République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Ibn Khaldoun - Tiaret

Faculté des Sciences et des Sciences de l'Ingénieur

Département d'Informatique

MEMOIRE

Présenté et soutenu publiquement
en vue de l'obtention du diplôme de

Magister en Informatique

par BOUDAA Boudjemaa

Thème

Les communautés pour une haute disponibilité des services Web maintenant la médiation sémantique dans les compositions

Dans le cadre de l'Ecole Doctorale:
Sciences et Technologies de l'Information et de la Communication (STIC)
Option : Systèmes d'Information et de Connaissance (SIC)

COMPOSITION DU JURY

[Président]	
- **Mr. BESSAID Abdelhafid**	**Maître de Conférences, Université d'Abou Bekr Belkaid, Tlemcen**
[Examinateurs]	
- **Mr. NOUALI Omar**	**Maître de Recherche, CERIST, Alger**
- **Mr. CHADLI Abdelhafid**	**Chargé de Cours, Université Ibn Khaldoun, Tiaret**
[Directeur de Mémoire]	
- **Mr. CHIKH Mohammed Amine**	**Maître de Conférences, Université d'Abou Bekr Belkaid, Tlemcen**

Année Universitaire: 2008 / 2009

Remerciements

Ce travail a été réalisé sous la direction de Monsieur CHIKH Mohammed Amine, Maître de conférences à l'université d'Abou Bekr Belkaid de Tlemcen, auquel j'exprime ma profonde reconnaissance pour la confiance et la liberté d'action qu'il m'a accordées pour la réalisation de ce travail.

Je souhaite adresser mes sincères remerciements aux personnes qui ont accepté la tâche délicate d'examiner ce mémoire et qui ont eu la patience d'évaluer ce travail : Mr. BESSAID Abdelhafid, Maître de conférences au niveau de l'université de Tlemcen, Mr. NOUALI Omar, Maître de recherche au niveau du CEntre de Recherche sur l'Information Scientifique et Technique (CERIST) à Alger et Mr. CHADLI Abdelhafid, chargé de cours à l'université d'Ibn Khaldoun à Tiaret.

Je souhaite également remercier infiniment Monsieur MAAMAR Zakaria, Professeur associé à l'université de Zayed Dubai, Emirats Arabes Unis, pour son assistance très précieuse et sa grande patience à l'égard de mon manque de savoir. Aussi, je tiens à remercier vivement le Doctorant HADJILA Fethallah pour son soutien et ses orientations considérables.

Mes pensées vont aussi à notre Directeur de Travail, en l'occurrence Mr. FARSI Toufik, pour son aide inestimable qui me permit de poursuivre mes études à l'université de Tlemcen. Je tiens à remercier aussi tous mes proches pour leurs encouragements et mes collègues de travail pour les moments partagés, notamment Messieurs BENAMARA Moussa Ex-instituteur de français et MAASKRI Mustapha pour leur contribution a l'élaboration de ce travail.

Enfin, je tiens à remercier toute ma famille, sans qui je ne serai rien et tous ceux, qui de près ou de loin, ont collaboré à l'aboutissement de ce mémoire.

A la mémoire de mes défunts parents, que dieu ait leur âme et les accueille dans son vaste paradis, qui m'ont quitté pendant la période de réalisation de ce travail.

A mon épouse et ma petite chérie Asmaa, auxquelles je demande pardon pour les absences que je leur ai imposées durant cette période. Puisse le temps à venir être plus clément et moins agité.

A tous ceux qui m'aiment et que j'aime très très fort.

I Sommaire

II Table de Figures

Résumé.

La haute disponibilité des services Web engagés dans des scénarios de composition est soutenue maintenant par l'approche basée sur l'utilisation des communautés regroupant des services selon leur fonctionnalité. Dans la composition, cette approche consiste à substituer le service Web échoué par un autre service Web adhérant à une communauté offrant une fonctionnalité équivalente à celle de ce service Web échoué en vue d'assurer sa disponibilité. En revanche, cette substitution peut provoquer une inconsistance sémantique au niveau de la composition et altérer la médiation initialement prise pour résoudre les hétérogénéités sémantiques entre les services Web composants. Dans ce mémoire, nous présentons une solution orientée contexte pour faire face à l'impact sémantique produit par la substitution. Avant de procéder à cette substitution, cette solution consiste à faire adopter la sémantique du service Web échoué, représentée par son contexte, par la communauté qui devra retourner le résultat d'exécution de la fonctionnalité suivant cette sémantique adoptée. L'avantage principal de notre solution est d'éviter, au maximum, la rupture de la médiation sémantique dans la composition en cas de remplacement de ses services Web pour les rendre hautement disponibles.

Mots clés: Service Web, composition, médiation sémantique, contexte, haute disponibilité, communauté, substitution.

Abstract.

The high availability of the Web services engaged in scenarios of composition is now supported by the approach based on the use of the communities gathering the services according to their functionality. In the composition, this approach consists in substituting the failed Web service by another Web service adhering at a community offering an equivalent functionality to that of this failed Web service in order to ensure its availability. On the other hand, this substitution can cause a semantic inconsistency at the level of the composition and deteriorate the mediation initially taken to solve semantic heterogeneities between the Web services components. In this research, we present a solution context directed to face the semantic impact produced by substitution. Before conducting this experience, this solution consists in adopting the semantics of the failed Web service, represented by its context, by the community which will have to turn over the result of execution of the functionality according to this adopted semantics. The main advantage of our solution is to avoid, to the maximum, the rupture of the semantic mediation in the composition in case of substitution of its Web services to make them highly available.

Key words: Web Service, composition, semantic mediation, context, high availability, community, substitution.

Introduction Générale

« Il faut dire que le courage est la vertu du commencement, de même que la fidélité est la vertu de la continuation et le sacrifice celle de la fin ».

Vladimir Jankélévitch

1 Contexte

Les services Web émergent comme une nouvelle génération des technologies du Web pour l'intégration des applications en déployant des interactions automatisées entre les applications d'entreprises réparties et hétérogènes. La notion de service Web désigne une application mise à disposition par un fournisseur et invocable sur Internet par des clients (utilisateurs ou autres services Web).

L'ambition portée par les services Web est de permettre une plus grande interopérabilité entre applications sur internet. L'on envisage ainsi des services Web capables, automatiquement, de se découvrir et d'être découverts, de négocier entre eux ou de se composer en des services plus complexes. La composition consiste à combiner plusieurs services Web dans un service composé pour répondre aux demandes complexes des utilisateurs qui ne peuvent être satisfaites par un simple service Web disponible.

Or, la médiation sémantique est sollicitée dans la composition afin d'éviter les hétérogénéités sémantiques et assurer un échange correct entre les services Web composants.

Ainsi, la haute disponibilité de ces services Web, participant à des compositions, est une exigence légitime par les utilisateurs pour répondre à leurs requêtes à n'importe quel moment. Et afin de soutenir cette exigence, plusieurs approches ont été imposées.

Nous nous intéresserons dans ce mémoire à l'approche de soutien de la haute disponibilité basée sur les communautés qui rassemblent les services Web selon leur fonctionnalité où le principe de cette approche est de substituer le service Web échoué dans la composition par un service Web de la communauté offrant une fonctionnalité équivalente à celle de ce service défaillant.

Cette substitution a pour objectif de garantir la continuité du service offert, mais d'un autre coté et suite à la sémantique particulière de ce service Web substitut, provoquera de nouvelles hétérogénéités sémantiques qui peuvent surgir au niveau de la composition entre ses services. La résolution de ces hétérogénéités sémantiques est appropriée pour éviter toute inconsistance sémantique dans la composition produite conséquemment à la substitution du service Web échoué.

2 Problématique

Les pratiques courantes dans les scénarios de composition exigent de diriger les conflits sémantiques entre les services Web par une approche de médiation sémantique. La médiation sémantique d'une composition des services Web doit évaluer l'impact sémantique quand au remplacement d'un service Web échoué par un autre service Web fonctionnellement similaire suivant l'approche basée-communauté et ce, pour supporter la haute disponibilité du service Web. Ce service Web remplaçant a une manière particulière de définir sémantiquement ses arguments d'entrée et de sortie dans un scénario de composition. C'est pour cela que dans le cas de substitution d'un service Web, il est considéré approprié de passer en revue les actions qui ont été initialement prises pour résoudre les conflits sémantiques dans une composition entre le service Web échoué et le reste des services Web. On peut résumer notre problématique à comment garantir la haute disponibilité des services Web dans une composition par l'approche basée-communauté des services Web tout en assurant en même temps la médiation sémantique de cette composition après la substitution des services Web défaillants.

3 Contribution

La substitution d'un service Web échoué membre d'une composition des services Web par un autre fonctionnellement équivalent adhérent à une communauté offrant une fonctionnalité semblable à celle de ce service échoué a pour but d'en soutenir la haute disponibilité selon l'approche basée-communauté sujet de notre mémoire. En outre, l'on doit assurer toujours la médiation sémantique dans cette composition entre le service Web remplaçant provenant de la communauté et le reste des services. La notion de communauté qui permet de regrouper les services Web de même fonctionnalité derrière une interface unique pour standardiser l'accès à ces derniers est prometteuse dans les futurs travaux de la sélection automatique des services Web participant à des compositions comme c'est le cas dans notre travail.

Notre contribution principale est de présenter une médiation sémantique orientée contexte pour résoudre les conflits sémantiques entre les services Web engagés dans une composition et ensuite, si l'un de ses éléments tombe en panne, l'on va procéder à sa substitution par un des services adhérant à une communauté fournissant une fonctionnalité comparable que celle du service Web défaillant.

Nous devrons résoudre encore une fois les conflits sémantiques qui peuvent venir du service suppléant ayant ses propres entrées/sorties avec le reste des services Web de la composition, tache pénible à chaque substitution. Et afin d'éviter cette lourde tache, on envisage une solution autre qu'adapter la médiation sémantique de la composition avec le nouveau service Web remplaçant. Les étapes de développement de notre solution de médiation sémantique seront examinées tout au long de ce mémoire à savoir :

- Réalisation d'un scénario de composition des services Web et la résolution de tous les conflits sémantiques qui peuvent y exister, par l'approche de médiation basée sur la notion de contexte des services Web pour faciliter l'échange correct des données.

- Lors de l'échec d'un des services Web de cette composition, sa substitution est nécessaire par un service membre d'une communauté pour garantir la haute disponibilité de ce service échoué. Notre idée, pour ne pas rompre la médiation sémantique dans la composition, exige que la communauté à laquelle appartient le service substitut doit adopter la sémantique du service tombé en panne représentée par son contexte et ce, afin qu'elle puisse retourner le résultat de l'exécution de la fonctionnalité avec cette même sémantique. Cette idée est orientée contexte pour rester toujours dans le même cadre de l'approche de médiation basée-contexte appliquée initialement à la composition et, par conséquent, a pour objectif de maintenir toujours cette même médiation sémantique prise et d'éviter de la modifier à chaque substitution d'un service Web échoué dans la composition.

- Pour démontrer la faisabilité de notre solution, l'on envisage aussi de réaliser une application montrant les étapes principales de cette contribution à savoir: l'adoption de la sémantique du service Web échoué ainsi que les conversions sémantiques nécessaires pour réaliser une haute disponibilité gardant la médiation sémantique telle quelle dans les compositions des services Web.

4 Organisation

Après cette introduction montrant la problématique de ce travail ainsi que notre contribution, le reste de ce mémoire est structuré comme suit :

Le Chapitre 1 donne une présentation générale sur la technologie des services Web en montrant leur fonctionnement, les standards sur lesquels ils reposent ainsi que leur composition qui est l'apport principal de cette nouvelle technologie.

Le chapitre 2 présente en détail les approches de soutien de la haute disponibilité des services Web. Il montre les inconvénients des approches basées sur la réplication et leurs remèdes dans l'approche basée sur les communautés des services Web.

Le chapitre 3 fait apparaitre la pertinence de la médiation sémantique entre les services Web afin de garantir entre eux l'échange correct des données. Il présente, à l'aide d'un exemple, les approches de cette médiation basées sur la notion de contexte, que ce soit au niveau de composition ou celui de communauté des services Web.

Le chapitre 4 expose notre contribution qui consiste à proposer un processus de médiation sémantique en cas de substitution du service Web défaillant dans la composition en vue d'échapper à l'impact sémantique provoqué par cette substitution et qui a pour but d'assurer la haute disponibilité des services Web. L'exemple pris auparavant fera l'objet de notre implémentation dans ce chapitre.

Et pour terminer ce mémoire, une conclusion générale qui exprime des apports de notre contribution ainsi que des perspectives de recherche envisagées.

I Chapitre 1 : Services Web, présentation générale

« Les découvertes récentes ont anéanti toutes nos illusions sur la simplicité de l'univers ».

Gustave Le Bon

I.1 Introduction

Les services Web prennent leur origine dans l'informatique distribuée et dans l'avènement du Web. Le but de l'informatique distribuée est de permettre à une application sur une machine d'accéder à une fonction d'une autre application sur une machine distante et ce, de la même manière que l'appel d'une fonction locale, indépendamment des plates-formes (.NET ou J2EE) et des langages utilisés (Java, C#, Phyton,…) et mêmes des systèmes d'exploitation (Windows, Unix, Solaris,…).

Actuellement, dans les technologies à base de composants distribués, le couplage entre les applications est fort et complexe pour assurer l'interopérabilité. Ainsi, il est peu compatible avec les pare-feux. A titre d'exemple, RMI (Remote Method Invocation) est une API (Application Program Interface) Java permettant de manipuler des objets distants créés en langage Java. Les communications s'effectuent grâce au protocole RMI-IIOP (Internet Inter-Orb Protocol), un protocole normalisé par l'OMG (Object Management Group) et utilisé dans l'architecture CORBA. La norme CORBA de l'OMG permet de manipuler des objets à distance avec n'importe quel langage. La mise en œuvre de CORBA nécessite une connaissance de la structure des objets manipulés par les applications clientes et par les applications fournisseurs. Des règles de transformation objets/messages doivent être définies au préalable par les partenaires [1]. Ces technologies restent donc souvent confinées à l'intérieur des entreprises [35].

Le second point qui a favorisé l'essor des services Web est le développement d'Internet par la technologie du World Wide Web qui est énormément sollicitée pour accomplir les activités quotidiennes, la structuration des données via XML et la recherche d'interopérabilité. Les entreprises publiaient déjà de l'information via des sites Web, utilisaient la messagerie et faisaient du commerce électronique. Maintenant, elles cherchent à utiliser les services Web pour leurs systèmes d'informations notamment leurs applications métiers (ressources humaines, ventes, finances, etc.).

Les systèmes d'information des entreprises doivent être souples et extensibles pour intégrer des modifications et des changements imposés par des besoins internes ou par des contraintes externes. Les services Web sont là pour réaliser cette exigence. Ces services Web représentent donc une nouvelle technologie qui dépasse les anciennes (CORBA, RMI…) pour faciliter l'interopérabilité des systèmes d'information, en se fondant sur les standards Internet comme XML [52] et http [40], d'où leur réputation croissante ces dernières années, notamment dans le développement de processus d'affaires inter-entreprises (souvent désignés par applications B2B).

Il existe aujourd'hui une grande variété de services Web capables de répondre aux différents types de requêtes des utilisateurs (prévisions météorologiques, différentes réservations comme pour l'hôtel ou les voyages, recherche, …).

Un service Web est un composant logiciel qui offre des prestations à travers une interface standardisée. La particularité des services Web réside dans le fait qu'elle utilise la technologie Internet comme infrastructure pour la communication et ceci en mettant en place un ensemble de standards basé sur le langage XML pour la description des caractéristiques fonctionnelles et non fonctionnelles de ces services. Le processus de standardisation touche actuellement trois couches du modèle de fonctionnement global : un protocole de communication permettant de structurer les messages échangés entre les services Web (SOAP[14]), une spécification de description des interfaces des services (WSDL[16]) et enfin une spécification de publication et de localisation de services (UDDI[48]). Actuellement, ce modèle supporte principalement des composants logiciels présentant des services sous forme d'une collection d'unités de traitement (opérations) dont l'invocation n'excède pas un échange simple de messages (généralement requête-réponse). A ce niveau de complexité des composants logiciels, la technologie des services Web est suffisante pour mettre en place des composants interopérables et facilement intégrables [47].

Dans ce chapitre, nous présentons les notions préliminaires des services Web : leur définition, leur fonctionnement, ainsi que les standards sur lesquels ils reposent et leur composition avec la nécessité de résoudre les conflits sémantiques qui peuvent surgir entre les services composants.

I.2 Définition d'un service Web

Comme pour tout concept important, il est fondamental de définir de manière rigoureuse la notion de service Web. Plusieurs définitions variant avec le temps ont été proposées. Nous présentons ici deux définitions standardisées par le World Wide Web Consortium (W3C).

Première définition:

« A Web Service is a software system identified by a URI, whose public interfaces and bindings are defined and described using XML. Its definition can be discovered by other software systems. These systems may then interact with the Web Service in a manner prescribed by its definition, using XML based messages conveyed by Internet protocols»[1].

Cette définition définit un service Web comme étant une application ou un composant logiciel vérifiant les propriétés suivantes [13,37] :

- Il est identifié par une URI[2];
- Ses interfaces et ses liens peuvent être décrits en XML;
- Sa définition peut être découverte par d'autres services Web;
- Il peut interagir directement avec d'autres services Web à travers le langage XML et en utilisant des protocoles Internet.

[1] http://www.w3.org/TR/2004/NOTE-wsa-reqs-20040211/
[2] Uniform Resource Identifier : Il s'agit d'un identifiant unique pour une ressource Web.

Deuxième définition:

« 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 »[3].

Cette dernière met en valeur les avantages principaux d'un service Web, à savoir [30]:

- 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 plates-formes d'implantation, qui renforcent l'interopérabilité[4] entre services.
- Son utilisation des normes actuelles du Web, qui permettent la réalisation des interactions faiblement couplées et favorisent aussi l'interopérabilité.

Pour résumer, on peut dire qu'un service Web « est une application accessible et que d'autres applications ou utilisateurs peuvent la trouver et l'invoquer pour satisfaire leurs divers besoins»[10].

Exemples des Services Web:

Plusieurs compagnies publient des interfaces publiques des services Web, à savoir:

- Google: http://www.google.com/apis, une application qui emploie une ontologie pour mettre en valeur une recherche Google.
- Prévisions météo via le service du National Digital Forecast Database XMLWeb: http://www.weather.gov/xml.
- Amazon: http://simplest-shop.com/camera (e-commerce)
- MapQuest (pour localiser un lieu et donner des itinéraires), FedEx, etc[5].

I.3 Architecture des services Web

L'architecture des services Web est une instance de l'architecture orientée Service (SOA :Service Oriented Architecture). L'architecture SOA (voir Fig. I.1) est un paradigme qui définit un système par un ensemble de composants logiciels distribués qui fonctionnent de concert afin de réaliser une fonctionnalité globale préalablement établie. Les composants dans un système distribué n'opèrent pas dans un même environnement de traitement et sont obligés de

[3] http://www.w3.org/TR/2004/NOTE-ws-gloss-20040211/

[4] L'interopérabilité ou l'interfonctionnement est le fait que plusieurs systèmes, qu'ils soient identiques ou radicalement différents, puissent communiquer sans ambigüité et opérer ensemble.

[5] Plus de définitions/illustrations/exemples dans : www.Webservices.org

communiquer par échanges de messages afin de solliciter des services dans le but d'accomplir le résultat souhaité.

La définition de l'architecture des services Web consiste à mettre en évidence les concepts, les relations entre ces concepts ainsi qu'un ensemble de contraintes qui assurent l'objectif premier des services Web à savoir l'interopérabilité [19].

I.3.1 Principaux concepts

Les principaux concepts intervenant dans l'architecture des services Web sont [47] :

- **Fournisseur du service** : d'un point de vue conceptuel, il désigne la personne ou l'organisation responsable juridiquement du service Web. D'un point de vue opérationnel, les fournisseurs de services peuvent également désigner le serveur qui héberge les services déployés.

- **Client du service** : comme pour le fournisseur, il représente une personne ou une organisation, client (utilisateur) potentiel du service Web comme il désigne également, d'un point de vue opérationnel, l'application cliente qui invoque le service.

- **Registre des services** : il représente la personne ou l'organisation responsable de l'hébergement du dépôt de publication des services. D'un point de vue fonctionnel, il représente l'entité logicielle qui joue le rôle d'annuaire entre les clients et les fournisseurs de services. Le concept registre ou dépôt de service est essentiel dans l'architecture services Web. Il joue un rôle central dans le processus de localisation des besoins et dans l'interopérabilité, car il est supposé fournir aux clients les informations techniques et sémantiques sur le fonctionnement du service et ceci dans des langages formels (interprétables par les machines).

- **Service** : le service est une notion abstraite pour désigner les fonctionnalités d'un agent (composant) logiciel qui implémente une fonctionnalité du service. Quand on parle de service, il est essentiel de faire la distinction entre le service vu d'un angle sémantique et sa réalisation qui est l'entité logicielle. Un seul service Web, par exemple, peut avoir plusieurs agents qui l'implémentent (utilisant des langages différents).

- **Description du service** : c'est la spécification du service exprimée dans un langage de description interprétable par les machines. Il existe deux niveaux de description de services. Une description technique dans laquelle le service est vu en termes de messages, de formats, de types, de protocoles de transport et d'une adresse physique. Cette description joue le rôle d'un contrat d'interaction entre le client et le service. La deuxième description concerne la sémantique du service. La description sémantique peut avoir une existence formelle comme elle peut prendre la forme d'un accord entre le fournisseur et le client. L'architecture des services Web impose à chaque entité logicielle accessible sur le Web d'éditer sa description afin d'assurer l'interopérabilité entre le client et le service.
- **Messages** : le message joue un rôle important dans l'architecture des services Web, c'est la plus petite unité d'échange de données entre les clients et les services. Le concept

message ne reflète aucunement la sémantique de son contenu, seuls la structure et les mécanismes de transport sont considérés dans l'architecture. La structure des messages qui permettent l'invocation des différentes unités fonctionnelles composant le service doit figurer dans la description du service.

- **Ressource** : le concept ressource désigne l'identifiant du service. Il constitue un point important dans l'architecture des services Web. Chaque service doit avoir une identification en terme d'adresse, généralement une URI. Les services Web, comme leur nom l'indique, doivent être accessibles à partir d'une adresse Web.

I.3.2 Fonctionnement

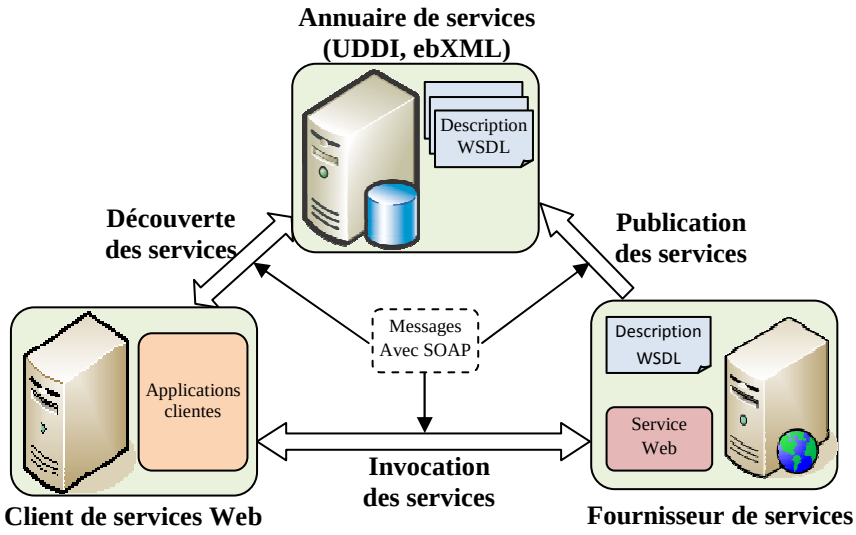


Fig. I.1 – Fonctionnement des services Web selon l'architecture SOA

Les étapes nécessaires à l'accès et à l'invocation (utilisation) d'un service Web selon l'architecture SOA présentée dans Fig. I.1 sont les suivantes [27,30] :

1) **Publier le service** : Le fournisseur de service déploie et publie la description WSDL[6] de son service dans des registres (annuaires) en vue d'être localisé par des clients;
2) **Découvrir le service** : Un client ayant des besoins spécifiques va rechercher le service correspondant à ses besoins à l'aide d'un annuaire spécialisé UDDI (ou autre tel que ebXML[7]) en spécifiant les critères.
3) **Récupérer la description du service** : le client reçoit en réponse à sa requête un document WSDL décrivant le service dont il a besoin.
4) **Invoquer le service** : sur la base des informations définies dans la description du service, une communication entre le client et le service Web se fait via le protocole SOAP. Le client appelle le service Web en précisant les paramètres divers de ce dernier.
5) **Recevoir la réponse** : Le client reçoit une réponse à sa demande.

[6] XML, SOAP, WSDL, UDDI seront étudiés ci-après.

[7] Une alternative permettant de fournir les mêmes fonctionnalités (http://www.ebxml.org).

I.3.3 Pile standard des couches

Ainsi, les architectures orientées service se construisent autour de plusieurs protocoles et langages, selon quatre couches de fonctionnalités (Fig. I.2) représentant une pile standard de protocoles des services Web [30,10] :

Fig. I.2 - Pile Standard des couches des Services Web

a) Couche Transport : permet de véhiculer les messages à travers le réseau. Pour la couche transport, le HyperText Transfert Protocol (HTTP) est devenu le standard. 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.

b) Couche Messages : assure la structuration et l'échange uniformes des messages, elle 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) et XML-Remote Procedure Call (XML-RPC) sont les deux protocoles utilisés pour cette couche, SOAP étant le protocole prédominant.

c) Couche Description : regroupe les détails nécessaires à l'invocation des services dans un document. Elle 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 de données contenues dans les messages sont décrits à l'aide du langage XML Schema [53]. Cette couche est prise en charge par le Web Service Description Language (WSDL)

d) Couche Publication : assure le regroupement, le stockage et la diffusion des descriptions de services Web et qui repose sur le protocole Universal Description, Discovery, and Intégration (UDDI).

Devant l'importance des termes XML, SOAP, UDDI et WSDL constituant les standards des services Web, la section suivante les expose d'une manière succincte.

I.4 Standards des services Web

XML, SOAP, WSDL et UDDI sont les technologies dominantes des services Web. Sans entrer dans tous les détails techniques, nous vous présentons ici les grandes lignes de chacune de ces technologies.

I.4.1 XML

Le XML(eXtended Markup Language) [45,52] est un langage permettant la représentation de données ainsi que de documents structurés sans l'utilisation de balises prédéfinies. Ainsi, ce langage peut être étendu de façon à y ajouter des balises spécialisées afin de décrire au mieux chaque type de données. Cette flexibilité confère à XML la popularité dont il jouit.

Historiquement, XML, développé par le W3C en 1996, a profité des meilleurs aspects de SGML[8] et est, depuis 1998, une recommandation du W3C. XML décrit un ensemble de règles syntaxiques, de façon à placer de manière cohérente et correcte les balises à l'intérieur d'un tel document. Un document XML qui respecte toutes ces règles imposées est dit bien formé.

L'avantage de toutes ces règles est de permettre la création de « ***Parsers*** » XML qui seront utilisés afin de lire et d'extraire les données d'un document XML[27].

Un exemple de document XML est donné dans Fig. I.3 :

```
<collection nom="Technologies Web">
        <livre id="1">
                <titre> Les Web services </titre>
                <auteurs>
                        <auteur>Hubert Kadima</auteur>
                        <auteur>Valérie Monfort</auteur>
                </auteurs>
                <sujet>Services Web </sujet>
        </livre>
</collection>
```

Fig. I.3 - Code source : Exemple de document XML bien formé

Les balises en gras sont appelées des **éléments**. Chaque entité peut avoir un ou plusieurs **attributs** (en *italique*) permettant d'ajouter de l'information supplémentaire aux éléments.

[8] Standard Generalized Markup Language.

I.4.1.1Namespaces XML

Les namespaces, ou espaces de noms XML, permettent de classer les éléments ainsi que les attributs en une collection, de façon à éviter les ambiguïtés de nommage telles que les collisions de noms d'éléments ou d'attributs. Chaque namespace est identifié par un URI et est déclaré en utilisant le mot réservé « **xmlns** » tout en respectant la syntaxe :

- « xmlns: prefix = "URI" » dans le cas d'une utilisation avec préfixe ou bien,
- « xmlns="URI" ».

