web médiateurs directement dans la composition permet de conserver le découplage existant entre les fonctionnalités offertes par les services et les fonctionnalités de médiation requises par la composition. De plus, ces approches restent indépendantes des langages et moteurs de composition, et respectent l'utilisation du paradigme orienté service. Enfin, l'extension des interfaces de description s'avère nécessaire pour décrire les propriétés non fonctionnelles des services Web et outrepasser les limitations du langage WSDL.

2.4 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 utilisés. Nous traitons séparément, dans la section 2.5, la médiation des paramètres d'entrée/sortie, qui décrivent les données échangées entre les services.

2.4.1 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 métamodèle présenté par la figure 2.2. L'élément `<portType>` fournit une description abstraite du service Web. Il décrit les fonctionnalités offertes par le service Web, grâce à des éléments `<operation>`, 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 « paramètres ». 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 paramètres.

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 Schema. 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 apparaître pour tous les éléments d'une interface. Nous étudions dans cette section les solutions de médiation correspondantes.

2.4.2 Classification et présentation des approches

Nous classifions les approches présentées dans cette section selon deux groupes :

1. Les approches du premier groupe sont intégrées dans une architecture qui utilise les descriptions sémantiques pour résoudre les hétérogénéités entre les services Web. Ainsi, l'adaptation au niveau des interfaces est facilitée par la résolution préalable des hétérogénéités sémantiques.
2. Les approches du deuxième groupe se passent de description sémantique et par conséquent, elles nécessitent une intervention manuelle pour résoudre les conflits sémantiques. Cependant, elles ont pour avantage de reposer sur le langage de description standard des services Web, WSDL.

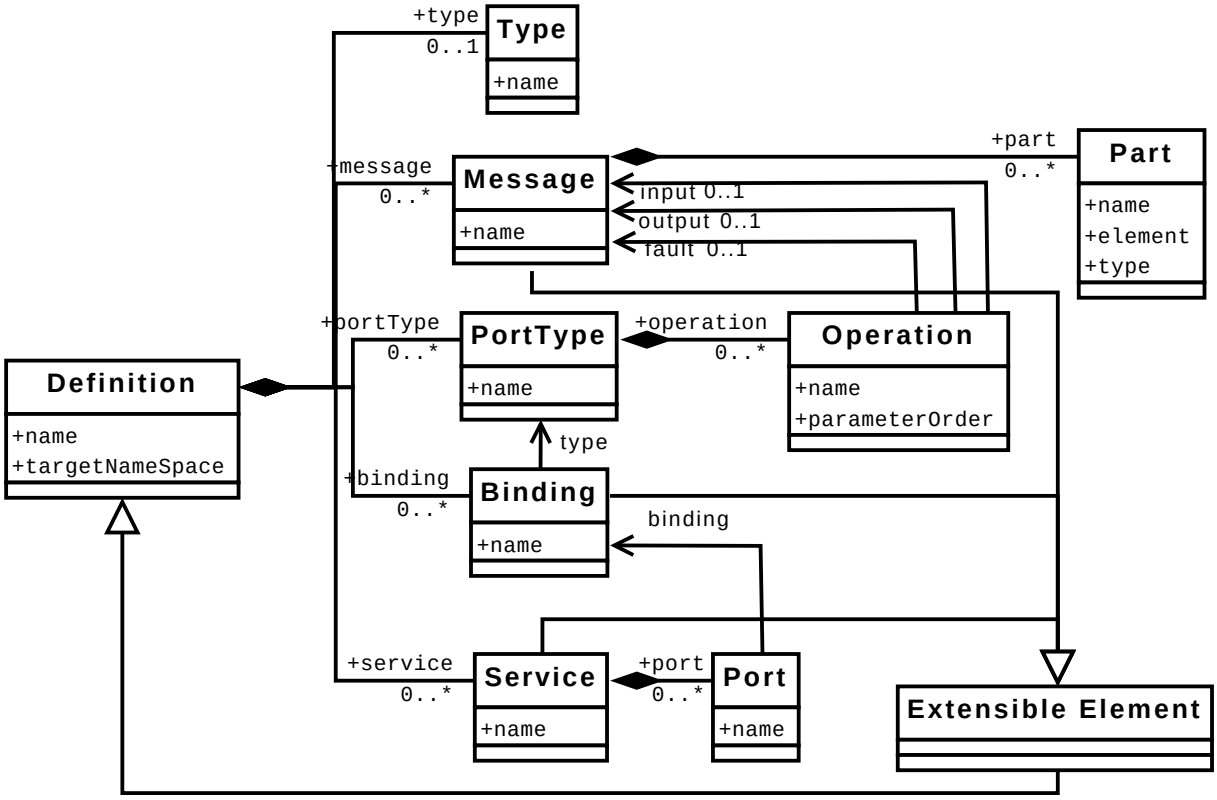

FIG. 2.2 – Métamodèle du langage WSDL

Approches exploitant l'information sémantique

Cabral et Domingue [23] avec IRS-III, ainsi que Haller et coll. [36] avec WSMX, établissent les correspondances entre les descriptions de services à l'aide d'un service Web médiateur. La sémantique des termes utilisés est intégrée dans les descriptions sémantiques des services. De cette manière, les auteurs résolvent les hétérogénéités d'interfaces en supposant que toutes les informations nécessaires sont décrites dans une ontologie commune, ce qui permet de résoudre les différences entre les services en raisonnant sur l'information sémantique contenue dans l'ontologie.

Cabral et Domingue effectuent la médiation à l'aide d'un service Web médiateur qui utilise un composant appelé « interpréteur d'orchestration ». Ce composant gère l'orchestration de plusieurs services Web au sein d'une composition. Il divise la tâche requise par la composition en plusieurs sous-tâches qui sont réalisées par des services Web, ou des sous-ensembles de services Web. Pour ce faire, le service Web médiateur fait appel à des médiateurs de fonctionnalité qui connectent les sous-tâches au sein de l'orchestration. Le médiateur de fonctionnalité fait correspondre la demande de fonctionnalité de l'application cliente avec une description sémantique de service fournie par l'architecture.

Haller et coll. [36] présentent WSMX, qui est une implantation du modèle conceptuel Web Service Modeling Ontology[8] (WSMO) [6]. WSMX est une architecture de médiation qui se concentre sur l'intégration de services Web. Le composant médiateur est une partie centrale de cette architecture, et est accessible comme un service Web. Il effectue la médiation au niveau des interfaces en suivant le format de description de services du modèle conceptuel WSMO. Les hétérogénéités entre les concepts utilisés pour la description des services sont résolues par des mécanismes de raisonnement, qui établissent des correspondances entre les concepts des différentes ontologies utilisées par les fournisseurs de services.

Approches n'exploitant pas l'information sémantique

Ponnekanti et Fox [62] proposent une solution pour résoudre les hétérogénéités d'interfaces dans le cadre de la substitution de services. Ils distinguent les hétérogénéités suivantes :

- méthode manquante (les méthodes additionnelles ne causent pas de problème),
- paramètre d'entrée ou de sortie additionnel ou manquant,
- domaines de valeurs et cardinalités des paramètres des services non compatibles,

et classifient ces hétérogénéités selon les types suivants : hétérogénéités structurelles, de valeur, d'encodage, sémantiques. Seules les hétérogénéités structurelles et de valeur, appelées $\mathcal{SV}$-incompatibilités, sont résolues dans ce travail. Comme l'utilisation de description sémantique explicite n'est pas envisagée, les auteurs adoptent les normes de conduite et règles de réutilisation des domaines de nommage (namespace) et de création de nouveaux domaines de nommage recommandées par Orchard [57]. Si ces normes de conduites sont effectivement suivies, alors les auteurs montrent qu'une compatibilité aux niveaux des valeurs et structures des données implique une compatibilité sémantique, que ce soit pour les valeurs échangées, les méthodes, les types ou les champs de types de données. L'approche proposée repose sur plusieurs éléments :

- Des outils d'analyse statique et dynamique (c'est-à-dire durant l'exécution), qui déterminent à partir de règles logiques si un service Web substitué est compatible avec une spécification de processus métier en comparant les paramètres d'entrée/sortie des opérations.
- Des composants semi automatiquement générés qui résolvent les incompatibilités et établissent les communications avec les services Web substitués.
- Un mécanisme de médiation appelé *multi-option types* qui facilite l'adaptation des paramètres d'entrée/sortie de services en précisant s'ils sont optionnels ou obliga-

[8] Ce modèle conceptuel de description sémantique des services Web est présenté dans le chapitre 3.

toires, ou lorsqu'ils manquent, s'ils sont remplaçables par des valeurs par défaut.

Benatallah et coll. [9] définissent, pour leur proposition de conception semi-automatique de services Web médiateurs, un « modèle de disparité de signature », qui s'occupe du cas dans lequel deux services Web fournissant la même fonctionnalité sont décrits avec des noms d'opération différents, ou un nombre, ordre ou type de paramètres d'entrées ou de sortie différents. Ils proposent un adaptateur générique et paramétrable selon les interfaces des services. Ils définissent aussi le « modèle de contraintes de paramètres » qui gère, comme son nom l'indique, les contraintes appliquées sur les paramètres d'entrée/-sortie. Par exemple, lorsqu'une valeur de paramètre délivrée par un service Web n'est pas autorisée pour le service Web qui la reçoit, une exception est levée.

Dans WebTransact, Pires et coll. [61], importent chaque élément `<portType>` d'une description WSDL via un service Web distant. Ce service Web distant fournit le lien entre l'interface décrite par le médiateur et l'interface du service Web qui implante la fonctionnalité. Il fournit les informations permettant la transformation des données entre le médiateur et le service Web. La transformation est réalisée comme une transformation d'élément à élément en utilisant XPath[9].

2.4.3 Conclusion

La médiation d'interfaces est généralement facilitée par l'addition d'information sémantique, qui permet de rendre explicite la signification des termes utilisés dans une description (par exemple WSDL). Dans le cas contraire, les correspondances entre interfaces sont établies de manière manuelle, car il n'est pas trivial d'automatiser le processus d'établissement des correspondances, aussi appelé « matching », sans posséder cette information sémantique. Une autre solution, comme dans les travaux de Ponnekanti et Fox, consiste à supposer que certaines règles sont respectées au niveau des méthodes de nommage. Dans ce cas particulier, les hétérogénéités sémantiques sont résolues, mais dans un contexte ouvert comme celui d'Internet, ce genre de solution reste difficilement applicable, en raison de la difficulté des fournisseurs de services d'adhérer à un vocabulaire commun. Nous présentons dans ce mémoire une alternative à ces propositions, visant à outrepasser les difficultés liées au contexte ouvert d'Internet.

[9]XPath est une syntaxe (non XML) pour désigner une portion d'un document XML. Initialement créé pour fournir une syntaxe et une sémantique aux fonctions communes à XPointer et XSL, XPath a rapidement été adopté par les développeurs comme un petit langage d'interrogation. (Source : Wikipedia)

2.5 Médiation de données pour les services Web

La médiation des données échangées entre services Web est un cas particulier des niveaux de médiation étudiés 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 placé 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.

2.5.1 Hétérogénéités des données échangées entre services Web

Comme mentionné précédemment, les services Web présentent des hétérogénéités au niveau des données qu'il échangent. Nous distinguons trois niveaux d'hétérogénéités distincts : syntaxique, structurel et sémantique, publiés dans nos travaux précédents [47, 50] :

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

Nagarajan et coll. proposent une classification différente dans [51]. 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 ignoré dans le domaine des services Web, car les hétérogénéités de ce type sont résolues par l'utilisation des langages de description liés aux services Web, c'est-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 présentés ci-dessus sont orthogonaux par rapport aux attributs (figure 2.3). 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 stocké 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 utilisés [35]. Les conflits de niveau sémantique liés 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.

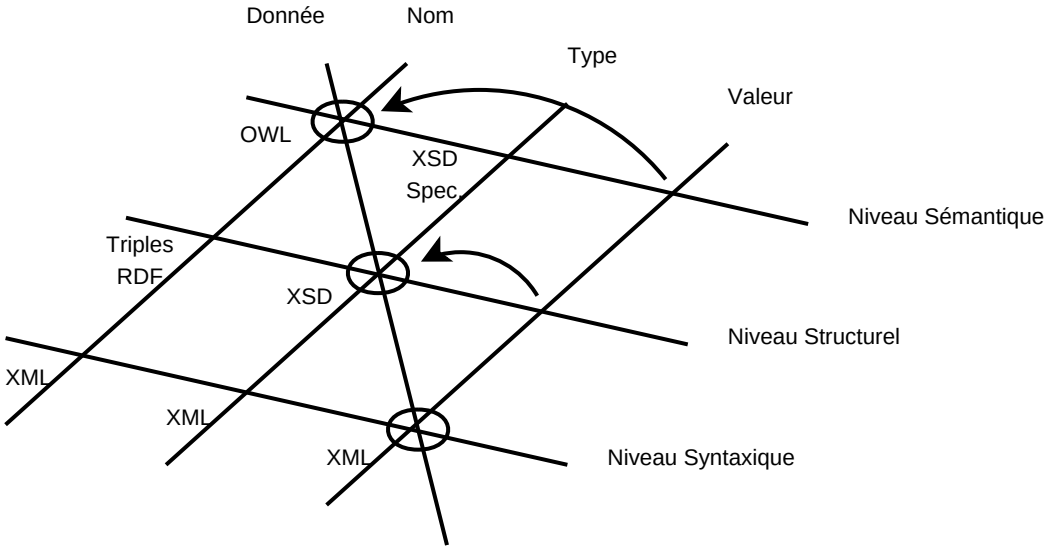

FIG. 2.3 – Niveaux d'hétérogénéités des données

2.5.2 Classification et présentation des approches

Les approches présentées dans cette section sont classifiées selon les niveaux d'hétérogénéités présentés ci-dessus. Les travaux traitant du niveau syntaxique ne sont pas traités, car l'usage du langage XML ne nécessite pas de médiation supplémentaire à ce niveau. En effet, la syntaxe unique du langage XML permet d'obtenir une représentation homogène des données au niveau syntaxique, quelle que soit la plateforme utilisée. Ainsi, nous distinguons les approches effectuant la médiation au niveau sémantique de celles qui se concentrent sur le niveau structurel.

Médiation au niveau sémantique

Tolk et Diallo [73] proposent une approche d'ingénierie des données orientée modèle pour adapter les données entre services Web. Leur objectif est de permettre l'interopérabilité sémantique des données échangées entre services Web composés, sur la base d'un modèle commun. Ils soulignent les lacunes du langage XML concernant la prise en charge de la sémantique attachée aux données, et ils utilisent des méthodes d'ingénierie pour construire manuellement un modèle commun, qui sert de référence aux services Web. La transformation des données vers le modèle commun est effectuée à l'aide de feuilles eXtensible Stylesheet Langage Transformation (XSLT) [10] étendues, qui sont aussi créées manuellement. Les auteurs n'apportent aucun détail concret concernant leur prototype, puisque la plupart des étapes de leur solution sont effectuées manuellement.

Cabral et Domingue [23] proposent aussi une solution de médiation de données au niveau sémantique. Ils supposent qu'à une annotation sémantique suffisamment explicite ne correspond qu'une seule représentation structurelle des données. Ainsi, leur solution de médiation repose sur un médiateur de données unique qui adapte l'information entre ontologies de domaine, entraînant la compatibilité des données au niveau structurel.

La médiation sémantique proposée par Cabral et Domingue repose sur un « broker », ou courtier, qui contient plusieurs types de médiateurs permettant de résoudre les hétérogénéités entre services Web. Le médiateur dédié aux hétérogénéités de données effectue la médiation entre ontologies. Ce médiateur utilise le Operational Conceptual Modeling Language (OCML)[11]. OCML est un langage de description de modèles de connaissances qui possède un mécanisme permettant d'établir des correspondances entre les éléments de plusieurs ontologies. Le fonctionnement du médiateur de données développé par Cabral et Domingue consiste à générer des correspondances entre les termes des ontologies, correspondances qui sont stockées dans une ontologie temporaire. Cette ontologie est une fusion des ontologies de départ, grâce à laquelle il est possible de convertir les données d'une ontologie à l'autre en passant par une représentation intermédiaire.

Haller et coll. [36] utilisent un médiateur sémantique pour effectuer la conversion des données d'une représentation à l'autre. Ce médiateur est constitué de deux composants : le premier composant identifie les similarités de représentation entre les concepts des ontologies et génère des correspondances entre ces derniers. Ces correspondances sont stockées pour être ensuite utilisées par le deuxième composant. C'est ce deuxième composant qui effectue la transformation des données durant l'exécution de la composition.

Bicer et coll. [15] proposent une infrastructure permettant l'interopérabilité sémantique

[10] XSLT est un langage de transformation XML.

[11] `http://kmi.open.ac.uk/projects/ocml/`

entre services Web du domaine médical. Leur proposition repose sur la notion d'archétype. Un archétype est une description formelle d'un concept de domaine, sous la forme de contraintes sur les données qui instancient le concept. Les archétypes sont décrits par des ontologies. L'idée consiste à annoter les messages décrits dans les documents WSDL avec un archétype, et de fournir les transformations entre les différents archétypes des ontologies utilisées par les fournisseurs de services Web. Ainsi, les données sont transformées d'un service à un autre via leurs représentations sémantiques.

Un archétype est composé de trois sections : un entête, une définition, et une ontologie. L'entête contient un identifiant unique pour l'archétype ainsi que des informations sur l'auteur et la version de la description. La définition contient les restrictions imposées sur les données sous forme de structure arborescente. La sémantique associée à la structure arborescente est décrite dans la section ontologie. Celle-ci contient une description formelle des significations des éléments de la structure arborescente, ainsi que les contraintes associées à ces éléments. La gestion des descriptions de services Web est effectuée par un médiateur. Les hôpitaux désirant publier leurs descriptions annotées les envoient au médiateur, qui est accédé par les clients lors de la recherche de services Web. Pour la transformation des données d'un archétype à l'autre, les auteurs ont réalisé un outil de création de correspondances entre archétypes. Une interface graphique permet à l'utilisateur(trice) de spécifier les correspondances entre les différents archétypes. Aussi, un outil d'annotation des descriptions de services est proposé.

Médiation au niveau structurel

Spencer et Liu [70] proposent une approche d'intégration des services Web sémantiques reposant sur des règles logiques de transformation des données. Ils considèrent des données compatibles au niveau sémantique, grâce à l'utilisation de langages de description sémantique tels que OWL-S[12]. Cependant, ils considèrent que les données, même sémantiquement compatibles, peuvent suivre des représentations structurelles différentes. Il y a donc un besoin de réconcilier les représentations de ces données au niveau structurel. Ces différences de représentation structurelle leur permettent d'identifier les tâches suivantes :

- correspondances entre les types de données,
- restructuration de l'arborescence au niveau du schéma des messages,
- appels de fonctions de conversion,
- et traduction entre ontologies.

[12]Ce langage est présenté dans le chapitre 3.

Les auteurs utilisent un compilateur de règles, qui raisonne à partir des descriptions et ontologies fournies par les services Web. Ce sont les descriptions WSDL et OWL-S, les ontologies référencées par ces descriptions, ainsi que les spécifications de conversion entre les différents types des données qui sont utilisées pour générer les règles de transformation. Si la conversion demandée n'est pas possible, alors les règles nécessaires ne seront pas créées, car les mécanismes d'inférence n'aboutiront pas à un résultat correct. Une fois que les règles ont été générées, une file d'inférence est insérée entre les services Web concernés durant l'exécution de la composition. Cette file d'inférence reçoit les données du premier service Web sur lesquelles elle applique les règles de conversion, et envoie le résultat au service Web suivant. Les détails de l'intégration de la file d'inférence dans un moteur d'exécution de composition ne sont pas donnés.