Les deux solutions sont équivalentes, si ce n'est que la première définit en plus un préfixe qui devra être utilisé avant chaque élément ou attribut du namespace. Le code de Fig. I.4 utilise un namespace avec préfixe [27].

```
<ns1 :livre xmlns :ns1 = "http ://www.mylibrary.com">
        <ns1 :titre>Les Web services</ns1 :titre>
        <ns1 :auteurs>
                <ns1 :auteur>Hubert Kadima</ns1 :auteur>
                <ns1 :auteur>Valérie Monfort</ns1 :auteur>
        </ns1 :auteurs>
</ns1 :livre>
```

Fig. I.4 - Code source : Exemple de namespace XML (préfixé)

I.4.1.2 Schémas XML

Un schéma XML [53] est un document XML permettant la définition d'une structure possible d'un document XML. Ce schéma permettra la validation du dit document en décrivant l'ensemble des éléments ainsi que des attributs pouvant apparaître dans ce dernier, l'ordre des éléments fils ainsi que leur contenu, les types de données des éléments et des attributs ainsi que leur valeur par défaut. Le schéma XML du Fig. I.5 correspond à celui du document XML du Fig. I.4 [27].

```
<xs :schema xmlns :xs=”http ://www.w3.org/2001/XMLSchema”
elementFormDefault=”qualified”>
        <xs :element name=”collection”>
                <xs :complexType>
                  <xs :sequence>
                      <xs :element maxOccurs=”unbounded” ref=”livre”/>
                        </xs :sequence>
                        <xs :attribute name=”nom”use=”required”/>
                </xs :complexType>
        </xs :element>
        <xs :element name=”livre”>
                <xs :complexType>
                        <xs :sequence>
                                <xs :element ref=”titre”/>
                                <xs :element ref=”auteurs”/>
                                <xs :element ref=”sujet”/>
                        </xs :sequence>
                        <xs :attribute name=”id”use=”required”type=”xs :integer”/>
                </xs :complexType>
        </xs :element>
        <xs :element name=”titre”type=”xs :string”/>
        <xs :element name=”auteurs”>
                <xs :complexType>
                        <xs :sequence>
                                <xs :element maxOccurs=”unbounded”ref=”auteur”/>
                        </xs :sequence>
                </xs :complexType>
        </xs :element>
        <xs :element name=”auteur”type=”xs :NCName”/>
        <xs :element name=”sujet”type=”xs :NCName”/>
</xs :schema>
```

Fig. I.5 - Code source : Exemple de schéma XML

I.4.2 SOAP

SOAP [14,39] ou encore Simple Object Access Protocol, est un standard du consortium W3C définissant un protocole RPC[9], tout comme RMI[10] et CORBA[11], mais utilisant XML pour structurer les messages. Il permet donc l'échange de messages entre entités SOAP dans un environnement distribué d'un émetteur à un récepteur pouvant passer par des intermédiaires. Un des grands avantages de ce protocole est qu'il n'est pas limité à un seul protocole de transport, contrairement à son ancêtre XML-RPC[12] qui, lui, est basé sur HTTP. En effet, alors que la grande tendance est à HTTP, on pourrait parfaitement utiliser SMTP, FTP ou toute autre technologie future pour l'implémentation de cette couche. Cette structure a été conçue pour être indépendante de tout modèle de programmation et autre sémantique spécifique d'implémentation.

Un message SOAP est une transmission unidirectionnelle entre des nœuds SOAP (d'un

[9] Remote Procedure Call : mécanisme permettant d'appeler des procédures situées sur un ordinateur distant.
[10] Remote Method Invocation : API de communication entre objets distants de Java de Sun
[11] Common Object Request Broker Architecture de OMG
[12] http://www.xmlrpc.com/

émetteur vers un récepteur SOAP), mais les messages SOAP peuvent être combinés par les applications pour implémenter les séquences plus complexes d'interactions (ex : Request-Reply (requête-réponse), échanges multiples "conversationnels").

I.4.2.1Messages SOAP

Deux formats de messages SOAP sont possibles [27,42,11], dépendant de l'utilisation ou non de pièces jointes. Comme illustré dans Fig. I.6, un message sans pièce jointe se compose de trois parties :

1) Une enveloppe (**Envelope**) qui est **obligatoire** et qui contient le nom du message ainsi que les namespaces XML-SOAP utilisés,
2) Un entête (**Header**) qui est **optionnel** et utilisé pour spécifier les comportements que doivent adopter les différents intermédiaires se situant entre l'expéditeur et le récepteur du message,
3) Un corps (**Body**) qui est **obligatoire** et qui contient les noms des différentes méthodes qui seront exécutées par le destinataire, ainsi que les paramètres associés, ou la réponse à une requête ou encore un message d'erreur.

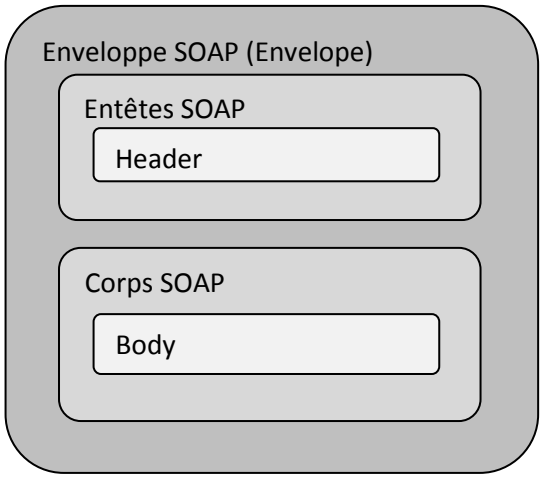


Fig. I.6 - Format d'un message SOAP sans pièce jointe

La version avec pièce jointe s'obtient aisément en ajoutant au format de base, à l'extérieur de l'enveloppe, des blocs de pièces jointes qui sont composés d'un entête MIME[13] ainsi que du contenu.

Dans les deux formats possibles, le message sera encapsulé par la couche de transport utilisée. Afin d'illustrer un échange Request-Reply utilisant SOAP, les Figures I.7 et I.8 montrent respectivement un message SOAP faisant appel à la méthode « SayHello » avec le paramètre "Ahmed", ainsi que le message de réponse associé contenant la réponse "Hello Ahmed".

[13] Multipurpose Internet Mail Extensions : précise comment mêler dans un même message différents types de données, sous le type particulier multipart/related.

```
<soap-env :Envelope xmlns :soap-nv=”http://schemas.xmlsoap.org/soap/envelope/”
                    xmlns :xsd=”http ://www.w3.org/2001/XMLSchema”
                    xmlns :ns1=”http ://test/”>
          <soap-env :Body>
                    <ns1 :SayHello>
                              <name>Ahmed</name>
                    </ns1 :SayHello>
          </soap-env :Body>
</soap-env :Envelope>
```

Fig. I.7 - Code source : Message de requête SOAP

```
<soap-env :Envelope xmlns :soap-env=”http ://schemas.xmlsoap.org/soap/envelope/”
                    xmlns :xsd=”http ://www.w3.org/2001/XMLSchema”
                    xmlns :ns1=”http ://test/”>
          <soap-env :Body>
                    <ns1 :SayHelloResponse>
                              <return>hello Ahmed</return>
                    </ns1 :SayHelloResponse>
          </soap-env :Body>
</soap-env :Envelope>
```

Fig. I.8 - Code source : Message de réponse SOAP

Finalement, il est à noter que des extensions peuvent être ajoutées au protocole SOAP de façon à assurer par exemple la sécurité, la fiabilité de transmission, le routage asynchrone, etc.

I.4.3 WSDL

A ce stade, dans la pile des couches des services Web, une problématique naît d’un point de vue de la description des services puisque le protocole SOAP ne s’occupe que de l’échange de messages. En effet, dans l’exemple d’échange Request-Reply (de la section I.4.2), comment l’utilisateur du service connaît-il les paramètres, les noms de méthodes de services distants et le type de réponse? La solution à cette problématique a été apportée par les sociétés Ariba, IBM et Microsoft, qui ont proposé la spécification WSDL (Web Services Description Language) [16] permettant la description de services Web et en particulier de leur interface au moyen de documents WSDL.

Ainsi, chaque prestataire de services va définir dans un document WSDL les différents types de messages qu’il peut recevoir et éventuellement ceux qu’il peut envoyer en guise de réponse.

En plus de définir ces derniers, WSDL définit également la localisation des services ainsi que les différents mécanismes pour accéder à ceux-ci puisque ces derniers sont accessibles via plusieurs protocoles différents.

I.4.3.1Structure d’un document WSDL

Un document WSDL est constitué d’une suite d’éléments décrivant un service Web sous forme d’un ensemble d’opérations exploitables depuis l’extérieur. Il est composé de deux parties

principales: une partie abstraite constituée de la définition des **Types**, des **Messages**, des **Opérations** (regroupement de messages) et des **Types de ports** (regroupement d'opérations), et une partie concrète constituée de la définition des liaisons (**Bindings**) et **Service**. Les informations de liaison définissent comment accéder au service en terme de protocoles, formats de messages et adressages. L'intérêt d'avoir ces deux parties, est que la partie concrète propose une ou plusieurs réalisations de la partie abstraite.

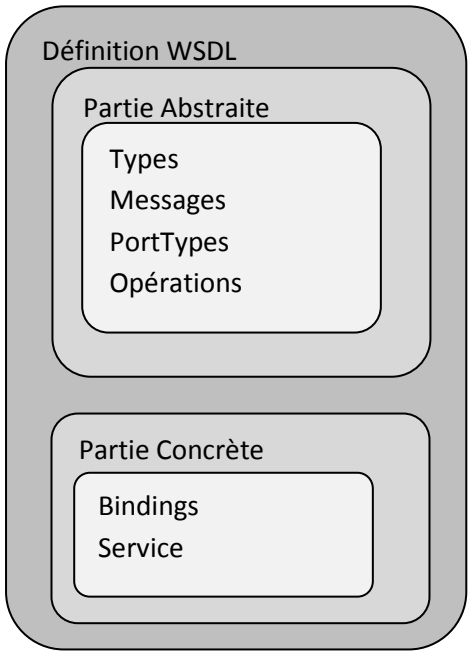


Fig. I.9 – Structure d'une description WSDL

Une description WSDL est un document XML où sa structure est illustrée à la Fig. I.9 et qui est constitué des éléments principaux suivants [11,27] :

- **Définition** : elle contient le nom du service qui est décrit dans le document ainsi que les namespaces utilisés. Il s'agit de l'élément « Root » du document.
- **Types** : il s'agit de la définition des types de données qui seront utilisés dans les messages. On y trouvera donc par exemple des schémas XML.
- **Messages**: il s'agit d'une description de la structure des messages qui seront échangés. On y retrouve deux types de messages : les messages entrants et les messages sortants. Chaque message peut être composé de plusieurs parties décrivant les différents paramètres de ces derniers.
- **PortTypes** : il s'agit d'un ensemble d'opérations, chacune se référant à un message entrant et sortant.
- **Opérations** : description d'une des actions supportées par le service Web. Une opération reçoit des messages et envoie des messages.
- **Bindings** : ils spécifient quels protocoles seront utilisés.
- **Service** : il s'agit des informations permettant la localisation du service.

Ainsi, lorsqu'un client veut accéder à un service Web, il va tout d'abord devoir se procurer la description WSDL de ce dernier. A l'aide de celle-ci, il aura toutes les informations nécessaires

pour créer les messages SOAP afin d'effectuer l'invocation et récupérer la réponse.

I.4.4 UDDI

UDDI [48], ou Universal Description Discovery and Integration, est la spécification d'un framework[14] permettant le stockage (publication) de descriptions de services Web ainsi que la découverte de ces derniers. Ce projet, lancé par les sociétés Microsoft, IBM et Ariba et soutenu initialement par plus d'une trentaine d'entreprises, est la réponse à la problématique du B2B[15] sur Internet ainsi qu'au succès du commerce électronique.

L'API UDDI est en fait un service Web qui est décrit via la technologie WSDL et qui permet d'accéder à un annuaire de services Web (annuaire UDDI) via le protocole SOAP.

Une analogie peut être faite entre un annuaire UDDI et un annuaire téléphonique : un annuaire UDDI va organiser l'information sur les services Web en trois catégories :

- **Les pages blanches** : celles-ci sont constituées des informations de description des entreprises fournissant les différents services Web. Une entrée de ces pages sera donc formée du nom de l'entreprise, de son adresse, des informations de contact ainsi que d'une description de celle-ci.
- **Les pages jaunes** : il s'agit de la classification des services Web selon des taxonomies standards comme par exemple les sortes de produits, les entreprises ou encore la localisation géographique.
- **Les pages vertes** : il s'agit des informations techniques concernant les différents services Web. Ceci se fera par l'intermédiaire de documents WSDL.

Pour ce faire, la spécification UDDI définit un modèle de données constitué des quatre entités principales suivantes [11,27] :

1) **L'entité Business (BusinessEntity)** : contient les informations concernant l'entreprise incluant son nom, sa description, son adresse ainsi que les informations de contact; il s'agit donc de l'équivalent des pages blanches.
2) **L'entité Service (BusinessService)** : cet élément décrit les services Web associés à l'entité business correspondante; il contient le nom, la description ainsi qu'une liste (optionnelle) de BindingTemplates; il s'agit donc de l'équivalent des pages jaunes.
3) **L'entité modèle de liaison (BindingTemplate)** : il décrit les informations techniques permettant de savoir où et comment accéder au service Web; il est donc ici question des pages vertes.
4) **L'entité modèle technique (tModel)** : il décrit les spécifications d'un service Web. Il contiendra donc une référence vers un fichier de description, généralement un document WSDL.

Afin de publier un ou plusieurs services Web dans un annuaire UDDI, il faut d'abord définir les tModels qui vont décrire les caractéristiques des services Web, comme l'interface du service

[14] Ensemble de bibliothèques qui permet de développer rapidement une application.
[15] Business-to-Business.

en WSDL. Il est à noter qu'un tModel peut déjà exister pour un service implémentant une interface standardisée qui a déjà été publiée par le consortium de standardisation correspondant.

Ensuite, il convient de publier les informations concernant l'entreprise (dans une BusinessEntity), celles concernant les services Web (dans une BusinessService) et finalement les informations techniques (contenant notamment une référence vers un tModel) pour les différents services. Fig. I.10 illustre une entrée dans un annuaire UDDI ainsi que les relations entre les différentes entités.

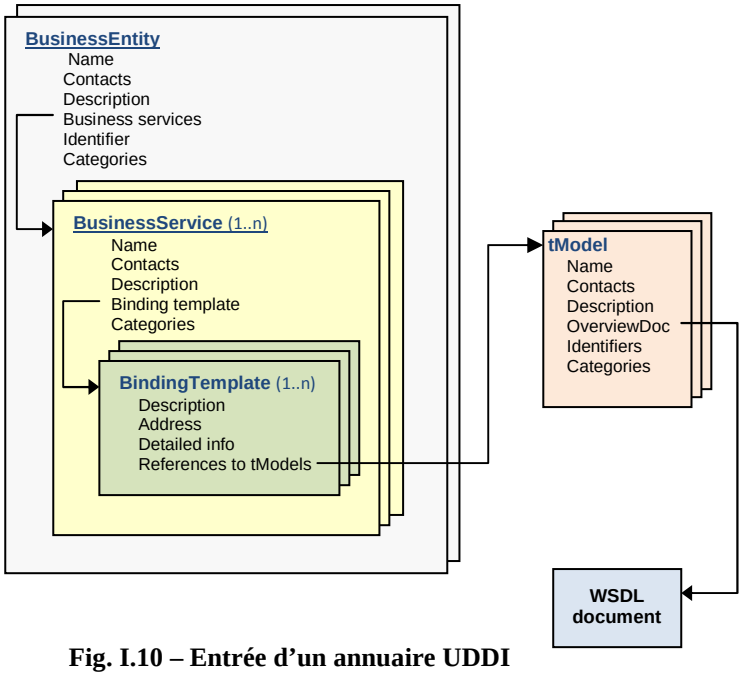


Fig. I.10 – Entrée d'un annuaire UDDI

I.5 Composition et Médiation Sémantique

L'apport principal des services Web est leur participation dans des compositions. La composition consiste à combiner les fonctionnalités de plusieurs services 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 satisfaire. Le processus métier est une représentation concrète des tâches à accomplir dans une composition.

Une composition des services Web est toujours associée à une spécification [10] qui décrit la liste de services Web composants participant à ce service composé, l'ordre d'exécution de ces services Web composants et les stratégies correctives en cas de manipulation d'exception.

L'exemple classique donné dans plusieurs travaux traitant les compositions des services Web est la planification de voyage. Fig. I.11 illustre un service Web composé, fourni par une agence de voyage.

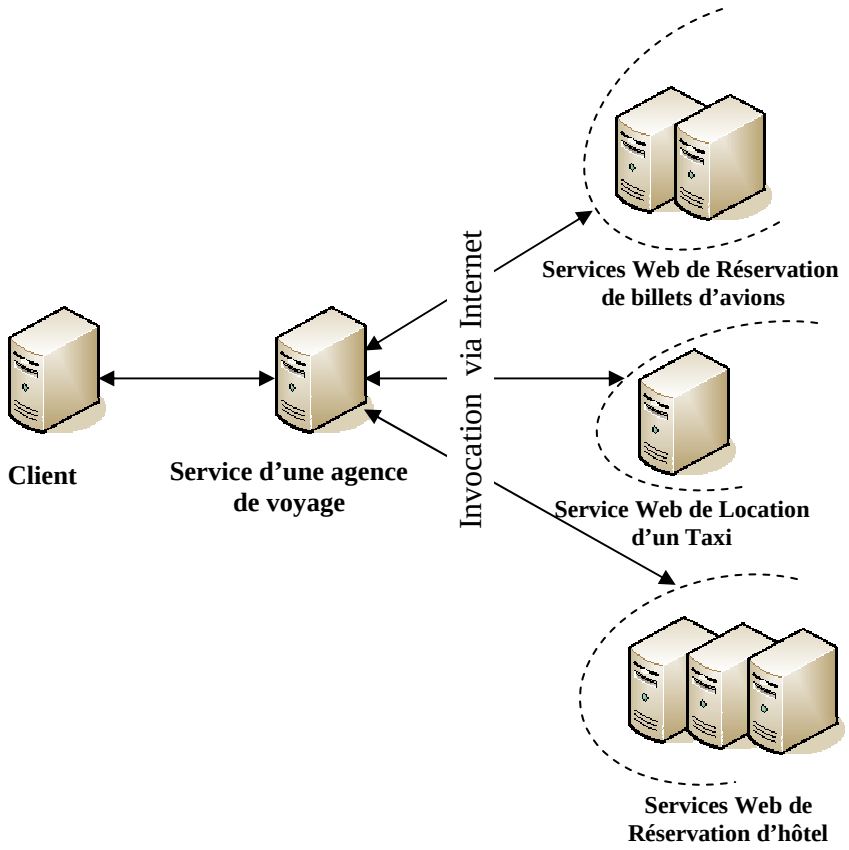


Fig. I.11 - Exemple de composition des services Web « Planification de voyage »

Dans cet exemple, une agence de voyage met à la disposition du client un service permettant la réservation d'un voyage. Ce service aura pour tâche d'effectuer les réservations nécessaires[16] après sélection des services convenables répondant aux différents critères à savoir ceux donnés par le client. Il est important de remarquer le fait que l'agence de voyage joue un double rôle : celui de prestataire de service envers le client et celui d'utilisateur des services de réservations divers. Ce type d'architecture est transparent pour le client qui n'utilisera que le service mis à sa disposition par l'agence de voyage.

Le déroulement d'une composition nécessite la réalisation des étapes suivantes [30]:

1) **La découverte des services Web** pouvant répondre aux besoins de la composition se fait généralement manuellement[17] par envoi de requêtes aux annuaires UDDI.

2) **L'organisation des interactions** entre les services Web participants est distinguée par les approches **orchestration et chorégraphie**. L'orchestration des services Web combine des services disponibles en ajoutant un coordonnateur central (l'orchestrateur) qui est responsable de l'invocation et de la combinaison des activités de la composition. La chorégraphie des services Web n'assume pas l'organisation d'une composition par un

[16] Tous les services offrant la même fonctionnalité telle que la réservation d'hôtel peuvent être regroupés dans des communautés. Chose qu'on va voir dans chapitre 2.
[17] De nombreux travaux en cours visent à automatiser cette étape.

coordonnateur central mais elle définit des tâches complexes via la définition de la conversation qui devrait être entreprise par chaque participant [5], d'une autre manière, l'orchestration organise les interactions entre plusieurs services Web (inter-service Web) et la chorégraphie, quant à elle, gère les échanges de messages avec un unique service Web (intra-service 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[18] et actuellement le WS-BPEL[19] s'est révélé comme un standard pour la composition des services Web. Le WS-BPEL est un langage d'orchestration basé XML permettant de décrire un processus métier de haut niveau via la description des interactions entre différents services Web dans une composition. Il tire ses origines de WSFL[20] et XLANG[21].

3) **L'exécution de la composition** est l'étape d'invocation effective des services Web participant 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 (structure, syntaxique, sémantique). 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 [30] où la résolution de ces hétérogénéités est appelée **médiation**. Cette médiation consiste à implanter des mécanismes supplémentaires par des médiateurs dans les compositions pour l'adaptation des données échangées entre les services Web. Le médiateur est un module logiciel qui exploite la connaissance codée sur quelques ensembles ou sous-ensembles de données pour créer l'information pour une couche plus élevée d'applications [30].

Dans le cadre de notre travail, l'on s'intéresse aux hétérogénéités sémantiques 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.

De nombreux nouveaux langages tel que:

- OWL-S: Web Ontology Language for Web services
- WSMO: Web Service Modeling Ontology
- DIANE: DIANE Elements (DE) et DIANE Service Description (DSD)

Ou langages d'extension en employant l'annotation comme:

- SESMA: SEmantic Service MArkup poue annoter les processus métier

[18] WSCI , WSFL , XLANG , WS-CDL , YAWL , XPDL , BPMN.
[19] Web Service Business Process Execution Language.
[20] Web Services Flow Language.
[21] XML Business Process Language

- WSDL-S:WSDL Semantic pour annoter le WSDL
- SAWSDL: Semantic Annotations for Web Services Description Language

Ont pour objectif de résoudre les problèmes de sémantique rencontrés par les services Web [30,31,18]. Ces langages intègrent la sémantique des données dans les descriptions des services Web[22], qui deviennent des **services Web sémantiques**. Un service Web sémantique étend la description d'un service Web en ajoutant des informations sémantiques (voir Fig. I.12) afin de permettre une meilleure découverte, sélection, composition et intégration.

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» [46].

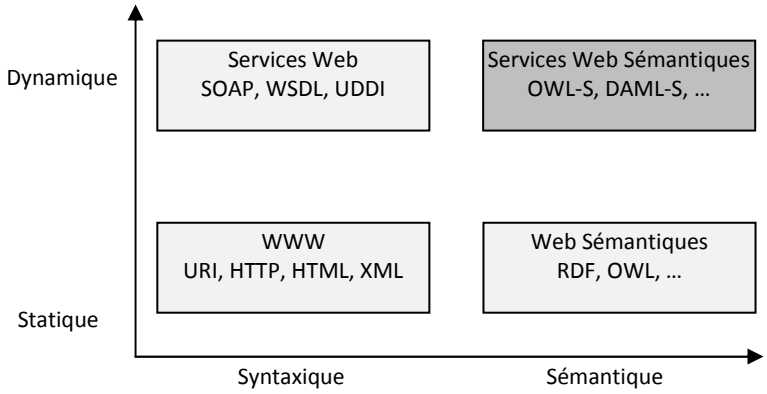


Fig. I.12 – Evolution vers les Services Web Sémantiques

I.5.1 Résolution des hétérogénéités sémantiques

Avant son exécution[23], 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. Et afin de résoudre ces hétérogénéités pour réaliser des échanges consistants, au niveau sémantique, entre des services Web participant à une composition, il faut adopter une approche de médiation sémantique[24].

[22] Pour un bon état de l'art des langages de description sémantiques, voir les travaux de Mrissa [30,31].

[23] On s'intéresse dans ce mémoire à la 3eme étape de la réalisation d'une composition.

[24] Le troisième chapitre est destiné à la présentation de ces hétérogénéités ainsi que leur résolution par la médiation sémantique.

I.6 Conclusion

Dans ce chapitre, nous avons présenté les concepts de base de la technologies des services Web, notamment leur composition dans des processus métiers pour répondre aux besoins complexes des utilisateurs.

Les services Web reflètent une approche de conception « orientée services », basée sur l'idée de construire des applications en déployant et en découvrant des services sur un réseau ou en invoquant des applications pour accomplir des taches. Les services Web sont en fait des applications modulaires et indépendantes qui peuvent être décrites, publiées, localisées et appelées à distance sur un réseau (Intranet, Internet,…) accessibles via des normes et standards Internet tels que XML et HTTP.

Par ailleurs, la bonne utilisation de ces services Web exige d'assurer leur disponibilité pour être accessibles à tout moment. La disponibilité des services Web, de nos jours, est un challenge scientifique dans le domaine du Web. Le deuxième chapitre est consacré à faire la lumière sur ce challenge ainsi que les approches de son soutien.

II Chapitre 2 : Haute disponibilité des services Web

« La science ne renverse pas à mesure ses édifices; mais elle y ajoute sans cesse de nouveaux étages et, à mesure qu'elle s'élève davantage, elle aperçoit des horizons plus élargis ».

Marcelin Berthelot

II.1 Introduction

L'utilisation de services Web rend possible la réalisation d'environnements technologiques distribués dans lesquels des applications ou des composantes applicatives peuvent interagir entre elles de façon homogène et ce, de manière indépendante de la plate-forme physique, des langages de programmation ou encore des systèmes d'exploitation.

A cet effet, les services Web sont devenus plus répandus comme technologie de noyau pour implémenter des processus métiers faiblement couplés, où la disponibilité de ces services est très importante (par exemple, les services permettant de contrôler un avion en vol).

En littérature, « il y a une différence énorme entre un système de calcul qui fonctionne et celui qui fonctionne mieux » [25]. Dans un environnement dynamique et ouvert tel que celui d'Internet, il est peu probable que les applications établies autour des composants logiciels tels que les services Web puissent être constamment disponibles en temps d'exécution. De multiples raisons (coupure électrique, surcharge, échec de matériel …) pourraient faire que ces applications s'effondrent, chose qui déclencherait l'exécution des plans correctifs afin de réaliser la continuité de l'opération.

En dépit de la popularité croissante des services Web comme technologie de base pour le développement des applications orientées services, elle reste à amplifier par la prise de contrôle dans des systèmes critiques (par exemple, médical, nucléaire, suivi du trafic aérien) où la disponibilité est un souci central. La disponibilité est un attribut de la fiabilité qui qualifie la promptitude d'une application [4].

Les solutions classiques réalisant la disponibilité sont basées sur la réplique [23,24,34,54] qui est la tendance dans le développement des services Web hautement disponibles ces dernières années. Brièvement, la réplique est la distribution des copies d'une application logicielle (par exemple, service Web) dans un réseau selon une certaine configuration (stratégie), et si l'une des copies défaille, les autres peuvent continuer de fournir le service.

Ce chapitre discourt de la haute disponibilité des services Web et montre que l'approche de réplication a des inconvénients, ce qui la rend très couteuse à utiliser. En revanche, il présente une nouvelle approche proposée par Maamar et al.[58,59] qui prouve comment nous pouvons soutenir la haute disponibilité des services Web par des communautés accueillant des services Web fonctionnellement équivalents au service Web original. La disponibilité des applications

établies autour des services Web n'est alors assurée non pas par des reproductions de service Web mais par une communauté des services Web similaires.

II.2 Définition de la disponibilité

La disponibilité est:

« Un attribut de fiabilité (disponibilité, sécurité, intégrité et maintenance) qualifie la promptitude d'un service. La fiabilité est la capacité d'éviter tous échecs de service inacceptables» [4].

« La capacité de fournir un certain niveau de service pendant une situation où un ou plusieurs composants d'un système ont échoué. L'échec peut être programmé (maintenance planifiée) ou non-programmé (panne). La réalisation d'une haute disponibilité est d'éliminer les seuls points de l'échec se résumant dans l'existence d'une seule source d'une ressource. L'élimination de l'échec exige inévitablement l'approvisionnement par des ressources additionnelles. Le but est que, quand un échec se produit, les utilisateurs peuvent encore accéder au service » [20].

« Le service Web est considéré disponible quand il peut répondre. Par exemple, un service Web donné n'est pas disponible s'il est invoqué et ne renvoie aucune réponse ou renvoie une erreur. La disponibilité de service est donnée par le rapport entre le nombre de fois que le service Web renvoie une réponse et tout le nombre de fois où le service Web est exécuté » [36].

Pour mesurer la disponibilité, on utilise souvent un pourcentage essentiellement composé de '9' (nines notation), par exemple :

- 99% désigne le fait que le service est **indisponible** moins de 3,65 jours par an
- 99,9%, moins de 8,75 heures par an
- 99,99%, moins de 52 minutes par an
- 99,999%, moins de 5,2 minutes par an
- 99,9999%, moins de 31,5 secondes par an
- 99,99999%, moins de 3,1 secondes par an
- etc.

Exemple : des clients utilisent régulièrement le portail du commerce électronique d'une entreprise présent sur la toile dont la panne peut causer une perte de productivité considérable et coûter beaucoup d'argent. Par exemple, eBay a perdu 5 millions de dollars en avril 2002 dus à une panne de serveur qui dura 22 heures; aucun plan approprié de rétablissement d'échec n'était prévu à ce moment-là [50].

Afin de supporter la haute disponibilité des services Web, plusieurs solutions ont été adoptées notamment celles basées sur la réplication.

II.3 Approches de soutien de la haute disponibilité

II.3.1 Approche basée-Réplication

Ces dernières années, la réplication est la technique principale pour fournir la haute disponibilité.