Bowers et Ludäscher [18] déploient une solution pour la transformation de données adaptée en particulier aux processus métiers scientifiques. Leur travail suppose une ontologie globale partagée par tous les utilisateurs. Ils définissent les notions de type sémantique et de type structurel. Le type sémantique correspond au concept qui permet de donner une signification aux données, et le type structurel correspond au schéma décrivant la structure des données. Leur objectif est d'adapter les différentes représentations structurelles que les services Web ont adoptées pour décrire un même type sémantique. Pour ce faire, ils se servent de correspondances qui doivent être préalablement établies entre les types structurels et les représentations sémantiques des données. Ces correspondances sont représentées comme des règles de transformation des données. Elles contiennent une requête q qui identifie les sous-structures du schéma, et les associe à un concept p de l'ontologie globale. Cependant, ce travail pose une hypothèse contraignante : les auteurs supposent que les types des données sont toujours compatibles. L'incompatibilité entre les différents types de données est un sujet de travaux futurs. Les auteurs intègrent leur solution dans un programme de création et d'exécution de processus métiers. Ils identifient les sous-ensembles des structures XML concernées par la médiation à l'aide de requêtes qui identifient les portions de messages à mettre en correspondance. Ces correspondances permettent l'adaptation des données aux structures XML ciblées.

Radetzki et Cremers [63] proposent un modèle de médiation à base de composants. Leur approche a pour but de supporter l'intégration de services Web au sens large. Concernant l'intégration de données, ils proposent une architecture formée de composants de médiation, qui sont sélectionnés et composés durant l'exécution de la composition. Leur solution repose sur une approche d'ingénierie logicielle afin d'assister les experts d'un domaine de connaissance dans la création de composition de services Web. Selon leur opinion, le processus de médiation doit commencer en tâche de fond pendant que le créateur de la composition modélise le processus métier. La médiation est effectuée à l'aide

d'ontologies, de médiateurs et de composants logiciels. Durant l'exécution, les composants médiateurs résolvent les hétérogénéités en utilisant d'autres fonctionnalités de médiation déployées comme des composants. Ces derniers peuvent être composés afin d'améliorer la tâche de médiation. Le prototype développé est une application Java™qui utilise la librairie Jena™, laquelle permet la gestion des profils de médiateurs et des ontologies qui utilisent tous les deux le langage OWL. Une base de données stocke les descriptions sémantiques de médiateurs, et une interface utilisateur permet de créer lesdits profils.

Lin et coll. [42] résolvent les hétérogénéités des données à l'aide d'un détecteur d'hétérogénéités, qui analyse le contenu des documents WSDL et génère des règles de médiation de données. Ces règles sont stockées dans un entrepôt de règle, qui est utilisé par un médiateur durant l'invocation des services. Les hétérogénéités prises en charge sont celles liées au type, à la structure et aux unités des données.

2.5.3 Analyse

La possibilité de rencontrer des incompatibilités structurelles entre des données sémantiquement compatibles sont clairement mises en évidence par les travaux de Spencer et Liu [70], ainsi que ceux de Bowers et Ludäscher [18]. Cette possibilité reste ignorée par d'autres travaux, comme celui de Cabral et Domingue [23], qui associent à un concept sémantique une représentation structurelle unique.

Malgré son application spécifique au domaine médical, le travail de Bicer et coll. [15] montre clairement que le processus de médiation des données se joue aussi bien au niveau sémantique que structurel : les correspondances sont établies au niveau sémantique, entre des concepts représentés d'une manière formelle dans des ontologies, et les conversions entre les concepts sont facilitées par la richesse du langage de description de l'ontologie. Une phase d'élévation enrichit des données au niveau sémantique, permettant à la conversion de tirer avantage des capacités de raisonnement apportées par le langage de description sémantique. Ensuite, une phase d'adaptation convertit les données vers la représentation structurelle désirée.

2.6 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.

Aussi, la plupart des approches se distinguent par l'utilisation de mécanismes à base de règles logiques. Ces règles logiques sont soit ajoutées manuellement par l'utilisateur(trice), soit déduites des descriptions sémantiques. 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.

Dans le chapitre suivant, nous décrivons les différentes propositions de langages et annotations permettant la description sémantique, et nous discutons de leurs contributions dans le cadre de la composition de services Web.

Chapitre 3

État de l'art de la description sémantique de services Web

Sommaire

3.1 Introduction

Malgré une large acceptation par la communauté des technologies de l'information, la pile de protocoles standard des services Web n'a pas été initialement développée pour satisfaire aux exigences de l'échange sémantique. Aussi, le langage de modélisation de processus métiers WS-BPEL reste concentré sur le côté syntaxique, sans considérer les aspects sémantiques mis en jeu durant la composition. Les problèmes d'hétérogénéités sémantiques sont par conséquent gérés de manière manuelle durant la phase de conception d'un processus métier. Pour atteindre l'interopérabilité sémantique, c'est-à-dire pour être en mesure de prendre en charge la signification des données qu'ils échangent, les services Web doivent être capables :

1. d'interpréter correctement la sémantique des données qu'ils envoient et reçoivent,
2. de décrire les fonctionnalités qu'ils fournissent en utilisant une sémantique explicite et compréhensible par les machines.

Afin de répondre à ces exigences, la communauté du Web sémantique propose des solutions reposant sur des ontologies de référence pour fournir une description explicite de la sémantique des services Web, lesquels sont ensuite appelés services Web sémantiques. L'objectif des services Web sémantiques est de faciliter les tâches liées à leur utilisation, telles que la découverte, la sélection, l'orchestration et l'invocation, par le biais de leurs descriptions qui rendent la sémantique explicite et compréhensible par les machines. Ainsi, le domaine des services Web sémantiques se situe au croisement du Web sémantique et des services Web. De nombreux langages et architectures ([44, 6, 60, 39, 46, 65]) sont proposés afin de décrire les services Web sémantiques. Nous proposons ci-dessous une classification des approches existantes, qui structure notre présentation des approches de description sémantique pour les services Web.

3.2 Classification et présentation des approches

La réalisation des conditions qui élèvent les services Web au rang de services Web sémantiques peut suivre deux approches. La première approche consiste à développer un langage complet qui décrit les services Web ainsi que leur sémantique d'un seul bloc. La deuxième approche consiste à annoter les langages existants avec de l'information sémantique. L'avantage principal de ce genre de solutions réside dans la facilité pour les fournisseurs de services d'adapter leurs descriptions existantes aux annotations proposées.

Nous classifions donc ces approches de la manière suivante : dans un premier temps, nous étudions les langages de description sémantique, puis dans un second temps nous détaillons les annotations de langages existants.

3.2.1 Langages de description sémantique

Web Ontology Language for services Web (OWL-S)

Le « Web Ontology Language for Web services » (OWL-S) [44] est un sous-ensemble du langage « Web Ontology Langage » (OWL) [66] dédié à la description sémantique de services Web. Il est compatible avec des formats de description syntaxiques tels que WSDL [43]. OWL [66] est le langage de référence dans le domaine des langages reposant sur la logique de description. Ce langage est construit à partir de RDF [40] qui est un modèle conceptuel permettant la description formelle des ressources du Web sous la forme de triplets : ressource, propriété, aussi appelée prédicat, et valeur. Par exemple, $(< Michael >, < est >, < un homme >)$ est un triplet qui a pour ressource $< Michael >$, pour prédicat $< est >$ et pour valeur $< un\ homme >$.

```
<!-- Description abstraite -->
      <process:Input rdf:ID=''InputLanguage">
            <process:parameterType  rdf:datatype=''&xsd;#anyURI">
                  &this;#SupportedLanguage
            </process:parameterType>
            <rdfs:label>Input Language</rdfs:label>
      </process:Input>
<!-- Description concrète -->
      <grounding:WsdlInputMessageMap>
            <grounding:owlsParameter rdf:resource=''#InputLanguage"/>
            <grounding:wsdlMessagePart rdf:datatype=''&xsd;#anyURI">
                  &groundingWSDL;#inputLanguage
            </grounding:wsdlMessagePart>
      </grounding:WsdlInputMessageMap>
```

Listing 3.1 – Extrait de Description OWL-S

OWL est une extension de RDF Schema [21], c'est-à-dire qu'à l'instar de RDF Schema, OWL définit un vocabulaire permettant de décrire des ontologies, tout en offrant des possibilités de description plus riches. Parmi les notions importantes apportées par OWL, citons les propriétés d'équivalence, d'identité de deux ressources, de différences entre deux ressources, de contraire, de symétrie, de transitivité et de cardinalité. Ces propriétés permettent l'utilisation de raisonneurs afin de déterminer des relations d'équivalence ou de subsomption entre des concepts de domaines de connaissances[13], et d'évaluer automatiquement la compatibilité entre différentes représentations sémantiques.

Une description OWL-S se compose de trois éléments : le *service profile*, le *process model*, et le *grounding*, qui décrivent respectivement *« que fait le service »*, *« comment le service fonctionne »* et *« comment accéder au service »* :

- Le *service profile* décrit les fonctionnalités des services Web, il est utile pour leur découverte et leur sélection.
- Le *process model* détaille la sémantique des données échangées, au niveau des messages échangés entre services Web.
- Le *grounding* spécifie l'encodage des données échangées, les protocoles de communication, ainsi que tous les détails concrets nécessaires à l'invocation du service.

OWL-S recommande la séparation entre les vues de haut et de bas niveau concernant la description des données échangées entre services Web, comme l'illustre la description des paramètres d'entrée d'un service Web présenté dans le listing 3.1.

La vue abstraite, dite de haut niveau, attache le service Web à des descriptions concep-

[13]La subsomption est une relation entre deux concepts, qui signifie que l'un des concepts est un sous-concept de l'autre, c'est-à-dire que les attributs du premier concept forment un sous-ensemble des attributs du deuxième.

tuelles en OWL, décrites dans des ontologies. La vue concrète, ou de bas niveau, décrit la représentation physique du service Web, c'est-à-dire les types de données et les protocoles utilisés. Cette séparation permet différentes représentations physiques du même concept, et renforce le rôle des ontologies dans la représentation abstraite de la sémantique des données.

Suivant les idées développées par les auteurs d'OWL-S, de nombreux travaux de recherche ont été proposés. Notamment, ODESWS [25] est une architecture de description et composition de services Web reposant sur les méthodes de résolution de problèmes, ou « Problem-Solving Methods » (PSM). Dans cette architecture, une fonctionnalité est représentée comme un problème, et la description des fonctionnalités offertes par les services permet d'évaluer les possibilités de résolution du problème.

Web Service Modeling Ontology (WSMO)

L'architecture Web Service Modeling Ontology (WSMO) [6], proposée par le laboratoire DERI, est une architecture conceptuelle, ou métamodèle, visant à expliciter la sémantique des services Web. Elle est organisée autour de quatre éléments principaux :

- **Les services Web** sont définis comme des « entités computationnelles » qui fournissent une fonctionnalité. Une description est associée chaque service, dans le but de décrire sa fonctionnalité, son interface, et ses détails internes.
- **Les Objectifs** servent à décrire les souhaits des utilisateurs(trices) en terme de fonctionnalités requises. Les objectifs sont une vue orientée utilisateur(trice) du processus d'utilisation des services Web, ils sont une entité à part entière dans le modèle WSMO. Un objectif décrit la fonctionnalité, les entrées/sorties, les préconditions et postconditions d'un service Web.
- **Les Médiateurs** sont utilisés pour résoudre de nombreux problèmes d'incompatibilité, telles que les incompatibilités de données dans le cas où les services Web utilisent différentes terminologies, les incompatibilités de processus dans le cas de la combinaison de services Web, et les incompatibilités de protocoles lors de l'établissement des communications.
- **Les Ontologies** fournissent la terminologie de référence aux autres éléments de WSMO, afin de spécifier le vocabulaire du domaine de connaissance d'une manière interprétable par les machines.

Contrairement à OWL-S, WSMO inclut les médiateurs comme des composants centraux de son architecture. Les hétérogénéités rencontrées lors de l'utilisation de services Web sont gérés par les types de médiation suivants :

- La **médiation de données** résout les incompatibilités de représentation des don-

nées.
- La **médiation de processus** est relative à la logique applicative de la composition.
- La **médiation de protocoles** adapte les différents protocoles de communication utilisés.

Ces types de médiation sont pris en charge par quatre familles de médiateurs :

- **Les GG-médiateurs** permettent d'effectuer la médiation entre deux objectifs, GG signifiant « goal-goal ». Cela signifie qu'ils permettent d'établir des correspondances entre objectifs, en se servant des ontologies d'objectifs disponibles dans le cadre de WSMO.
- **Les WG-médiateurs**, avec WG pour « Web service-goal », établissent les correspondances entre les fonctionnalités offertes par les services Web et les requêtes des utilisateurs(trices), qui sont toutes les deux définies comme des objectifs. L'intérêt de ce type de médiateurs est d'aider la découverte et la sélection de services Web.
- **Les WW-médiateurs**, avec WW pour « Web service-Web service », établissent les correspondances entre services Web. Leur tâche est de résoudre les conflits au niveau des données, du protocole, et du processus de composition. Ils sont mis en œuvre lors de l'orchestration des services Web au sein d'une composition.
- **Les OO-médiateurs**, avec OO pour « ontology-ontology », sont destinés à résoudre les conflits entre ontologies. Les autres types de médiateurs énoncés ci-dessus, mais aussi n'importe quel élément de l'architecture WSMO qui pourrait utiliser les ontologies, peut utiliser un OO-médiateur pour résoudre un conflit sémantique. Le travail des OO-médiateurs consiste à établir des correspondances entre les terminologies contenues dans les différentes ontologies pour les intégrer en une représentation homogène des données. Cette représentation homogène permet de résoudre les hétérogénéités sémantiques et de répondre aux requêtes soumises par les composants de l'architecture WSMO.

DIANE Elements (DE) et DIANE Service Description (DSD)

Les langages DIANE Elements (DE) et DIANE Service Description (DSD) sont des langages orientés objet construits à partir d'une analyse des conditions requises pour la description des services Web sémantiques, et des difficultés de OWL-S et WSMO à remplir ces conditions [39].

Les langages DIANE Elements (DE) et DIANE Service Description (DSD) utilisent les notions d'ensembles configurables et de logique floue pour améliorer la découverte sémantique de services Web. DE est un langage général d'ontologie qui contient des caractéristiques spécifiques visant à améliorer la description de services Web sémantiques.

Ce langage orienté objet hérite de la F-logique pour décrire les concepts d'attributs, de types de données primitifs (empruntés à XML Schema), et d'élément restreints (seulement huit types primitifs). La F-logique est un langage déductif orienté objet de représentation des connaissances, qui combine la sémantique déclarative et l'expressivité des langages déductifs avec les capacités de modélisation du modèle orienté objet. La séparation entre schéma et instances est aussi empruntée à la logique de description.

Le langage DSD quant à lui, utilise les constructions fournies par DE afin de décrire les services Web. Il est construit autour des notions de description de requête et d'offre de service : la description d'un service constitue une offre et une demande utilisateur constitue une requête. L'approche proposée fournit une architecture globale pour la description de services Web sémantiques, avec un fort support de raisonnement, qui emprunte de nombreux avantages apportés par les approches OWL-S et WSMO.

3.2.2 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]. Ces extensions sont détaillées ci-dessous.

Annotation du processus métier

SEmantic Service MArkup (SESMA). Le langage SESMA [60] 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.

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 [46]. Ces extensions contiennent des références aux concepts décrits dans les ontologies de description du domaine de connaissance associé au service Web, afin de spécifier la sémantique des messages, mais aussi les préconditions et effets des opérations. WSDL-S vise à fournir une approche « allégée » 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) [65] 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 utilisé pour la description sémantique. SAWSDL fournit les mécanismes permettant d'attacher des concepts décrits dans des ontologies aux annotations des descriptions WSDL. Cette proposition étend WSDL-S, elle peut 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'extensibilité sont définis par défaut : l'attribut « modelReference » permet l'association entre un composant WSDL et un concept d'une ontologie, et les attributs « liftingSchemaMapping » et « lowering-SchemaMapping » sont ajoutés aux définitions de types pour spécifier les correspondances entre les éléments du schéma des données et l'information sémantique de l'ontologie.

Annotation des registres

Extension sémantique de UDDI. Paolucci et coll. [58] 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 [1], 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.[29] 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 que 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.

3.3 Conclusion

Dans ce chapitre, nous avons présenté les propositions de description sémantique des services Web. En effet, le chapitre précédent a montré l'importance du niveau sémantique pour la réconciliation de services Web participant à une composition, par les nombreuses utilisations de descriptions sémantiques pour les besoins de la médiation [23, 36, 70, 15]. En effet, comme le montrent la multiplicité et l'évolution rapide des langages et annotations proposés, la résolution des conflits sémantiques, principalement par la description explicite des données, paraît clairement constituer la prochaine étape vers l'interopérabilité entre systèmes distribués.

Ainsi, les approches de description sémantique des services Web suivent un objectif commun : décrire de manière explicite et gérer la sémantique associée aux services Web, car cette sémantique reste absente de leur pile standard de protocoles. Malgré cet objectif commun, deux orientations complètement différentes caractérisent les approches étudiées :

– Les langages sémantiques tentent de s'imposer comme de nouvelles normes de description. Ils souhaitent remplacer les langages actuels comme WSDL, et inscrire les

services Web dans le cadre du Web sémantique. Ces langages apportent les nombreux avantages liés à la description sémantique, et bénéficient des critiques apportées sur les langages précédents. Cependant, ils nécessitent des descriptions plus complexes qui demandent des connaissances importantes lors de la conception d'une description.

– Les annotations enrichissent les langages existants afin de décrire la sémantique des services Web. Ces approches sont avantageuses dans la mesure où elles exploitent les éléments d'extensibilité des langages existants, et restent compatibles avec ces derniers. En effet, il est contraignant de devoir renoncer aux normes existantes afin de bénéficier des avantages apportés par les approches proposées.

Notons que la plupart des approches décrites, excepté WSMO, se concentrent sur la découverte et la correspondance de concepts plutôt que sur la médiation. Même dans l'approche WSMO, le concept de médiation se limite à l'établissement de correspondances sémantiques entre termes d'ontologies. Cela suppose que les services Web découverts ont adhéré à une ontologie commune, et adoptent la représentation des données de cette ontologie. Cela suppose aussi une adaptation de la sémantique locale des services Web à la sémantique de l'ontologie partagée. Autant de suppositions difficiles à soutenir dans le contexte des services Web et d'Internet. Dans notre contribution, nous tentons d'apporter une réponse à ce verrou scientifique, par l'utilisation du contexte pour la représentation sémantique des données.

Deuxième partie

Conception et réalisation

Chapitre 4

Approche orientée contexte pour l'échange de données entre services Web

Sommaire

4.1 Introduction

Dans les chapitres précédents, nous avons mis en valeur le rôle primordial de l'information sémantique dans le processus de médiation. Nous avons analysé les limites des approches existantes concernant la représentation sémantique des données, ainsi que les besoins requis pour effectuer la médiation des données dans une composition.

À partir des propositions étudiées, nous avons constaté que l'interopérabilité sémantique entre services Web ne peut être atteinte que lorsque les exigences suivantes sont vérifiées :

- Des mécanismes de médiation sont insérés dans le processus de composition. De nombreuses approches ont été proposées : à base d'adaptateurs, de communautés, de services Web et de descriptions étendues. Nous avons montré les avantages apportés par les solutions à base de services Web par rapport aux autres propositions, et la nécessité d'utiliser des descriptions étendues pour les besoins de la sémantique.
- Les propriétés sémantiques des services Web sont explicitement décrites. Les solutions de médiation actuelles reposent sur des extensions des langages existants ou des descriptions sémantiques.
- Les différences entre les représentations sémantiques locales sont uniformisées ou résolues à l'aide de mécanismes de médiation. Les solutions actuelles supposent une adhésion des services Web à une ontologie commune ou utilisent des médiateurs entre ontologies.

Dans la suite de ce document, nous proposons une solution de médiation sémantique, qui a pour objectif de répondre aux exigences ci-dessus afin de résoudre les hétérogénéités sémantiques entre services Web dans le cadre d'une composition. Nous décrivons dans ce chapitre notre approche pour améliorer l'échange des données entre services Web au niveau sémantique. Cette approche repose sur la notion de contexte[14] pour la représentation des données, et se compose de quatre parties :

1. La première partie motive le besoin de médiation sémantique des données à l'aide d'un scénario de planification de voyage en ligne. Les hétérogénéités sémantiques liées au contexte des données sont présentées dans le cadre de cet exemple.
2. La deuxième partie propose une classification des hétérogénéités sémantiques des données. Nous présentons la notion de propriété sémantique afin de décrire le contexte des données échangées entre services Web et de structurer notre classification.
3. La troisième partie présente les fondements de l'utilisation du contexte pour dé-

[14]Nous utilisons la notion de contexte présentée dans le chapitre 1, où le contexte est décrit comme un ensemble de métadonnées décrivant les différents aspects sémantiques d'un objet.

crire la sémantique des données, via une analyse chronologique mettant en valeur l'évolution des approches qui ont inspiré notre travail. À travers cette analyse, les points forts et inconvénients des approches existantes sont mis en lumière, et les fondements de notre modèle sont détaillés.

4. La quatrième partie présente notre modèle orienté contexte pour la représentation des données échangées entre services Web [48]. Ce modèle bénéficie des apports des parties précédentes, et se construit autour de deux notions fondamentales : le *contexte* et l'*objet sémantique*, dont nous spécialisons les définitions pour la composition de services Web.

4.2 Exemple de motivation

Dans cette section, nous examinons les besoins relatifs à l'interprétation des données échangées entre services Web composés, à l'aide d'un scénario de planification de voyage en ligne. Ce simple exemple de composition permet d'illustrer concrètement notre problématique. Cependant, il est important de noter que les perspectives d'application de notre travail sont beaucoup plus vastes et concernent tous les domaines dans lesquels la composition de services Web peut être envisagée.

4.2.1 Scénario de planification de voyage

Considérons la planification d'un voyage au Japon, qui comprend la réservation de billets d'avion et la location d'un véhicule pour la durée du voyage. Cette planification nécessite la coordination de plusieurs services Web au sein d'une même composition, dans notre cas :

- un service Web de réservation de billets d'avion,
- un service Web de location de véhicule,
- un service Web pour additionner les prix des services précédents.

Les étapes de sélection des services et de conception du processus métier ont été réalisées au préalable, par exemple à l'aide d'un éditeur graphique[15]. Le langage WS-BPEL est utilisé pour décrire le processus métier. WS-BPEL [26] est le langage le plus utilisé actuellement. Il bénéficie d'une grande communauté d'utilisateurs, et est considéré comme le langage le plus techniquement avancé, car il réunit les avantages des langages XLang [72] et WSFL [41]. Cependant, malgré l'inclusion du langage XPath dans les fonctionnalités

[15]Tel que Oracle BPEL Designer, IBM WebSphere Studio Application Developer, IBM BPWS4J Editor, Vergil VCAB Composer, ou Active Endpoints ActiveWebflow Designer.

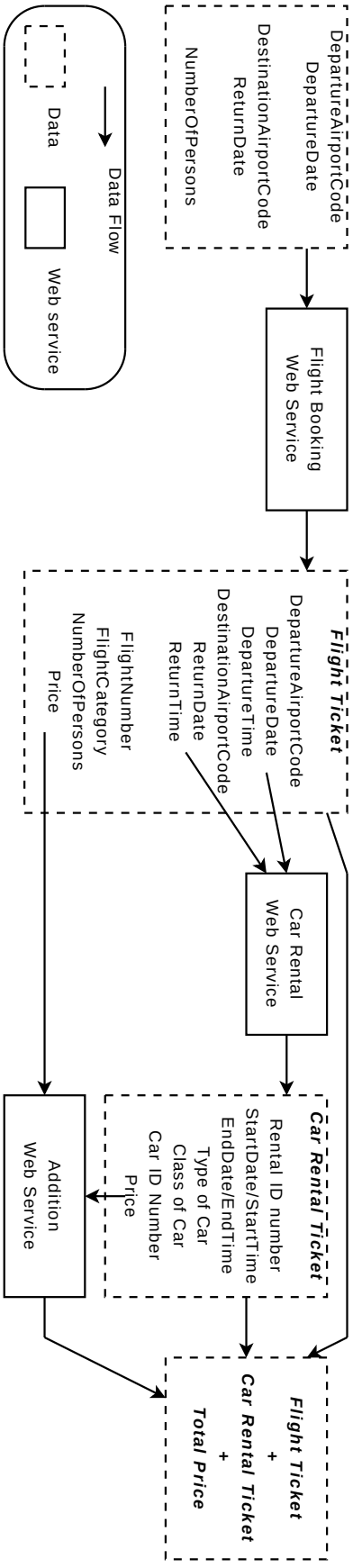

FIG. 4.1 – Processus métier de planification de voyage

de WS-BPEL, le traitement des données directement dans le processus métier est limité aux nombres entiers [26], ce qui justifie l'utilisation d'un service Web supplémentaire pour additionner les prix des deux autres services.

Cette composition de services Web est représentée par le processus métier de la figure 4.1. De nombreuses 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.

4.2.2 Hétérogénéités liées 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 « Flight Booking » assure la réservation de billets d'avion,
- le service « Car Rental » prend en charge la location d'un véhicule pendant la durée du séjour,
- le service « Addition » calcule le prix total généré par les deux services.

Ces services présentent les hétérogénéités de représentation des données suivantes, illustrées par la figure 4.2 :

- le service Web de réservation de billets d'avion est un service Européen, qui traite les prix en Euro et avec un facteur multiplicateur[16] de 1.
- en revanche, le service Web de location de véhicule fonctionne avec la devise locale qui est le Yen Japonais, et utilise un facteur multiplicateur de 1000. Ainsi, la valeur retournée par ce service Web doit être multipliée par 1000 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 ». De plus, les représentations de dates et d'heures diffèrent. Le service Web de réservation de billets d'avion adhère à un format Européen (`dd.mm.yyyy` et `12 :00 AM/PM`), alors que le service Web de location de véhicule adopte une notation Japonaise (`yy.mm.dd` et `24 :00`). 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

[16] Un facteur multiplicateur est un nombre utilisé comme multiplicateur pour la mise à l'échelle (traduit de WordNet, `http://wordnet.princeton.edu/`).

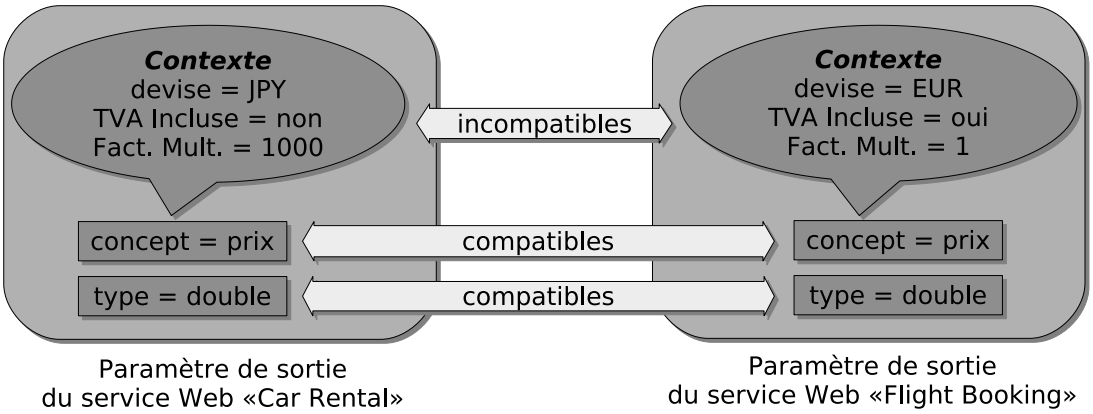

FIG. 4.2 – Conflits entre les contextes des services « Flight Booking » et « Car Rental »

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.

Actuellement, la communauté des services Web ignore les hétérogénéités sémantiques liées au contexte :

- Lorsque la sémantique des données est prise en charge dans une composition, la solution classique consiste à utiliser une ontologie commune qui fixe le contexte des données, et ne permet qu'une interprétation unique pour un concept donné. Ainsi, dans notre exemple, une ontologie serait utilisée pour décrire le concept de « prix », qui serait interprété selon une devise unique, par exemple « Dollar Américain », et un facteur multiplicateur unique, par exemple « 1 ». Les services « Flight Booking », « Car Rental » et « Addition » devraient alors être adaptés à ce contexte. Cette solution nécessite une adaptation de la sémantique des services Web au contexte imposé par l'ontologie, ce qui réduit les possibilités d'interactions entre services Web. En effet, il est fastidieux pour les fournisseurs d'adapter manuellement la sémantique de chaque service Web à celle de chaque ontologie avec laquelle le service peut être amené à interagir.
- Lorsque la sémantique des données n'est pas prise en charge par la composition, la conversion des données d'un contexte à l'autre est effectuée manuellement durant l'étape d'orchestration. Cette solution demande aux concepteurs de composition de larges connaissances, concernant à la fois les langages de traitement des données utilisés par le langage de composition (tels que XPath ou XSLT pour le langage WS-BPEL), et les différents contextes dans lesquels les données doivent être interprétées. De plus, cette solution n'est pas adaptable à une approche dynamique qui sélectionne les services Web juste avant chaque exécution, car l'échange des données

d'un contexte à un autre sans adaptation produirait des non-sens sémantiques.

L'exemple développé ci-dessus montre l'importance du contexte pour l'interprétation des données. C'est à partir de cette notion de contexte que nous construisons notre approche de médiation. Afin de faciliter la résolution des hétérogénéités sémantiques des données échangées entre les services Web, nous commençons par classifier les types d'hétérogénéités sémantiques ci-après. Nous introduisons tout d'abord la notion de propriété sémantique pour décrire le contexte des données et structurer notre classification.

4.3 Classification des hétérogénéités sémantiques

4.3.1 Notion de propriété sémantique

Le rôle d'une ontologie est de décrire un domaine de connaissances, en explicitant les concepts utilisés ainsi que les relations entre ces concepts. Afin d'obtenir une interprétation homogène de la part des utilisateurs, les éléments du contexte permettant d'interpréter correctement les concepts de l'ontologie sont fixés. Ces éléments, que nous appelons des « propriétés sémantiques », décrivent les divers aspects et caractéristiques sémantiques d'un concept. Dans l'exemple de planification de voyage développé précédemment, le facteur multiplicateur et la devise associés au concept de prix sont des propriétés sémantiques, comme illustré par la figure 4.3.

Chaque propriété sémantique est identifiée par un nom et possède une valeur qui est une instance d'un type (entier, chaîne de caractères, etc.). Aussi, une propriété sémantique peut voir sa signification précisée par une ou plusieurs propriétés sémantiques qui lui sont rattachées. Le contexte est ainsi constitué d'un ensemble de propriétés sémantiques qui peut se représenter d'une manière arborescente. Nous classifions les hétérogénéités qui peuvent se présenter entre des contextes différents selon trois types, qui sont liés aux caractéristiques des propriétés sémantiques : les hétérogénéités de valeur, les hétérogénéités structurelles, et les hétérogénéités sémantiques.

4.3.2 Types d'hétérogénéités

Hétérogénéités de valeur

Ces hétérogénéités concernent les valeurs des propriétés sémantiques. Comme présenté dans l'exemple section 4.2, les prix sont généralement supposés avoir un facteur multiplicateur de 1. Cependant, certaines organisations gèrent les prix avec un facteur multiplicateur

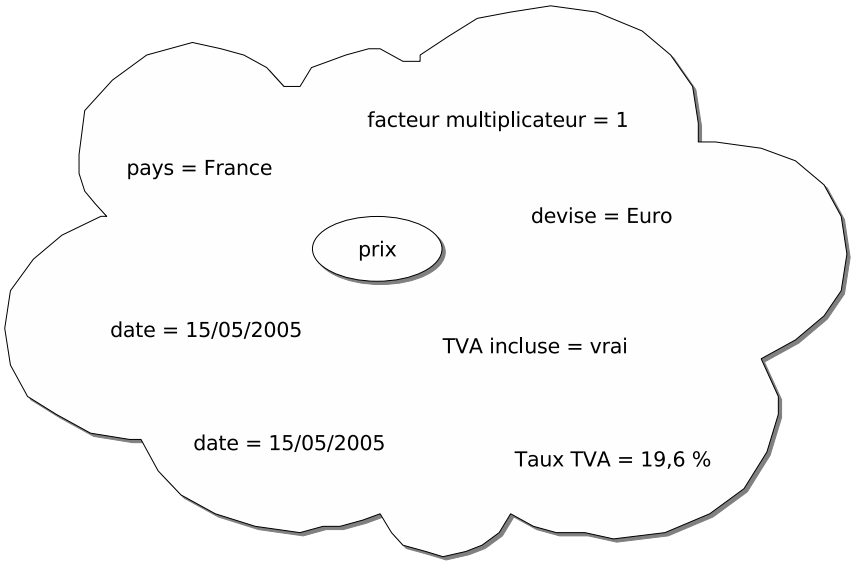

FIG. 4.3 – Un exemple de « prix » et son contexte

de 1000 par souci de lisibilité. Dans ce cas, nous constatons que les valeurs de la propriété sémantique *facteur multiplicateur* du concept *prix* diffèrent. Nous identifions ces hétérogénéités par le terme « hétérogénéités de valeur », car elles concernent les valeurs des propriétés sémantiques.

Afin de résoudre de telles hétérogénéités, il est nécessaire de décrire explicitement la propriété sémantique, le domaine de valeurs qui lui est associé, et les implications sur l'objet sémantique d'une conversion entre deux valeurs. Dans notre exemple, le passage d'un facteur multiplicateur à un autre implique la multiplication du prix par le facteur multiplicateur d'origine, puis sa division par le facteur multiplicateur ciblé. La conversion d'une longueur en unité de mesure anglaise (pouces) vers une unité de mesure française (centimètres) nécessite la multiplication de la valeur par $2,54$.

Hétérogénéités structurelles

Selon les services Web, la représentation structurelle des propriétés sémantiques associées aux concepts d'un domaine d'application peut changer. En effet, le concept de *prix* par exemple, peut être décrit suivant différentes structures, selon les besoins des agences de réservation de vol. Certaines agences pourraient employer des facteurs multiplicateurs différents et ignorer la monnaie utilisée, parce que leurs partenaires emploient tous la même devise, tandis que d'autres agences pourraient faire des suppositions contraires, à savoir un seul facteur multiplicateur, mais des monnaies différentes. Les structures de représentation du contexte doivent être explicitement décrites, pour qu'il soit possible de

déterminer si deux structures de contexte sont compatibles. Dans le cadre des services Web, la compatibilité des structures de représentation de données est atteinte lorsque la structure de sortie du service émetteur des données est suffisamment riche pour subvenir aux besoins de la structure d'entrée du service destinataire.

Hétérogénéités sémantiques

Les hétérogénéités sémantiques sont liées à la signification du vocabulaire utilisé pour décrire les propriétés sémantiques. Elles restent invisibles aussi longtemps que le contexte est implicite, mais elles apparaissent lorsque ce dernier est explicitement décrit. Par exemple, pour décrire la propriété sémantique indiquant si la taxe sur la valeur ajoutée (TVA) est incluse dans le prix, un fournisseur pourrait employer le mot « VATIncluded », tandis qu'un autre pourrait employer le mot « TVAIncluse ». Afin de résoudre de tels conflits sémantiques, la description des correspondances entre les termes utilisés par les propriétés sémantiques est nécessaire. Ces correspondances pourraient être insérées directement dans l'ontologie de domaine, cependant, nous estimons que ce type d'information doit être décrit séparément, car une ontologie de domaine est un accord sur une représentation commune des connaissances d'un domaine, et n'a pas pour but de restreindre les suppositions locales des utilisateurs sur l'interprétation des données. Nous proposons une solution pour décrire ces hétérogénéités sémantiques dans la section 5.3.

4.4 Description sémantique orientée contexte : synthèse

En général, l'utilisation de la notion de contexte offre la possibilité de personnaliser et d'adapter les applications à différents environnements, vues sur les données, circonstances extérieures, etc. La définition usuelle du terme *contexte* englobe *« tout élément interne ou externe, relatif à l'application ou à l'utilisateur(trice), ou même complètement extérieur, qui pourrait modifier le déroulement d'une interaction » [28].*

Le contexte a aussi été exploité pour apporter une solution efficace aux problèmes d'hétérogénéités sémantiques dans le domaine des bases de données. Dans les travaux de Sciore et coll. [67] et de Sheth et Kashyap [37], le contexte est utilisé pour décrire les différents aspects sémantiques d'un objet, sous la forme d'un ensemble de métadonnées. Ces travaux apportent des perspectives intéressantes pour le domaine des services Web.

Avant de construire notre description sémantique orientée contexte, il est nécessaire d'établir les fondations de cette notion. Pour ce faire, nous effectuons ci-après un rappel des

travaux reposant sur le contexte pour décrire la sémantique des données. Nous adoptons une vue chronologique afin de mettre en valeur l'évolution de ces travaux dans le temps ainsi que le positionnement de notre contribution, et nous analysons les avantages et difficultés d'une adaptation de la notion de contexte au domaine des services Web.

4.4.1 Notion de valeur sémantique

L'utilisation du contexte pour la description des aspects sémantiques trouve son origine en 1994 avec les travaux de Sciore et coll. [67]. Afin de modéliser la sémantique des données et de faciliter les échanges entre systèmes d'information hétérogènes, les auteurs introduisent le concept de *valeur sémantique* comme unité d'échange. Une valeur sémantique est représentée par l'association d'une valeur simple et d'un contexte. Le contexte est défini abstraitement comme :

> *« L'ensemble des données relatives au sens sémantique, aux propriétés et à l'organisation de la valeur sémantique [67] ».*

Le contexte d'une donnée est modélisé par un ensemble fini et récursif de méta-attributs qui peuvent contenir des valeurs différentes. Il fait partie de l'environnement des données (data environment), et se compose d'un schéma et d'une spécification. Le schéma décrit les méta-attributs, leurs propriétés, et d'une manière générale la structure du contexte, alors que la spécification spécifie les valeurs prises par une partie ou l'ensemble des méta-attributs dans un contexte précis. Il est possible que certains méta-attributs ne possèdent pas de valeurs dans certains contextes, laissant des aspects d'une valeur sémantique non spécifiés. Par exemple, la valeur 5 et le contexte ($currency = EUR$) forment une valeur sémantique, ce qui nous permet d'interpréter la valeur 5 comme étant une quantité à compter en Euro.