II.3.1.1 Définition de la réplication

La réplication se rapporte à l'utilisation des ressources récurrentes, telles que des composants software ou hardware, pour améliorer la fiabilité, la tolérance aux fautes ou la performance. La réplique implique typiquement[25]:

(i) La réplique dans l'espace dans lequel la même donnée est stockée sur de multiples dispositifs de stockage ou la même tâche de calcul est exécutée sur des dispositifs multiples.
(ii) La réplique dans le temps où une tâche de calcul est exécutée à plusieurs reprises sur un seul dispositif.

Dans notre cas, la réplication consiste à distribuer des copies (reproductions) d'un service Web dans un réseau. Si une reproduction échoue, les autres peuvent continuer de fournir le service.

Les reproductions sont distribuées selon une stratégie spécifique qui indique, par exemple, le nombre de reproductions, leurs endroits appropriés et la manière de sorte à ce qu'elles interagissent l'une sur l'autre avec la copie originale. Il y a trois stratégies différentes de la réplication : réplication active, réplication semi-active et réplication passive [24], visant toutes à garantir une cohérence forte [26] entre les copies d'un composant en vue d'assurer la tolérance aux fautes[26].

II.3.1.2 Stratégie active

Toutes les copies reçoivent la même séquence totalement ordonnée de requêtes des clients, les exécutent de manière déterministe et renvoient la même séquence totalement ordonnée de réponses. Un traitement est déterministe si pour les mêmes données en entrée, il donne toujours le même résultat.

La réplication active se définit comme suit [43]:

1. Toutes les copies reçoivent la même séquence (ensemble totalement ordonné) de requêtes.
2. Toutes les copies traitent les requêtes de manière déterministe.
3. Toutes les copies émettent la même séquence de réponse.

[25] L'encyclopédie en ligne de Wikipedia.

[26] Quelles que soient les précautions prises, l'occurrence de fautes est inévitable (erreur humaine, malveillance, vieillissement du matériel, catastrophe naturelle, etc.). Cela ne veut pas dire qu'il ne faut pas essayer de prévenir ou d'éliminer les fautes, mais les mesures prises peuvent seulement réduire la probabilité de leur occurrence et assurer en même temps la fourniture du service malgré l'occurrence de fautes.

L’avantage principal de la réplication active est sa simplicité (par exemple, le même code partout) et sa transparence pendant l’échec. Si une copie échoue, le reste des copies peut continuer de fournir le service d'une manière transparente pour le client.

Selon le niveau de fiabilité, le client peut reprendre son traitement après obtention de la première réponse d'une reproduction, de la majorité des réponses ou de toutes les réponses.

Mais l’inconvénient majeur de cette stratégie est la contrainte déterministe, c.-à-d. que l’exécution d’une action non-déterministe par toutes les copies aboutit à des résultats différents malgré les mêmes données d’entrée.

II.3.1.3 Stratégie semi-active

Toutes les copies traiteront toutes les requêtes et les actions non-déterministes seront exécutées seulement par une copie désignée comme maître.

Le maître diffuse aux autres ses synchronisations après l’exécution d’une action non-déterministe. Enfin, seul le maître renvoie le résultat aux clients. Le service Web maître est appelé ***leader*** et les autres services Web secondaires sont appelés ***suiveurs.***

La réplication semi-active se décrit comme suit [43]:

1. Toutes les copies reçoivent le même ensemble de requêtes.
2. Toutes les copies traitent toutes les requêtes. Le leader traite une requête dès qu’il la reçoit. Par contre un suiveur doit attendre une notification du leader pour pouvoir traiter une requête. Lorsqu’il y a des sources d’indéterminisme, le leader envoie ses choix aux suiveurs.
3. Le leader est le seul à émettre les réponses au client.

II.3.1.4 Stratégie passive

Cette technique consiste à ce qu’une seule copie reçoive la requête d’un client et l’exécute. Cette copie est désignée sous le nom de copie primaire. Les autres copies sont appelées copies secondaires ou copies de sauvegarde. Les clients soumettent des requêtes à la copie primaire pour les traiter. Et pour assurer la cohérence, cette dernière envoie son nouvel état résultant à toutes les copies secondaires.

Cependant, les copies secondaires, oisives, surveillent la copie primaire. En cas de sa défaillance, l'une d’elles lui succède comme nouveau primaire. Dans ce cas-ci, l’échec n'est pas totalement transparent du point de vue client. C'est que la dernière requête du client nécessite d’être re-soumise (la première requête échouée avant de répondre) et l’exigence d’avoir un mécanisme de mise à jour d’état pour les copies secondaires (la nouvelle copie primaire a pu avoir déjà reçu l'état produit après l’exécution de la requête par la copie primaire échouée).

La réplication passive se résume comme suit [43] :

1. Le client envoie la requête à la copie primaire.
2. La copie primaire exécute la requête.
3. La copie primaire coordonne avec les autres copies en envoyant l'information de mise à jour aux secondaires.
4. La copie primaire envoie la réponse au client.

Cette méthode permet de dupliquer les services Web effectuant des traitements déterministes mais aussi non déterministes car il y a une diffusion de l'état de la copie primaire vers les autres copies et qui est la seule habilitée à exécuter les requêtes des clients.

II.3.1.5 Inconvénients

	Déterminisme du Serveur est nécessaire	**Déterminisme du Serveur n'est pas nécessaire**
Échec du serveur n'est pas transparent pour le client		**Passive**
Échec du serveur est transparent pour le client	**Active**	**Semi-Active**

Fig. II.1 – Stratégies de réplication dans un système distribué

Actuellement, la réplication représente l'approche courante pour réaliser la haute disponibilité des services Web qui préconise l'utilisation des copies qui sont contrôlées selon des stratégies et héritent le jeu quand le service Web original échoue.

Mais d'après ce qui précède, plusieurs problèmes particuliers surgissent dans les stratégies de cette approche synthétisées dans voir Fig. II.1 [34], et d'une manière générale, on peut résumer les inconvénients majeurs de la réplication dans :

- La synchronisation récurrente de tous les états des copies après chaque exécution d'une requête.
- La mise à jour du code de ces copies quand un changement s'effectue.
- Le cout de ces deux opérations.

Le remplacement des copies de services Web avec des services Web pareillement fonctionnels offre des solutions aux inconvénients que comportent ces stratégies. Nous étudierons dans ce mémoire une nouvelle approche de soutien de la haute disponibilité selon les stratégies existantes de réplique et qui doivent être ajustées. Le principe de cette approche est basé sur l'accumulation des services Web qui ont la même fonctionnalité dans une simple communauté.

II.3.2 Approche basée-Communauté

Maamar et al. [58,59] proposent pour soutenir la haute disponibilité des services Web leur propre approche basée sur le concept de communauté des services Web. C'est une approche qui se résume à regrouper plusieurs services Web fournissant une fonctionnalité équivalente comme « HotelBooking[27]» dans une seule communauté, indépendamment de qui les ont développées, leurs localisations et leurs fonctionnements. La notion de communauté a pour but d'apporter une réponse au problème de substitution de service Web. La substitution consiste à remplacer un service Web défaillant qui ne répond plus aux exigences de qualité de service (QoS) requises dans une composition par un autre fonctionnellement similaire tout en satisfaisant les besoins d'une composition des services Web. Nous consacrons cette section pour détailler cette approche de disponibilité basée-communauté. Mais avant cela, la notion de communauté doit être explicitée.

II.3.2.1 Définition de la communauté des services Web

Dans le dictionnaire de Longman[28] :

- « La communauté est un groupe de personnes qui vivent ensemble et/ou sont unies par des intérêts communs, une religion, une nationalité, etc.»

Pour ce qui concerne les services Web :

- **Benatallah et al, 2003**: « définissent la communauté comme une collection de services Web avec une fonctionnalité commune même si ces services Web ont des propriétés non-fonctionnelles distinctes comme différents fournisseurs et différents paramètres de QoS[29] » [7]
- **Medjahed et Bouguettaya, 2005** : emploient la communauté comme un moyen d'offrir une organisation ontologique des services Web ayant le même domaine d'intérêt [9].
- Enfin, selon **Maamar et al. 2007** : La définition d'une communauté dépasse « recueillir les services Web similaires dans une communauté » et considère une communauté « en tant qu'un moyen de fournir une description commune d'une fonctionnalité désirée (par exemple, `FlightBooking`[30]) sans se rapporter explicitement à un quelconque service Web concret (c-à-d. déjà connu, par exemple : `EKFlightBooking`) qui réside dans cette communauté et implémente cette fonctionnalité en temps d'exécution [57].

Exemple: On peut recueillir les services Web Google et Yahoo dans une seule communauté avec une fonctionnalité commune, en l'occurrence "la recherche d'une donnée".

[27] Réservation d'hôtel.
[28] http://www.ldoceonline.com/
[29] Qualité de Service.
[30] Réservation de billets d'avion

II.3.2.2 Architecture

Cette section présente compendieusement les concepts de base de la communauté, à savoir son architecture et ses opérations. Plus de détails sont donnés aux [57,60,12,21].

L'architecture développée par Maamar et al., dans Fig. II.2, illustre les communautés des services Web[31] et la manière dont elles sont structurées et reliées aux fournisseurs des services Web et aux registres UDDI (ou tout autre type de registre telque ebXML).

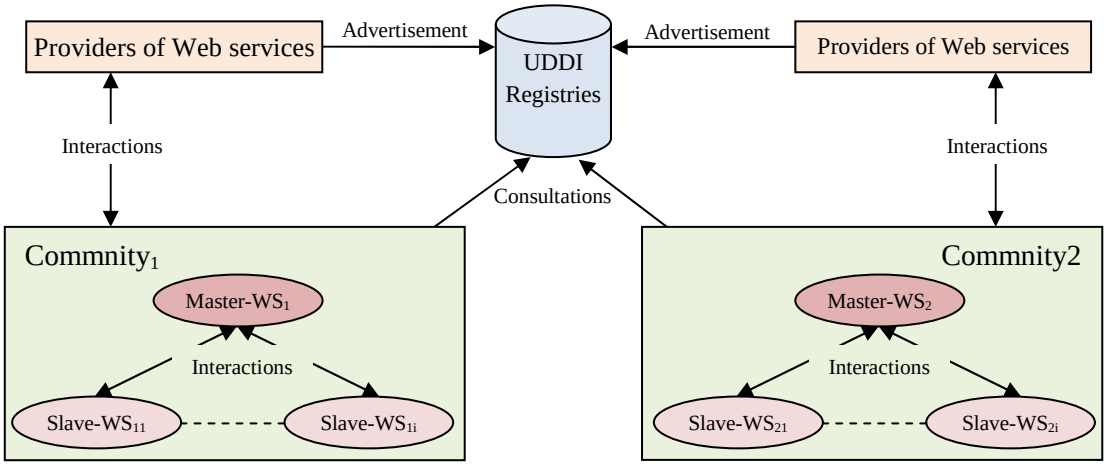


Fig. II.2 – Architecture des communautés des services Web

Dans cette architecture, un service Web spécial promu au grade de Maître (Master) conduit une communauté ; les autres services Web de la communauté étant libellés maintenant comme Esclaves (Slaves)[32]. Les interactions entre Maître et Esclaves sont formulées selon des protocoles spécifiques résumés dans la section II.3.2.3.

Caractéristiques de l'architecture

L'architecture présentée dans Fig. II.1 a des caractéristiques, et dans ce qui suit, on en montre les intéressantes :

- La manière traditionnelle de définir, d'annoncer, et d'invoquer des services Web reste toujours la même alors que ces services Web sont maintenant des éléments des communautés.
- Les fonctionnalités que les registres UDDI offrent habituellement aux fournisseurs et aux utilisateurs des services Web sont toujours les mêmes.
- La sélection des services Web (depuis) des communautés est transparente aux utilisateurs et se produit indépendamment de la manière dont ces services Web sont recueillis dans les communautés.

[31] Les deux communautés montrées dans la figure, elles pourraient par exemple offrir les fonctionnalités de HotelBooking (réservation dans un hôtel) et CarRental (location d'une voiture pendant un séjour), respectivement.
[32] Des architectures internes du Maître et Esclave sont données dans [60].

Responsabilités du Service Web Maître

Parmi les responsabilités du service Web Maître, on cite les suivantes:

- Attirer des services Web pour s'engager dans la communauté qu'il dirige en utilisant des récompenses, tout en vérifiant leurs qualifications (par exemple, QoS) avant l'acceptation définitive. En conséquence, il interagit avec l'annuaire UDDI d'une façon régulière pour connaitre les nouveaux services Web.
- Convaincre des services Web de rester plus longtemps dans sa communauté.
- Nommer les services Web qui participeront aux scénarios de composition des services Web en exécutant le protocole Contrat-Net (voir ci-après la section II.3.2.3), et supporter le fonctionnement de leurs pairs si nécessaire.

Conception du service Web Maître

Dans une communauté, le service Web Maître est désigné de deux manières différentes :

- La première manière est d'avoir un service Web dédié qui jouera le rôle du Maître pendant l'existence d'une communauté. Ce service Web est indépendamment développé (par exemple, par le concepteur d'application) à partir d'autres services Web qui sont publiés dans les registres UDDI. Le service Web menant une communauté ne participe jamais à n'importe quelle composition. Par conséquent, ce service Web est seulement chargé de la gestion de la communauté.
- La deuxième manière est d'identifier un service Web entre les services Web qui peuplent déjà une communauté. Cette identification a pu se produire à titre volontaire ou après élection entre les services Web. Afin de restreindre la non-participation temporaire d'un service Web pour satisfaire une requête d'utilisateur, ce dernier nécessite d'être compensé par d'autres pairs qui le désignent comme Maître. L'appel pour des élections parmi les services Web d'une communauté se produit régulièrement, mais le fardeau sur les mêmes services Web menant une communauté doit d'être réduit au minimum ou évité.

Puisque les services Web de la deuxième manière de désignation du service Web Maître nécessitent d'être augmentés avec des fonctionnalités supplémentaires qui sont spécifiquement consacrées à la gestion de la communauté (sans compter les fonctionnalités offertes par ces services Web), il est convenable de développer un service Web indépendant pour jouer le rôle du Maître (selon la première manière).

II.3.2.3 Fonctionnement

Une communauté a une nature dynamique ; elle est établie et démantelée suivant des scénarios spécifiques et les services Web entrent et partent à leur convenance.

Le fonctionnement d'une communauté est divisé en quatre parties [59,57,21] comme montré dans Fig. II.3: gestion de la communauté, attraction et rétention des services Web, sélection des services Web pour participer à une composition et orientation des stratégies de la réplique pour

la haute disponibilité[59].

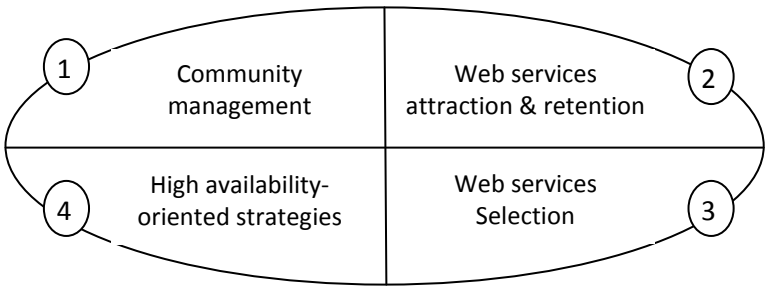


Fig. II.3 – Fonctionnement de communauté des services Web

Gestion de la communauté

Deux opérations essentielles gèrent une communauté:

a) **Etablissement** : Une communauté des services Web est établie essentiellement pour assembler des services Web qui ont la même fonctionnalité. Cet établissement se produit en deux étapes comme suit:

- Premièrement, le concepteur définit la fonctionnalité de la communauté (par exemple, FlightBooking), en la liant à une ontologie spécifique. Cette liaison est cruciale puisque les fournisseurs emploient différentes terminologies pour décrire la fonctionnalité de leurs services Web. Par exemple, « FlightBooking », « FlightReservation » et « AirTicketBooking » ont une fonctionnalité identique qui est bien la réservation de vol.

- Deuxièmement, il déploie le service Web Maître qui mènera cette communauté et qui assure ses responsabilités comme rapporté plus haut.

b) **Démantèlement** : Le démantèlement ou le désassemblage d'une communauté est une autre activité du service Web Maître. Ce dernier surveille tous les évènements qui se déroulent dans sa communauté telle que l'arrivée de nouveaux services Web, le départ des services Web existants, la sélection des services Web pour joindre des compositions des services Web, et la sanction des services Web en cas de mauvais comportement.

Si le service Web Maître remarque que :

- Le nombre de services Web dans la communauté est inférieur à un certain seuil déterminé, et que
- Le nombre des requêtes de participation dans des scénarios de compositions provenant des utilisateurs sur une certaine période de temps est inférieur à un autre seuil prédéfini,

Alors la communauté doit être démantelée. Les deux seuils sont déterminés par le concepteur. Les services Web éjectés d'une communauté peuvent rejoindre d'autres communautés (ou en créer une nouvelle) tout en évaluant la similarité des fonctionnalités

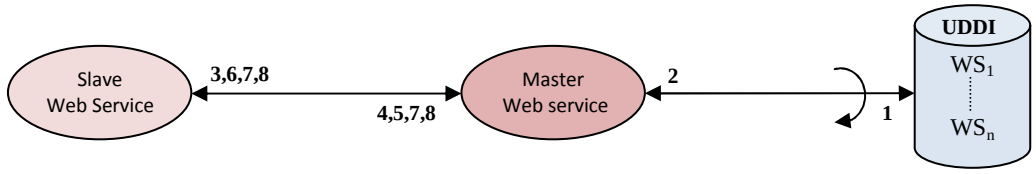
entre ces services Web et les communautés existantes.

Le tableau suivant résume les rôles des seuils [60].

Nombre des services Web dans la(les)		Commentaire	Action recommandée
Communauté	**Compositions**		
Petit	Grand	Configuration efficiente	Maintenir les services Web et en inviter de nouveaux
Grand	Petit	Configuration pauvre	Éjecter les services Web dont la participation baisse et en inviter de nouveaux
Petit	Petit	Configuration très pauvre	Démanteler la communauté
Grand	Grand	Configuration désirée	Maintenir la même stratégie

Hamdi et al. [55] a montré que la réputation[33] entre aussi comme argument dans l'établissement ou le démantèlement d'une communauté. Une bonne réputation d'une communauté peut jouer le rôle d'un facteur motivant les services Web pour faire partie d'elle, comme une mauvaise réputation aurait pu être un argument valide pour désassembler la communauté.

Attraction et rétention des services Web



Chronologie

1. Requête de consultation (opération continué) ; 2. URL de WSDL ;
3/4. Requête/réponse d'adhésion à la communauté ; 5/6. Requête/réponse d'enregistrement du WS ;
7/8. Requête/réponse de retrait du WS

Fig. II.4 – Attraction, rétention et éjection des services Web

L'attraction de nouveaux services Web et la rétention de ceux déjà existants dans la communauté sont de la responsabilité du service Web Maître. Une communauté pourrait disparaître si le nombre de services Web qui résident en elle chute au-dessous d'un certain seuil.

a) **Attraction des services Web :** fait que le service Web Maître consulte régulièrement les registres UDDI en recherchant de nouveaux services Web. Quand un service Web candidat est identifié selon sa fonctionnalité, le service Web Maître entre en interaction avec son fournisseur. Le but des interactions est de demander au

[33] La réputation est un processus continu qui est fondé sur plusieurs facteurs comme la satisfaction, la fiabilité, l'efficacité, et les expériences antérieures [55].

fournisseur d'enregistrer son service Web dans la communauté de ce service Web Maître (opérations 1 à 6 de Fig. II.4). Quelques arguments à employer pendant les interactions incluent le taux élevé de participation des services Web existants dans les compositions, et l'efficience de ces services Web dans le traitement des requêtes d'utilisateurs.

b) Rétention des services Web : maintenir des services Web dans une communauté pendant une longue période de temps est un bon indicateur de ce qui suit :

- Les services Web dans une communauté sont en concurrence, ils exposent une attitude coopérative[34],
- Les services Web (fournisseur) sont satisfaits de leur taux de participation dans les compositions.

c) Ejection des services Web : l'attraction et la rétention de services Web mène à un troisième scénario concernant les services Web auquel une demande de quitter la communauté a été formulée. Un service Web Maître doit répondre à une telle requête après avoir évalué les critères suivants :

- Le service Web a une nouvelle fonctionnalité qui ne concorde pas parfaitement avec celle de la communauté,
- Le service Web est peu ou non fiable.

Dans les différentes occasions, ce service Web ne réussit pas à participer dans des compositions à cause de problèmes opérationnels récurrents ou d'une faible qualité de service (opérations 7 et 8 de Fig. II.4).

Sélection des services Web pour participer à une composition

Dans une communauté, les interactions entre le service Web Maître et les services Web Esclaves concernant la participation aux compositions sont encadrées conformément au protocole Contrat-Net (CN) [41]. Le but principal de ces interactions est d'identifier le service Web Esclave qui participera à une composition des services Web. Ce protocole est établi sur l'idée de contracter et de sous-traiter les travaux entre deux types d'agents connus sous le nom « Initiateur et Participant ». A tout moment un agent peut être initiateur, participant, ou les deux.

L'ordre des étapes dans le protocole Contrat-Net est comme suit :

(i) Initiateur envoie un appel d'offres aux participants qui peuvent réaliser certains travaux;
(ii) Chaque participant passe en revue l'appel d'offre et soumissionner si cela s'intéresse (c.-à-d., le travail est faisable) ;
(iii) Initiateur choisit la meilleure offre et attribuera un contrat à ce participant ;
(iv) Enfin, l'initiateur rejette les autres offres.

[34] Voir [56] pour un exemple de « comportement coopératif » quand les services Web supportent les uns les autres lors d'un cas de substitution.

L'application du protocole CN à une communauté des services Web se produit comme suit : Quand un utilisateur choisit une communauté en raison de sa fonctionnalité qui satisfait ses besoins, son service Web Maître respectif entre en contact avec lui. Le but est d'identifier un service Web Esclave spécifique, qui sera responsable de mettre en application cette fonctionnalité (Fig. II.5).

(i) Le service Web Maître envoie un appel d'offres à tous les services Web Esclaves pour l'implémentation de la fonctionnalité.
(ii) Les services Web Esclaves évaluent leurs statuts et leurs disponibilités pour exécuter la requête du service Web Maître, où les services Web intéressés répondent à l'appel.
(iii) Le service Web Maître examine toutes les offres avant de choisir le meilleur service Web Esclave selon ses préférences (QoS, disponibilité, coût, équité…), ensuite Il notifie le service gagnant.
(iv) Le reste des services Web Esclaves qui ont montré un intérêt mais n'ont pas été choisis, sont également notifiés.

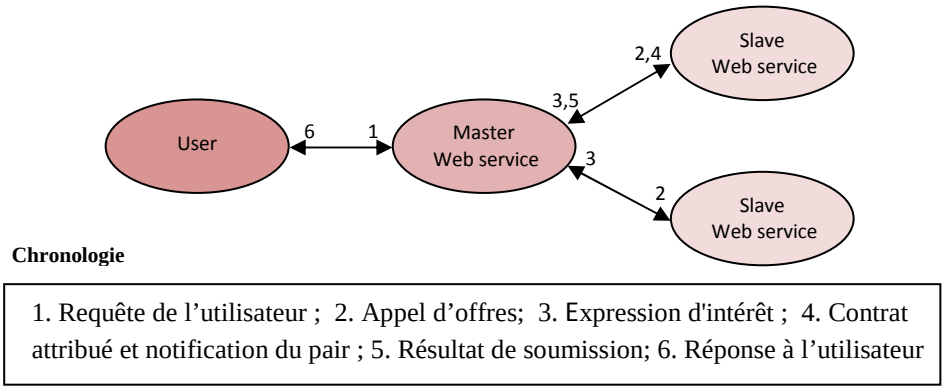


Fig. II.5 – Interactions suivant le Protocole Contract-Net

Orientation des Stratégies de réplique pour la haute disponibilité

Nous allons voir plus tard que les services Esclaves non sélectionnés (selon CN_{etape4}) ont un rôle important en soutenant le fonctionnement approprié du service Web Esclave choisi pour satisfaire à un besoin de l'utilisateur. Ces services Web Esclaves non sélectionnés agiront en tant que reproductions dans différentes stratégies de réplique.

II.3.2.4 Soutien de la haute disponibilité en utilisant les communautés

Dans la section II.3.1, nous avons présenté l'approche de la réplique comme une solution habituelle pour assurer la disponibilité des applications. Pour les applications établies autour des services Web, Maamar et al. proposent une autre approche [58,59] de soutien de la disponibilité des services Web qui consiste à substituer les reproductions à l'aide d'une simple communauté des services Web, où ils montreront que (i) la cohérence entre les états des reproductions et (ii) la mise à jour des codes des reproductions en cas de changement, n'ont pas une grande importance dans l'approche basée-communauté. Avant de monter comment une communauté soutient la haute disponibilité des services Web selon les stratégies de réplique existantes, nous détaillerons ci-après quelques points à employer dans ces stratégies.

Types et détection des échecs

Le développement de toute approche qui soutient la haute disponibilité des composants logiciels comme les services Web, dépend étroitement de deux facteurs. Le premier facteur concerne les types d'échec qui empêcheraient un composant logiciel de continuer ses opérations, et qui exige de prendre les mesures nécessaires en réponse à ces échecs. Le deuxième facteur concerne les mécanismes pour détecter qu'un composant logiciel est en difficulté, et qui exige aussi de prendre des mesures appropriées en réponse à ces échecs. Dans ce sens, Maamar et al. ont mis les types d'échec et les facteurs de leur détection dans le cadre de leur approche de la haute disponibilité basée-communauté [59].

Types d'échec

Plusieurs types d'échecs existent s'étendant à des accidents où un service Web cesse de fonctionner ou fournit des résultats incorrects. Certains des types les plus communs d'échecs sont énumérés dedans [4]. Parmi eux nous citerons:

- **L'échec de Contenu**: se produit quand la sortie d'un service Web dévie de la fonction supposée qu'elle implémente.
- **L'échec de Timing (tôt ou tard):** se produit quand un résultat de service Web n'est pas produit dans un intervalle de temps spécifique.
- **L'échec Silencieux** : est de type Contenu et Timing, se produit quand un service Web cesse de fonctionner et ne produit aucune sortie ou continue de produire des mêmes sorties quelque soient les entrées.

Détection d'échec

Traitement des échecs soulève la question de comment savoir que des échecs se sont produits. Dans la littérature, plusieurs façons de détecter les échecs sont identifiées [26] comme test d'acceptation, le chien de garde (Watchdog), le Temps mort (Timeout), et le Vote (Voting). Parmi elles, deux approches ont pu s'adapter avec les services Web :

- **Watchdog**: Les services Web génèrent des signaux réguliers que les composants spéciaux connus sous le nom de Watchdog reçoivent automatiquement et « consomment » tant qu'ils sont vivants. Le manque de signaux en une certaine période indique l'occurrence d'échec potentiel.
- **Temps mort (Timeout)**: les utilisateurs des services Web réagissent au cas où les réponses ne seraient pas fournies dans le délai de temps convenu.

II.3.2.5 Fondements de déploiement des stratégies de la réplique

Les aspects suivants sont associés à l'approche de disponibilité basée-communauté indépendamment de la stratégie de réplique utilisée (active, passive, ou hybride). Quelques aspects sont intrinsèques à la définition et le fonctionnement des services Web, alors que d'autres sont intrinsèques aux structures et opérations des communautés (section II.3.2).

Flux de contrôle et opérationnel

En utilisant les diagrammes d'état [15], nous spécifions un service Web par l'utilisation de deux types de flux [58,59] : de contrôle et opérationnel (Fig. II.6). Les deux flux interagissent entre eux (Fig. II.7).

- **Le flux de contrôle** : décrit la logique d'affaire "**business logic**"[35] qui implémente la fonctionnalité d'un service Web. Cette description dépend strictement de la façon de développement des services Web qui changent d'un domaine d'application à un autre.

- **Le flux opérationnel** : guide la progression d'exécution de flux de contrôle d'un service Web. A cet effet, un ensemble d'états établis dans le flux opérationnel sont employés, à savoir : *activated, done et suspended*. Ces états sont communs à tous les services Web indépendamment de leurs types, fonctionnalités, origines, et endroits.

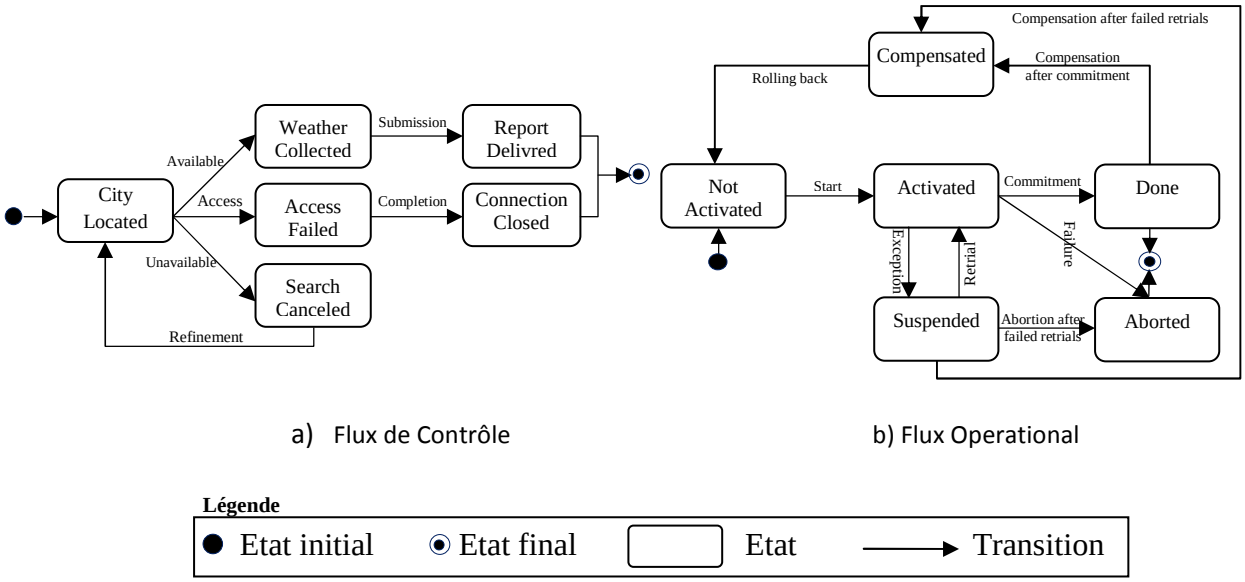
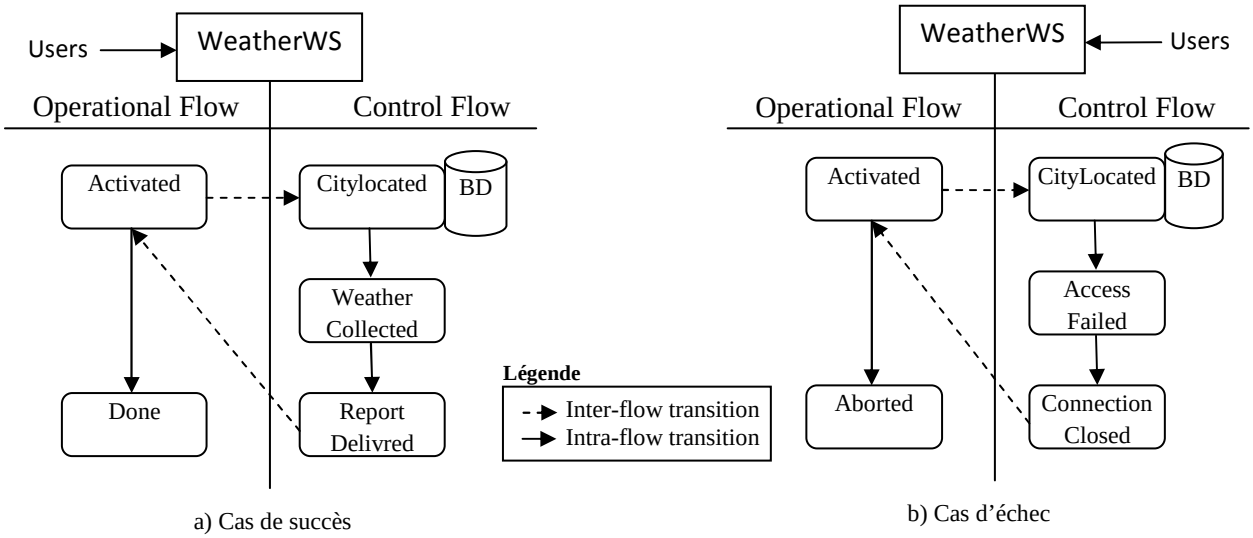


Fig. II.6 – Flux de contrôle et opérationnel du service Web « WeatherWS »

A cause de la synchronisation des états de quelques stratégies de l'approche réplication (la synchronisation doit se produire continûment), les deux flux (de contrôle et opérationnel) devront être synchronisés puisque toutes les reproductions sont des copies de l'application originale et ainsi, ont les mêmes flux de contrôle et opérationnel. Par contre, dans l'approche de disponibilité basée-communauté, la synchronisation concerne seulement le flux opérationnel et pas le flux de contrôle.

[35] Chaque service Web a une logique d'affaire qui est un processus d'un ensemble d'actions à exécuter.



a) Cas de succès

b) Cas d'échec

Fig. II.7 – Interactions entre les flux de contrôle et opérationnel du « WeatherWS »

Pour illustrer l'utilisation des flux de contrôle et opérationnel et comment ils interagissent ensemble, on prend l'exemple du service Web de « *WeatherWS* ». Sa fonctionnalité est de retourner les prévisions météorologiques de cinq (05) jours pour une certaine ville. Quelques états dans le flux de contrôle de *WeatherWS* sont *CityLocated*, *WeatherCollected*, et *ReportDelivered* (Fig. II.6 (a)), et quelques états dans le flux opérationnel de *WeatherWS* à savoir *Activated*, *Compensated*, et *Done* (Fig. II.6 (b)). Il convient de noter que le flux de contrôle est strictement spécifique à *WeatherWS*. Cependant, d'autres services Web dans la communauté où *WeatherWS* réside offrent la même fonctionnalité que *WeatherWS*. Ces services Web ont différents flux de contrôle en termes d'actions à prendre et leur chronologie.

Après cette présentation des flux contrôle et opérationnel de *WeatherWS*, nous allons voir maintenant comment ces deux flux interagissent entre eux en utilisant deux cas. Deux types de transitions sont montrés dans Fig. II.7 : intra-flux (ligne continue) et inter-flux (ligne discontinue)[36].

Dans Fig. II.7 le déclenchement de *WeatherWS* est indiqué par l'état *Activated* dans le flux opérationnel. Dans cet état, *WeatherWS* prend la requête de l'utilisateur, et à cause de la transition inter-flux *(Activated, CityLocated), WeatherWS* commence à chercher la ville demandée en employant une base de données dédiée. Ceci fait maintenant *WeatherWS* prend l'état *CityLocated* dans le flux de contrôle. Après l'exécution des actions nécessaires dans cet état, deux cas sont enregistrés :

- **Cas a** : tout marche bien et le rapport des prévisions météorologiques de 05 jours est délivré à l'utilisateur. Et avec la transition inter-flux *(ReportDelivered, activated), WeatherWS* accomplit son opération avec succès par une transition de l'état *Activated* à l'état *Done* dans le flux opérationnel.

- **Cas b :** se produirait au contraire du cas a, l'accès à la base de données échoue. Ceci est indiqué dans le flux de contrôle de *WeatherWS* par les états *: AccessFailed* et

[36] Pour des raisons de simplicité, les noms des transitions entre les états sont omis.

ConnectionClosed. Par la transition inter-flux *(ConnectionClosed, Activated)*, *WeatherWS* termine son opération avec échec par une transition de l'état *Activated* à l'état *Aborted* dans le flux opérationnel.

Fig. II.7 a) et b) illustrent comment des transitions d'état dans le flux opérationnel d'un service Web sont affectées par les états que ce service Web prend dans le flux de contrôle. Par exemple, les transitions *WeatherWS* de l'état « *activated* » à l'état « *Done* » ou « *Aborted* » basé sur les résultats d'exécution obtenus après le déclenchement du flux de contrôle en utilisant les états *ReportDelivered* ou *ConnectionClosed*. L'état à prendre dans le flux opérationnel dépend des transitions inter-flux.

Service Web Esclave de support (Backup)

Conformément au protocole Contrat-Net [57,32,33], tous les services Web Esclaves qui ont positivement répondu à l'appel d'offres du service Web Maître reçoivent une notification de leur non-sélection (excepté un seul à qui la sélection est notifiée). Le service Web Esclave sélectionné prend le rôle du primaire, signifiant qu'il exécute la fonctionnalité dont l'utilisateur a besoin. Dans l'approche de soutien de la disponibilité basée-communauté, une notification supplémentaire (notification de non-sélection) est appropriée pour supporter, dans les compositions et en temps d'exécution, la haute disponibilité du service Web Esclave sélectionné (qui a gagné l'offre), c.-à-d. que sa participation à une composition est confirmée. La notification supplémentaire a pour but d'informer les services Web Esclaves non sélectionnés de leur participation potentielle à cette composition comme des supports pour le service Web Esclave primaire en temps d'exécution. Cette notification est importante pour les prochaines opérations dans une communauté. En effet, elle permet aux services Web Esclaves d'avoir deux choix pour répondre au service Web Maître : accepter ou refuser de prendre le rôle de support en évaluant leurs futurs engagements dans d'autres compositions quand les prochains appels d'offres du service Web Maître arrivent.

- **En cas d'acceptation**, si le service Web Esclave primaire échouait (cessait de fonctionner ou fournissait des résultats incorrects), les services Web Esclaves de support seraient tous des candidats potentiels pour jouer le rôle de primaire afin d'assurer l'exécution continue de la fonctionnalité. Seulement un seul service Web Esclave de support sera sélectionné. Il s'appellera **service Web Esclave de support-primaire** (Primary Backup Slave Web service) qui a été au commencement choisi comme Esclave de support et maintenant comme Esclave primaire. Il convient de noter que les services Web Esclaves de support ne doivent pas intervenir (c.-à-d., être activés) si le fonctionnement du service Web primaire se produit sans échec[37]. Ces services Web pourraient avoir manqué quelques occasions de participation à d'autres compositions.
- **En cas de rejet**, le service Web Maître pourrait adopter une stratégie différente qui se propose de convaincre certains des services Web Esclaves non sélectionnés d'agir en tant que support en utilisant les récompenses [21].

[37] Une question de recherche intéressante ici, est comment récompenser les services Web Esclaves quand ils n'interviennent pas.

II.3.2.6 Ajustement des stratégies de réplication

L'approche de la haute disponibilité basée-communauté propose l'ajustement des stratégies de la réplique (active, passive et hybride) examinées dans la section II.3.1, selon ses caractéristiques [58,59]. Cette section récapitule ces ajustements en se basant sur le travail de He dans [50].

Stratégie active

Connue par réplication machine, elle est non-centralisée et exige que toutes les reproductions[38] reçoivent et traitent dans le même ordre les requêtes d'utilisateur. La cohérence est garantie étant donné qu'elles produisent avec la même séquence des requêtes les mêmes résultats. Cette supposition signifie que les reproductions traitent les requêtes d'une manière déterministe. Les clients n'entrent pas en contact avec une reproduction particulière, mais adressent toutes les reproductions en tant que groupe. Le privilège majeur de la stratégie active est sa simplicité et sa transparence pendant l'échec. Nous nous rappelons que le service Web Maître fournit la liste de services Web Esclaves de support au service Web Esclave primaire et tous ces services exécutent concurremment la requête d'utilisateur. Si le service Web primaire échoue, les autres services de support continuent de traiter la requête. En outre l'un d'entre eux doit être choisi maintenant pour agir en tant que primaire (support-primaire) de sorte qu'il puisse fournir des réponses à l'utilisateur. Les réponses retournées par les services Web peuvent être différentes; c'est le problème du déterminisme qui est l'obstacle capital devant la stratégie active de réplication [50].

Fig. II.8 emploie WeatherWS pour illustrer les interactions dans la stratégie active de la réplication. Dans cette figure, les lignes continues correspondent aux interactions qui ont lieu entre les utilisateurs et les services Web et entre les flux opérationnel et de contrôle de WeatherWS (primaire et de support). Les lignes discontinues correspondent aux interactions qui ont lieu entre les flux opérationnels de WeatherWS et de WeatherWSs[39] de support. La ligne discontinue (1) illustre WeatherWS qui envoie la requête d'utilisateur à ses supports et lance leur exécution. Nous rappelons que le service Web Maître fournit la liste de services Web esclaves de support au service Web esclave primaire. Après, tout les WeatherWSs traitent concurremment cette requête. Si le WeatherWS primaire échoue, c.-à-d. prend l'état "Failure" dans son comportement opérationnel respectif (Fig. II.6 (b)), les autres services de Web esclaves de support continuent de fonctionner. En outre, l'un d'entre eux doit d'être choisi maintenant pour agir en tant que primaire de sorte qu'il puisse fournir des réponses à l'utilisateur. Nous rappelons aussi que WeatherWSs retournent des réponses différentes puisque la contrainte déterministe est automatiquement détendue.

Dans l'approche de disponibilité basée-communauté, l'application de la stratégie active aux services Web soulève les questions suivantes (Qi):

[38] Les reproductions sont tous les services Web Esclaves (primaire et services de support) qui soutiennent le fonctionnement du primaire.
[39] WeatherWSs sont tous les services Web qui supportent le service Web WeatherWS.

- Q1 : Comment lancer l'exécution de tous les services Web de support puisque le service Web Maître ne les a pas sélectionnés pour satisfaire la requête de l'utilisateur.

- Q2 : Comment la synchronisation d'état et la maintenance (mise à jour en cas de changement du code) sont réalisées entre tous les services Web de support.

- Q3 : Quand un service Web primaire échoue, quelle est la technique qui devrait être utilisée pour choisir le futur service Web primaire et qui doit fournir des résultats.

- Q4 : Quels sont les endroits appropriés pour positionner les services Web de support dans un réseau.

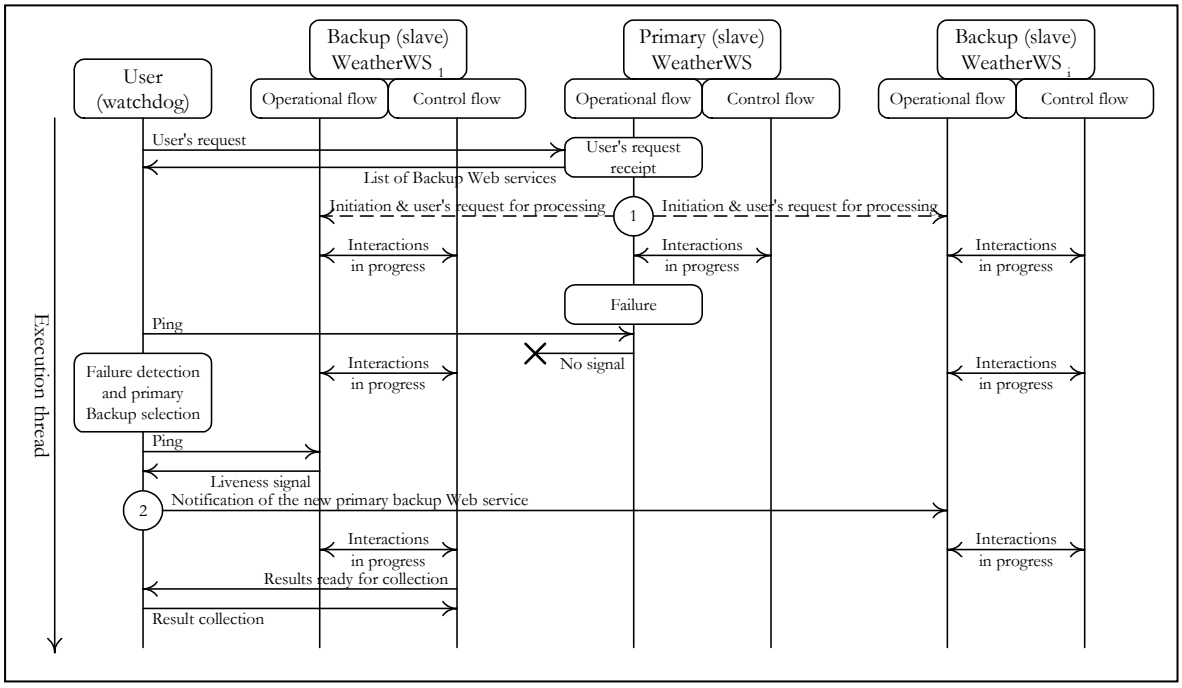


Fig. II.8 – Interactions suivant la stratégie active de la réplication

L'adoption de l'approche de disponibilité basée-communauté fournit des réponses (Ri) à ces questions [58,59].

- R1 à Q1: Puisque toutes les reproductions doivent fonctionner simultanément, le service Web Esclave choisi (c.-à-d., primaire) qui traitera la requête de l'utilisateur, doit (i) transmettre cette requête aux autres services Web qui supportent son fonctionnement et (ii) lancer leurs exécutions.

- R2 à Q2 : La synchronisation d'état entre les services Web de support peut se produire uniquement au niveau du flux opérationnel, quoiqu'elle ne soit pas obligatoire à cause des différences qui existent entre les flux de contrôle. Ceci exclut les flux de contrôle de la synchronisation d'état. En outre, il n'y a pas besoin de maintenir le code entre les services Web de support. Chaque service Web de support a son propre code pour implémenter sa logique d'affaire et qui est différent des codes des autres.

- R3 à Q3 : Comme les services Web de support peuvent probablement ne pas renvoyer les mêmes réponses bien qu'ils reçoivent les mêmes requêtes d'utilisateurs, la contrainte déterministe est automatiquement détendue. Les services Web sont indépendants entre eux; il est improbable qu'ils accèdent aux mêmes sources d'information et traitent les requêtes d'utilisateurs de la même manière. Il convient de noter que les réponses retournées par les services Web de support peuvent être semblables, partiellement différentes (chevauchement), ou totalement différentes.

  Une technique bien connue pour choisir une reproduction qui fournira des résultats de nouveau à l'utilisateur est le vote [38]. Le vote apparait sous différentes formes, dans l'une d'entre elles est le vote de majorité où les résultats à employer doivent être « suffisamment égaux ». Des reproductions produisant les résultats qui diffèrent de la majorité sont habituellement considérées en tant qu'échouées et seront masquées. La stratégie de majorité échoue s'il n'y a aucun consensus ou les résultats convenus sont incorrects.

- R4 à Q4 : La question d'endroit ne se pose pas puisque les endroits des services Web de support sont initialement adoptés par défaut.

Stratégie passive

Connue également par réplication support-primaire (primary backup), elle suggère que les utilisateurs soumettent leurs requêtes directement à une application spécifique (habituellement la copie originale), qui est le support-primaire. Cette application traite les requêtes d'utilisateurs et envoie régulièrement des requêtes de mise à jour aux autres services Web de support. Ces derniers sont en attente pour recevoir des requêtes de mise à jour, les mettre en application afin de synchroniser leurs comportements respectifs avec le comportement d'application originale. Dans la stratégie passive, il est obligatoire de garantir l'ordre d'exécution des requêtes de mise à jour et d'assurer la continuité des opérations au cas où l'application primaire échouerait en dirigeant le fil d'exécution vers un des supports. En raison de la distance dans le statut qui existe entre le support-primaire et le reste des supports, la réplique passive souffre d'une reconfiguration élevée, comme coût, quand les échecs surgissent.

L'adoption de la stratégie passive dans l'approche de la haute disponibilité basée-communauté se produit comme suit. Tout d'abord, le service Web Esclave désigné par le service Web Maître pour satisfaire une demande sera le support-primaire. Il satisfait la requête d'utilisateur et transmet des requêtes de mise à jour aux autres services Web de support (ceux qui ont confirmé joueront le rôle de support) dans sa communauté. Ces requêtes de mise à jour sont limitées seulement aux flux opérationnels de ces services Web de support. Puisque le flux opérationnel agit comme un contrôleur sur le flux de contrôle, n'importe quelle mise à jour dans ce flux opérationnel nécessite automatiquement le reflet sur le flux de contrôle (Fig. II.7).

Cependant, les différences dans le flux de contrôle qui existent entre le service Web de support-primaire et les autres services Web de support empêchent ce reflet de se produire.

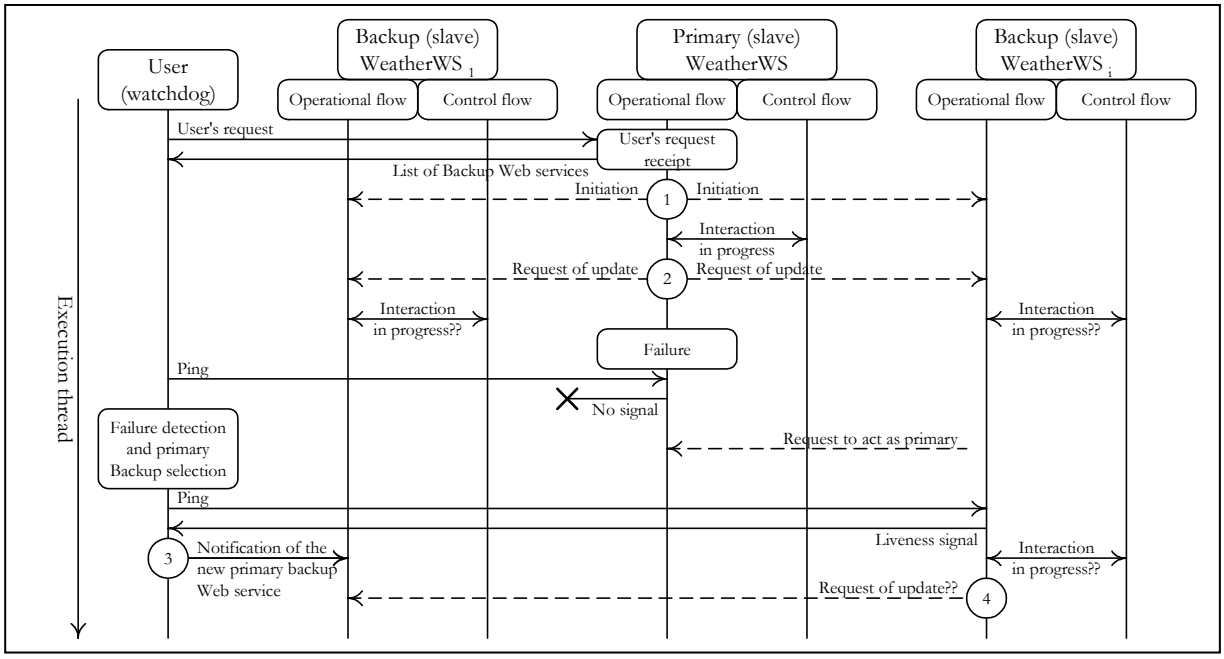


Fig. II.9 – Interactions suivant la stratégie passive de la réplication

L'inaptitude de la stratégie passive dans le cadre de l'approche de la haute disponibilité basée-communauté est due à la difficulté de mettre à jour les états de tous les services Web où les requêtes de mise à jour ont seulement une influence sur le flux opérationnel sans le flux de contrôle.

Fig. II.9 emploie encore WeatherWS pour illustrer les interactions dans la stratégie passive de la réplication. Dans cette figure, les lignes continues correspondent aux interactions qui ont lieu entre les utilisateurs et les services Web et entre les flux opérationnel et de contrôle de WeatherWS (primaire et de support). Les lignes discontinues correspondent aux interactions qui ont lieu entre les flux opérationnels de WeatherWS et de WeatherWSs de support. La ligne discontinue (1) illustre le WeatherWS support-primaire soumettant initialement les requêtes à ses supports avant qu'il commence à traiter la requête d'utilisateur. Dans la ligne discontinue (2), le service Web primaire invite les services Web de support à mettre à jour leurs flux opérationnels respectifs, ce qui signifie aussi bien de mettre à jour leurs flux de contrôle respectifs. Cette mise à jour est montrée avec la ligne continue **« interaction in progress ?? »**. Le symbole ?? souligne l'impossibilité de mettre à jour les flux de contrôle de ces services Web de support. Ce premier exemple montre l'inaptitude de la stratégie passive. En effet, les actions que le service Web primaire effectue n'accomplissent pas les actions que les services Web de support doivent aussi bien accomplir. Le deuxième exemple se produit quand le service Web primaire WeatherWS échoue, c.-à-d. prend l'état "Failure" dans son comportement opérationnel respectif (Fig. II.6 (b)). En conséquence et comme exposé par la ligne discontinue (3), l'un des services Web de support prend la relève dans le processus d'exécution et devient le nouveau support primaire. Cependant, parce que les flux de contrôle dans le support-primaire échoué et dans le nouveau support-primaire WeatherWS sont différents, ceci empêche la mise à jour du flux de contrôle de ce nouveau support primaire WeatherWS et ainsi, l'impossibilité de continuer le traitement de la requête d'utilisateur (ligne discontinue (4)). Ces deux exemples illustrent pourquoi la stratégie passive de la réplication est peu convenable pour les services Web dans le cadre de l'approche de la haute disponible basée communauté.

Stratégie semi-active (ou hybride)

La réplication hybride manipule la contrainte déterministe de la réplication active en adressant toutes les actions indéterministes[40] à un composant spécifique appelé *leader*. Le Leader les exécute et envoie ses résultats à tous les autres services Web suiveurs pour les mettre à jour. Dans l'approche de la disponibilité basée-communauté, l'indéterminisme existe par défaut car tous les services Web renvoient des résultats différents. En effet, elle est comme la stratégie de passive, où la mise à jour des états entre tous les services Web est impossible, la stratégie de réplique hybride n'est pas appropriée aux services Web dans le contexte de l'approche de la disponibilité basée-communauté.

D'après ce qui précède, on constate que l'approche de disponibilité basée-communauté s'adapte uniquement [58,59] avec la réplication active et qui doit être ajustée par cette approche.

II.4 Conclusion

Les services Web seront dans un avenir proche la technologie choisie pour le développement des missions critiques et pour les applications économiques légèrement couplées où la disponibilité de ces services sera indispensable. Dans ce chapitre, nous avons examiné la haute disponibilité des services Web en présentant une nouvelle approche de soutien de cette dernière qui tourne principalement autour du concept de la communauté. La disponibilité est la proportion en temps qu'un service Web est fonctionnel. Et la communauté est une structure recueillant des services Web fonctionnellement semblables, de sorte que ces services Web devraient pouvoir se remplacer l'un l'autre, en cas de panne, en temps d'exécution pour assurer la disponibilité du service.

Les anciennes approches de la haute disponibilité emploient des reproductions du service Web original pour le soutenir en cas d'échec en temps d'exécution. Ces reproductions fonctionnent suivant des stratégies de réplication à savoir active, passive et hybride pour fournir les réponses. Malheureusement, ces stratégies connaissent des inconvénients et sont spécifiquement adaptées pour les cas où les reproductions sont les copies exactes du service Web original.

L'ajustement des stratégies de réplication active, passive et hybride dans le cadre de la nouvelle approche de disponibilité qui est basée sur les communautés a seulement montré la convenance de la première stratégie à cause de la nature du fonctionnement des services Web en utilisant les flux de contrôle et opérationnel. Dans la réplique active, nous avons identifié le service Web primaire, les services Web de support et le service Web de support-primaire. Ils désignent, respectivement, le service Web sélectionné qui implémente la fonctionnalité requise pour satisfaire une requête d'utilisateur, l'ensemble de services Web qui appuient ce service Web choisi et le service Web qui soutient concrètement ce service Web choisi quand il échoue.

Par l'utilisation des communautés, plusieurs avantages immédiats sont obtenus. Par exemple, la gestion du code entre les reproductions des services Web en cas de changements n'est pas

[40] Le change de devise est indéterministe à cause des fluctuations en taux de change. La soumission d'une somme d'argent semblable à une telle application renvoie chaque fois presque une valeur différente.

nécessaire, puisque ces reproductions ont des codes différents (mais suggèrent toujours la même fonctionnalité). Un deuxième avantage est la possibilité d'interviewer les communautés en recherchant des services Web similaires au lieu des registres traditionnels tels que UDDI.

Cette approche qui est basée sur la substitution du service Web défaillant peut provoquer des hétérogénéités sémantiques dans une composition où l'on doit les régler par une des techniques de médiation sémantique comme on l'a vu dans le premier chapitre. Avant cela et dans le chapitre qui suit, nous allons détailler une technique de médiation par la présentation d'une approche de médiation sémantique orientée contexte, pour résoudre les conflits sémantiques entre les services Web.

III Chapitre 3 : Médiation sémantique entre les services Web

« La coopération n'est pas l'absence de conflits, mais un moyen de les gérer ».

Deborah Tannen

III.1 Introduction

Dans le premier chapitre l'on a vu la nécessité d'effectuer la médiation sémantique dans une composition afin de résoudre toutes les hétérogénéités pouvant se trouver entre ses services Web dont chacun à une représentation sémantique spécifique différente des autres. Or, les solutions actuelles se basent sur les ontologies comme un moyen d'opérer cette médiation.

Ce chapitre est considéré comme un état de l'art des travaux de Mrissa et al. dans la médiation sémantique basée sur la notion de contexte et ce, en vue d'une représentation sémantique des données permettant leur échange correct entre les services Web.

Nous motivons, d'abord, le besoin de médiation sémantique des données à l'aide d'un scénario de planification de voyage en ligne en montrant les hétérogénéités sémantiques, ensuite et avant de détailler l'approche de médiation adoptée comme solution pour résoudre ces hétérogénéités que ce soit au niveau de composition ou celui de communauté, nous exposerons le modèle contextuel de description des données sur lequel cette médiation repose.

III.2 Scénario d'illustration

Afin d'illustrer l'importance de la médiation sémantique, nous préférons prendre comme scénario de composition l'exemple mentionné dans [31] planifiant un voyage en ligne au Japon, en l'adaptant avec quelques spécificités relatives au sujet de ce mémoire.

III.2.1 Spécification de composition de planification de voyage

La spécification de notre composition de planification de voyage comprend trois services voir le schéma Fig.III.1 où l'on exhibe juste quelques paramètres d'entrée et de sortie des services aidant à la compréhension de notre travail[41].