Effectuer la médiation d'une valeur sémantique d'un contexte à un autre revient à modifier les valeurs de ses méta-attributs. La valeur 5 peut être convertie en Yen Japonais en lui appliquant une fonction de conversion, qui permette de changer la valeur du contexte de EUR à JPY. Pour effectuer ces modifications, les auteurs introduisent le « médiateur de contexte ». Ce dernier effectue la médiation des données d'un contexte à l'autre, et repose sur une ontologie commune pour interpréter le vocabulaire utilisé par les contextes des données. Afin d'appliquer les concepts présentés au modèle relationnel, Sciore et coll. introduisent une extension du langage SQL, appelés Context-SQL (C-SQL), dans lequel les contextes des valeurs sémantiques sont explicitement décrits et pris en compte. Les méta-attributs nécessaires sont déclarés dans les tables des bases de données comme des attributs normaux ; ils sont rattachés aux attributs ou méta-attributs qu'ils décrivent. Ce

travail fournit la première approche orientée contexte, qui subira plusieurs améliorations par la suite.

4.4.2 Notion d'objet sémantique

En 1999, Bornhövd 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 [17, 16] et [34]. L'idée dans ces travaux est de reprendre les fonctionnalités du modèle initial tout en utilisant un formalisme logique orienté objet adaptable aux données semi-structurées pour représenter l'information et son contexte.

L'architecture COIN proposée par Goh et coll. [34] repose sur trois composants principaux :

- Un **modèle de domaine** décrit les connaissances du domaine concerné au niveau sémantique. Il contient des objets primitifs et des objets sémantiques, qui sont respectivement des instances de types primitifs et de types sémantiques. Les types primitifs sont du type chaîne de caractères, entiers, réels, etc., et les types sémantiques sont des types complexes qui supportent la description du contexte.
- Des **axiomes d'élévation** établissent les correspondances entre les attributs des sources de données et les concepts décrits dans le modèle de domaine. Ils élèvent des objets simples au niveau sémantique en identifiant le type sémantique correspondant à l'objet concerné et en supportant l'instanciation de l'objet sémantique.
- Des **axiomes de contexte** décrivent les contextes associés aux émetteurs et récepteurs des données, et définissent le contexte d'interprétation des données. Deux groupes d'axiomes sont distingués : le premier groupe définit la sémantique des données en associant des valeurs aux éléments du contexte, et le deuxième groupe d'axiomes spécifie les méthodes de conversion associées aux éléments du contexte dans le but de permettre la conversion de l'objet sémantique entre plusieurs contextes.

Le modèle développé par Goh et coll. a pour but de fournir un cadre pour la médiation automatique de requêtes sur des bases de données. Le formalisme logique adopté pour la médiation de requête est l'abduction. L'abduction est une forme de raisonnement hypothétique qui retrouve des faits à partir de résultats et de règles. Par exemple, étant donné un résultat X et une règle précisant qu'Y implique X, alors un raisonnement par abduction infère l'existence du fait Y comme une explication possible de X. L'architecture COIN utilise la logique abductive, ce qui la différencie du travail de Bornhövd [17, 16], qui présente un modèle similaire à celui de Goh mais adopte une logique déductive classique pour effectuer la médiation de requêtes.

4.4.3 Ajout des perspectives ontologique et temporelle

En 2003, Firat présente eCOIN, une extension du modèle COIN qui inclut les hétérogénéités ontologiques et temporelles comme partie intégrante de l'architecture de médiation proposée [33]. Les hétérogénéités ontologiques incluent les différences de définition dans des ontologies différentes décrivant des concepts similaires. Les hétérogénéités temporelles identifient les changements de sémantique qui se présentent lorsque les données appartiennent à des intervalles de temps durant lesquels la sémantique des données est modifiée.

La solution proposée pour résoudre les hétérogénéités ontologiques repose sur des correspondances entre les concepts appartenant à des ontologies différentes. Ces correspondances sont établies à l'aide d'équations, et d'un solveur d'équations permettant de résoudre les hétérogénéités grâce à un raisonnement par contraintes. Les hétérogénéités temporelles sont présentées comme des travaux futurs.

L'aspect temporel est abordé dans le travail de Zhu et coll. [82]. Les auteurs enrichissent les ontologies des applications avec la notion de temps, en utilisant le modèle COIN. Les concepts temporels sont décrits dans une ontologie. Grâce à ces concepts, les contextes des données sont enrichis avec des indications temporelles. Les contraintes temporelles sont explicitement décrites, afin d'adapter les données aux changements de sémantique sur des périodes déterminées. Les valeurs des éléments du contexte sont donc multiples dans le temps, mais à un moment donné chaque élément du contexte possède une valeur unique.

4.4.4 Analyse des travaux précédents et positionnement

La représentation sémantique orientée contexte permet de transformer les données au niveau sémantique, et aussi de sortir du dilemme entre faible et fort couplage, bien connu des bases de données. En effet, les approches fortement couplées ont des difficultés à fournir des vues multiples d'une source de données, car l'administrateur(trice) système centralise les transformations entre les vues ; et les approches faiblement couplées demandent aux utilisateurs(trices) de comprendre les hétérogénéités présentes entre les données pour les résoudre lors de l'intégration d'une nouvelle source de données.

L'adoption d'une représentation sémantique orientée contexte permet de confier au médiateur de contexte le travail de résolution des hétérogénéités, sans surcharger la tâche de l'utilisateur(trice) ni celle de l'administrateur(trice) système. Seule est requise la présence d'une ontologie globale permettant de préciser le vocabulaire utilisé. De plus, les travaux de Firat [33] et Zhu et coll. [82], apportent des perspectives pour la résolution des conflits d'ontologies et temporels des données.

Ainsi, l'adoption d'une approche orientée contexte offre les avantages suivants dans le cadre de la composition de services Web :

- aucune représentation unique des données n'est favorisée, ce qui n'impose pas aux fournisseurs de services d'adapter leurs sémantiques locales ;
- la résolution des hétérogénéités sémantiques n'est pas confiée aux utilisateurs(trices) ni à l'administrateur(trice) système ;
- les hétérogénéités sémantiques sont explicitées d'une manière interprétable par les machines, ce qui ouvre la perspective d'automatisation des conversions de données par le moyen de fonctions.

Cependant, l'adoption du contexte dans le domaine des services Web n'est pas une tâche triviale, car le paradigme de communication et la représentation des données sont différents. Pour utiliser le contexte dans le domaine des services Web, il faut prendre en considération les contraintes suivantes :

- les services peuvent difficilement former un agrément sur une ontologie commune, car nous sommes placés dans le contexte ouvert d'Internet, où les partenaires engagés dans les compositions peuvent changer très rapidement ;
- les services séparent la description sémantique (abstraite) des données de la description syntaxique (concrète) ;
- un échange de données entre services Web est unidirectionnel. Il a pour origine la sortie du service émetteur et pour destination l'entrée du service récepteur.

En considérant les exigences imposées par l'utilisation du contexte pour une application dans le domaine des services Web explicitées ci-dessus, nous proposons ci-après notre modèle de représentation orienté contexte pour décrire les données entre services Web composés.

4.5 Représentation des données échangées entre services Web : un modèle orienté contexte

4.5.1 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. Cette contrainte justifie notre adoption d'une définition de la notion de contexte plus restrictive que celle des travaux présentés ci-dessus, à savoir :

« Le contexte d'une donnée englobe ***tout élément*** *interne ou externe,*

relatif à la donnée ou même complètement extérieur, qui est ***nécessaire à l'interprétation correcte de la donnée*** *».*

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 données,
- et le contexte des données 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 notre propre 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. Nous proposons de modéliser le contexte comme une arborescence de méta-attributs, qui décrivent de manière explicite les différentes propriétés sémantiques nécessaires à une interprétation correcte des données.

4.5.2 Définition de l'objet sémantique

Comme expliqué dans le chapitre 3, 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 Schema, 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 :

$$S = (c, v, t, C), \text{ où}$$

– c est le concept auquel l'objet sémantique se réfère,
– 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 autodescriptive. Une définition formelle d'un contexte C est :

$$C = \{(c_1, v_1, t_1, C_1), \ldots, (c_n, v_n, t_n, C_n)\}, n \in \mathbb{N},$$

où $(c_i, v_i, t_i, C_i), 1 \leq i \leq n$, sont les modifieurs qui décrivent les différentes propriétés sémantiques de S. Ainsi, il est possible de former des descriptions récursives, et de représenter le contexte des objets sémantiques avec des structures arborescentes composées de modifieurs. Notre définition de l'objet sémantique est résumée par la figure 4.4.

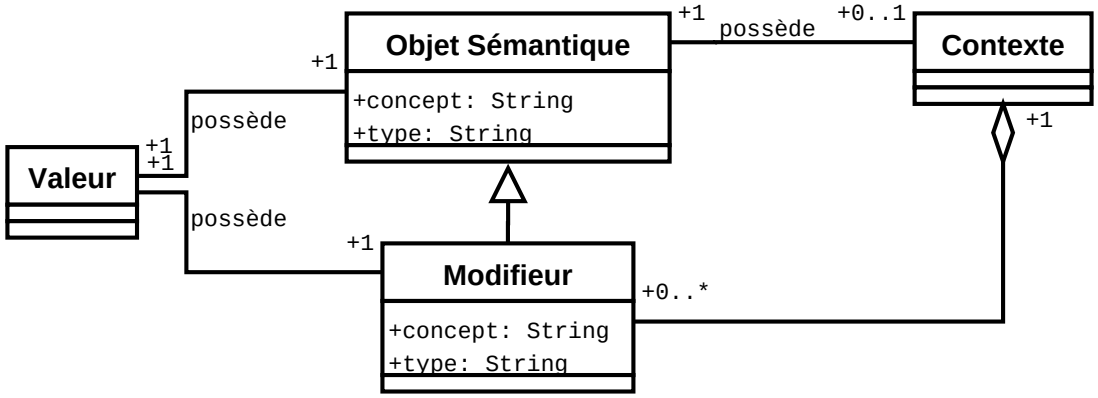

FIG. 4.4 – Représentation UML de l'objet sémantique

4.5.3 Modifieurs statiques et dynamiques

À 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, ces modifieurs pouvant être indistinctement statiques ou dynamiques. L'utilisation de règles logiques dans le fonctionnement de notre architecture de médiation est expliquée dans le chapitre 5.

Par exemple, la valeur du modifieur « *Format de date* » de l'exemple illustré par la figure 4.5 peut être déduite lorsque l'on connaît la valeur du modifieur « *Pays* », qui appartient au même contexte. La règle logique qui permet cette déduction peut être décrite comme suit :

« Si *pays* = *France*, 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 ».

De manière formelle, étant donné un modifieur M appartenant à un contexte C tels que $M = (c_m, v_m, t_m, C_m) \in C$, alors M est qualifié de *dynamique* si et seulement si :

$$\forall v_m \in M,\ \exists f : \{Dom(t_m) \times \cdots \times Dom(t_m)\} \mapsto Dom(t_m) \wedge \exists \{M_1, \ldots M_i, \ldots, M_n\},$$
$$s.t.\ M_i = (c_i, v_i, t_i, C_i) \in C \wedge M_i \neq M \wedge f(v_1, \ldots, v_i, \ldots, v_n) = v_m.$$

avec $1 \leq i \leq n$ et $i, n \in \mathbb{N}$.

Illustration. La figure 4.5 donne un exemple d'objet sémantique $\boldsymbol{S}$ qui peut être envoyé au service Web « Addition » de notre scénario de planification de voyage en ligne :

– 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.

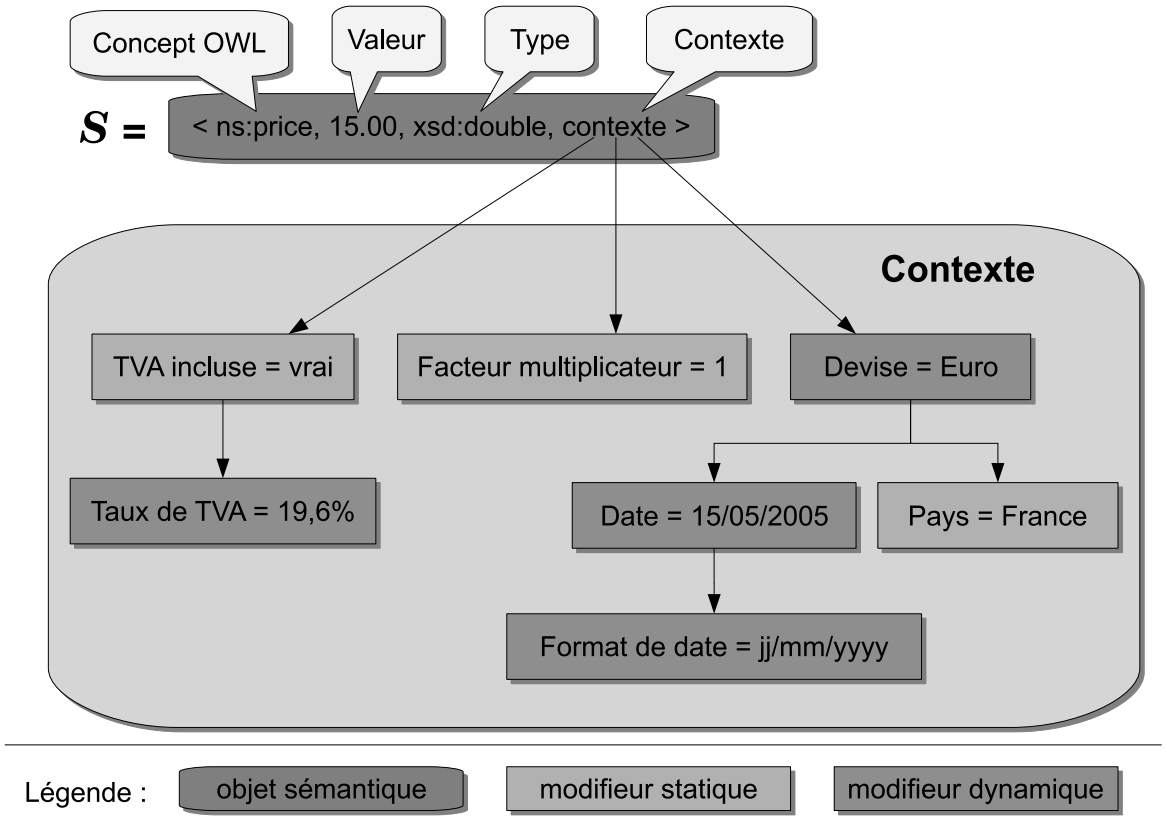

FIG. 4.5 – Représentation de l'objet sémantique S avec son contexte

4.5.4 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. Nous classifions ci-dessous les différentes propriétés des fonctions de conversion pour les objets sémantiques et identifions différentes catégories de fonctions, selon les attributs de l'objet sémantique sur lesquels les fonctions s'appliquent.

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.

Fonctions de conversion totales et non totales. Une fonction de conversion totale convertit de et vers n'importe quelle valeur de son domaine de définition. Les fonctions de conversion d'unités de distance sont totales. Par exemple, la fonction qui convertit des pouces en centimètres est totale, car 1 pouce vaut $2,54$ centimètres, et il est possible de convertir n'importe quelle valeur d'une unité à l'autre.

La conversion de précision est un exemple de fonction de conversion non totale. En effet, la valeur 1.25762 peut être convertie en une valeur avec une seule décimale de précision, elle devient alors 1.2, mais elle ne peut pas être convertie de nouveau vers une valeur de meilleure précision.

Fonctions de conversion avec pertes et sans pertes. Une fonction est dite sans pertes si elle peut être appliquée plusieurs fois sur le même objet sans perte d'information. Une fonction d'archivage de données est dite sans pertes, car les données originales peuvent être retrouvées par la suite. Cependant, une fonction qui convertit une image au format « bitmap » (BMP) vers le format « Joint Photographic Experts Group » (JPEG) est une fonction avec pertes, à cause de l'algorithme de compression de ce dernier format.

Fonctions de conversion monotones croissantes. Une fonction monotone croissante a pour particularité de préserver l'ordre original des valeurs. Par exemple, la fonction de conversion entre les échelles de température Celsius et Fahrenheit est une fonction monotone croissante. On appelle aussi ces fonctions des morphismes.

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*.

Fonctions de conversion contextuelles. Ces fonctions s'appliquent sur les modifieurs des objets sémantiques. Elles changent l'interprétation de l'objet sémantique auquel les modifieurs sont rattachés, et par la même occasion la valeur même de l'objet sémantique. Ces fonctions sont stockées comme des règles et peuvent requérir des accès à des sources de données présentes sur Internet. Par exemple, les fonctions de conversion de prix d'une devise à l'autre peuvent appeler des fournisseurs de taux de change via Internet afin de convertir les prix en utilisant des taux de change actuels.

Fonctions de conversion de type. Elles changent uniquement le type de représentation de la valeur de l'objet sémantique. Ces fonctions peuvent être stockées dans une librairie de conversion associée au système de représentation des données utilisé, et ne sont pas sujettes à de fréquents changements. Dans notre cas, les possibilités offertes par le langage XML Schema sont exploitées pour effectuer ces conversions, comme par exemple, la conversion d'un entier en chaîne de caractères.

Fonctions de conversion de concept. Ces fonctions s'appliquent sur le concept de l'objet sémantique. Elles sont utilisées lorsque deux concepts utilisés par des ontologies différentes sont mis en relation. L'architecture que nous présentons dans le chapitre suivant nous permet de nous passer de cette catégorie de fonctions, par l'utilisation d'ontologies spécifiques pour représenter le contexte des données, comme détaillé dans la section 5.3.

4.6 Conclusion

Dans ce chapitre, nous avons proposé une approche pour l'échange contextuel des données entre services Web. Cette approche est motivée par un scénario de planification de voyage en ligne, et construite sur la base d'une classification des hétérogénéités sémantiques des données échangées entre services Web. Aussi, une analyse des avantages et inconvénients liés à l'utilisation du contexte pose les fondations de notre approche, qui repose sur notre modèle contextuel de représentation des données. Ce modèle est inspiré des modèles orientés contexte existants, mais il a été spécifiquement conçu pour représenter les données échangées entre services Web. Certains aspects, comme la modélisation des contraintes temporelles, n'ont pas encore été abordés et sont des perspectives de travaux futurs. Dans le chapitre suivant, nous exploitons les avantages de notre modèle à travers la mise en œuvre de notre architecture de médiation sémantique pour la composition de services Web.

Chapitre 5

Médiation orientée contexte pour les services Web

Sommaire

5.1 Introduction

Nous avons présenté dans la partie I les différents axes de recherche et travaux relatifs à notre problématique de médiation sémantique pour les services Web. Nous avons montré les limites des approches de médiation et de description sémantique existantes. Dans le chapitre 4, nous avons proposé une approche pour l'échange de données entre services

Web, reposant sur un modèle orienté contexte de représentation sémantique des données construit autour de la notion d'objet sémantique. Dans ce chapitre, nous exploitons les fonctionnalités offertes par cette notion d'objet sémantique et le modèle développés précédemment pour construire une architecture de médiation pour les services Web.