- Un service Web de réservation de billets d'avion **FlightBooking**, qui est un service européen traitant les prix en « **Euro** »(devise désignée par EUR) avec une **TVA incluse** d'un taux de « **19,6%**», un facteur multiplicateur[42] de « **1** » et dont la date est de format « **jj/mm/aaaa** ».

[41] Il y a d'autres paramètres comme: villes de départ et d'arrivée, nombre de places, classe,….

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

- Un service Web de réservation d'une chambre au niveau d'un hôtel **HotelBooking**, qui est un service japonais utilisant le «**Yen**»(JPY), avec une **TVA non incluse** de «**9.3%**», un facteur multiplicateur de« **1000** » et dont le format de date est « **aaaa/mm/jj** ».
- Un service Web **EuroBanking** pour calculer la somme des prix des services précédents offerts en « **Euro** » avec une TVA de «**19,6%**» et un facteur multiplicateur égal à « **1** » afin de s'acquitter de ce voyage.

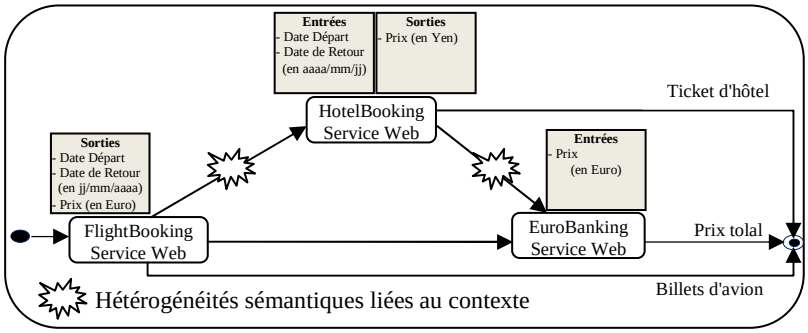


Fig. III.1 – Composition de planification de voyage avec les données échangées entres ses services Web

Le processus métier de cette composition est réalisé par le langage WS-BPEL[43] où plusieurs hétérogénéités [31] sémantiques[44] peuvent survenir pendant l'échange des données entres ses services composants à savoir:

a) Lorsque les dates sont envoyées du service Web « FlightBooking » au service Web « HotelBooking », d'où elles doivent être ajustées selon un format unique pour les interpréter correctement.

b) Lorsque le prix est envoyé du service Web « HotelBooking » au service Web « EuroBanking », où les prix utilisés doivent être convertis dans la même devise et dans le même facteur multiplicateur en prenant compte l'inclusion ou non de la TVA avant d'être envoyés au service Web «EuroBanking» pour les additionner en vue de s'en acquitter au niveau de la banque[45].

Ces hétérogénéités sémantiques sont liées au contexte des services Web qui doit être inclus dans la description des données échangées pour leur interprétation exacte.

III.2.2 Hétérogénéités sémantiques liées au contexte

Nous nous intéressons dans ce travail aux hétérogénéités sémantiques, notamment celles liées au contexte des services Web. L'exemple développé ci-dessus (Fig. III.1) montre l'importance

[43] Qui peut exécuter sur n'importe quel moteur de composition tel que ActiveBpel Engine (www.activevos.com).

[44] Dans les compositions des services Web, Il peut y avoir des hétérogénéités syntaxiques et structurelles qui sont hors sujet de ce mémoire. Pour plus d'information, veuillez vous reportez au [28,30].

[45] On remarque bien qu'il n y a pas de conflits entre FlightBooking et EuroBanking pendant l'échange des prix car ayant le même contexte. Voir ci-après cette notion de contexte.

du contexte pour l'interprétation correcte des données. Nous présentons ci-après, la définition du contexte ainsi que la classification des hétérogénéités sémantiques liées au contexte.

III.2.2.1 Définition du contexte

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 [6] 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 a été exploité en premier lieu pour apporter une solution efficace aux problèmes d'hétérogénéités sémantiques dans le domaine des bases de données. Dans les travaux de Sciore et al. [17] et de Sheth et Kashyap [49], 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.

Mrissa et al. ont bénéficié de cette notion de contexte pour l'appliquer dans le domaine des services Web, d'où leur définition du contexte comme suit :

« 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*** »[30].

Exemple 1 : Dans Fig. III.2, le « Format de la date » est considéré comme contexte de la donnée « Date »

Exemple 2: Dans Fig. III.2, le Facteur multiplicateur et la Devise[46] sont des éléments de contexte associés à la donnée (concept) « prix » permettant son interprétation correcte et homogène par les utilisateurs.

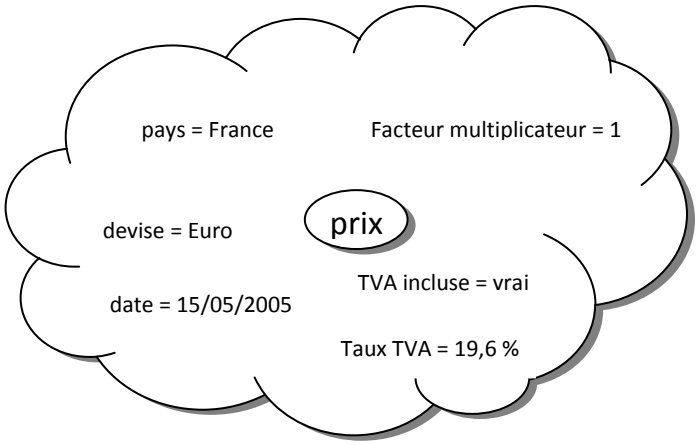


Fig. III.2 - Un exemple de « Prix » et son contexte

[46] Nous les appelons par la suite propriétés sémantiques (section III.2.2.2).

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.

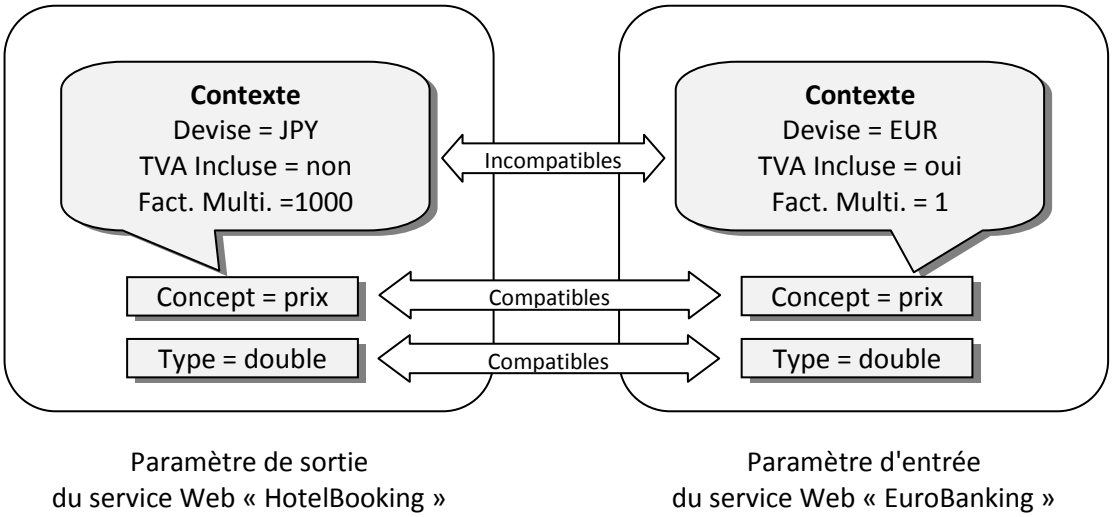


Fig. III.3 - Conflits entre les contextes « HotelBooking » et « EuroBanking »

Dans Fig. III.3, malgré les compatibilités qui existent dans les concepts sémantiques utilisés (concept « prix ») et dans les types de données (type «double»), les données doivent être interprétées différemment, car elles sont placées dans des contextes différents où la description WSDL ne peut les décrire par le format actuel[47].

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.

III.2.2.2 Définition de la propriété sémantique

L'utilisation d'une ontologie a pour but de décrire un domaine de connaissances, en explicitant les concepts utilisés ainsi que les relations entre ces concepts. Les propriétés sémantiques sont les données du contexte qui décrivent les divers aspects et caractéristiques sémantiques d'un concept de l'ontologie. Dans l'exemple de planification de voyage développé précédemment, le facteur multiplicateur et la devise associés au concept de prix sont des propriétés sémantiques (Fig. III.2).

Chaque propriété sémantique comme toute donnée[48] est caractérisée par :

- un nom
- une structure y compris le type (entier, chaîne de caractères, etc.)
- une valeur qui est une instance d'un type.

[47] Ci-après, on verra que Mrissa et al. proposent une extension du WSDL par son annotation.
[48] Sachant que chaque donnée est caractérisée par le triplet : nom, type et valeur [29].

III.2.2.3 Classification des hétérogénéités sémantiques

Suivant ces caractéristiques des propriétés sémantiques, les hétérogénéités sémantiques qui peuvent se présenter entre les différents contextes son classées comme suit [30,31] :

Hétérogénéités de nomination

Appelées aussi hétérogénéités sémantiques, les hétérogénéités de nomination sont liées à la signification du vocabulaire utilisé pour décrire les propriétés sémantiques.

Exemple: Pour décrire la propriété sémantique indiquant si la taxe sur la valeur ajoutée (TVA) est incluse dans le prix, un service Web anglais pourrait employer le mot «VATIncluded», tandis qu'un autre français pourrait employer le mot «TVAIncluse».

Dans ce cas, deux termes différents ont la même signification (synonymes). Dans d'autre cas, le même terme peut signifier deux choses différentes (homonymes) [28].

Afin de résoudre de tels conflits sémantiques, la description des correspondances entre les termes utilisés par les propriétés sémantiques est nécessaire en utilisant des ontologies.

Hétérogénéités structurelles

La représentation structurelle des propriétés sémantiques associées aux concepts d'un domaine d'application peut changer selon les services Web.

Exemple: Le concept de prix peut être décrit suivant différentes structures et selon les besoins des agences de réservation de vol. Certaines agences pourraient employer des facteurs multiplicateurs différents et ignorer la devise utilisée, parce que leurs partenaires emploient tous la même devise alors que d'autres agences pourraient utiliser un seul facteur multiplicateur, mais des devises 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 de valeur

Ces hétérogénéités concernent les valeurs des propriétés sémantiques. Dans notre exemple les prix sont gérés avec un facteur multiplicateur dont la valeur varie entre 1 et 1000.

La résolution de telles hétérogénéités nécessite de décrire explicitement la propriété sémantique, le domaine de valeurs qui lui est associé et les fonctions de conversion entre deux valeurs.

Exemple 1 : pour calculer le prix d'une devise à une autre, on utilise le facteur multiplicateur en multipliant le prix par le facteur multiplicateur d'origine, puis sa division par le facteur multiplicateur ciblé.

Exemple 2: 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.

Exemple 3 : Expression de dégrée de la température en Celsius et en Fahrenheit.

III.3 Modèle orientée contexte pour la description des données

L'interprétation correcte de la sémantique des données envoyées et reçues lors d'un échange des données entre les services Web en interaction exige en premier lieu de décrire les données de ces services avec une sémantique explicite et compréhensible par les machines.

Afin de répondre à cette exigence, 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 [29].

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 décrivent la sémantique attachée à chaque service Web. Ainsi, le domaine des services Web sémantiques se situe au croisement du Web sémantique et des services Web[49]. De nombreux langages et architectures existent afin de décrire les services Web sémantiques qui ont été classifiés selon deux catégories :

1) La classe des langages sémantiques qui tentent de s'imposer comme de nouvelles normes de description en remplacement des langages actuels comme WSDL. Ce sont des langages complets qui décrivent les services Web ainsi que leur sémantique (OWL-S, WSMO et DIANE).
2) La classe des annotations qui consiste à annoter les langages existants avec de l'information sémantique (annotations par SESMA, WSDL-S et SAWSDL).

L'avantage principal de ce dernier genre de solution réside dans la facilité pour les fournisseurs de services d'adapter leurs descriptions existantes aux annotations proposées.

C'est à partir de cette notion de contexte et cette idée d'annotation que Mrissa et al. construisent leur modèle orienté contexte [29,31] pour la représentation des données échangées entre les services Web en adoptant le principe d'annotation du fichier de description WSDL. Et cela, afin de permettre la résolution des hétérogénéités sémantiques qui peuvent apparaitre entre ces services Web en interaction.

Dans ce qui suit, nous présentons les principaux concepts de ce modèle, ainsi que la manière d'intégrer le contexte dans les descriptions des services Web.

[49] Comme l'on a vu dans la section I.5.

III.3.1 Principaux Concepts

Ce modèle contextuel [29,31] est basé principalement sur les notions de l'objet sémantique et les modifieurs statiques et dynamiques pour décrire les données avec leurs contextes.

II.3.1.1 Objet sémantique

Un objet sémantique est un objet de donnée avec une sémantique explicite décrite par son contexte.

Un objet sémantique S est un quadruplet [29,30,31], représenté de la manière suivante :

S=(*c*,*v*,*t*,*C*), où

- "***c***" est le concept auquel l'objet sémantique se réfère, un concept définit la famille ontologique (ontologie de domaine) à laquelle cette donnée appartient, en d'autres termes la classe abstraite dont cette donnée est une instance.
- "**v**" est la valeur de l'objet sémantique, une valeur est la donnée elle-même. Cette valeur est une instance du type de la donnée.
- "**t**" est le type dans lequel cette valeur est décrite, un type 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.
- "**C**" est le contexte de l'objet sémantique, un contexte apporte des précisions sur l'interprétation de la donnée.

Ce contexte est lui-même constitué d'objets sémantiques appelés ***modifieurs***, car ils appartiennent au contexte. Les modifieurs ont la capacité de modifier la signification de l'unique objet sémantique auquel ils sont associés. La définition de l'objet sémantique est résumée par le diagramme de classes donné dans Fig. III.4.

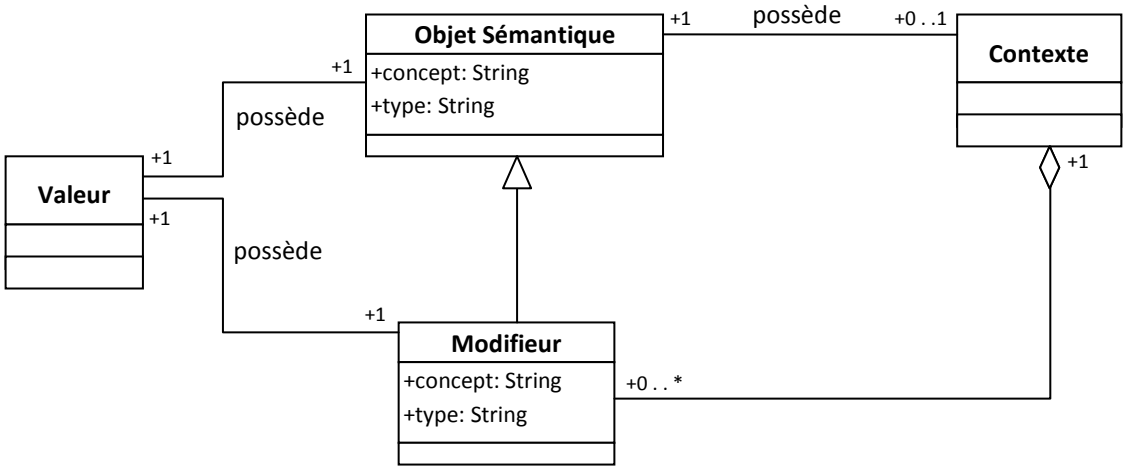


Fig. III.4 - Représentation UML de l'objet sémantique

II.3.1.2 Modifieurs statiques et dynamiques

Les Modifieurs sont des objets sémantiques qui participent à un contexte. Par conséquent, le contexte est vu comme un arbre de modifieurs où les feuilles sont des modifieurs avec un contexte nul [29,31]. Les modifieurs peuvent être de type statique ou dynamique :

- 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[50], des valeurs prises par un ensemble d'autres modifieurs appartenant au même contexte, ces modifieurs pouvant être indistinctement statiques ou dynamiques.

Exemple : La valeur du modifieur dynamique « *Format de date* » de l'exemple illustré par la Fig. III.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 = jj/mm/aaaa ».

Par contre, aucune information contenue dans ce contexte ne nous permet de déduire la valeur du modifieur statique « TVAincluse ».

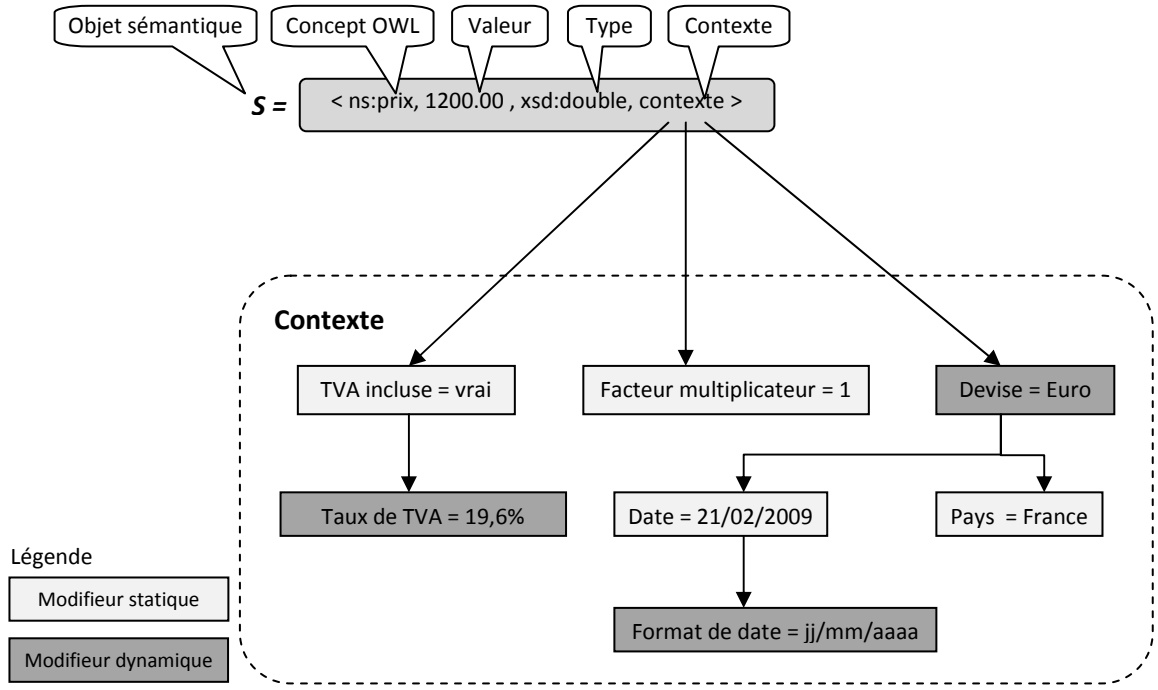


Fig. III.5 - Représentation de l'objet sémantique S avec son contexte

[50] L'utilisation de règles logiques dans le fonctionnement de l'architecture de médiation est expliquée ci-après.

Fig. III.5 donne un exemple d'objet sémantique S relatif au prix qui peut être envoyé au service Web «EuroBanking» de notre scénario de planification de voyage en ligne :

- L'attribut **ns:prix** renvoie au concept « prix » de l'ontologie symbolisée par l'espace de nommage « ns »,
- L'attribut **1200.00** 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 statiques additionnels (**Date et Pays**)

III.3.2 Intégration du contexte dans la description des services Web

Le modèle orienté contexte de représentation sémantique des données construit autour de la notion d'objet sémantique décrit précédemment permettra 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 les modifieurs, dont la sémantique est décrite dans une ontologie contextuelle.

Cependant, pour pouvoir exploiter cette information sémantique supplémentaire, il est nécessaire de fournir une méthode pour l'intégrer dans la pile standard de protocoles des services Web en utilisant les ontologies contextuelles et l'annotation du langage de description des services Web (WSDL).

III.3.2.1 Ontologies de domaine et contextuelles

La caractéristique majeure de ce modèle contextuel de Mrissa et al. est l'attachement des ontologies de contexte aux concepts des ontologies de domaine [29,31].

Les ontologies de domaine

Une ontologie de domaine contient les concepts les plus généraux d'une représentation partagée d'un domaine de connaissances et leurs relations conçus avec une approche descendante« Top-Down »[51]. Les experts du domaine de connaissance devraient spécifier les ontologies de domaine et limiter au minimum la description supplémentaire des concepts de domaine facilitant ainsi l'adhésion des fournisseurs de services. Ainsi, les ontologies de domaine devraient être les plus similaires possibles pour servir l'interopérabilité.

[51] Qui consiste à définir d'abord les concepts les plus généraux d'une ontologie pour obtenir une représentation partagée du domaine de connaissances.

Cependant, les hétérogénéités sémantiques entre les services Web doivent toujours être manipulées. Ceci est faisable grâce aux ontologies contextuelles.

Les Ontologies contextuelles

Une ontologie contextuelle est attachée à chaque concept d'ontologie du domaine. Ce type d'ontologie a pour but de décrire le contexte d'un concept de l'ontologie de domaine par modélisation de toutes les différentes propriétés sémantiques des concepts du domaine qui ne sont pas décrites dans l'ontologie du domaine. Afin de construire l'ontologie contextuelle, une approche ascendante « Bottom-Up »[52] est adoptée.

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.

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.

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.

Pourquoi l'utilisation des ontologies contextuelles

L'utilisation du contexte vient d'une simple prétention: l'on précise qu'il y a une difficulté pour que les services Web adhèrent à plusieurs communautés[53] ou participent à plusieurs compositions. En effet, chaque communauté suit une ontologie spécifique qui n'est pas toujours conforme avec les ontologies des autres communautés d'une part et, d'autre part, chaque composition contient divers services Web dont les ontologies sont différentes. En revanche, les services Web suivent implicitement ou explicitement les sémantiques locales de leurs fournisseurs d'où il est très difficile d'obtenir un agrément commun sur une représentation partagée des connaissances d'un domaine, notamment dans un environnement ouvert comme celui d'Internet.

[52] Qui part des vues locales des utilisateurs pour suivre un processus de généralisation jusqu'à obtenir une représentation cohérente du domaine de connaissances.

[53] Chapitre 2 a détaillé l'architecture des communautés des services Web, ainsi que son fonctionnement.

Par conséquent, l'adhérence d'un service Web à une nouvelle communauté (ou la participation à une nouvelle composition) exige un changement dans le code d'implémentation (hard-coded) du service ou la conception d'un emballage (Wrapper) qui agit au nom du service Web original, afin de se soumettre à l'ontologie de la communauté (ou de la composition). Une telle condition pénible s'applique à chaque nouvelle communauté (composition) quand un service Web y adhère (participe).

Le modèle contextuel a pour objectif de simplifier la tâche des fournisseurs de service Web en adhérant (participant) aux nouvelles communautés (compositions), en réduisant l'ontologie de domaine au minimum, et en fournissant des ontologies contextuelles additionnelles pour manipuler la sémantique locale des différents prestataires de service.

III.3.2.2 Annotation contextuelle du fichier WSDL

Le Modèle contextuel repose sur l'annotation des parties de messages décrites dans les documents WSDL des services Web [29,31]. 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 (entrée et sortie) en objets sémantiques par l'ajout du contexte permet d'effectuer la médiation sémantique.

Comme expliqué dans le chapitre 1, chaque opération <**operation**> 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**>, appelés aussi « 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.

L'annotation enrichit les paramètres d'entrée et de sortie contenus dans des documents WSDL en annotant les éléments <**part**> avec un attribut **context**[54] qui décrit les noms et valeurs des modifieurs (voir le méta-modèle WSDL, en mettant en valeur l'annotation contextuelle encadrée en pointillés dans Fig. III.6) et en utilisant une liste de noms qualifiés[55]. Nous associons le premier nom qualifié de la liste au concept de l'ontologie de domaine auquel le paramètre est rattaché, (*c*) dans la définition de l'objet sémantique. Cette information nous permet d'identifier le concept de domaine concerné par l'annotation. 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.

Et les règles logiques permettront d'inférer les valeurs des modifieurs dynamiques nécessaires pour compléter le contexte *C*, pendant l'échange de données.

[54] La spécification WSDL autorise l'ajout d'attributs supplémentaires.

[55] Un nom qualifié (type Qname) 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).

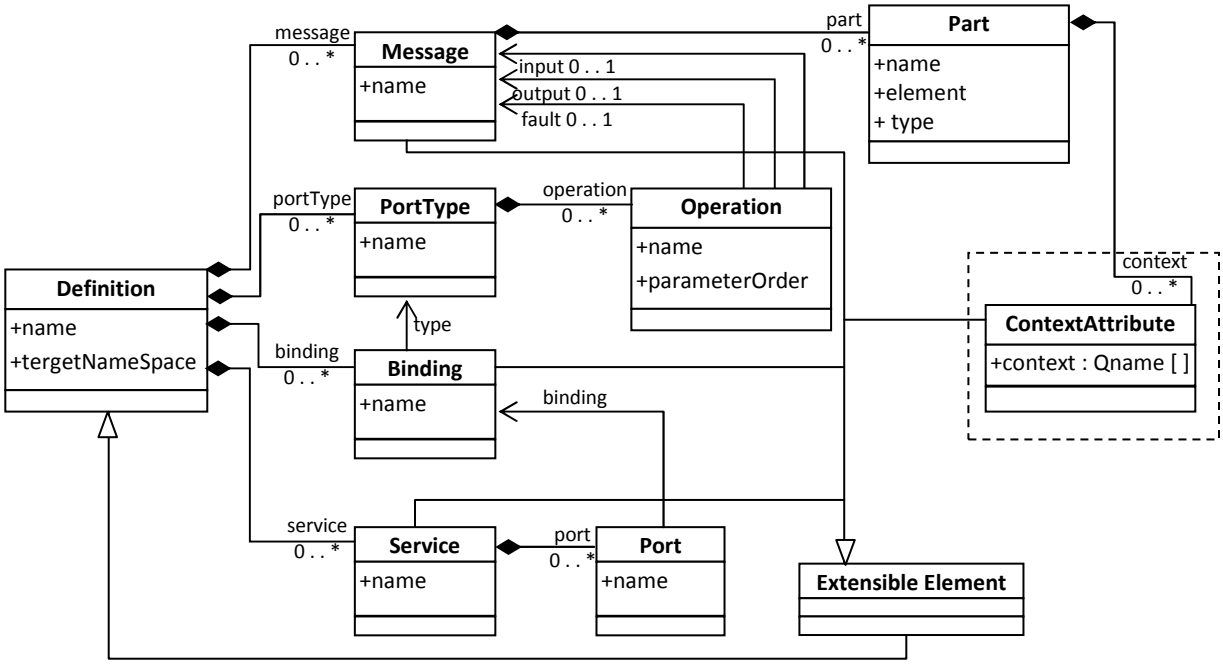


Fig. III.6 - Représentation du contexte dans le méta-modèle WSDL

III.3.2.3 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 [30].

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. Une classification donnée par [30] montre les différentes propriétés des fonctions de conversion pour les objets sémantiques et identifie différentes catégories de fonctions, selon les attributs de l'objet sémantique sur lesquels les fonctions s'appliquent.

Propriétés des fonctions de conversion

Les propriétés des fonctions de conversion entre objets sémantiques sont :

a) 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.

Exemple 1 : 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.

Exemple 2 : la conversion de précision est une fonction non totale. En effet, la valeur 1.24762 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.

b) 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.
Exemple 1: Une fonction d'archivage de données est dite sans pertes car les données originales peuvent être retrouvées par la suite.
Exemple 2 : Une fonction qui convertit une image au format « BMP[56]» vers le format «JPEG [57]» est une fonction avec pertes à cause de l'algorithme de compression de ce dernier format.

c) Fonctions de conversion monotones croissantes. Une fonction monotone croissante a pour particularité de préserver l'ordre original des valeurs.
Exemple : La fonction de conversion entre les échelles de température Celsius et Fahrenheit est une fonction monotone croissante.

Catégories des fonctions de conversion

Il y a trois catégories de fonctions de conversions, selon l'attribut de l'objet sémantique auquel elles s'appliquent à savoir:

a) 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.
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 les taux de change actuels.

b) 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.
Exemple : la conversion d'un entier en chaîne de caractères.

c) 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'utilisation des ontologies contextuelles permet de passer cette catégorie de fonctions comme détaillé dans la section III.3.2.1.

[56] Bitmap
[57] Joint Photographic Experts Group

III.4 Médiation orientée contexte pour les services Web

Exploitant les fonctionnalités offertes par cette notion d'objet sémantique et le modèle contextuel développés précédemment pour construire une architecture de médiation permettant aux services Web d'échanger leurs données, nous procéderons à l'échange des données dans une communauté et celui dans une composition des services Web sans conflit. Dans les deux cas, le principe de médiation est commun en utilisant le même module de médiation [30,31] illustré dans Fig. III.7.

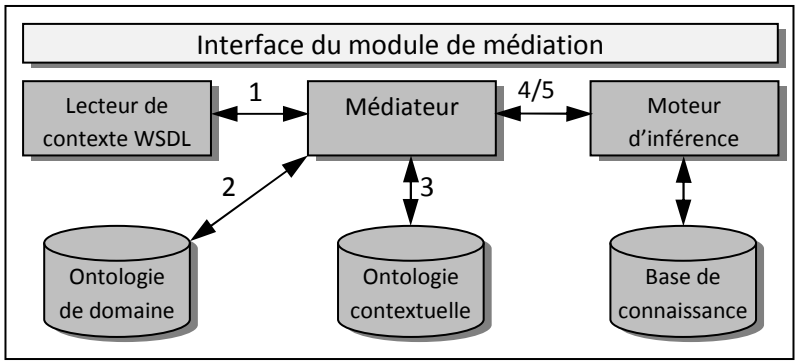


Fig. III.7 - Vue d'ensemble du module de médiation

Ce module est composé de :

- Une interface pour communiquer avec les services Web (Maître ou composant),
- Un composant noyau appelé médiateur qui orchestre les différentes étapes de processus de médiation décrites ci-dessous,
- Un composant appelé lecteur de contexte WSDL pour lire les fichiers WSDL annotés,
- Des entrepôts pour des ontologies de domaine et contextuelle.
- Un moteur d'inférence et un entrepôt de connaissance pour stocker les règles de conversion des données et de construction du contexte.

Dans une communauté, ce module de médiation est implanté au niveau du service Web Maître (Fig. III.8(a)) afin d'assurer un échange correct des données entre l'utilisateur et les services Web Esclaves qui exécutent les requêtes. Par contre, il est inséré, sous un service Web médiateur dynamiquement généré, dans le processus de composition entre chaque deux services Web qui participent à une composition et qui pourraient présenter des hétérogénéités de contexte entre eux (Fig. III.8(b)) afin de lever l'inconsistance des données lors de l'échange.

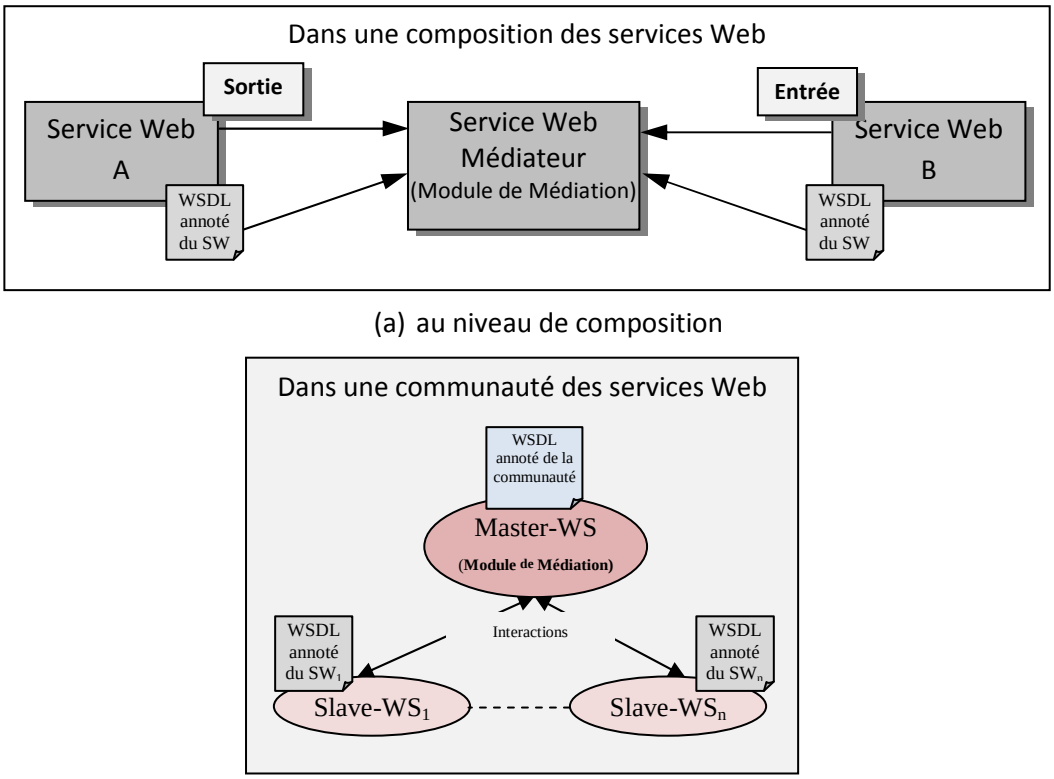


Fig. III.8- Module de médiation dans composition et communauté

III.4.1 Médiation sémantique au niveau des compositions

Pour obtenir une interprétation correcte de données échangées dans une composition des services Web différents, une médiation sémantique est nécessaire. L'approche de médiation[58] proposée ci-après par Mrissa et al. [30,31] a pour objectif de résoudre toutes les hétérogénéités sémantiques qui peuvent se produire au niveau de composition, tout en limitant la tâche des fournisseurs de services et des concepteurs de composition 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 comme décrire de manière explicite la sémantique utilisée par leurs services Web tout en annotant leurs fichiers WSDL, ainsi que l'unanimité sur les noms des concepts sémantiques utilisés par leurs services dans les ontologies de domaine et contextuelle pour permettre l'interopérabilité.

III.4.1.1 Intégration de la médiation sémantique dans la composition

Considérons que les étapes prérequises pour la mise en place de médiation ont été réalisées, c'est-à-dire que les fichiers WSDL des services Web sont correctement annotés avec l'information contextuelle, et que les ontologies de domaine et contextuelles requises sont disponibles. 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 contextualiser le processus métier de la composition c.-à-d. modifier le processus

[58] L'architecture de cette approche est étendue comme perspective de recherche dans la conclusion de ce travail.

métier en vue de l'intégration des services Web médiateurs nécessaires (Fig. III.8 (a)) à la médiation sémantique des données par la prise en charge du contexte.

Contextualisation des processus métiers

La contextualisation d'un processus métier se constitue selon les étapes suivantes [30,31]:

- Un algorithme de détection des 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.

1) Détection des hétérogénéités potentielles

Un algorithme de contextualisation, présenté dans [31], est réalisé pour analyser le processus métier afin de localiser dans les flux de données explicites et implicites les conflits sémantiques entre les services Web. Dans notre exemple, les hétérogénéités potentielles liées au contexte sont :

a) lorsque les dates sont envoyées au service Web « HotelBooking », et
b) lorsque les prix sont envoyés au service Web « EuroBanking».

2) 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[59] a détecté des hétérogénéités sémantiques potentielles.

Chaque service Web médiateur généré propose une opération appelée **mediateX2Y**, où X et Y sont remplacés par les noms des messages d'entrée et sortie des opérations. Ces dernières sont spécifiées dans les fichiers WSDL des services mis en relation via le service Web médiateur. Les entrées de l'opération de médiation sont les valeurs qui doivent être transformées dans la représentation requise, lesquelles sont spécifiées par les annotations des fichiers WSDL. Les sorties de l'opération sont les valeurs transformées, qui ont la signification désirée par l'opération cible dans la composition.

3) 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 de contextualisation. Le code BPEL inséré utilise les URI de référence (endpoint) des services Web médiateurs générés dynamiquement. Le processus métier contextualisé est prêt à être exécuté par n'importe quel moteur d'exécution BPEL classique.

[59] Cet algorithme est hors sujet à ce travail, et pour de plus amples informations, veuillez consulter [31].

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.

III.4.1.2 Processus de médiation dans une composition

Dans le processus de composition contextualisé, un service Web médiateur, en contenant le module de médiation, est placé entre chaque deux services Web composants qui présentent des hétérogénéités de contexte. Les étapes de fonctionnement de ce service Web médiateur sont détaillées comme suit [31] :

1. **Lecture des fichiers WSDL Annotés**

Les fichiers WSDL des deux services Web sont téléchargés par le service Web médiateur. Ces fichiers sont lus et analysés par le lecteur de contexte WSDL pour extraire les annotations des parties de messages que le service Web médiateur reçoit et envoie. Les annotations extraites font référence au concept de l'ontologie de domaine et aux éléments du contexte (représentés dans des ontologies contextuelles) nécessaires à une interprétation correcte des données. Le médiateur génère un modèle du document WSDL en mémoire et identifie les éléments annotés.

2. **Identification de vocabulaire du domaine**

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. Le service Web médiateur vérifie que les concepts utilisés par les deux services Web correspondent, c'est-à-dire qu'ils vérifient une relation de subsomption ou d'équivalence, laquelle peut être vue comme un cas particulier de subsomption réciproque entre deux concepts.

3. **Identification du contexte**

Ensuite, pour chaque service Web, une représentation en mémoire sous forme de structure arborescente des objets sémantiques est construite à partir de l'ontologie contextuelle. Les annotations contextuelles d'un concept 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'elles prennent dans le contexte du service Web.

Considérant toujours que les fournisseurs de services Web insèrent correctement l'information relative à leur contexte dans l'ontologie contextuelle avant d'annoter les fichiers WSDL, 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.

4. **Construction des Objets sémantiques**

Le service Web médiateur communique avec un moteur d'inférence pour déduire les valeurs

des modifieurs dynamiques appartenant au contexte en utilisant les règles logiques stockées dans la base de connaissances du moteur d'inférence et ce, afin de compléter la construction les objets sémantiques dans la mémoire avant de procéder à la conversion des données.

5. **Conversion des données**

Une autre fois, le service Web médiateur communique avec le moteur d'inférence pour effectuer cette conversion des données dans la représentation contextuelle requise. A partir des étapes précédentes, nous obtenons deux arbres de représentation du contexte en mémoire qui ont des feuilles. Ces feuilles sont des modifieurs contenant des valeurs, formant ainsi le contexte. Le service Web médiateur compare chaque élément du contexte l'un à l'autre et tente d'établir des correspondances d'équivalence entre les modifieurs des deux contextes grâce à l'information contenue dans l'ontologie contextuelle. Ensuite, il demande au moteur d'inférence de vérifier la convertibilité des valeurs des modifieurs correspondants et d'effectuer leur conversion, si possible. La conversion des valeurs est effectuée à l'aide de fonctions de conversions telles que décrites dans la section III.3.2.3. 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.

III.4.2 Médiation sémantique au niveau des communautés

La médiation au niveau des communautés des services Web est établie par le service Web Maître [32,33] qui contient un module de médiation (Fig. III.8 (b)). Ce module de médiation permet au service Web Maître de traiter des requêtes entrantes de l'extérieur de la communauté.

Grâce au module de médiation, le service Web Maître peut agir en tant que médiateur[60]. Après réception d'une requête d'utilisateur, il utilise le module de médiation pour convertir le message de la requête selon la sémantique du service Web Esclave. Et après réception de la réponse de ce service, le service Web Maître utilise le module de médiation encore une fois pour convertir le message de la réponse suivant la sémantique de la communauté avant de l'envoyer de nouveau à l'utilisateur.

III.4.2.1 Processus de médiation dans une communauté

Typiquement, un utilisateur qui souhaite interagir avec une communauté pour l'accomplissement de ses objectifs, découvre et choisit le fichier WSDL de la communauté par l'intermédiaire du registre UDDI. Puis, le programme client emploie ce fichier WSDL pour établir une requête et l'envoyer au point d'accès de la communauté (endpoint) qui est le service Web Maître. Ce dernier a pour but de manipuler les interactions avec le client afin de cacher la complexité de la communauté, tandis que l'architecture basée-communauté est complètement transparente du point de vue de ce dernier.

[60] Le service Web Maître est également responsable d'autres tâches comme rapporté dans le chapitre 2.

Cependant, et comme nous l'avons vu dans le chapitre 2, le service Web Maître n'implémente pas lui-même la fonctionnalité de la communauté[61]. Son rôle est de choisir un des services Web Esclaves appartenant à la communauté selon les préférences d'utilisateur[62], et expédier la requête du client à ce service Web Esclave. Ensuite, il doit retourner la réponse du service Web Esclave au client. Le processus de médiation sémantique orienté contexte se déroule de la façon suivante [32,33]:

1. **Lecture du Fichier WSDL Annoté**

    a) Le processus choisit un service Web Esclave et cherche sa description WSDL.
    b) Il analyse ensuite les paramètres d'entrée et de sortie de l'opération WSDL choisie de ce service Web Esclave.
    c) Pour chaque paramètre, il extrait le concept du domaine et les modifieurs statiques contenus dans le WSDL annoté. En supposant que le module médiation sémantique est configuré pour une communauté spécifique et a toujours des représentations, dans la mémoire, des paramètres d'entrée/sortie de la communauté en tant qu'objets sémantiques, alors il n'est pas nécessaire de chercher le fichier WSDL de la communauté.

2. **Identification de vocabulaire du domaine**

    a) Il communique avec l'ontologie de domaine pour identifier les concepts de domaine contenus dans l'annotation.
    b) Si les termes ne sont pas trouvés, une exception est levée et la prochaine étape est confirmée.

3. **Identification du contexte**

    a) Il communique avec l'ontologie contextuelle pour identifier les modifieurs contenus dans l'annotation.
    b) Si les termes ne sont pas trouvés, une exception est déclenchée et l'étape suivante est confirmée.

4. **Construction des Objets sémantiques**

    a) Usant de l'information contenue dans le WSDL annoté, le module de médiation convertit les paramètres annotés de WSDL en objets sémantiques.
    b) Il interagit avec le moteur d'inférence afin de déduire les valeurs des modifieurs dynamiques disponibles dans le contexte.

5. **Conversion des données**

    a) À ce stade, le module de médiation possède deux objets sémantiques dans-mémoire

[61] On verra, dans le chapitre 4, comment cette information nous aidera à la réalisation de notre contribution.
[62] En appliquant le protocole Contract-net (Chapitre 2)

qui ont des contextes différents et doit convertir les données du contexte de la communauté au contexte du service Web Esclave. Pour ce faire, il interagit avec l'entrepôt de connaissance qui stocke des règles de conversion et procède à la conversion des données de la sémantique de la communauté à la sémantique du service Web Esclave

b) Si la conversion n'est pas possible, une exception est levée et les données seront expédiées au service Web Esclave.

A la réception de la réponse, la tâche du module de médiation est renversée :

1. Il lit la description WSDL du service Esclave qu'il interagit avec lui, identifie les informations du domaine et du contexte contenues dans l'annotation et construit les objets sémantiques.
2. il interagit avec le moteur d'inférence qui convertit de nouveau les données de la sémantique du service Web Esclave à la sémantique de la communauté.
3. Si la conversion échoue, il soulève une exception et expédie les données au client de la communauté.

III.5 Conclusion

Dans ce chapitre, nous avons présenté en premier lieu l'approche de médiation orientée contexte de Mrissa et al. pour échanger correctement des données entre les services Web au niveau des compositions ou celui de communauté. Cette approche repose sur un modèle contextuel de représentation des données, construit autour de la notion de l'objet sémantique. Ensuite, nous avons détaillé les différents processus de médiation dans la composition et dans la communauté des services Web, en se basant sur ce modèle contextuel pour résoudre les hétérogénéités sémantiques qui peuvent apparaitre entre les services Web en interaction. Le déroulement de ces processus de médiation est basé essentiellement sur la réalisation de deux principales taches, l'intégration du contexte dans la description des services par l'annotation en utilisant des ontologies contextuelles couplées à des ontologies de domaine, et l'intégration du module de médiation dans la composition, à l'intérieur d'un service Web médiateur entre chaque deux services connus des conflits et, dans la communauté, par son implantation au niveau du service Web Maître.

Ces travaux de médiation cités dans ce chapitre sont à l'origine de la solution proposée à notre problématique qui sera détaillée dans le chapitre suivant où nous décrirons notre contribution en démontrant sa faisabilité par son implémentation dans le cadre de notre exemple de planification de voyage.

IV Chapitre 4 : Haute disponibilité maintenant la médiation

« Un échec apparent peut contenir les graines d'un succès qui germera en son temps, et donnera des fruits toute l'éternité ».

Frances Ellen Watkins Harper

IV.1 Introduction

Dans le deuxième chapitre, nous avons présenté l'approche de Maamar et al. pour soutenir la haute disponibilité des services Web en substituant le service Web défaillant par un autre fonctionnellement similaire. Quant au troisième, une exposition est étalée des travaux de Mrissa et al. dans la médiation sémantique aussi bien pour les compositions comme pour les communautés des services Web en utilisant un modèle orienté contexte pour la description des paramètres d'entrée/sortie échangées entre ces services Web et ce, afin de garantir un échange correct des données.

Dans ce chapitre, l'on va exposer clairement la problématique de ce mémoire à l'aide de notre exemple de planification de voyage lors de la substitution d'un service Web de la composition tombé en panne et cela pour supporter sa haute disponibilité, en montrant l'impact sémantique de cette substitution sur la médiation de la composition, ainsi que notre solution envisagée pour y remédier, inspirée des approches de la médiation sémantique susmentionnées. Et pour prouver la faisabilité de cette solution, nous avons procédé à la réalisation d'une application relative à notre scénario de composition.

IV.2 Exposition détaillée de la problématique

Après réalisation de la composition de notre exemple, où l'on a réglé toutes les hétérogénéités sémantiques liées aux contextes des services Web[63] suivant l'approche de médiation sémantique basée-contexte de Mrissa et al. [31], notamment pour les entrées/sorties "Prix" et "Date", tout en réalisant les étapes nécessaires du processus de médiation au niveau des compositions citées dans la section III.4.1. Ces étapes débutent par l'annotation des services Web et se terminent par l'insertion des services Web médiateurs nécessaires (Fig. IV.1) équipés de modules de médiation (Fig. III.7) et cela pour garantir un échange consistant entre les services Web composants. Les médiateurs ajoutés sont :

- Un service Web médiateur entre « FlightBooking » et « HotelBooking » pour régler les conflits des dates échangées et qui ont des formats différents.
- Un autre service Web médiateur entre « HotelBooking » et « EuroBanking » pour régler les conflits des prix envoyés en on les convertissant de son contexte Japonais au contexte Européen.

[63] Voir Fig. III.1

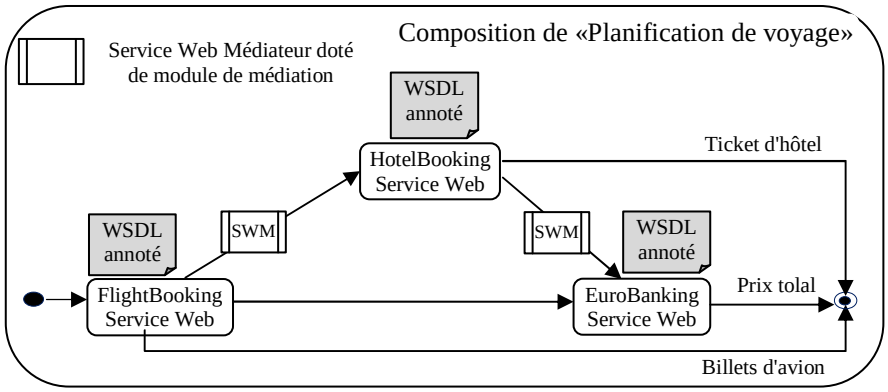


Fig. IV.1 – Résolution des hétérogénéités par la médiation sémantique basée-contexte

Ensuite, l'on a remarqué que le service de réservation de billets d'avion FlightBooking à un certain moment ne répond pas à l'exigence de la composition, alors que les autres services attendent de lui des réponses[64] pour faire exécuter la composition. Tombé en panne, et pour assurer la continuation dans son service, l'on a procédé à appliquer l'approche de soutien de la haute disponibilité en utilisant les communautés des services Web comme proposée par Maamar et al. dans [58,59].

En appliquant cette approche, on a fait appel à la communauté nommée « **FlightBooking** » (Fig. IV.2) qui fournit la même fonctionnalité que celle du service Web échoué[65]. Après avoir exécuté le protocole Contract-Net (détaillé dans la section II.3.2.3) par son Maître « **MasterFlightBooking** », le service Web Esclave « **UKFlightBooking »** a gagné l'offre pour substituer le service Web échoué.

Dans Fig. IV.2, nous remarquons aussi que le service Web « **EKFlightBooking »** n'a pas bien répondu à l'appel de soutien de son Maître alors que le service Web « **USFlightBooking »** a répondu positivement à l'appel mais n'a pas été choisi par le Maître à cause de sa qualité de service (QoS) qui est inférieur à celle de « **UKFlightBooking »**. Or, il peut rester comme support pour le service gagnant (service primaire) en cas de sa défaillance suivant l'approche de la disponibilité basée-communauté (voir la section II.3.2).

UKFlightBooking est un service Web anglais qui a ses propres paramètres (La date est de format « **mm/jj/aaaa »** et le prix est en « **livre (Pound) »** dont la devise est représentée par **GBP** avec un facteur multiplicateur de « **1** » et une TVA « **non incluse »** de « **17.5% »**). Cette substitution tend à garantir la haute disponibilité du service défaillant.

[64] La détection de sa défaillance est réalisée par la composition qui joue le rôle d'un "Watchdog" [59] expliquée auparavant dans II.3.2.4.

[65] Dans notre exemple, le service Web échoué ne fait pas partie de cette communauté.

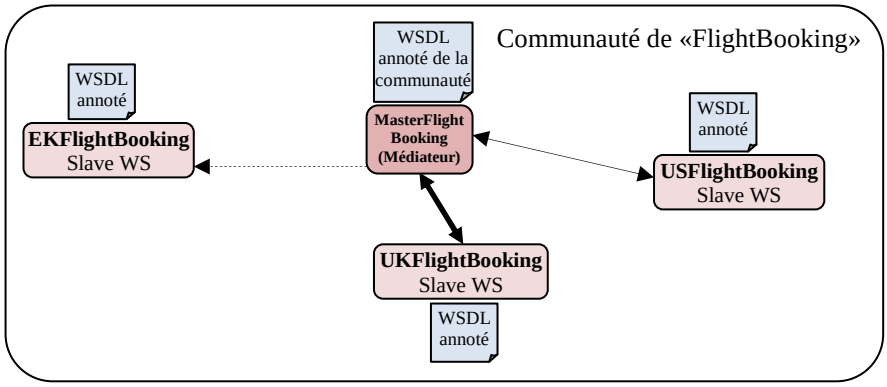


Fig. IV.2 – Communauté « FlightBooking »

A ce moment, les résultats retournés par ce service substitut doivent être convertis[66] selon le contexte de la communauté[67]. Dans notre cas, le service Web Maître de cette communauté adopte un Fichier WSDL annoté avec un contexte Américain différent de celui de UKFlightBooking uniquement dans la devise des prix qui est en « **Dollar »** (devise désignée par **USD**).

Par la suite, ce service remplaçant rentre en interaction avec les reste des services Web de la composition y compris les services Web médiateurs insérés par le processus de la médiation sémantique orientée contexte c.-à-d. leurs contextes qui rentrent en interaction (avec HotelBooking pour le Concept "Date" et avec EuroBanking pour le concept "Prix"). Après comparaison des paramètres entrées/sorties de ces services en interaction, l'on a découvert qu'il y avait de nouvelles hétérogénéités sémantiques que la médiation actuelle ne peut résoudre avec ses médiateurs déjà intégrés car ils ont des contextes différents de celui du service substitut converti par le contexte de sa communauté (Fig. IV.3). Il faut donc revoir de nouveau ces actions de médiation initialement prises pour régler ces nouvelles hétérogénéités dues à la substitution appliquée suivant cette nouvelle approche de la haute disponibilité qui emploie les communautés.

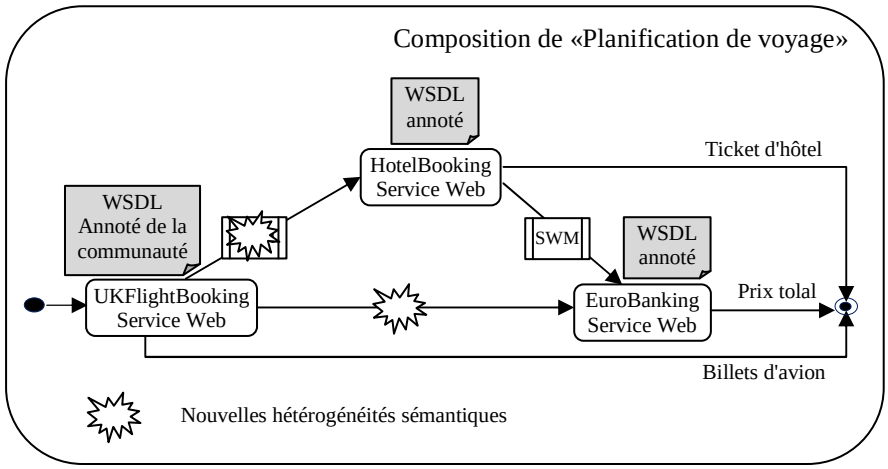


Fig. IV.3 – Apparition des nouvelles hétérogénéités après la substitution de « FlightBooking »

[66] Comme les requêtes doivent être converties.

[67] Comme on l'a vu dans III.4.2 pour la médiation sémantique au niveau des communautés.

IV.2.1 Impact sémantique de la substitution

Dans notre scénario, la substitution [56] par ce nouveau service Web provoquera une inconsistance sémantique dans la composition quand à l'échange des dates et des prix qui ont des contextes différents (Fig. IV.3) ce qui implique de relancer, une autre fois, et en temps d'exécution, le processus de médiation sémantique de cette composition en prenant en considération le contexte de ce nouveau service, c'est-à-dire son WDSL annoté[68], ainsi que la création des nouveaux Web service médiateurs à savoir :

- Entre UKFlightBooking et EuroBanking pour éviter le conflit de prix.
- Et l'adaptation du service Web médiateur déjà créé et inséré dans la composition entre UKFlightBooking et HotelBooking pour prendre en charge les conflits des dates.

Chose pénible à chaque substitution d'un service Web échoué. Suite à cela, et pour y remédier en évitant cette lourde tache de mise à jour de médiation initialement prise, on propose dans ce qui suit une solution basée toujours sur le principe de la médiation orientée contexte tout en tirant profit des travaux de Mrissa et al. [29,31,32,33] et résumés dans chapitre 3.

IV.3 Médiation sémantique en cas de substitution

Pour faire face à cet impact engendré par la substitution du service Web défaillant et garantir une haute disponibilité du service, nous présenterons une solution qui consiste, avant le remplacement du service Web défaillant, à faire adopter la sémantique de ce dernier par la communauté (plus précisément par son service Web Maître) soutenant ce service défaillant comme étant sa sémantique. Et étant donné que nous travaillons dans un cadre sémantique orientée contexte, le service Web Maître va procurer le contexte du service défaillant pour annoter son fichier WSDL.

Nous argumentons cette adoption contextuelle par le fait que :

- Le service Web Maître est développé par le concepteur d'application [57,60,12,21] indépendamment de tout fournisseur de services Web, c.-à-d. que le Maître n'a pas de fournisseur qui lui appartient et qui réclame une telle adoption.
- Le Maître de la communauté n'implémente pas la fonctionnalité de cette dernière et, par conséquent, ne participe jamais à une composition des services Web. Il est chargé seulement de la gestion de la communauté (comme on l'a vu dans la section II.3.2). Ces caractéristiques permettront au service Web Maître d'adopter un autre contexte autre que le sien sans qu'il y ait une influence sur la médiation sémantique de la composition tandis que ce dernier, en aucun cas, ne fait partie de la composition en offrant une fonctionnalité [57,60,12,21].

[68] Il s'agit du WSDL annoté de sa communauté.

- Cette Adoption ne change pas les propriétés non fonctionnelles du service Web Maître, elle touche seulement les propriétés fonctionnelles qui sont décrites dans son document WSDL, à savoir les paramètres d'entrée/sortie à travers le changement qui sera effectué sur ce document. Les 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, la qualité de service et le support des transactions sont des propriétés non fonctionnelles [30].

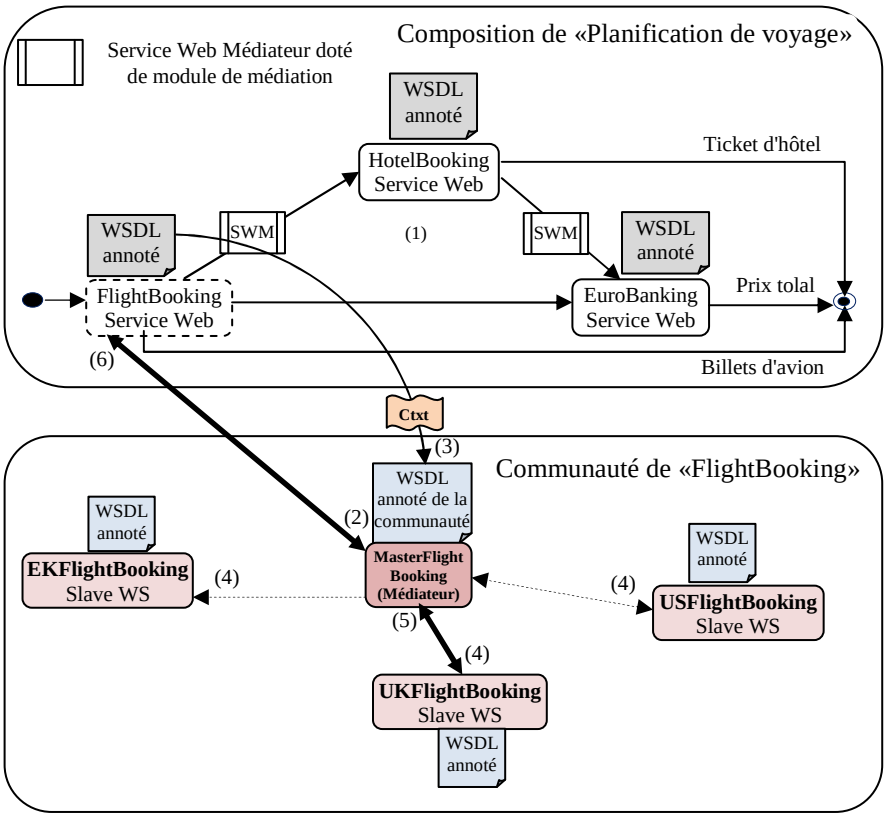
La réalisation de notre solution concrétise plusieurs objectifs à la fois, à savoir :

1. Assurer la haute disponibilité du service Web comme stipulé dans l'approche basée-communauté et cela par la substitution du service défaillant par un autre service Web de la communauté.
2. Eviter de rompre la médiation sémantique de la composition initialement prise pour régler les hétérogénéités sémantiques entre les services Web participant à la composition, et ce, malgré la substitution du service défaillant par un autre service Web sémantiquement différent de ce service défaillant.
3. Permettre le soutien du service Web substitut, en cas de sa défaillance, par les autres services Web Esclaves de support qui ont répondu positivement à l'appel sans aucun changement au niveau de la médiation sémantique de la composition. Même en cas de reprise de fonctionnement du service Web échoué, il peut continuer à fournir sa fonctionnalité dans la composition sans qu'il y ait d'inconvénients.
4. Enfin, notre solution est orientée contexte et ne sort pas du cadre général de l'approche de la médiation sémantique basée sur la notion de contexte. Chose qui permettra à notre solution de s'effectuer d'une manière transparente pour le client durant l'exécution de la composition.