En effet, rappelons qu'une composition peut être confrontée à de nombreuses hétérogénéités. Notamment, des hétérogénéités sémantiques peuvent se présenter entre les données échangées par les services Web. Une médiation des données 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 tâche 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 [49] 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. Nous illustrons le fonctionnement de l'architecture proposée avec un processus de composition WS-BPEL correspondant au scénario développé dans la section 4.2.

5.2 Architecture conceptuelle

L'architecture conceptuelle soutenant notre architecture se compose de trois couches principales, présentées par la figure 5.1 :

- **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.
- **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.
- **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.

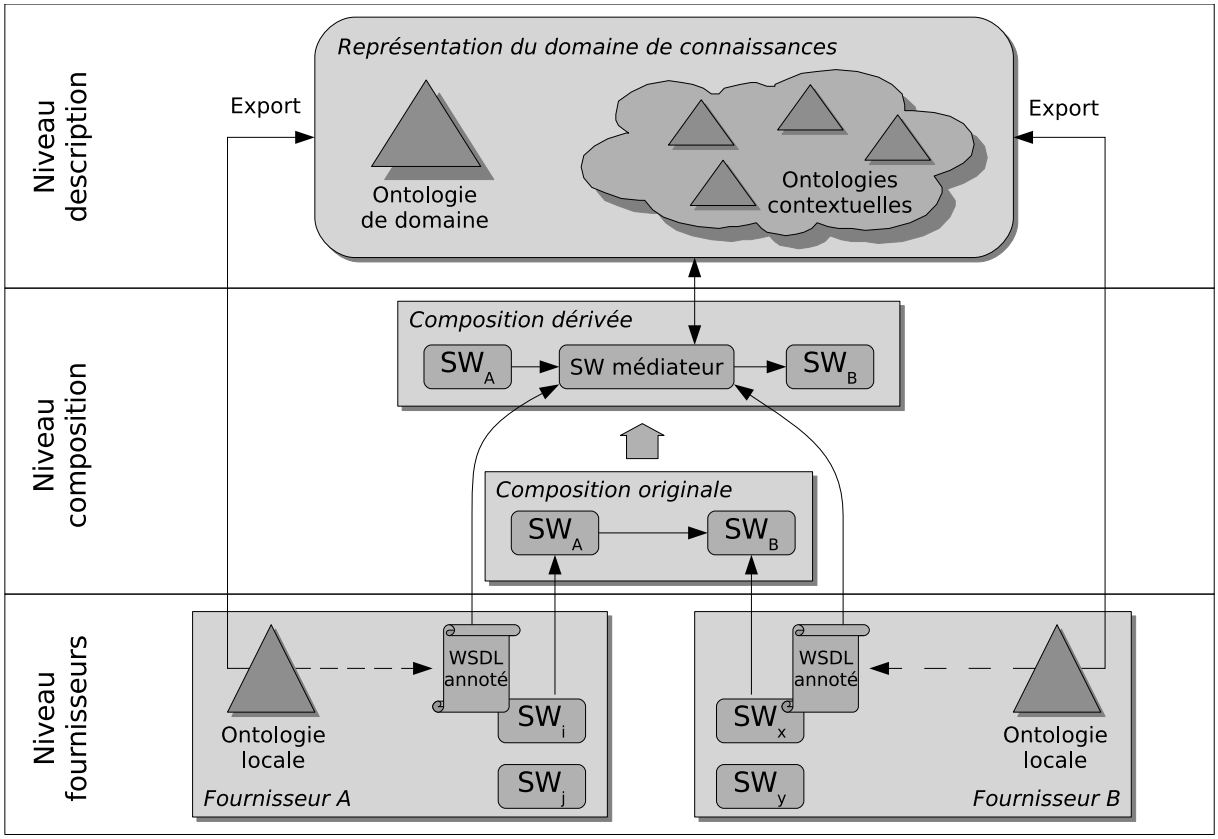

FIG. 5.1 – Architecture conceptuelle de l'approche de médiation proposée

Un des objectifs de notre architecture de médiation est de limiter les tâches affectées aux fournisseurs de services, afin de faciliter son adoption par un grand nombre de fournisseurs. Cependant, sa mise en place requiert certaines actions inévitables de la part des fournisseurs. Nous organisons ces actions comme suit :

1. Les fournisseurs doivent décrire de manière explicite la sémantique utilisée par leurs services Web. Cette tâche reste à leur charge, car ils sont les plus aptes à décrire la sémantique de leurs services. En effet, une annotation de la description d'un service par un acteur autre que le fournisseur du service contiendrait une probabilité d'erreurs qui ne garantirait pas un fonctionnement optimal de notre architecture. Considérant que de nombreux services existent déjà, et ont adopté le langage de description standard de services WSDL, nous nous orientons vers une annotation des descriptions de services avec l'information nécessaire pour décrire la sémantique qui leur est associée. Cette solution permet aux services existants d'intégrer notre architecture de médiation, sans requérir de la part des fournisseurs la conception

d'une description supplémentaire de leurs services. De plus, cette solution reste compatible avec le format WSDL, qui est la norme actuelle de description de services Web, facilitant l'interopérabilité avec les architectures existantes et par conséquent l'adhésion des fournisseurs à notre solution.

2. Les fournisseurs 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. En effet, ils doivent décrire explicitement les particularités locales d'interprétation de ces concepts, tout en établissant des correspondances avec les interprétations locales des autres fournisseurs. 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, les ontologies contextuelles associées. Cette mise à jour permet d'établir les correspondances avec les représentations locales des autres fournisseurs. L'avantage de cette solution est de limiter le rôle de l'ontologie de domaine à une description minimum des concepts communs, facilitant ainsi l'adhésion des fournisseurs de services. De plus, la mise à jour d'une ontologie contextuelle est seulement nécessaire lors de la première adhésion d'un service dont la représentation locale est inconnue de l'ontologie contextuelle. Tous les services adoptant une sémantique locale existante et décrite dans l'ontologie contextuelle réutilisent l'information existante et n'ont aucune mise à jour à effectuer.

Aussi, des mécanismes de médiation, nécessaires à la résolution des hétérogénéités des données échangées entre les services, doivent être intégrés dans les compositions. Dans ce but, nous proposons un algorithme qui permet d'insérer des services Web médiateurs dans une composition. Dans la suite de ce chapitre, nous détaillons ces solutions concrètes permettant la mise en place de notre architecture de médiation, et présentons les avantages apportés par leur utilisation.

5.3 Intégration du contexte dans la description des services Web

Notre modèle de description des objets sémantiques présenté dans la section 4.5 remplit les conditions permettant de décrire les messages échangés entre services Web ainsi que leur sémantique. Le concept sémantique attaché aux données est décrit dans une ontologie de domaine, et son contexte est explicitement détaillé en utilisant des méta-attributs addi-

tionnels 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.

5.3.1 Stratégies de conception des ontologies

Afin de présenter les motivations à l'origine de notre utilisation d'ontologies contextuelles, nous effectuons un bref rappel des objectifs et spécificités des stratégies existantes de conception d'ontologies. La conception d'une ontologie de domaine peut être vue selon différentes perspectives, illustrées par des approches respectivement appelées approche descendante, ou « top-down », ascendante, ou « bottom-up » et une combinaison des deux appelée « middle-out » [54, 55].

Adopter une approche « top-down » consiste à définir d'abord les concepts les plus généraux d'une ontologie pour obtenir une représentation partagée du domaine de connaissances. Ensuite, cette représentation est ajustée et spécialisée aux besoins locaux des utilisateurs de l'ontologie. Cette stratégie s'avère très efficace dans un monde clos, où le nombre d'utilisateurs et les différences entre leurs représentations sont limités. Cependant, elle montre ses limites dans un environnement ouvert comme Internet, où le nombre d'utilisateurs est potentiellement très élevé et en constante fluctuation, et où les différences de représentation du domaine de connaissances peuvent être très grandes.

Pour ce genre de situation, il est plus recommandé d'adopter une approche « bottom-up », qui part des vues locales des utilisateurs pour suivre un processus de généralisation jusqu'à obtenir une représentation cohérente du domaine de connaissances. Par rapport à l'approche « top-down », cette solution est plus efficace dans le cadre d'Internet, mais en revanche, elle s'avère peu efficace dans un environnement clos, où il est plus simple et rapide de se mettre d'accord sur un modèle commun.

Quant à l'approche « middle-out », elle renforce les concepts intermédiaires de l'ontologie en groupes identifiables, et suit à la fois le processus de généralisation de l'approche « bottom-up » et le processus de spécialisation de l'approche « top-down ». D'une manière générale, il est recommandé de combiner ces trois approches pour obtenir une méthode efficace de conception d'ontologies [55]. Dans notre cas, nous associons deux types d'ontologies aux deux premières approches : les ontologies de domaine sont associées à l'approche « top-down » et les ontologies contextuelles à l'approche « bottom-up ». Les motivations et avantages de notre proposition sont décrits dans la section suivante.

5.3.2 Ontologies de domaine et contextuelles

Analyse et proposition

Nous observons que les hétérogénéités présentées dans la section 4.2 concernent les suppositions locales et implicites des services Web concernant l'interprétation des concepts d'un domaine de connaissances, plus que le domaine de connaissances en soi. Ces suppositions, qui forment le contexte, sont liées aux situations culturelles, géographiques, et temporelles des services Web, c'est-à-dire quand, où et comment ils ont été conçus, déployés et exécutés. Ainsi, nous proposons de distinguer des ontologies contextuelles et de domaine pour représenter séparément ces deux types de connaissances.

Dans la plupart des cas, quand plusieurs utilisateurs essaient de se mettre d'accord sur une ontologie de domaine commune, ils sont déjà placés dans des contextes différents. En particulier dans un environnement ouvert comme celui d'Internet, il est très difficile d'obtenir un agrément commun sur une représentation partagée des connaissances d'un domaine. Cette situation est principalement due aux différents contextes dans lesquels les participants sont placés.

Ainsi, nous distinguons deux inconvénients liés à l'utilisation actuelle 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.

En conséquence, nous définissons les objectifs suivants, qui ont pour but de faciliter la conception d'ontologies, l'adhésion des services Web aux ontologies, mais aussi la réconciliation sémantique des services Web durant leur composition :

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 adhèrent à l'ontologie de domaine.

Définition des ontologies contextuelles

Afin de remplir ces objectifs, nous définissons la notion d'**ontologie contextuelle**. 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. Par 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.

Cette séparation entre ontologies de domaine et ontologies contextuelles 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. C'est à ce moment que les relations d'homonymie et autres hétérogénéités sémantiques sont explicitées dans l'ontologie. Par 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 5.2 montre la séparation entre ontologies contextuelles et de domaine, et clarifie la terminologie utilisée pour la description du contexte.

Intégration des modifieurs

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

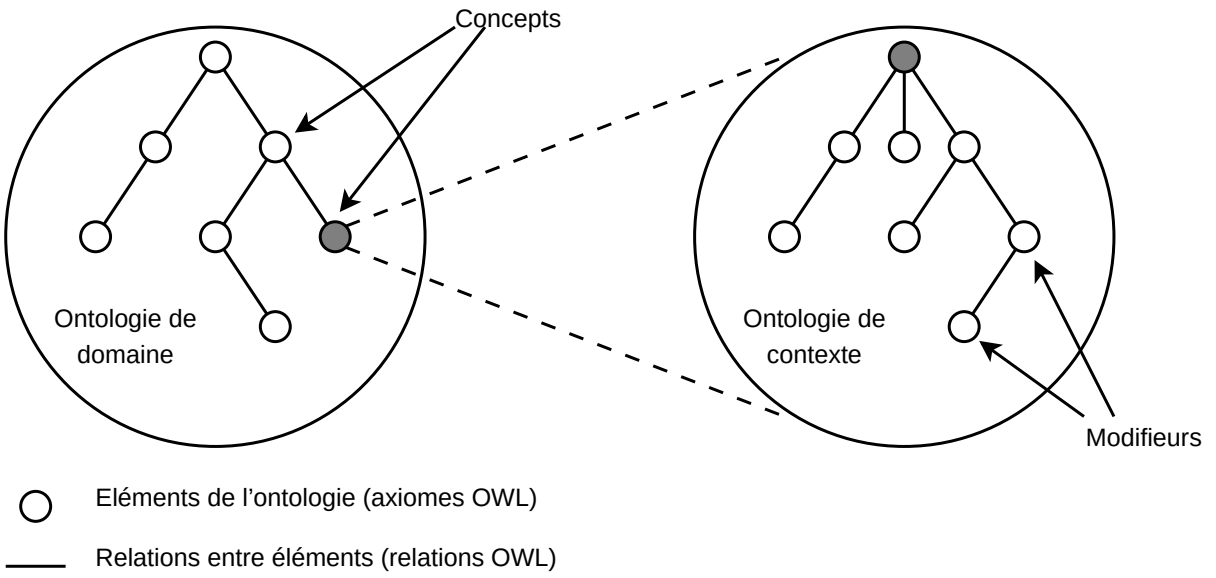

FIG. 5.2 – Séparation des ontologies contextuelles et de domaine

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 protocoles 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. De plus amples détails sont donnés sur ces règles dans la section 6.4. Nous présentons ci-dessous notre méthode pour l'insertion des noms et valeurs des modifieurs statiques dans une description WSDL.

5.3.3 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.

Dans un premier temps, nous allons étudier plus en détail le métamodèle de WSDL pour définir à quel endroit il est nécessaire de placer l'annotation, puis dans un deuxième

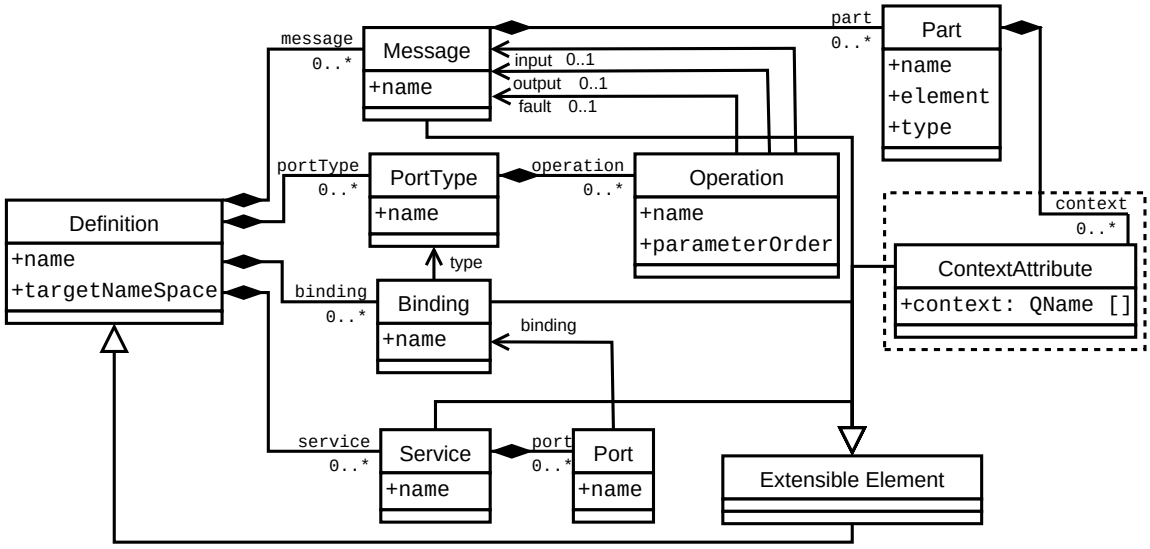

FIG. 5.3 – Représentation du contexte dans le métamodèle WSDL

temps nous développerons notre annotation afin qu'elle respecte les éléments d'extensibilité fournis par la spécification WSDL.

Comme expliqué dans le chapitre 2, 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 [24]. Notre annotation profite d'une telle possibilité d'extension pour que les fichiers WSDL annotés puissent fonctionner indifféremment avec tous les clients de services Web, qu'ils prennent en compte l'annotation que nous proposons ou pas. La figure 5.3 détaille le métamodèle WSDL, en mettant en valeur notre annotation contextuelle, encadrée en pointillés.

Nous annotons les éléments `<part>` d'une description WSDL avec un attribut `context` qui décrit les noms et valeurs des modifieurs, en utilisant une liste de noms qualifiés[17]. 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

[17]Un nom qualifié est le nom d'un élément, ou d'un attribut, défini par la concaténation d'un nom local, précédé en option d'un préfixe d'espace de nommage et d'un caractère deux-points. (Glossaire W3C)

permet d'identifier le concept de domaine concerné par l'annotation. Les éléments suivants réfèrent aux instances des modifieurs statiques qui caractérisent l'objet sémantique. Ces éléments sont décrits dans l'ontologie contextuelle associée au concept.

Le listing 5.1 illustre l'extension proposée avec le service Web « Car Rental » de la section 4.2. Le premier élément de notre annotation identifie le concept « Price » 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 la France, que la TVA est incluse dans le prix, et que le facteur multiplicateur utilisé est 1.

```
<?xml version=``1.0" encoding=``UTF-8"?>
<wsdl:definitions ...>...
    <wsdl:message name=``CarRentalTicket">
        <wsdl:part name=``inputPrice" type=``xsd:double"
            ctxt:context=``dom1:Price ctxt1:France
            ctxt1:VATIncluded ctxt1:ScaleFactorOne"/>
    </wsdl:message>...
</wsdl:definitions>
```

Listing 5.1 – Car Rental Annotation Snippet

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.

5.3.4 Intégration du contexte : conclusion

L'utilisation d'ontologies contextuelles et d'annotations de descriptions WSDL présente de nombreux avantages dans le cadre de notre problématique :

- 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.

Dans la section suivante, nous détaillons la solution que nous avons développée pour intégrer notre 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, les annotations contenues dans les descriptions des services, ainsi qu'avec un moteur d'inférence et son entrepôt de règles pour résoudre les hétérogénéités sémantiques présentes dans un processus métier existant.

5.4 Intégration de la médiation dans la composition

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 ; de déployer des médiateurs pour résoudre les hétérogénéités potentielles ; et de modifier le processus métier pour qu'il intègre ces médiateurs. Afin de remplir ces objectifs, nous proposons une méthode de contextualisation de processus métier. Nous appelons contextualisation d'un processus métier 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.

Nous décrivons ces éléments ci-après, et présentons tout d'abord les avantages apportés par une solution orientée service.

5.4.1 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.

Le problème majeur lié à l'utilisation de services Web pour la médiation est l'adaptation de leurs interfaces aux types de données qu'ils reçoivent et envoient. Nous apportons une solution à ce problème en déployant les services Web médiateurs juste avant l'exécution de la composition, et en exploitant l'information contenue dans les descriptions des services originaux. Les étapes de notre solution de médiation sont détaillées ci-dessous.

5.4.2 Étapes de Contextualisation des processus métiers

Pour des raisons de simplicité et d'illustration, nous utilisons l'exemple développé en section 4.2 pour présenter les étapes de contextualisation d'un processus métier. Le processus métier décrit par la figure 5.4 décrit la logique de composition de la figure 4.1. Il a été démontré précédemment que plusieurs hétérogénéités liées au contexte gênent l'exécution correcte de cette composition. Ces hétérogénéités sont dues à l'utilisation de différents formats de représentation pour les dates, heures et monnaies.