Dans ce qui suit nous détaillons cette solution.

IV.3.1 Processus de médiation sémantique

Dans notre travail précédent [8], nous avons développé un processus de médiation en cas de substitution schématisé dans Fig. IV.4. Ce processus est constitué principalement de deux étapes, l'adoption du contexte du service Web défaillant et la conversion entres les contextes.



Chronologie

1. Résolution de toutes les hétérogénéités de la composition par la médiation ; 2. Appel à la substitution après l'échec; 3. Extraction et adoption de contexte du SW échoué; 4. Appel d'offres suivant le protocole CN ; 5. Exécution et conversion de la réponse du SW substitut; 6. Réponse retournée à la composition avec le même contexte du service Web échoué.

Fig. IV.4 – Vue d'ensemble de la solution proposée

IV.3.1.1 Extraction et adoption du contexte

L'adoption de la sémantique du service Web défaillant représentée par son contexte se réalise en deux étapes à savoir : l'extraction de ce contexte à partir du fichier WSDL de description des données annoté lors de la médiation sémantique initialement prise et l'adoption du contexte de ce service défaillant par le service Web Maître de la communauté appelée pour soutien :

1. Téléchargement et lecture du WSDL annoté du service Web échoué (dans notre cas FlightBooking) par le lecteur de contexte WSDL faisant partie du module de médiation sémantique (Fig. III.7) intégré au niveau du service Web Maître de la communauté, et ce, afin d'extraire son contexte résumé dans les annotations de ses paramètres d'entrée/sortie. Les annotations extraites sont des objets sémantiques qui font référence aux concepts de l'ontologie de domaine et aux éléments du contexte rattachés à ces concepts (ce sont les modifieurs statiques).

2. Téléchargement et lecture du WSDL annoté du service Web Maître (dans notre cas MasterFlightBooking) par le lecteur de contexte WSDL en vue d'annoter ses paramètres d'entrée/sortie par les annotations déjà extraites à partir du service Web échoué. Cette annotation est faite après avoir trouvé les correspondances nécessaires entre les concepts de domaine concernés des deux services Web (échoué et Maître) en interrogeant l'ontologie de domaine, de sorte que le Maître devra interpréter les entrées et les sorties par ce contexte du service Web échoué.

IV.3.1.2 Conversion contextuelle

Les étapes de la conversion dans notre processus de médiation (Fig. IV.4) sont décrites comme suit:

Après appel à la communauté FlightBooking supportant la haute disponibilité du service qui est tombé en panne pour le remplacer par un autre fonctionnellement équivalent, et avant de choisir ce dernier, l'on exige au service Web Maître de la communauté d'adopter la sémantique du service échoué, représentée par son contexte, comme une sémantique de la communauté, c.-à-d le WSDL du Maître doit être annoté par le contexte du service Web échoué et cela, après son extraction comme mentionnée ci-dessus, où toutes les entrées et sorties à la/de la communauté doivent être converties selon ce nouveau contexte par le module de médiation intégré au niveau du Maître.

Suite à ses responsabilités, le service Web Maître lance un appel d'offres à tous les services Web Esclaves de la communauté pour implémenter (exécuter) la fonctionnalité que le service Web échoué a été chargé de fournir, tout en appliquant le protocole Contract-Net [41,32,33].

Lorsque le service Web Esclave est sélectionné qui a gagné l'offre (dans notre cas UKFlightBooking) pour satisfaire la demande de la composition en transmettant la réponse au Maître, et avant de la ressortir, il l'adapte et la convertit suivant la sémantique de la communauté qui est finalement la sémantique du service Web défaillant en utilisant le module de médiation (Fig. III.7), où, à ce niveau, la médiation se déroule comme explicité dans le processus de médiation au niveau des communautés cité dans la section III.4.2, à savoir:

- Téléchargement et lecture par le lecteur de contexte WSDL des annotations des paramètres des entrées/sorties contenues dans des fichiers WSDL de la communauté et du service Web Esclave choisi.
- Identification de vocabulaire du domaine en communiquant avec l'ontologie de domaine pour identifier les concepts contenus dans ces annotations.
- Identification des contextes en communiquant avec l'ontologie contextuelle pour identifier le modifieurs statiques ainsi que leurs valeurs contenues dans ces annotations.
- Construction des objets sémantiques en déduisant les valeurs des modifieurs dynamiques disponibles dans les contextes à partir des valeurs des modifieurs

statiques connues précédemment en sollicitant le moteur d'inférence qui emploie les règles logiques stockées dans la base de connaissance du module de médiation. Par exemple, la règle logique suivante:

« Si pays = UK, alors Format de date = mm/jj/aaaa ».

- A ce stade, le module de médiation possède deux objets sémantiques différents dans la mémoire, et il doit convertir les données du contexte du service Web Esclave au contexte de la communauté en sollicitant encore une fois le moteur d'inférence qui utilise des fonctions de conversions entre les différentes valeurs des modifieurs. Ces fonctions de conversions sont stockées comme des règles logiques dans la base de connaissance.

Après cette conversion entre les deux objets sémantiques, la réponse est retournée par la sémantique adoptée par la communauté et la composition traite cette réponse comme elle venue du service Web défaillant or, il n'y aura pas de détection de nouveaux conflits sémantique au niveau de cette composition et par conséquent il n'y aura pas de nécessité de revoir la médiation sémantique dans la composition qui reste valable même après la défaillance d'un de ses services Web et sa substitution par un autre service Web.

Dans Fig. IV.5 Nous avons modélisé ces étapes de notre processus de médiation par un diagramme de séquence, appliqué à notre scénario, en montrant tous les messages échangés entre les trois acteurs : FlightBooking (service Web échoué), MasterFlightBooking (service Web Maître) et UKFlightBooking(service Web Esclave substitut)

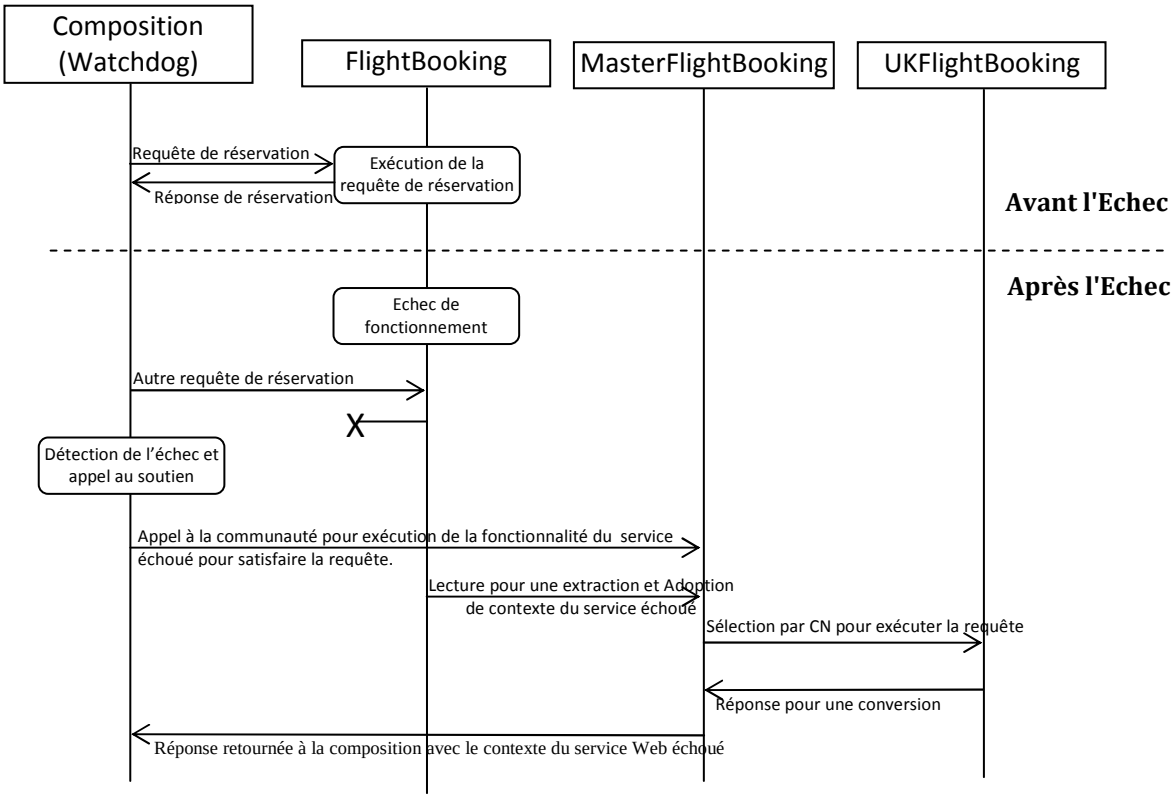


Fig. IV.5 – Diagramme de séquence du processus de médiation en cas de substitution

Il reste à noter que comme, dans tout travail utilisant la notion de communauté des services Web, nous remarquons que le service Web Maître prend un rôle primordial dans notre solution de médiation sémantique proposée dans le cadre de l'approche basée-communauté qui a pour objectif de soutenir la haute disponibilité des services Web en cas de panne en utilisant les communautés. Mais une communauté peut à son tour tomber en panne suite à l'échec de son service Web Maître qui est responsable de sa gestion. Cette problématique n'a été traitée que récemment[69] par Subramanian dans [44] qui à donné une stratégie pour assurer la haute disponibilité d'une communauté par choisir un autre service Web Maître temporaire parmi les services Esclave de la communauté en parcourant un algorithme d'élection nommé (FBA : Fast Bully Algorithm).[70]

IV.4 Implémentation

En vue de prouver la faisabilité de notre solution, l'on a développé une application limitée à notre exemple (Fig. IV.4) suivant le diagramme de séquence ci-dessus (Fig. IV.5). Cette application est réalisée en Java™ (JDK1.5[71]) comme langage de programmation et Eclipse 3.3[72] comme environnement de développement. Une capture d'écran du menu principal de cette application est donnée dans Fig. IV.6.

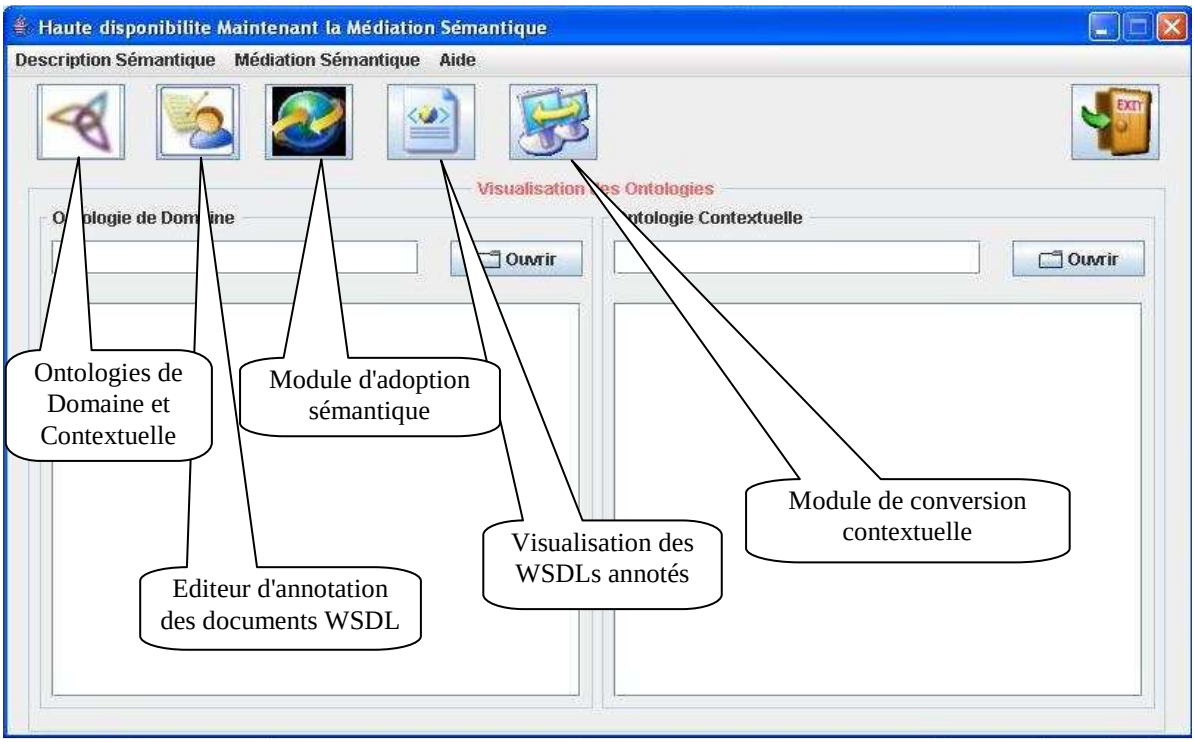


Fig. IV.6 – Capture d'écran du Menu principal de l'application

[69] En avril 2009.

[70] L'on propose ici un thème de recherche intéressant pour qui souhaite travailler sur l'impact sémantique entre la communauté et l'extérieur suite à l'application de cet algorithme.

[71] http://java.sun.com/

[72] www.eclipse.org/

Avant de détailler la conception de cette application, nous voudrions préciser que notre processus de médiation en cas de substitution s'exécute en temps réel et sans l'intervention de l'utilisateur par ce que nous somme toujours dans la troisième étape de déroulement d'une composition étape d'exécution (voir section I.5). L'utilisateur est seulement chargé, au préalable, de la création des ontologies de domaine et contextuelles et l'annotation des documents WSDL. Et sachant que notre processus connait déjà les URIs de nos services Web pour les invoquer, il va intervenir en cas de remplacement d'un service Web défaillant d'une manière indépendante. Mais en vue de montrer ses étapes de déroulement, l'on a procédé dans notre application à les suivre par menus.

Après la création des services Web de notre application l'on a procédé à leur déploiement dans Axis [3] installé sous le serveur Apache Tomcat [2], et les annotations des fichiers WSDL sont faites, chacun par le contexte de son fournisseur, par l'éditeur d'annotation de Mrissa et al. [31] conçu dans ce sens et intégré dans notre application. 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.

Les ontologies contextuelles et de domaine servant à ces annotations ont été conçues avec l'éditeur Protégé[73] pour décrire les concepts et les contextes requis pour un bon déroulement de notre exemple, ainsi les relations qui existent entre ces concepts.

Dans cette application, deux modules sont majeurs, à savoir:

1. Le module d'extraction et d'adoption du contexte du service Web échoué par le service Web Maître.
2. Le module de conversion contextuelle qui est responsable de la conversion sémantique entre les contextes des différents services Web en interaction.

Les ontologies créées précédemment seront employées par ces deux modules pour identifier des termes employés dans les annotations contextuelles et pour trouver les relations d'équivalence ou de subsomption[74] entre des concepts de domaine de connaissances. Le premier module les utilise pour sélectionner les concepts de domaine à annoter par le contexte adopté dans le fichier WSDL du service Web Maître, et le second pour construire les objets sémantiques sujet d'une éventuelle conversion contextuelle.

Cette identification des concepts et leurs relations sont faites grace à Jena™ API [22]. Jena est un framework sémantique pour java. Il permet d'écrire des programmes destinés au web sémantique et donc de manipuler des fichiers tels que RDF, RDF-S, DAML-OIL, OWL. Son utilisation, dans notre cas, permettra de transformer les ontologies de domaine et contextuelles sous forme d'une collection d'objets (dans le sens Orienté Objet du terme) afin de les interroger.

Le fonctionnement des deux modules repose sur l'API WSDL4J notamment ses classes [51] pour stocker les modèles de document WSDL en mémoire et gérer les éléments d'extensibilité

[73] Version 3.4 : http://protege.stanford.edu

[74] 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.

du contexte.

Notre module d'extraction et d'adoption utilise cette API notamment sa classe ***WSDLReader*** pour lire et télécharger en mémoire le fichier WSDL annoté du service défaillant accessible par la classe ***Definition*** pour extraire son contexte en utilisant la méthode ***GetExtensionAttribute*** de la classe ***Part***, ensuite et il annote à l'aide de la méthode ***SetExtensionAttribute*** les paramètres d'entrées/sorties correspondants du document WSDL du service Web Maître par ce contexte extrait. L'enregistrement du nouveau WSDL du service Web Maître annoté par le contexte du service échoué est fait par la classe ***WSDLWriter.***

Et le module de conversion contextuelle a pour tache d'effectuer toutes les conversions nécessaires entre les contextes des services Web Esclave substitut et le Maître de la communauté en suivant le processus de médiation détaillé dans la section IV.3.1.2.

IV.4.1 Exemple

Pour mieux comprendre notre application, nous présenterons ici les différentes étapes de notre solution illustrée dans Fig. IV.4. et Fig. IV.5 appliquées sur notre exemple de planification de voyage en ligne.

a) <u>Construction des ontologies de domaine et contextuelle</u>:

Ont été construites à l'aide de l'outil Protégé tout en signalant que l'assistance des experts de domaine est obligatoire pour une meilleure réalisation de ces ontologies. Fig. IV.7 est une capture d'écran de l'ontologie contextuelle rattachée au concept « Price » de l'ontologie de domaine.

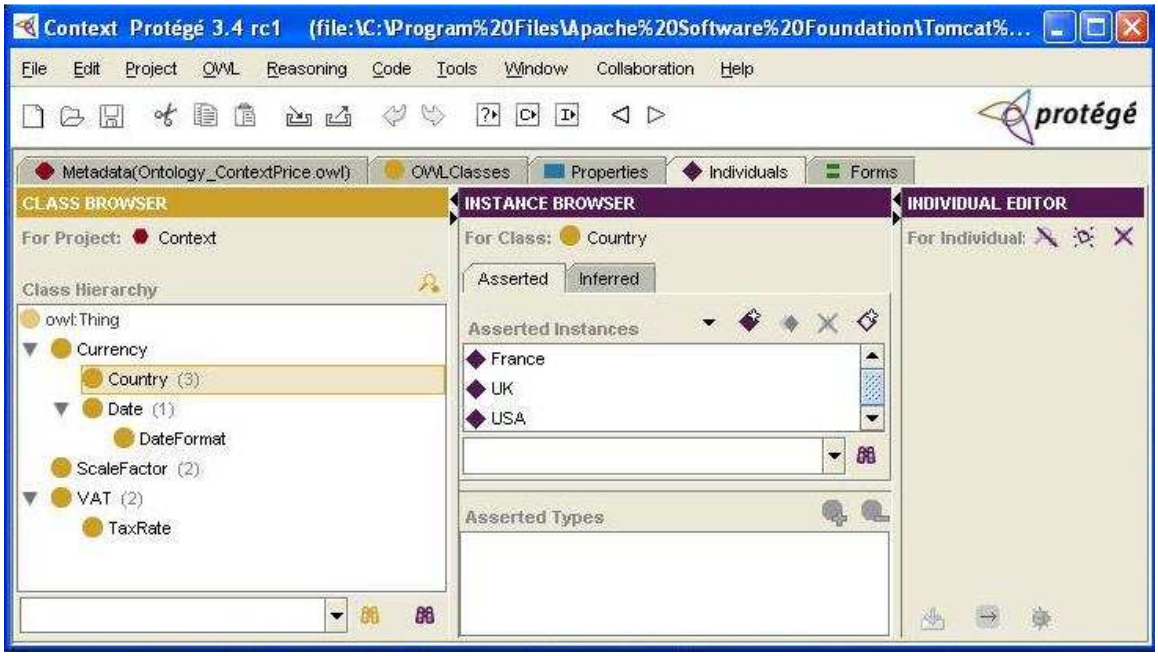
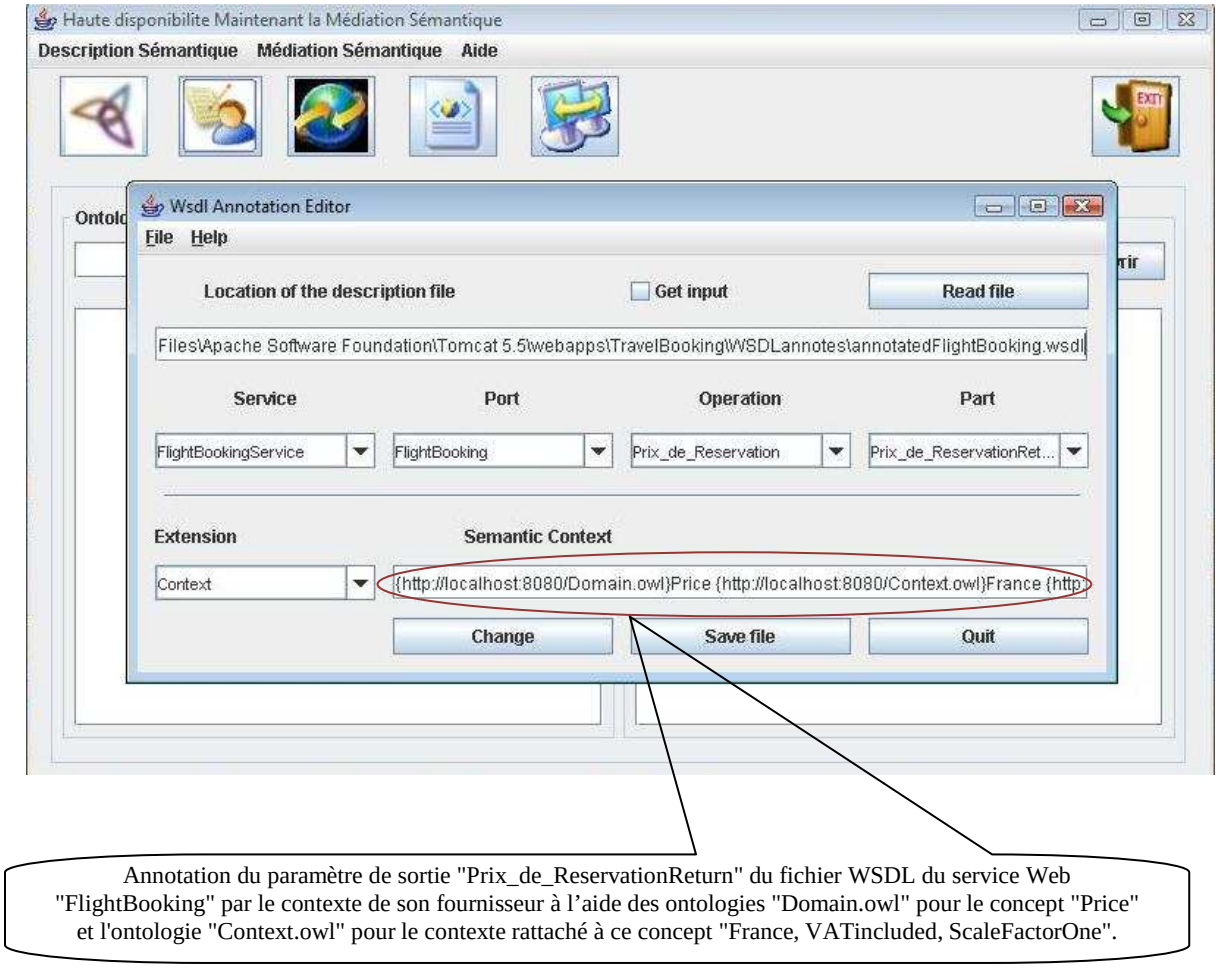


Fig. IV.7 – Ontologie contextuelle réalisée par Protégé

b) Annotation des services Web:

L'annotation des fichiers WSDL des services Web (FlightBooking, MasterFlightBooking et UKFlightBooking) est réalisée par l'éditeur d'annotation de Mrissa et al [31]. Cette annotation concerne uniquement les modifieurs statiques étant donné que les modifieurs dynamiques seront automatiquement déduits à partir de ces modifieurs statiques en utilisant des règles logiques comme susmentionnés dans la section IV.3.1.2. Cette annotation est faite par l'ajout des noms des instances de ces modifieurs statiques dans l'ontologie contextuelle où des espaces de nommage[75] sont ajoutés dans les déclarations du document WSDL afin d'identifier les termes utilisés par les annotations. Une capture d'écran de l'annotation du service FlightBooking est donnée dans Fig. IV.8.



Annotation du paramètre de sortie "Prix_de_ReservationReturn" du fichier WSDL du service Web "FlightBooking" par le contexte de son fournisseur à l'aide des ontologies "Domain.owl" pour le concept "Price" et l'ontologie "Context.owl" pour le contexte rattaché à ce concept "France, VATincluded, ScaleFactorOne".

Fig. IV.8 – Annotation du fichier WSDL "annotatedFlightBooking.wsdl"

[75] "ctxt1" pour référencer l'ontologie de domaine et "ctxt2" pour celle contextuelle.

c) Extraction et Adoption du contexte:

Après l'échec du service Web FlightBooking, et après téléchargement en mémoire des modèles de document WSDL des services Web de celui échoué et de celui du Maître MasterFlightBooking, l'on annote le contexte de ce dernier par celui du service FlightBooking.

Fig. IV.9 montre deux fichiers WSDL avant de procéder à l'extraction et l'adoption du contexte du service échoué, et Fig. IV.10 détaille les étapes de cette opération d'adoption contextuelle[76] (écrans 1, 2 et 3).

Après, l'on sauvegarde le modèle en mémoire modifié par la nouvelle annotation dans le fichier WSDL du MasterFlightBooking (écran 4).

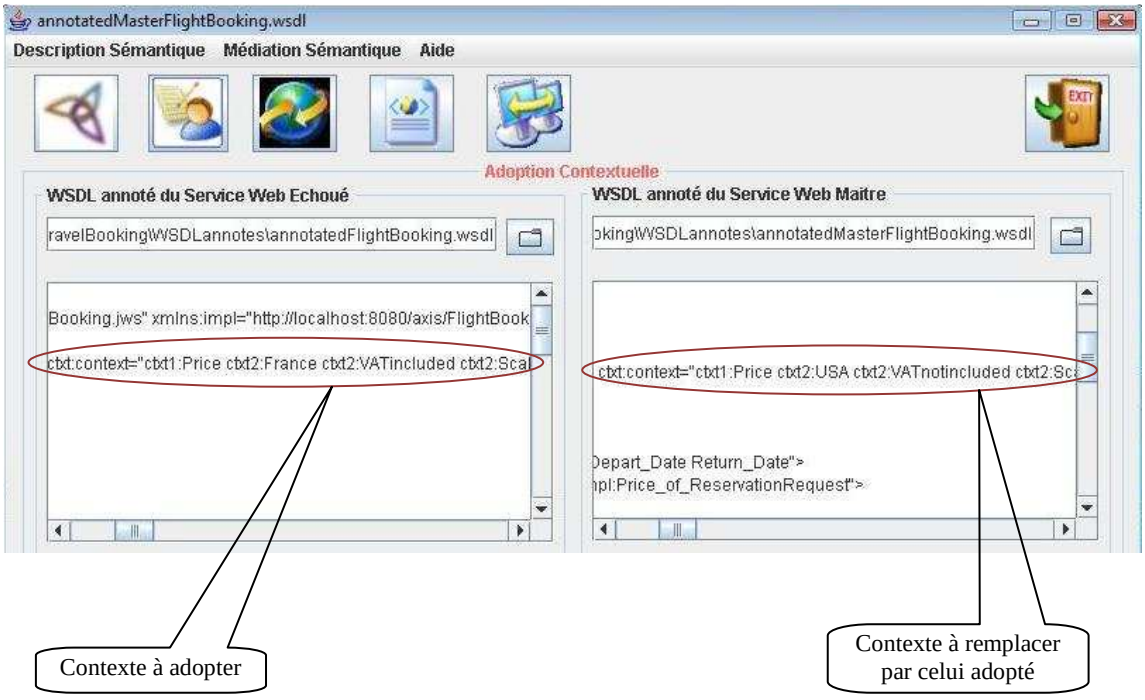


Fig. IV.9 – Les documents WSDL de "FlightBooking et MasterFlightBooking"

[76] Dans l'écran 3, la valeur du Facteur multiplicateur(ScaleFactorOne) est commune pour les deux contextes. C'est le cas de notre exemple. Par contre, elle peut être différente dans d'autres cas (Par exemple concernant le service Web japonais HotelBooking de notre scénario).

Fig. IV.10 – Etapes d'extraction et d'adoption contextuelle

d) Conversion contextuelle:

Après l'exécution de la fonctionnalité du service défaillant FlightBooking par le service Web remplaçant UKFlightBooking, et avant de retourner le prix de la réservation aux autres services Web de la composition, le module de médiation doit convertir cette dernière à son contexte qui est à son tour le contexte du service FlightBooking échoué.

A cet effet, et après avoir lu et parcouru les étapes de processus de médiation explicité dans IV.3.1.2 on aura en mémoire deux objets sémantiques pour le concept « Price » (voir Fig. IV.11) représentant le contexte émetteur du UKFlightBooking et contexte destinataire celui du Maître MasterFlightBooking (contexte du FlightBooking échoué). Notre module doit réaliser les conversions nécessaires entre ces deux contextes à l'aide de fonctions de conversions.

Pour des raisons de simplicité l'on a fixé, dans notre application, ces fonctions de conversion[77] ainsi que les règles logiques pour déduire les modifieurs dynamiques à partir des modifieurs statiques. Par exemple, la fonction qui permet de calculer la valeur d'un prix avec la devise de destination consiste à multiplier sa valeur source par le taux de change actuel.

[77] On peut utiliser le moteur d'inférence Drools (http://www.drools.org/) pour stocker ces fonctions dans sa base de connaissances.

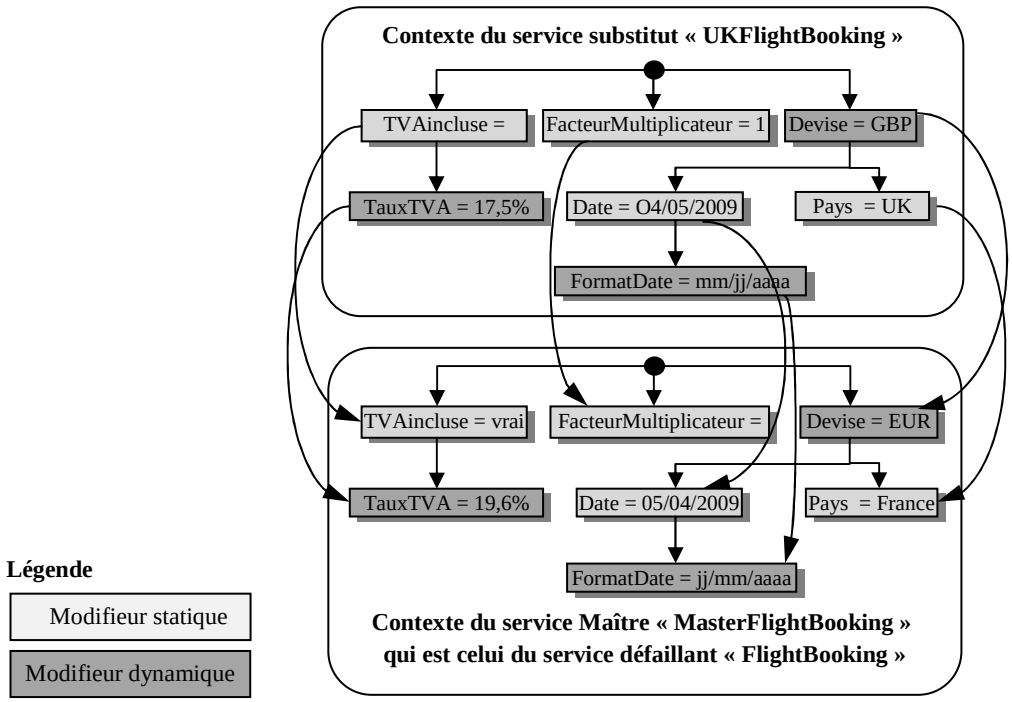


Fig. IV.11 – Conversion entre deux contextes de l'objet sémantique "Price"

Si l'on prend la valeur du prix retournée par le service UKFlightBooking d'un montant de **1200£**. Cette valeur, et après les conversions nécessaires[78], sera de **1575,20**€ dans le contexte du service Web Maître qui est le contexte du service défaillant (voir Fig. IV.12). Par cela, il n'aura aucun conflit sémantique au niveau du processus métier de notre composition et les autres services Web composant cette composition ainsi que les services Web médiateurs initialement insérés considèreront cette valeur comme elle est venue du service échoué FlightBoking dont l'échec est transparent pour eux.

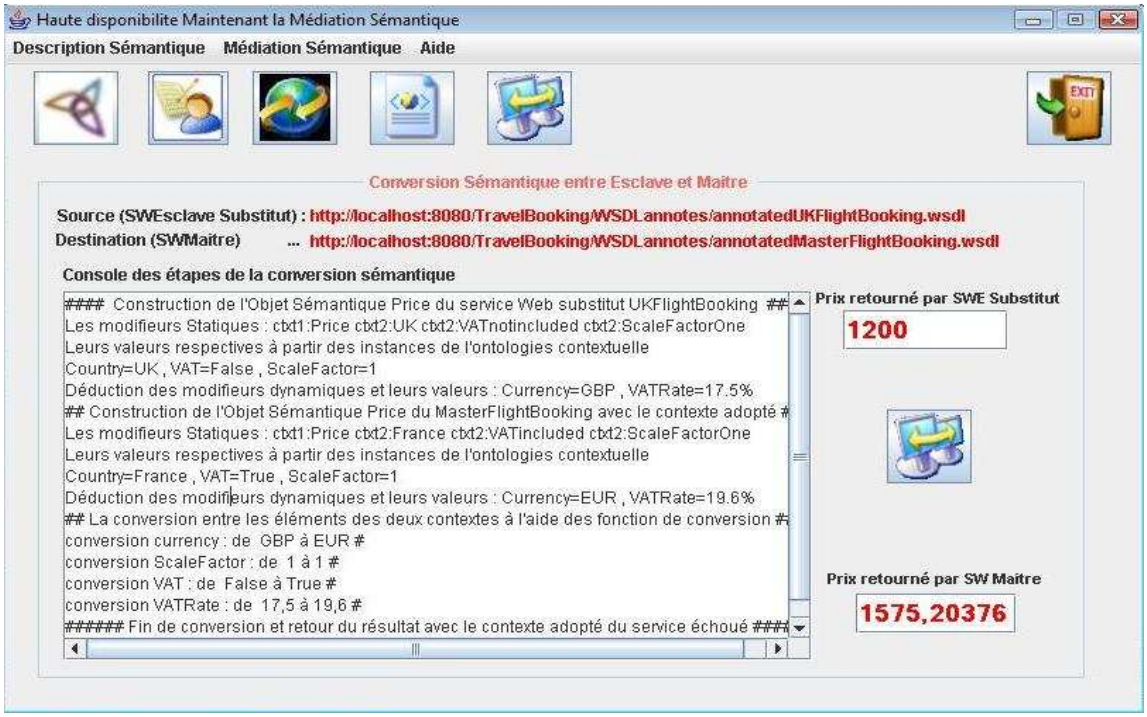


Fig. IV.12 – Conversion contextuelle entre « UKFlightBooking et MasterFlightBooking »

[78] Après l'application du taux de change du 04/05/2009 (1 GBP=1.9755 EUR) et l'inclusion de la TVA de faux au vrai avec un taux de 19,6%.

IV.5 Conclusion

Dans ce chapitre, nous avons présenté un processus de médiation sémantique comme solution quand à la substitution d'un service Web échoué par un autre fonctionnellement équivalent appartenant à une communauté et cela, pour soutenir la haute disponibilité du service échoué dans la composition dont les conflits sémantiques entre ses services composants ont déjà été résolus auparavant par l'approche de la médiation sémantique orientée contexte. Cette solution consiste, en premier lieu, d'exiger au service Web Maître de la communauté doté du module de médiation d'adopter la sémantique du service Web défaillant représentée par son contexte, et ensuite son module de conversion doit effectuer les échanges nécessaires entre les différents contextes des services Web Esclave substitut et Maître de la communauté.

Notre solution a été démontrée par une application sur notre scénario d'illustration pour prouver sa faisabilité. Cette solution, en cas de substitution, permettra d'éviter toute inconsistance sémantique dans la composition.

Conclusion Générale

« Par la sagesse on n'entend pas seulement la prudence dans les affaires, mais une parfaite connaissance de toutes les choses que l'homme peut savoir ».

René Descartes

1 Résume de la contribution

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.

En revanche, les challenges scientifiques tels que la disponibilité sont toujours à relever. Dans ce mémoire, nous nous sommes intéressés à la disponibilité des services Web qui est un attribut de fiabilité et qui qualifie la promptitude d'une application. L'approche proposée par Maamar et al. basée sur le fonctionnement des communautés des services Web pour soutenir la haute disponibilité a démontré des solutions aux inconvénients posés par les solutions traditionnelles basées sur la réplication et qui se sont avérées très coûteuses. Cette approche consiste à la substitution du service échoué par un autre de même fonctionnalité membre d'une communauté. Mais à cause de ce service remplaçant qui a ses propres paramètres d'entrée/sortie sémantiquement différents de ceux du service défaillant, cette substitution est sujette à l'apparition des nouvelles hétérogénéités sémantiques dans une composition des services Web après qu'ils aient été résolus initialement. L'objectif recherché à travers la résolution sémantique est de permettre aux machines d'interpréter les données traitées et de saisir leur signification de manière automatique. Dans ce cadre et pour faire face à cette inconsistance sémantique provoquée par la substitution, notre contribution vient proposer un processus de médiation sémantique orienté contexte tout en tirant profit des travaux de Mrissa et al. dans la médiation sémantique entre les services Web. Ce processus consiste à exiger au Maître de la communauté soutenant le service Web défaillant, d'adopter la sémantique de ce service défaillant représentée par son contexte pour qu'il puisse faire ressortir le résultat après l'exécution de la fonctionnalité par le service substitut de la communauté, suivant la sémantique adoptée. Par cela, la réponse retournée est considérée comme venue du service Web échoué. Par conséquent, il n'y aura pas d'inconsistance sémantique au niveau de la composition tout en lui maintenant la médiation sémantique précédemment prise.

Notre travail de recherche a abouti à détailler les étapes de ce processus de médiation sémantique en cas de substitution du service Web échoué afin d'assurer sa haute disponibilité, basées essentiellement, comme première étape, sur l'extraction et l'adoption du contexte du service défaillant par le Maître de communauté soutenant ce dernier, en téléchargeant et en lisant par un lecteur de contexte WSDL les descriptions annotées du service Web défaillant, et ce, afin d'extraire les annotations de ses paramètres d'entrée/sortie. Ensuite, on procédera à l'annotation

des paramètres correspondants du service Web Maître de la communauté par ces annotations contextuelles extraites, de sorte que ce Maître devra convertir les requêtes et les réponses par ce contexte adopté du service Web échoué.

Comme deuxième étape, le flux de données envoyé au service défaillant dans le cadre de notre composition, sera adressé maintenant à la communauté concernée pour assurer la disponibilité du service, et l'implémentation de la fonctionnalité du défaillant va être faite par l'un des services de cette communauté qui accepte d'agir comme support (service primaire remplaçant), d'où une conversion entre les deux contextes des paramètres d'entrée/sortie des deux services Web, le service Esclave de support et le service Maître de la communauté, et cela à l'aide d'un module de médiation intégré au niveau du Maître de la communauté.

Par la suite, la réponse du service substitut convertie selon le contexte adopté par la communauté, plus précisément par son Maître, est transmise à la composition, où elle est traitée comme arrivée du service défaillant et par conséquent, il n y aura pas d'impact sémantique au niveau de la médiation sémantique initialement prise dans cette composition et il n y aura rien à changer à ce niveau.

2 Perspectives envisagées

Nous pensons que l'idée de notre solution qui consiste à faire adopter la sémantique du service Web échoué dans une composition des services Web par le Maître d'une communauté comme une sémantique pour elle, peut être appliquée à d'autres approches de médiation sémantique entre les services Web que celle de l'approche orientée contexte comme présentée dans ce mémoire, notamment les approches qui ont comme principe l'annotation des fichiers de description des données pour une prise en charge de la sémantique des paramètres d'entrée/sortie. A cet effet, toute application de ce principe sur une autre nouvelle approche de médiation sémantique donnera à notre solution une valeur ajoutée afin qu'elle soit recommandée dans l'approche de soutien de la haute disponibilité des services Web en utilisant les communautés.

Aussi, notre solution est à l'origine de la proposition d'une extension de l'architecture conceptuelle de la médiation sémantique dans les compositions des services Web proposée par Mrissa et al. [31,30], tout en lui ajoutant une couche représentant les communautés (Couche colorée en gris dans Fig. 1) comme standard pour sélectionner les services Web participants à des compositions selon Benslimane et al. dans [12]. L'on envisage, comme travaux futurs, de concrétiser cette architecture étendue.

Cette architecture étendue de médiation représentée par Fig. 1 est composée de quatre couches à savoir:

i. **Couche fournisseurs :** comprend les services disponibles, regroupés par fournisseur. Chaque fournisseur possède une sémantique locale, symbolisée dans l'architecture conceptuelle par une ontologie.

ii. **Couche communautés :** recueille tous les services Web de la même fonctionnalité qui peuvent être de différents fournisseurs. Cette couche est importante afin d'adopter une

seule sémantique par communauté pour l'ensemble des services Web adhérant à cette dernière, de sorte qu'au niveau sémantique, la composition des services Web devient la composition des communautés.

iii. **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.

iv. **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 du domaine.

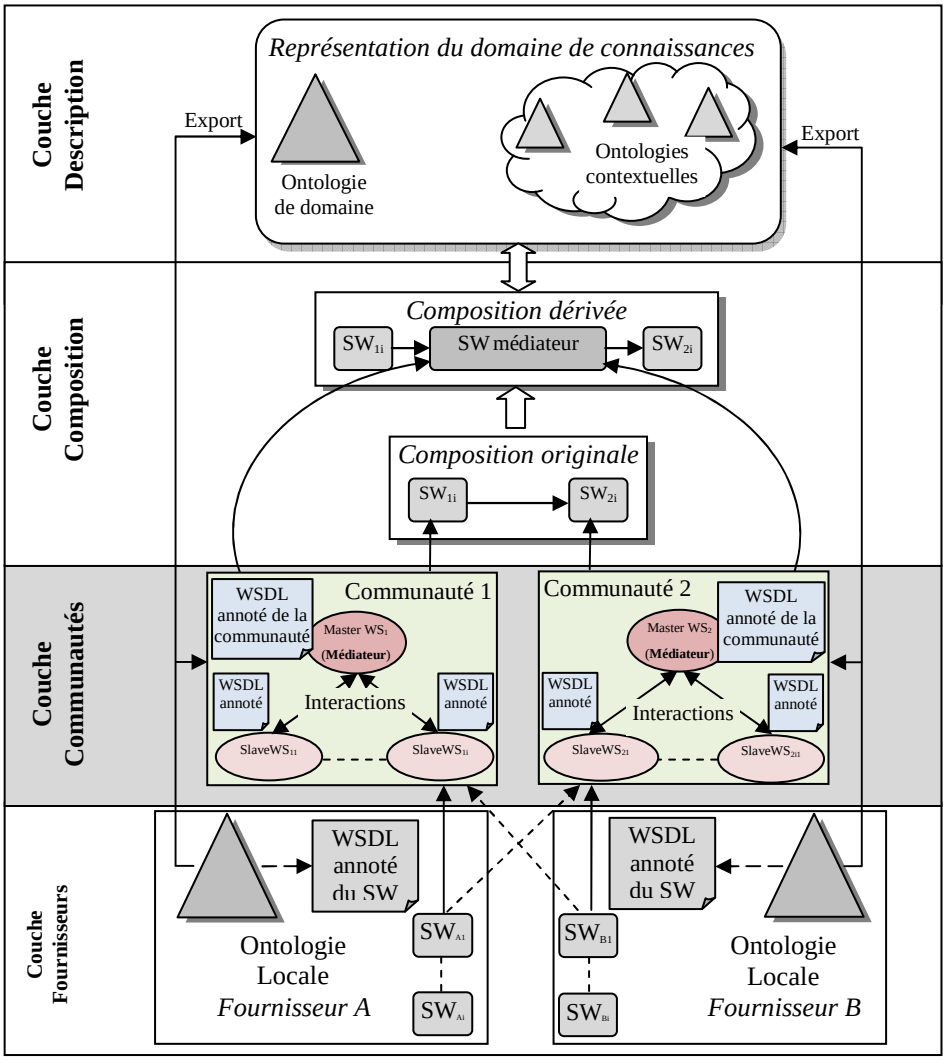


Fig. 1 – Architecture étendue de médiation sémantique

3 Conclusion

Ce travail a été réalisé après réflexion sur la levée de challenge sémantique de l'approche de soutien de la haute disponibilité des services Web par les communautés en se fondant sur la substitution du service défaillant par un autre fonctionnellement équivalent adhérant à une communauté garantissant la continuité dans la fourniture du service. Nous avons présenté une solution orientée contexte pour faire face à l'impact sémantique qui peut être provoqué lors de cette substitution et cela, pour pouvoir, dans une composition où les hétérogénéités sémantiques entre ses services composants ont été résolus auparavant par une architecture de médiation sémantique, soutenir la haute disponibilité du service. Cet impact a surgi à cause de la différence qui peut exister, au niveau sémantique, entre les deux contextes différents des services défaillant et substitut.

Cette solution consiste, avant de procéder à la conversion entre les différents éléments de contextes, à exiger au Maître de la communauté doté par le module de médiation d'adopter la sémantique du service Web défaillant, plus précisément son contexte pour, en cas de substitution, éviter de rompre la médiation sémantique dans la composition initialement prise.

L'avantage principal de cette solution est d'échapper au maximum des conflits sémantiques qui peuvent apparaître dans une composition des services Web en cas de substitution.

Notre contribution permettra à l'approche de la haute disponibilité des services Web basée sur les communautés de gagner le challenge de tout conflit sémantique qui peut être provoqué après la substitution des services Web défaillants.

Bibliographie

[1] A. Aït-Bachir. "ArchiMed : un canevas pour la détection et la résolution des incompatibilités des conversations entre services Web", Thèse de Doctorat à l'université Université Joseph Fourier - Grenoble 1, France. 28 Novembre 2008.

[2] Apache Software Foundation. http://tomcat.apache.org/, version 5.5.27, (dernière visite: 01 Avril 2009).

[3] Apache Software Foundation. Axis. http://ws.apache.org/axis/, version 1.4, (dernière visite: 01 Avril 2009).

[4] A. Avizienis, J.C. Laprie, B. Randell and C. Landwehr. "Basic Concepts and Taxonomy of Dependable and Secure Computing", IEEE, 2004.

[5] A. Bucchiarone and S. Gnesi. "A Survey on Services Composition Languages and Models", International Workshop on Web Services Modeling and Testing (WS-MaTe 2006), http://www.selab.isti.cnr.it/ws-mate/Bucchiarone_WS-MaTe.pdf.

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

[7] B. Benatallah, Z. Q. Sheng and M. Dumas. "The Self-Serv Environment for Web Services Composition", IEEE, 2003.

[8] B. Boudaa. "Vers une médiation sémantique pour les compositions des services Web dont les disponibilités sont soutenues par les communautés", In Proceedings of WWS'2009, CERIST, Alger, Algérie, 21 et 22 Février 2009.

[9] B. Medjahed and A. Bouguettaya. "A Dynamic Foundational Architecture for Semantic Web Services", Distributed and Parallel Databases, Kluwer Academic Publishers, 2005.

[10] C. Ghedira, Z. Maamar and D. Benslimane. "On Composing Web Services for Coalition scenarios - Concepts and Operations". Information & security Journal, 2005.

[11] C. Jean-Marie. "Services Web avec SOAP, WSDL, UDDI, ebXML", Editions Eyrolles, 2002.

[12] D. Benslimane, Z. Maamar, Y. Taher, M. Lahkim, M.C. Fauvet and M. Mrissa. "A Multi-Layer and Multi-Perspective Approach to Compose Web Services", (AINA'2007), Niagara Falls, Ontario, Canada, 2007.

[13] D. Booth, H. Haas, F. McCabe, E. NewCorner, M. Champion, C. Ferris and D. Orchard. W3C working group note – "Web services architecture", February 2004.

[14] 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/.

[15] D. Harel and A. Naamad. "The STATEMATE Semantics of Statecharts", ACM Transactions on Software Engineering and Methodology, Vol. 5 No 4, pp. 293-333.

[16] 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.

[17] 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.

[18] F. Hadjila et M.A Chikh. "Une approche orientée agent pour la composition sémantique des services Web", LANIA'2007, Chlef, Algérie.

[19] G. Julien. "Planification multi-agent pour la composition dynamique de services Web", Rapport de stage – Master 2 Recherche, Université Joseph Fourier – Grenoble 1, le 12 Juin 2006.

[20] H. Simon. "Creating Web Farms with Linux (Linux High Availability and Scalability)", December 2001, For Presentation in Tokyo, Japan.
[21] J. Bentahar, Z. Maamar, D. Benslimane and P. Thiran. "Using Argumentative Agents to Manage Communities of Web Services", (WAMIS'2007), Niagara Falls, Ontario, Canada, 2007.
[22] Jena 2 A Semantic Web Framework, http://jena.sourceforge.net/, version 2.5.7. (dernière visite: 01 Avril 2009).
[23] J. Osrael, L. Froihofer, and K. Goeschka. "What Service Replication Middleware Can Learn from Object Replication Middleware", (MW4SOC'2006), Melbourne, Australia, 2006.
[24] J. Salas, F. Pérez-Sorrosal, M. Patiño Martínez, and R. Jiménez-Peris." WS-Replication: A Framework for Highly Available Web Services".(WWW'2006),Edinburgh, Scotland, 2006.
[25] K. Birman, R.van Renesse and W. Vogels. "Adding High Availability and Autonomic Behavior to Web Services", In Proceedinds of the 26th International Conference on Software Engineering (ICSE'2004), IEEE Computer Society, Edinburgh, Scotland, pp.17-26.
[26] K. H. Kim (Kane). "Issues Insufficiently Resolved in Century 20 in the Fault-Tolerant Distributed Computing Field", In Proceedings of the 19th IEEE Symposium on Reliable Distributed Systems (SRDS'2000), IEEE Computer Society, Nurnberg, Germany, pp. 106-115.
[27] K. Hubert et M. Valérie. "Les Web services - Techniques, démarches et outils : XML-WSDL-SOAP-UDDI-Rosetta-UML", Edition Dunod 13 mars 2003.
[28] M. Mrissa, C. Ghedira, D. Benslimane, Z. Maamar, and J. Fayolle. "A mediation framework for Web services in a peer-to-peer environment". (AICCSA'2005), Cairo, Egypt, 2005.
[29] M. Mrissa, C. Ghedira, D. Benslimane and Z. Maamar. "A context model for semantic mediation in Web services composition", Springer, 2006.
[30] M. Mrissa. "Médiation Sémantique Orientée Contexte pour la Composition de Services Web", Thèse de Doctorat à l'université Claude Bernard Lyon I, France. Novembre 2007.
[31] M. Mrissa, C. Ghedira, D. Benslimane, Z. Maamar, F. Rosenberg and S. Dustdar. "A Context-based Mediation Approach to compose semantic Web services", ACM, 2008.
[32] M. Mrissa, P. Thiran, C. Ghedira, D. Benslimane and Z. Maamar. "Using Context to Enable Semantic Mediation in Web Service Communities", (CSSSIA'2008), Beijing, China, 2008.
[33] M. Mrissa, S. Dietze, P. Thiran, C. Ghedira, D. Benslimane, Z. Maamar. "Context-based Semantic Mediation in Web Service Communities", in book "Weaving Services, Location, and People on the WWW", Springer, 2009 (à paraitre).
[34] M. Wiesmann, F. Pedone, A. Schiper, B. Kemme, and G. Alonso. "Understanding Replication in Databases and Distributed Systems", (ICDCS'2000), Taipei, Taiwan, 2000.
[35] N. Christophe. "Définition des services-Web, Application aux IFC", Edition LE2I – Université de Bourgogne – CNRS, 11 Octobre 2002.
[36] N. Laranjeiro and M. Vieira. "Towards Fault Tolerance in Web Services Compositions". In Proceedings of The 2nd International Workshop on Engineering Fault Tolerant Systems (EFTS'2007) held in conjunction with The 6th Joint Meeting of the European Software Engineering Conference and The ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE'2007), Dubrovnik, Croatia, 2007.
[37] N. Milanovic. "Contract-based Web Service Composition", Thèse de Doctorat, Université Humboldt – Berlin, 13 juin 2006.

[38] P. R. Lorczak, A. K. Caglayan and D. E. Eckhardt. "A Theoretical Investigation of Generalized Voters for Redundant Systems", Digest of Papers from the Nineteenth International Symposium on Fault-Tolerant Computing, pp. 444-451.
[39] P. Vanneau, M. Samson et J. Donel. "JINI – SOAP : Présentation, SOAP", 3WC, http://www.w3schools.com/soap/soap_fault.asp et http://www.w3.org/TR/soap/.
[40] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext transfer protocol – HTTP/1.1, 1999.
[41] R.G. Smith. "The contract Net Protocol: High Level Communication and Control in Distributed Problem Solver", IEEE, 1980.
[42] S. Benmokhtar. "Synthese ad-hoc de services Web dans les environnements de l'informatique diffusé", Rapport de stage, DEA Systèmes Informatiques Repartis(SIR), Université Pierre et Marie Curie, 2004.
[43] S. drapeau et L. Claudia Roncancio. "Concepts et techniques de duplication", France Telecom R&D / Laboratoire IMAG – LSR, Grenoble, France, http://www-lsr.imag.fr/Les.Publications/conceptsTechniquesDuplication.ps.
[44] S. Subramanian. "Highly-Available Web Service Community", in Proceedings. of the 6th International Conference onInformation Technology : New Generations, IEEE Computer Society Press, Las Vegas, USA, 2009.
[45] T. Bray, J. Paoli, and C. M. Sperberg-McQueen. "Extensible markup language (xml)", World Wide Web Journal, 2(4) :27–66, 1997.
[46] T. Gruber. "What is an ontology?", 2000 ,http://www-ksl.stanford.edu/kst/what-is-an-ontology.html.
[47] T. Melliti. "Interopérabilité des services Web complexes. Application aux systèmes multi-agents", Thèse de Doctorat, Université Paris IX Dauphine, 8 décembre 2004.
[48] UDDI Specification Technical Committee, "Universal Description, Discovery, and Integration of Business for the Web. Technical report", October 2001. http://www.uddi.org.
[49] V. Kashyap and A. Sheth. "Semantic and Schematic Similarities Between Database Objects : A Context-Based Approach", The Very Large Data Base Journal, 5(4), 1996.
[50] W. He. "Recovery in Web Service Applications", In Proceedings of The IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE'2004), IEEE Computer Society, Washington, DC, USA, pp. 25-28.
[51] WSDL4J Project Homepage. "Web Services Description Language for Java" Toolkit (WSDL4J), version 1.6.2, http://sourceforge.net/projects/wsdl4j (dernière visite: 01 Avril 2009).
[52] W3C Working Group. "eXtensible Markup Language (XML) 1.0", 2006. http://www.w3.org/TR/2006/ REC-xml-20060816/.
[53] W3C Working Group. XML Schema Part 2 : Datatypes Second Edition. Technical report, October 2004. http://www.w3.org/TR/xmlschema-2/.
[54] X. Ye and Y. Shen. "A Middleware for Replicated Web Services", (ICWS'2005), IEEE, Orlando, Florida, USA, 2005.
[55] Y. Hamdi, Z. Maamar, J. Bentahar, N. Sahli, S. Elnafar and P. Thiran. "On the Reputation of Communities of Web Services", in Proceedings of the 8th International Conference on New Technologies of Distributed Systems, collection ACM International Conference Proceedings Series, Lyon, 2008.
[56] Y. Taher, D. Benslimane, M.C. Fauvet and Z. Maamar. "Towards an Approach for Web Services Substitution", (IDEAS'2006), IEEE, Delhi, India, 2006.
[57] Z. Maamar, M. Lahkim, D. Benslimane, P. Thiran and S. Sattanathan. "Web Services Communities - Concepts & Operations", (WEBIST'2007), Barcelona, Spain, 2007.

[58] Z. Maamar, Z. Quan Sheng and D. Benslimane "Sustaining Web Services High Availability Using Communities", The 3rd International Conference on Availability, Reliability, and Security (ARES 2008), Barcelona, Spain, Mar 4-7, 2008.
[59] Z. Maamar, Z. Quan Sheng, S. Tata, D. Benslimane and M. Sellami. "Towards An Approach to Sustain Web Services High-Availability Using Communities of Web Services", International Journal of Web Information Systems (IJWIS), 2008 (à paraitre).
[60] Z. Maamar, S. Subramanian, P. Thiran, D. Benslimane, and J. Bentahar. "An Approach to Engineer Communities of Web Services: concepts, architecture, operation, and deployment", International Journal of E-Business Research, 5(4), 2009 (à paraitre).