Dans le but de réaliser les étapes de contextualisation d'un processus métier, nous considérons que les étapes prérequises pour la mise en place de notre architecture de

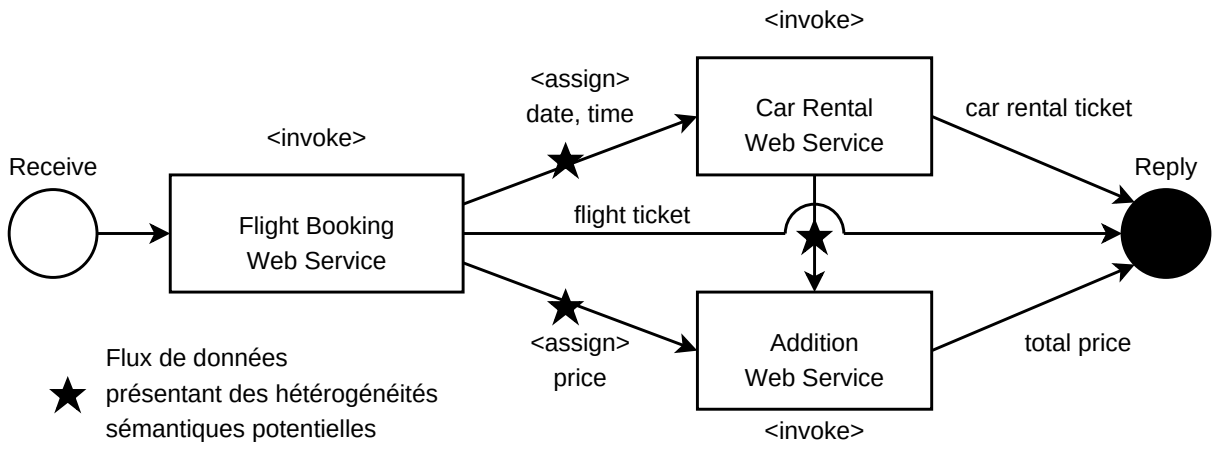

FIG. 5.4 – Représentation du processus métier original

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 (voir figure 5.1). La contextualisation d'un processus métier se constitue des étapes suivantes, résumées dans la figure 5.5 :

Localisation des hétérogénéités potentielles

Un algorithme, présenté en section 5.4.3, analyse le processus métier pour localiser les flux de données explicites et implicites. Dans notre exemple illustré par la figure 5.5, les flux de données intéressants[18], c'est-à-dire ceux qui peuvent présenter des hétérogénéités liées au contexte, sont : a) lorsque les prix sont envoyés au service Web d'addition, et b) lorsque les dates et heures sont envoyées au service Web « Car Rental ».

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), détaillé en section 6.3.

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

[18]Ces flux de données sont indiqués par une étoile.

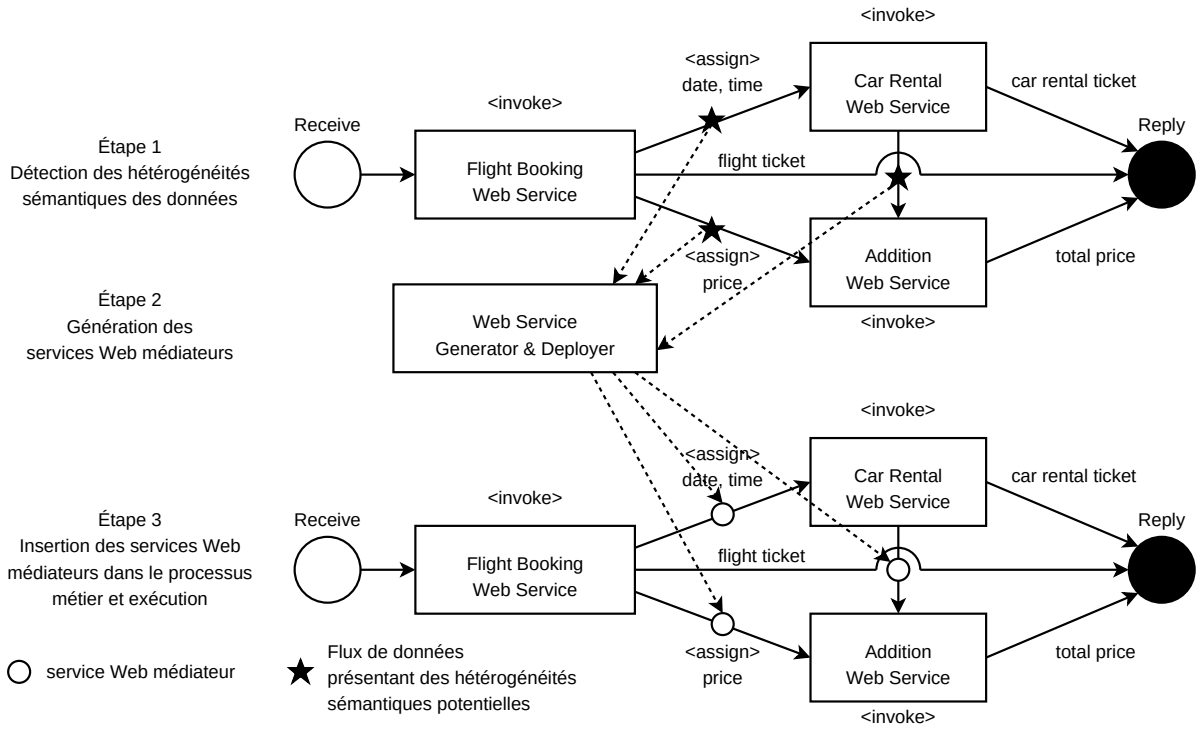

FIG. 5.5 – Étapes de génération du service Web médiateur

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 générés lors de la deuxième étape sont insérées dans le code BPEL original en suivant l'algorithme 5.4.3. 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 5.2. 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. Le fonctionnement des services Web médiateurs est détaillé en section 6.4.

5.4.3 Génération dynamique de processus contextualisés

WS-BPEL a été conçu pour la « programmation à large échelle ». Par conséquent, il ne décrit pas toujours 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.

Premièrement, 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'où les données viennent et où 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 5.2.

```
<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 5.2 – Invocation du médiateur dans WS-BPEL

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>`. Ensuite, 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 5.2, le service Web médiateur est inséré dans la composition BPEL afin d'intercepter et d'adapter le flux de données. La génération du service Web médiateur à partir des descriptions WSDL des services Web source et cible est décrite en détail dans la section 6.3.

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>`.

Dans les deux cas, 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 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 5.2.

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 à 16 montrent la détection des éléments `<invoke>` consécutifs dans une séquence. La fonction $getInvokeChildren(sequence)$ prends 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`

Algorithm 1 Algorithme de Contextualisation BPEL

```
1: for all assign ∈ findElements(< assign >) do
2:    in ← getFromElement(assign)
3:    out ← getToElement(assign)
4:    if in ∈ < variable > ∧ 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(seq)^2) do
11:      if getOutputVar(a) = getInputVar(b) ∧ 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(flow)^2) do
19:      if getOutputVar(a) = getInputVar(b) then
20:         if a.hasChild(< source >) ∧ 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
```

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 (ligne 11), 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.

5.4.4 Intégration de la médiation : conclusion

Dans cette section, nous avons détaillé notre solution pour l'intégration de composants de médiation au sein d'un processus métier. Tout d'abord, nous avons mis en lumière les avantages apportés par l'utilisation de médiateurs développés sous la forme de services Web. Ensuite, nous avons présenté les différentes étapes de contextualisation d'un processus métier, permettant la mise en place des composants nécessaires à la médiation sémantique orientée contexte pour la composition. Ces étapes comprennent la détection des hétérogénéités sémantiques potentielles dans la composition, la génération de services Web médiateurs, et la mise à jour du processus métier décrivant la composition initiale par l'ajout des appels aux services Web médiateurs.

Le développement de notre solution a principalement été confronté au problème du type des données adopté par les interfaces des services. En effet, un médiateur doit être compatible avec les entrées et sorties des services entre lesquels il s'interpose. Pour résoudre ce problème, nous avons développé un générateur de services Web qui déploie les services Web médiateurs et l'interface requise pour assurer la compatibilité avec les ser-

vices existants. Aussi, un algorithme d'analyse de processus métier a été proposé afin d'insérer les services Web médiateurs aux points stratégiques de la composition.

5.5 Conclusion

Le chapitre précédent a introduit notre modèle de description sémantique orientée contexte des entrées/sorties de services Web. Dans ce chapitre, nous avons présenté notre architecture de médiation contextuelle, qui tire profit de ce modèle 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 adopté une approche orientée service. Nous avons proposé une méthode de contextualisation de composition, qui permet s'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.

Les composants réalisant l'implantation de notre architecture ont été développés et testés dans le cas d'application de notre scénario de planification de voyage. Nous décrivons ces composants dans le chapitre suivant, et démontrons la faisabilité de notre approche en illustrant leur fonctionnement dans le cadre de notre exemple de planification de voyage.

Chapitre 6

Réalisation du prototype

6.1 Introduction

La réalisation de notre architecture de médiation a été validée par le développement de trois composants :

- un outil d'annotation de services Web accessible via une interface graphique,
- un générateur de services Web médiateurs développé en collaboration avec l'équipe Vitalab de l'université technique de Vienne,
- et un service Web médiateur générique déployable entre services Web composés.

Ces trois composants ont été développés sous l'environnement de développement Java™, à partir de l'exemple développé en section 4.2, afin de prouver la faisabilité de notre architecture. Dans ce chapitre, nous détaillons le fonctionnement de chaque composant, ainsi que les détails concrets de leur implantation.

6.2 Outil d'annotation de document WSDL

6.2.1 Cas d'utilisation

Notre outil de lecture/écriture de fichiers WSDL permet l'ajout, la modification et la suppression des annotations contextuelles décrites dans la section 5.3.3. Il permet aux fournisseurs de services Web d'annoter des fichiers WSDL avec l'information contextuelle nécessaire pour réaliser la médiation à l'aide de services Web médiateurs.

Deux formats d'annotation sont pris en charge. Le format d'annotation développé dans notre travail est le format par défaut, mais il est aussi possible d'éditer les documents

écrits dans le format WSDL-S [46], une autre annotation de WSDL présentée dans le chapitre 3. Notre outil nécessite l'adresse URL (Unique Resource Location) du fichier WSDL à annoter, ainsi qu'une intervention de l'utilisateur via l'interface graphique pour annoter le document. Il enregistre les modifications de l'utilisateur(trice) dans le document WSDL original. Le fonctionnement du programme que nous avons développé est illustré par le diagramme de cas d'utilisation de la figure 6.1.

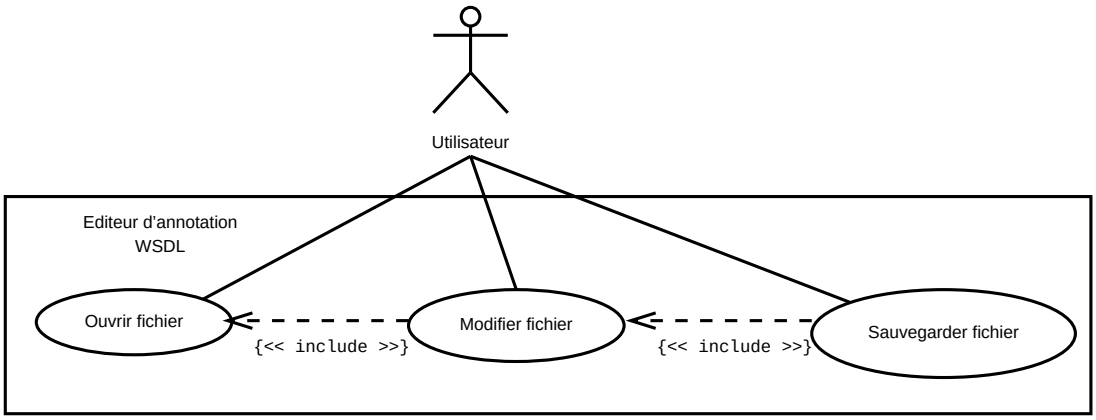

FIG. 6.1 – Diagramme de cas d'utilisation de l'éditeur d'annotation WSDL

6.2.2 Fonctionnement détaillé de l'interface graphique

Notre programme repose sur l'API WSDL4J [81] pour stocker les modèles de document WSDL en mémoire et gérer les éléments d'extensibilité du contexte. Il fonctionne de la manière suivante : l'utilisateur(trice) sélectionne un fichier par le menu « File-Open » de l'application qui ouvre une fenêtre de navigation dans le système de fichiers local. Le chemin complet du fichier sélectionné est ensuite affiché dans le champ correspondant. Ce chemin est éditable manuellement pour faciliter les changements de localisation du fichier. Lorsque le fichier n'est pas trouvé, n'est pas un fichier WSDL ou est mal formaté, le message d'erreur correspondant s'affiche. Lorsque l'utilisateur(trice) clique sur le bouton *ReadFile*, le programme lit le fichier à l'aide de la classe WSDLReader qui génère un modèle en mémoire du document WSDL, accessible via la classe $Definition$ fournie par l'API WSDL4J.

Une fonction parcourt les éléments du modèle pour remplir les champs de l'interface graphique avec les éléments correspondants. Des menus déroulants permettent de choisir entre les différents services, ainsi que les ports (descriptions abstraites), opérations, et parties de messages décrits dans le fichier WSDL. Une case à cocher permet de choisir si l'on souhaite annoter les messages d'entrée ou de sortie. Tous les éléments de la description

WSDL sont pris en charge par des classes spécifiques de l'API WSDL4J.

En ce qui concerne l'annotation, ce sont les éléments d'extensibilité des parties de messages qui sont utilisés pour stocker l'information sémantique de la manière décrite dans la section 5.3.3. Cette annotation est réalisée à l'aide d'une classe *Part*, qui possède une fonction appelée *setExtensionAttribute*. Cette fonction est utilisée pour enregistrer les annotations ajoutées par l'utilisateur(trice) en mémoire. Lorsque cela est nécessaire, des espaces de nommage sont ajoutés dans les déclarations du document WSDL afin d'identifier les termes utilisés par les annotations. Cet ajout est pris en charge par la classe *Definition* qui possède une fonction *addNamespace* permettant d'ajouter des espaces de nommage au document. Deux boutons de sauvegarde *Change* et *Savefile* assurent respectivement la sauvegarde en mémoire de la modification en cours et l'écriture du modèle en mémoire dans le fichier WSDL.

La figure 6.2 montre une capture d'écran de l'interface utilisateur de notre programme. Elle présente l'édition du fichier « HotelBooking.wsdl ». Le contexte du paramètre de sortie de l'opération *HotelBooking* est ajouté à la description. L'annotation indique que le paramètre de sortie *estimatePriceReturn* fait référence au concept *price* de l'ontologie de domaine localisée à *http* : *//domain.onto*, et indique le modifieur *Euro* qui appartient à l'ontologie contextuelle localisée à *http* : *//context.onto*. Dans cette ontologie contextuelle, *Euro* est décrit comme une instance du modifieur *currency* qui indique la devise d'un prix.

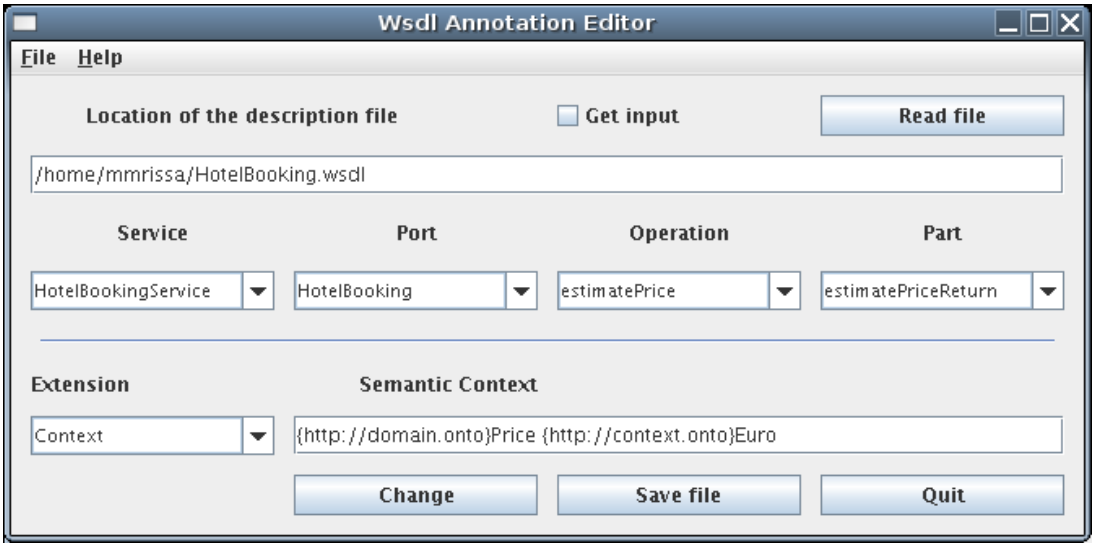

Fig. 6.2 – Capture d'écran de l'éditeur d'extension WSDL

6.3 Génération de services Web médiateurs

Durant la contextualisation du processus métier WS-BPEL original, un ou plusieurs services Web médiateurs doivent être générés. En conséquence, nous avons développé un générateur de code flexible et réutilisable reposant sur un modèle objet indépendant de la plateforme d'accueil. À partir de ce modèle indépendant, nous avons réalisé un modèle spécifique à la plateforme Java™.

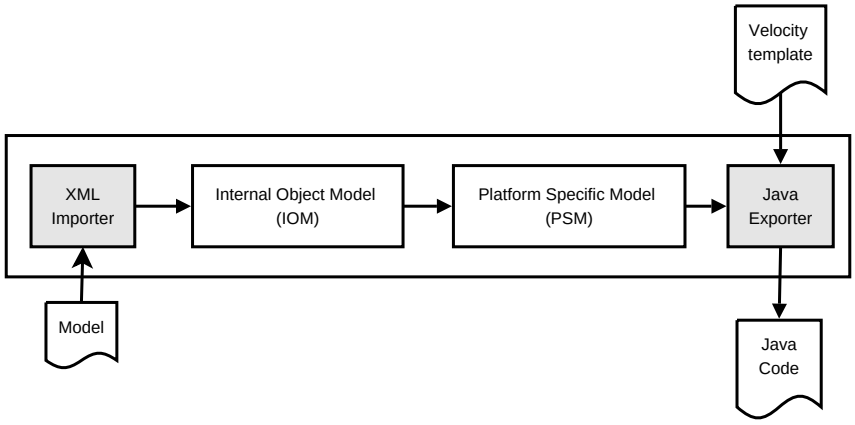

Fig. 6.3 – Présentation du générateur de services Web médiateurs

Les éléments majeurs du générateur de code sont décrits figure 6.3. Un modèle de description spécifié en XML est utilisé en entrée du générateur de code. Le fichier d'entrée est parsé par un composant `XMLImporter`, qui construit le modèle objet indépendant (Internal Object Model IOM) correspondant à la description XML du modèle. Le modèle objet indépendant IOM est ensuite transformé en un modèle spécifique à la plateforme (Platform Specific Model PSM), dans notre cas en utilisant la plateforme Java™. Le composant `JavaExporter` opère directement sur le modèle spécifique à la plateforme et parcourt chaque classe et interface afin de générer le code Java™. La génération du code est supportée par le moteur de modèle Apache Velocity [3]. À partir de ce générateur de code à base de modèle, nous avons développé un composant spécial appelé `Axis2CodeGenerator` afin de générer un service pour l'environnement d'exécution Axis 2 [2]. Ce composant effectue les étapes suivantes :

1. Génération et compilation dynamique du code :

   Le générateur de code mentionné ci-dessus est utilisé pour générer et compiler dynamiquement une classe Java™qui sera déployée comme un service Web. Toutes les librairies nécessaires à la compilation du code seront ajoutées dynamiquement.

2. Génération du descripteur de déploiement :

Axis 2 nécessite un descripteur de déploiement spécial appelé `services.xml` qui spécifie la classe principale réalisant la logique du service Web. De plus, chaque opération de la classe qui sera exposée en tant qu'opération de service Web doit être spécifiée. Axis2 implante des services Web purement « document-style » en exploitant une approche top-down (ou « WSDL-first »). Dû au fait que nous générons l'implantation du service directement, nous suivons une approche « bottom-up », qui est généralement utilisée pour les services Web « RPC-style ». En conséquence, nous utilisons un gestionnaire de message spécifique, appelé `RPCMessageReceiver` qui est spécifié dans le descripteur de déploiement. Ce dernier est responsable de la conversion du style document du service Web tel que requis par Axis2 vers la structure RPC utilisée en interne, en exposant une classe Java™en tant que service Web.

3. Paquetage et déploiement du code :

   Le code compilé avec le descripteur de déploiement et les librairies requises sont empaquetés ensemble dans un fichier .aar (Archive Axis2). Le nouveau modèle de déploiement d'Axis2 permet une étape de déploiement très simple. Le fichier archive créé dans l'étape précédente est simplement copié dans le répertoire de déploiement d'Axis2, spécifié dans les propriétés de notre système. Le déploiement lui-même est géré par le moteur d'exécution d'Axis2.

6.4 Fonctionnement des services Web médiateurs

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. Examinons les étapes de médiation effectuées par les services Web médiateurs avec l'exemple présenté section 4.2. Nous considérons le flux de données entre le service« Car Rental » 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 « price » envoyée par le service Web « Car Rental ». Ensuite, il effectue les étapes décrites par la figure 6.4 avant d'envoyer le résultat de ses calculs dans une partie de message « price », au service Web « Addition ».

Dans la première étape, les fichiers WSDL des services Web « Car Rental » et « Addition » sont téléchargés par le service Web médiateur. Ces fichiers sont analysés 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,

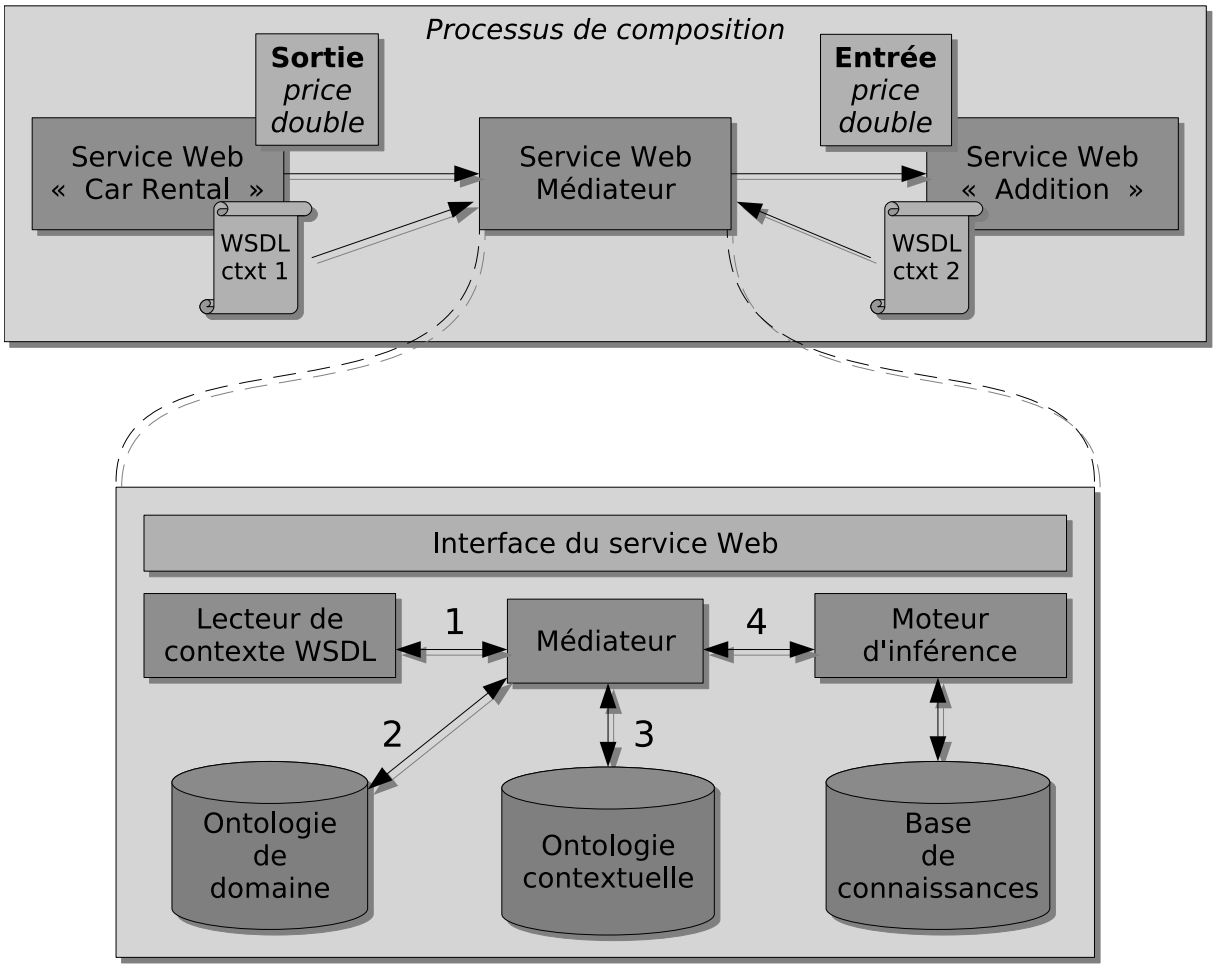

FIG. 6.4 – Vue détaillée du service Web médiateur

les parties de message nommées *price* de sortie et d'entrée des services Web « Car Rental » 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. Un tel exemple d'annotation a été présenté dans le listing 5.1 de la section 5.3.3 avec le service Web « Car Rental ». Le médiateur utilise l'API WSDL4J afin de générer un modèle du document WSDL en mémoire et d'identifier les éléments annotés. Nous avons créé un objet *ContextReader* qui identifie et extrait les annotations nécessaires du document.

Dans la deuxième étape, le service Web médiateur identifie les 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 *price* de l'ontologie de domaine identifiée par le namespace *dom*1 du fichier WSDL. Le service Web médiateur vérifie que

les concepts utilisés par les services Web « Car Rental » 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[19]. Actuellement, nous utilisons la bibliothèque Jena™ pour identifier les termes dans les ontologies et établir les correspondances.

Dans la troisième étape, pour chaque service, une représentation en mémoire sous forme de structure arborescente du contexte de l'objet sémantique *price* est construite à partir de l'ontologie contextuelle. Les annotations contextuelles du concept *price* 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. Cette étape est aussi effectuée à l'aide de la bibliothèque Jena™ qui permet de manipuler des représentations OWL.

Dans le listing 5.1, l'attribut *ctxt1 :France* est une instance du concept *country* de l'ontologie contextuelle *ctxt*1. Aussi, l'attribut *ctxt1 :ScaleFactorOne* est une instance du concept *scaleFactor* de l'ontologie contextuelle. 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.

Dans la quatrième étape, 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 $France$ donnée par la description WSDL, le moteur d'inférence en déduit que le modifieur dynamique *currency* du service possède la valeur $Euro$ en questionnant sa base de connaissances, qui contient une règle associant la devise $Euro$ au pays $France$. Ainsi, l'attribut contextuel *currency* se voit affecté la valeur $Euro$. 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. À partir des étapes précédentes, nous obtenons deux arbres de représentation du contexte en mémoire, qui ont des feuilles. Ces feuilles sont

[19]Pour un bon état de l'art des techniques d'intégration sémantiques, voir les travaux de Noy [53].

des modifieurs contenant des valeurs, formant ainsi le contexte, comme présenté par la figure 6.5.

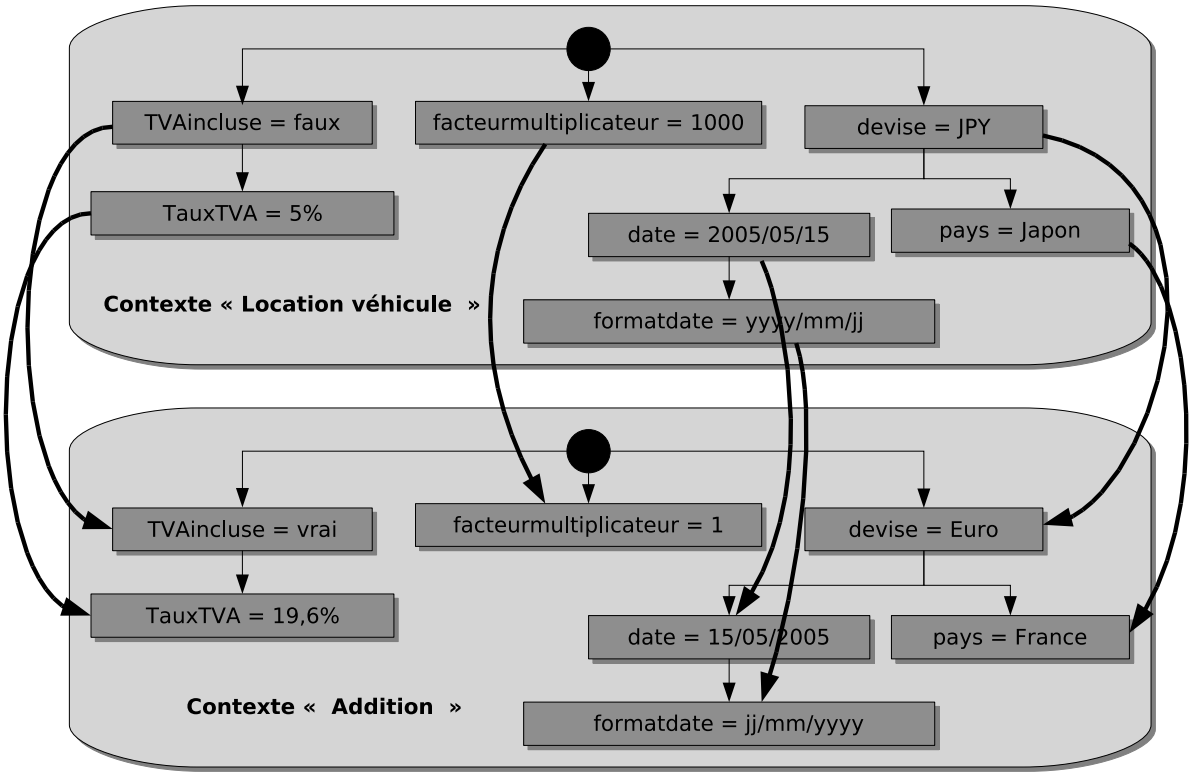

FIG. 6.5 – Conversion des éléments du contexte des services « Car Rental » et « Addition »

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 4.5.4. 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 un telle règle n'existe pas, la conversion est annulée et une exception est levée.

Considérons notre exemple de la section 4.2, avec les valeurs des prix V et leurs facteurs multiplicateurs SF. La règle logique permettant de gérer le modifieur $scalefactor$ est stockée 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 et *target* est le contexte vers lequel les données doivent être converties. Ainsi, durant l'exécution de la composition, le moteur d'inférence reçoit en entrée : $scalefactor$, 1000, 1 et la valeur V_{source} qui est convertie dans le facteur multiplicateur approprié. Si un facteur multiplicateur manque, une règle logique dans la base de connaissances suppose un facteur multiplicateur de 1 et la conversion est encore possible. La base de connaissances contient aussi les règles de conversion qui permettent une conversion dynamique entre des valeurs de modifieurs. Dans notre exemple, le prix est converti dans la devise appropriée en appelant un composant distant qui fournit des taux de change actuels. Une telle conversion doit être dynamique, afin de répondre aux exigences de la perspective temporelle du contexte. Les différentes étapes de fonctionnement du médiateur sont illustrées ci-dessous par le listing de sortie du programme (version simplifiée).

```
run:
Input value = 105.0
Retrieving document at '/home/mmrissa/dev/files/HotelBooking2.wsdl'.
Retrieving document at '/home/mmrissa/dev/files/EuroBanking2.wsdl'.
estimatePriceReturn(double) matches price(double)

Compare parts.
Semantic types {http://some.example.com}Price match.
Building context structures...
http://...PriceContext.owl#dateFormat
http://...PriceContext.owl#isIncluded
http://...PriceContext.owl#taxRate
http://...PriceContext.owl#currencyCode
http://...PriceContext.owl#countryName
http://...PriceContext.owl#scaleFactor
Country = France
Country = Japan
Converting http://...PriceContext.owl#VATIncluded modifier
from false to true
Converting http://...PriceContext.owl#currency modifier
from JPY to EUR
Converting http://...PriceContext.owl#scaleFactor modifier
from 1000 to 1
Converting http://...PriceContext.owl#countryName modifier
from Japan to France
Converting http://...PriceContext.owl#TaxRate modifier
```

```
from 9.3 to 14.7
Converting http://...PriceContext.owl#dateFormat modifier
from yyyy.mm.dd to dd.mm.yyyy

Converted value = 854.0944195051381
BUILD SUCCESSFUL (total time: 5 seconds)
```

6.5 Déploiement du prototype

Nous avons conduit notre expérimentation à partir du scénario présenté tout au long de ce document. Notre exemple de composition a été déployé dans un serveur Apache Tomcat[20]. Notre service Web médiateur utilise la bibliothèque Jena 2[21] et le moteur d'inférence Drools[22] pour accéder aux ontologies contextuelles et de domaine et pour effectuer les conversions de données. Notre prototype inclut des ontologies contextuelles et de domaine simplifiées qui décrivent les concepts et contextes requis pour un bon fonctionnement de l'exemple[23]. Ces ontologies ont été conçues avec l'outil Protégé™.

Nous avons déployé tous les services Web dans un serveur Apache Axis [4], et ils ont été composés dans un processus WS-BPEL, hébergé par un moteur BPWS4J[24].

Notre prototype effectue la conversion des données durant l'exécution de la composition, lui permettant d'aboutir à un résultat sémantiquement correct. Dans notre exemple, les concepts de prix sont identifiés dans l'ontologie de domaine, et leurs contextes sont adaptés durant l'exécution : les différents facteurs multiplicateurs, formats de dates, taux de TVA inclus ou non sont adaptés au fil de la composition.

Nous envisageons des évaluations de performance et des tests additionnels permettant d'observer plus précisément les limites de notre approche. Cependant, de tels tests requièrent de nouvelles ontologies de domaine et contextuelles validées par des experts. Dans le cadre de notre travail, nous avons limité notre expérimentation à l'exemple illustratif, comme preuve de la faisabilité de notre approche. Notre travail en cours concerne aussi l'intégration de notre algorithme de contextualisation avec une des implantations de la spécification WS-BPEL telles que ActiveBPEL™ ou Apache Ode™.

[20] `http://tomcat.apache.org/`
[21] `http://jena.sourceforge.net/`
[22] `http://www.drools.org/`
[23] Disponible sur `http://www710.univ-lyon1.fr/~mmrissa/`
[24] `http://www.alphaworks.ibm.com/tech/bpws4j/`

6.6 Conclusion

Dans ce chapitre, nous avons présenté les divers composants qui forment l'implantation de notre approche de médiation sémantique orientée contexte. Nous avons mis en valeur la faisabilité de notre approche et son applicabilité en l'illustrant avec notre scénario de planification de voyage en ligne.

Nous avons fourni, via notre outil d'annotation de document WSDL doté d'une interface graphique, les mécanismes d'annotations qui permettent d'enrichir les descriptions WSDL avec l'information sémantique nécessaire. De plus, nous avons observé le fonctionnement des mécanismes de génération automatique de services Web développés en collaboration avec l'équipe de recherche Vitalab de l'université technique de Vienne. Enfin, nous avons examiné en détail le fonctionnement interne des services Web médiateurs, et montré les avantages apportés par notre architecture, notamment grâce à l'utilisation des règles stockées dans une base de connaissances, et d'ontologies de domaine et contextuelles développées pour les besoins de notre application.

Chapitre 7

Conclusion générale

7.1 Résumé de la contribution

Avec le développement rapide des technologies de l'information, la recherche de l'interopérabilité est de nos jours une problématique centrale des systèmes d'information distribués. Ce domaine de recherche est favorisé par l'adoption de l'architecture orientée service comme modèle de développement, et particulièrement par les services Web qui combinent les avantages de ce modèle aux langages et technologies développés pour Internet. Les services Web ont permis une avancée significative dans l'automatisation des interactions entre systèmes distribués. Notamment, la composition de services Web est considérée comme un point fort, qui permet de répondre à des requêtes complexes en combinant les fonctionnalités de plusieurs services au sein d'une même composition.

Cependant, afin de pallier le manque des langages et protocoles actuels mis en place par la communauté informatique, nous avons vu que les travaux liés à l'interopérabilité dans le cadre de la composition de services Web sont particulièrement orientés vers le niveau sémantique. L'objectif recherché à travers l'utilisation de la sémantique est de permettre aux machines d'interpréter les données traitées et de saisir leur signification de manière automatique. Cet objectif est concrètement atteint par le déploiement d'ontologies de domaine, qui sont des descriptions explicites et partagées de la sémantique associée aux données. Les ontologies servent de référence aux descriptions de services Web, facilitant ainsi l'adaptation des données entre ces derniers.

De nombreux langages et annotations ont été proposés pour la description sémantique. Parallèlement, de nombreuses propositions de médiation ont été construites à partir d'approches de description sémantique. En effet, les perspectives apportées par l'utilisation de la sémantique sont très prometteuses. Cette dernière permet d'automatiser les tâches

de sélection, d'orchestration et de coordination des services Web dans une composition, tâches qui sont toujours effectuées manuellement à l'heure actuelle.

Néanmoins, les services sont sujets à de nombreuses hétérogénéités, d'autant plus que dans le contexte dynamique d'Internet, il est difficilement envisageable d'imposer une représentation sémantique unique et partagée par tous les services Web. La sémantique locale adoptée par chaque service doit être prise en charge, et les différences de représentation des données doivent être résolues par l'utilisation de mécanismes de médiation qui doivent être implantés au sein de la composition.

C'est dans le but de répondre à ces problématiques que nous avons mené nos recherches. Nos travaux sont orientés vers la sémantique, et plus particulièrement vers une proposition de médiation sémantique orientée contexte pour les services Web. Nous avons étudié les travaux existants relatifs à la médiation et à la description sémantique de services Web, afin d'établir notre proposition.

Notre contribution repose sur l'utilisation du contexte. Tout d'abord, nous avons mis en évidence les avantages que le contexte peut apporter à la description sémantique de services Web. En effet, l'utilisation du contexte permet de rendre explicite la sémantique locale utilisée par chaque service Web et de l'intégrer dans un cadre permettant la médiation d'un contexte à l'autre, grâce à l'utilisation de règles de conversion. Les contextes des services Web sont découverts par le biais de leurs descriptions sémantiques, et des règles logiques sont utilisées par un composant de médiation spécifique, le médiateur de contexte, pour effectuer les conversions entre les différents contextes mis en relation dans une composition.

Ainsi, notre travail de recherche a abouti à plusieurs propositions, qui forment une architecture de médiation pour services Web composés. Cette architecture repose sur un modèle de description orienté contexte, une annotation du langage de description des services Web WSDL, et un mécanisme de médiation orienté service reposant sur des règles logiques, décrits ci-dessous :

- Le modèle que nous avons développé est construit autour de la notion d'objet sémantique qui a été introduit dans le domaine des bases de données, et propose une adaptation de cette notion aux besoins de la technologie des services Web. Notre travail repose sur une classification des hétérogénéités sémantiques liées au contexte des services Web, qui nous a servi de référence pour proposer notre modèle, ainsi que sur les propositions existantes dans ce domaine.
- Notre choix d'annoter le langage WSDL est motivé par la facilité d'adaptation des services Web existants qu'apporte une telle approche, mais aussi par les éléments d'extensibilité fournis par la spécification WSDL, qui permettent aux documents

annotés de rester compatibles avec le format de description standard. Nous proposons pour cette étape un outil permettant d'assister les utilisateurs(trices) dans ce processus d'enrichissement sémantique des descriptions WSDL.

– Enfin, notre architecture de médiation permet de résoudre en partie les problèmes de composition de service, en automatisant la résolution des hétérogénéités sémantiques sans intervention manuelle autre que l'annotation sémantique des descriptions de services Web et la mise à jour d'ontologies contextuelles. Cette architecture intègre le médiateur de contexte comme un service Web, qui participe à la composition et adapte le flux de données entre services Web composés en exploitant les éléments du contexte présents dans les fichiers de description annotés.

7.2 Perspectives envisagées

La réalisation de notre architecture de médiation nous a permis de dégager plusieurs perspectives de travail. Nous avons démontré la faisabilité de notre proposition en l'illustrant à l'aide d'un exemple de planification de voyage en ligne, cependant une étude des différents domaines d'application possibles montrerai plus justement sa généricité. Cette étude nécessite le développement des ontologies de domaine et contextuelles nécessaires pour effectuer la médiation entre services Web.

La mise à jour des ontologies contextuelles est actuellement effectuée de manière manuelle : les propriétés sémantiques et leurs relations sont ajoutées par les fournisseurs de services lors de l'adhésion à l'ontologie de domaine. Des interfaces graphiques reposant sur des outils de raisonnement pourraient être utilisées pour assister la mise à jour des ontologies contextuelles. Les relations entre les propriétés sémantiques présentes dans les ontologies pourraient être calculées grâce à des algorithmes de mesure de similarité sémantique, et proposées aux fournisseurs.

Notre modèle de description orienté contexte peut être étendu pour intégrer une perspective temporelle, comme cela a été fait pour les modèles développés dans le domaine des bases de données. L'ajout d'une dimension temporelle permettrait d'évaluer la validité d'une description de service au moment de l'exécution, et de mettre à jour une sémantique obsolète en identifiant les conversions à appliquer selon son évolution dans le temps.

Une autre perspective de travail est l'exploitation de l'information contextuelle durant la sélection de services. Un filtrage sémantique permettrait d'éliminer durant l'étape de conception de la composition des services sémantiquement incompatibles avec ceux qui ont été déjà sélectionnés.

Enfin, la spécialisation de notre service Web médiateur en plusieurs services, chacun

dédié à un domaine de connaissances particulier, permettrait d'améliorer la gestion des capacités de conversion par la séparation des spécialisations des médiateurs, ainsi que l'exécution des compositions par la sélection de médiateurs adaptés à leurs besoins.

7.3 Conclusion

Ce travail est le résultat de notre réflexion sur les limites actuelles des solutions de médiation et de description sémantique pour la composition de services Web. Nous avons proposé une approche de médiation construite sur notre analyse des nombreuses propositions existantes. 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 orienté service.

Nous apportons un compromis, 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. De plus, nous facilitons le travail des fournisseurs en proposant une annotation des descriptions de services qui reste compatible avec le format de description WSDL.

La notion de contexte, autour de laquelle nous avons construit notre proposition 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 concernés par l'interopérabilité des données.

Bibliographie

[1] 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.

[2] Apache Software Foundation. Axis2. `http://ws.apache.org/axis2/` (last accessed : 29 Mai 2006).

[3] Apache Software Foundation. Velocity – Java-based template engine. `http://jakarta.apache.org/velocity/` (last accessed : 29 Mai 2006).

[4] Apache Software Foundation. Apache axis, 2003. `http://ws.apache.org/axis`.

[5] A. Arkin, S. Askary, S. Fordin, W. Jekeli, K. Kawaguchi, D. Orchard, S. Pogliani, K. Riemer, S. Struble, P. Takacsi-Nagy, I. Trickovic, and S. Zimek. Web service choreography interface (wsci) 1.0. W3C Note, August 2002. `http://www.w3.org/TR/wsci`.

[6] S. Arroyo and M. Stollberg. WSMO Primer. WSMO Deliverable D3.1, DERI Working Draft. Technical report, WSMO, 2004. `http://www.wsmo.org/2004/d3/d3.1/`.

[7] G. Athanasopoulos, A. Tsalgatidou, and M. Pantazoglou. Interoperability among heterogeneous services. In Society [69], pages 174–181.

[8] L. Aversano, G. Canfora, and A. Ciampi. An algorithm for web service discovery through their composition. In *ICWS '04 : Proceedings of the IEEE International Conference on Web Services (ICWS'04)*, page 332, Washington, DC, USA, 2004. IEEE Computer Society.

[9] 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.

[10] B. Benatallah, M.-S. Hacid, A. Léger, C. Rey, and F. Toumani. On automating web services discovery. *VLDB J.*, 14(1) :84–96, 2005.

[11] B. Benatallah, M.-S. Hacid, C. Rey, and F. Toumani. Semantic Reasoning for Web

Services Discovery. In *WWW2003 Workshop on E-Services and the Semantic Web, Budapest, Hungary*, May 2003.

[12] B. Benatallah, Q. Z. Sheng, and M. Dumas. The self-serv environment for web services composition. *IEEE Internet Computing*, 7(1) :40–48, 2003.

[13] D. Berardi, D. Calvanese, G. D. Giacomo, M. Lenzerini, and M. Mecella. A foundational vision of e-services. In C. Bussler, D. Fensel, M. E. Orlowska, and J. Yang, editors, *WES*, volume 3095 of *Lecture Notes in Computer Science*, pages 28–40. Springer, 2003.

[14] A. Bernstein and M. Klein. Discovering services : Towards high-precision service retrieval. In C. Bussler, R. Hull, S. A. McIlraith, M. E. Orlowska, B. Pernici, and J. Yang, editors, *WES*, volume 2512 of *Lecture Notes in Computer Science*, pages 260–276. Springer, 2002.

[15] V. Bicer, O. Kilic, A. Dogac, and G. B. Laleci. Archetype-based semantic interoperability of web service messages in the health care domain. *International Journal of Semantic Web and Information Systems (IJSWIS)*, 1(4) :1–23, October 2005.

[16] C. Bornhövd. Mix - a representation model for the integration of web-based data, tech. rep. dvs98-1. Technical report, DVS1, Dep. CS, Darmstadt University of Technology, Germany, Nov. 1998.

[17] C. Bornhövd. 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.

[18] S. Bowers and B. Ludäscher. An ontology-driven framework for data transformation in scientific workflows. In E. Rahm, editor, *DILS*, volume 2994 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 2004.

[19] D. Box, D. Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn, H. F. Nielsen, S. Thatte, and D. Winer. Simple object access protocol (SOAP) 1.1. Technical report, The World Wide Web Consortium (W3C), 2000. `http://www.w3.org/TR/SOAP/`.

[20] T. Bray, J. Paoli, and C. M. Sperberg-McQueen. Extensible markup language (xml). *World Wide Web Journal*, 2(4) :27–66, 1997.

[21] D. Brickley and R. Guha. Resource description framework (rdf) schema specification 1.0. Candidate recommendation, March 2000. `http://www.w3.org/TR/2000/CR-rdf-schema-20000327`.

[22] C. Bussler. Semantic web services : Reflections on web service mediation and composition. In *WISE*, pages 253–260. IEEE Computer Society, 2003.

[23] 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.*, December 12th 2005.

[24] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana. Web Services Description Language (WSDL) 1.1. W3c note, The World Wide Web Consortium (W3C), March 2001. `http://www.w3.org/TR/wsdl`.

[25] Ó. Corcho, A. Gómez-Pérez, M. Fernández-López, and M. Lama. ODE-SWS : A Semantic Web Service Development Environment. In I. F. Cruz, V. Kashyap, S. Decker, and R. Eckstein, editors, *SWDB*, pages 203–216, 2003.

[26] F. Curbera, Y. Goland, and J. K. et al. Business Process Execution Language for Web Services (BPEL4WS) Version 1.1, May 2003.

[27] A. K. Dey. Understanding and using context. *Personal and Ubiquitous Computing*, 5(1) :4–7, 2001.

[28] A. K. Dey, G. D. Abowd, and D. Salber. A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications. *Human-Computer Interaction Journal, Special Issue on Context-Aware Computing*, 16(1), 2001.

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

[30] M. Dumas, M. Spork, and K. W. 0002. Adapt or perish : 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.

[31] S. Dustdar and W. Schreiner. A Survey on Web services Composition. *International Journal of Web and Grid Services (IJWGS)*, 1(1) :1–30, 2005.

[32] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext transfer protocol – HTTP/1.1, 1999.

[33] A. Firat. *Information Integration Using Contextual Knowledge and Ontology Merging.* PhD thesis, Massachusetts Institute of Technology, Sloan School of Management, 2003.

[34] 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.

[35] T. Gruber. What is an ontology ? `http://www-ksl.stanford.edu/kst/what-is-an-ontology.html`, 2000.

[36] A. Haller, E. Cimpian, A. Mocan, E. Oren, and C. Bussler. Wsmx - a semantic service-oriented architecture. In I. C. Society, editor, *ICWS*, pages 321–328. IEEE Computer Society, 2005.

[37] V. Kashyap and A. P. Sheth. Semantic and schematic similarities between database objects : A context-based approach. *VLDB J.*, 5(4) :276–304, 1996.

[38] N. Kavantzas, D. Burdett, G. Ritzinger, T. Fletcher, Y. Lafon, and C. Barreto. Web services choreography description language version 1.0. W3C Candidate Recommendation, November 2005. `http://www.w3.org/TR/ws-cdl-10/`.

[39] M. Klein, B. König-Ries, and M. Müssig. What is needed for semantic service descriptions - a proposal for suitable language constructs. *International Journal on Web and Grid Services (IJWGS)*, 1(3/4) :328–364, 2005.

[40] O. Lassila and R. R. Swick. Resource description framework (rdf) : Model and syntax specification. Recommendation, World Wide Web Consortium, February 1999. `http://www.w3.org/TR/REC-rdf-syntax/`.

[41] F. Leymann. Web services flow language (wsfl 1.0), May 2001.

[42] B. Lin, N. Gu, and Q. Li. A requester-based mediation framework for dynamic invocation of web services. In Society [69], pages 445–454.

[43] D. Martin, M. Burstein, O. Lassila, M. Paolucci, T. Payne, and S. McIlraith. Describing Web Services using OWL-S and WSDL. Technical report, DAML-S Coalition, October 2003. `http://www.daml.org/services/owl-s/1.0/owl-s-wsdl.html`.

[44] D. L. Martin, M. Paolucci, S. A. McIlraith, M. H. Burstein, D. V. McDermott, D. L. McGuinness, B. Parsia, T. R. Payne, M. Sabou, M. Solanki, N. Srinivasan, and K. P. Sycara. Bringing Semantics to Web Services : The OWL-S Approach. In J. Cardoso and A. P. Sheth, editors, *SWSWPC*, volume 3387 of *Lecture Notes in Computer Science*, pages 26–42. Springer, 2004.

[45] B. Medjahed, B. Benatallah, A. Bouguettaya, and A. K. Elmagarmid. Webbis : An infrastructure for agile integration of web services. *Int. J. Cooperative Inf. Syst.*, 13(2) :121–158, 2004.

[46] J. Miller, K. Verma, P. Rajasekaran, A. Sheth, R. Aggarwal, and K. Sivashanmugam. WSDL-S : Adding Semantics to WSDL - White Paper. Technical report, Large Scale Distributed Information Systems, 2004. `http://lsdis.cs.uga.edu/library/download/wsdl-s.pdf`.

[47] 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 Society.

[48] M. Mrissa, C. Ghedira, D. Benslimane, and Z. Maamar. A context model for semantic mediation in web services composition. In D. W. Embley, A. Olivé, and S. Ram, editors, *ER*, volume 4215 of *Lecture Notes in Computer Science*, pages 12–25. Springer, 2006.

[49] M. Mrissa, C. Ghedira, D. Benslimane, and Z. Maamar. Towards Context-based Mediation for Semantic Web Services Composition. In *Proceedings of the Eighteenth International Conference on Software Engineering and Knowledge Engineering (SEKE'2006)*, San Francisco, California, July 2006.

[50] M. Mrissa, C. Ghedira, D. Benslimane, Z. Maamar, and J. Fayolle. A mediation framework for web services in a peer-to-peer 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 Society.

[51] M. Nagarajan, K. Verma, A. P. Sheth, J. Miller, and J. Lathem. Semantic interoperability of web services - challenges and experiences. In *ICWS*, pages 373–382. IEEE Computer Society, 2006.

[52] X. T. Nguyen and R. Kowalczyk. An agent based qos conflict mediation framework for web services compositions. In *IAT*, pages 522–528. IEEE Computer Society, 2006.

[53] N. F. Noy. Semantic integration : a survey of ontology-based approaches. *SIGMOD Rec.*, 33(4) :65–70, 2004.

[54] N. F. Noy and C. D. Hafner. The state of the art in ontology design : A survey and comparative review. In *AI Magazine*, volume 18, pages 53–74, 1997. `http://aaai.org/Library/Magazine/Vol18/18-03/Papers/AIMag18-03-006.pdf`.

[55] N. F. Noy and D. Mcguinness. Ontology development 101 : A guide to creating your first ontology. *Stanford KSL Technical Report KSL-01-05*, 2000. `http://www.ksl.stanford.edu/people/dlm/papers/ontology101/ontology101-noy-mcguinness.html`.

[56] Object Management Group (OMG). Bpmn 1.0 : Omg final adopted specification. `http://www.bpmn.org/`, February 2006. `http://www.bpmn.org/`.

[57] D. Orchard. Versioning xml vocabularies, 2003.

[58] M. Paolucci, T. Kawamura, T. Payne, and K. Sycara. Importing the Semantic Web in UDDI. In *Proceedings of E-Services and the Semantic Web Workshop*, 2002.

[59] M. Paolucci, T. Kawamura, T. R. Payne, and K. P. Sycara. Semantic matching of web services capabilities. In I. Horrocks and J. A. Hendler, editors, *International*

Semantic Web Conference, volume 2342 of *Lecture Notes in Computer Science*, pages 333–347. Springer, 2002.

[60] 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.

[61] P. F. Pires, M. R. F. Benevides, and M. Mattoso. Mediating heterogeneous web services. In *SAINT*, pages 344–347. IEEE Computer Society, 2003.

[62] S. Ponnekanti and A. Fox. Interoperability among independently evolving web services. In H.-A. Jacobsen, editor, *Middleware*, volume 3231 of *Lecture Notes in Computer Science*, pages 331–351. Springer, 2004.

[63] U. Radetzki and A. B. Cremers. Iris : A framework for mediator-based composition of service-oriented software. In *ICWS*, pages 752–755. IEEE Computer Society, 2004.

[64] S. Ran. A model for web services discovery with qos. *SIGecom Exch.*, 4(1) :1–10, 2003.

[65] 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/`.

[66] G. Schreiber and M. Dean. Owl web ontology language reference. `http://www.w3.org/TR/2004/REC-owl-ref-20040210/`, February 2004.

[67] E. Sciore, M. Siegel, and A. Rosenthal. Using semantic values to facilitate interoperability among heterogeneous information systems. *ACM Trans. Database Syst.*, 19(2) :254–290, 1994.

[68] J. R. Searle. *Speech Acts : An Essay in the Philosophy of Language*. Cambridge University Press, Cambridge, 1969.

[69] I. C. Society, editor. *2006 IEEE International Conference on Services Computing (SCC 2006), 18-22 September 2006, Chicago, Illinois, USA*. IEEE Computer Society, 2006.

[70] B. Spencer and S. Liu. Inferring data transformation rules to integrate semantic web services. In S. A. McIlraith, D. Plexousakis, and F. van Harmelen, editors, *Int'l Semantic Web Conference*, volume 3298 of *Lecture Notes in Computer Science*, pages 456–470. Springer, 2004.

[71] Y. Taher, D. Benslimane, M.-C. Fauvet, and Z. Maamar. Towards an approach for web services substitution. In P. Ghodous, R. Dieng-Kuntz, and G. Loureiro, editors, *IDEAS*, pages 166–173. IOS Press, 2006.

[72] S. Thatte. Xlang : Web services for business process design, 2001. http://www.gotdotnet.com/team/xml_wsspecs/xlang-c/default.htm.

[73] A. Tolk and S. Y. Diallo. Model-based Data Engineering for Web Services. *IEEE Internet Computing*, 9(4), July/August 2005.

[74] UDDI Specification Technical Commitee. Universal Description, Discovery, and Integration of Business for the Web. Technical report, October 2001. http://www.uddi.org.

[75] W. M. P. van der Aalst and A. H. M. ter Hofstede. Yawl : yet another workflow language. *Inf. Syst.*, 30(4) :245–275, 2005.

[76] W3C Working Group. W3c working group note, February 2004. http://www.w3.org/TR/ws-gloss/.

[77] W3C Working Group. XML Schema Part 2 : Datatypes Second Edition. Technical report, W3C, October 2004. http://www.w3.org/TR/xmlschema-2/.

[78] G. Wiederhold. Mediators in the architecture of future information systems. *IEEE Computer*, 25(3) :38–49, 1992.

[79] S. K. Williams, S. A. Battle, and J. E. Cuadrado. Protocol mediation for adaptation in semantic web services. Technical report, Hewlett-Packard, May 2005. http://www.hpl.hp.com/techreports/2005/HPL-2005-78.html.

[80] Workflow Management Coalition (WFMC). Workflow standard : Workflow process definition interface - xml process definition language (xpdl). Technical report (WFMC-TC-1025), 2002. Workflow Management Coalition, Lighthouse Point, Florida, USA.

[81] WSDL4J Project Homepage. Web Services Description Language for Java Toolkit (WSDL4J) v. 1.5, 2005. http://sourceforge.net/projects/wsdl4j.

[82] H. Zhu, S. E. Madnick, and M. Siegel. Reasoning about temporal context using ontology and abductive constraint logic programming. In H. J. Ohlbach and S. Schaffert, editors, *PPSWR*, volume 3208 of *Lecture Notes in Computer Science*, pages 90–101. Springer, 2004.