République **A**lgérienne **D**émocratique et **P**opulaire
Ministère de l'**E**nseignement **S**upérieur et de
la **R**echerche **S**cientifique
Université Ibn Khaldoun – Tiaret –

Faculté des Sciences et de la Technologie et Sciences de la Matière

Département d'**I**nformatique

MEMOIRE EN VUE DE L'OBTENTION DU DIPLÔME DE MAGISTER

SPECIALITE : **I**nformatique
OPTION : **S**ystèmes d'**I**nformation et de **C**onnaissances (SIC).

Présenté par :
HAMDANI Mostefa

SUJET DU MEMOIRE :

Service Web & Base de données.
Optimisation de la composition des DaaS par Invocations Parallèles de Services Web.

SOUTENU LE 03/11/2012 Devant Le Jury Composé de :

Mr. Amar BALLA	Professeur	ESI - Alger	Président
Mr. Djillali BEHLOUL	MC -A-	Université USTHB-Alger	Examinateur
Mr. Youcef DAHMANI	MC -A-	Université Ibn Khaldoun- Tiaret	Examinateur
Mr. Youcef AKLOUF	MC -A-	Université USTHB Alger	Directeur de mémoire

ANNEE UNIVERSITAIRE 2011/2012

ملخص

نتحدث من خلال هذه المذكرة على تركيب نوع من خدمات الويب التي تهتم بالبحث عن المعطيات الموجودة في قواعد بيانات مختلفة و موزعة. إن الاتصال و طلب الخدمة من الويب سيرفس ينتج عنه طلب معلومات من مصادر البيانات، وهو ما يحتاج في الغالب إلى تركيب وإدماج مجموعة من خدمات الويب (ويب سيرفس) .

الطرق الكلاسيكية الموجودة حتى الآن، والمستعملة في تركيب خدمات الويب لا تأخذ بعين الاعتبار المعنى أو المدلول الذي يمكن أن يربط بين معطيات الادخال و الاخراج. لذلك يقترح بعض الباحثين طرقا وبرامج جديدة توصل إلى هذا المبتغى .

اقتراحنا في هذه المذكرة يتمثل في تسريع عملية البحث عن المعلومات ، و تركيب خدمات الويب وفق أنماط متوازية، وتطبيق هذا المبدأ على الهندسة المقترحة من طرف برهامجي و آخرين في أبحاثهم المتعلقة بهذا الميدان .

كلمات البحث : خدمات ويب، تركيب خدمات الويب ، خدمات ويب الوصول إلى البيانات، إدماج البيانات، التوازي.

Résumé

Dans ce mémoire, nous intéressons à la composition de service web DaaS (DaaS Data-as-a-Service Web services) pour les besoins de partage de données dans les environnements distribués.

L'invocation d'un Service web DaaS a comme conséquence l'exécution d'une requête au-dessus des sources de données. Dans la plupart des cas, les requêtes des utilisateurs exigent la composition de plusieurs services; les méthodes de composition de services web classiques ne permettent pas de saisir le rapport sémantique entre les entrées/sorties d'un service web DaaS, et en conséquence, elles ne sont pas adaptées pour composer ce type de service. Certains chercheurs proposent des nouvelles architectures permettant de saisir cette sémantique.

Notre proposition vise l'optimisation de la composition de DaaS et la recherche d'informations en implémentant des appels parallèles de service web DaaS dans l'architecture de Barhamgi et al.

Mots clés: Services Web DaaS, Services Web d'accès aux données, composition des services Web, intégration des donnés, parallélisme.

Abstract

We are interested in the composition of web service DaaS (Data-as-a-Service Web services) for the purposes of data sharing in distributed environments.

Invoking a Web Service DaaS has as result the execution of a query over data sources. In most cases, users' queries require the composition of several services, standard methods of web service composition does not capture the semantic relationship between the input / output of a Web service DaaS, and consequently they are not suitable for this type of service call. Some researchers propose new architectures to capture this semantic.

Our proposal is to add the concept of parallelism to the architecture of Barhamgi and al. to optimize the composition of web services DaaS and information retrieval.

Keywords: DaaS Web services, Web services data access, Web services composition, data integration, parallelism.

Remerciements

Je tiens à remercier sincèrement Mr. AKLOUF Youcef, qui, en tant que Directeur de mémoire, s'est toujours montré à l'écoute et très disponible tout au long de la réalisation de ce mémoire, ainsi pour l'inspiration, l'aide et le temps qu'il a bien voulu me consacrer et sans qui ce mémoire n'aurait jamais vu le jour.

Je tiens également à remercier Mr. BARHAMGI Mahmoud pour son aide et ses conseils.

Mes remerciements s'adressent également au Professeur Balla Amar qui a bien voulu accepter de présider le jury de mon mémoire.

J'adresse également mes remerciements à Mr. DAHMANI Youcef pour son aide, sa gentillesse et son soutien tout au long de mes études universitaires et pour avoir accepté de faire partie de ce jury.

Je tiens également à remercier Mr. BEHLOUL Djillali pour avoir accepté de faire partie de ce jury, et pour juger mon travail.

Un immense merci à FEKIER A, DEKKICHE A, YOUSEFI Z et DERDJINI D qui m'auront soutenu dès le début et m'auront aidé dans les périodes de doute.

Enfin, j'adresse mes plus sincères remerciements à mes proches et mes amis pour les moments partagés, ainsi que ma famille, et surtout mes parents, sans qui je ne serai rien.

A la mémoire de mon père

A ma mère

A ma femme et mon fils Ahmed

A mes frères et sœurs

A tous mes amis

Table des matières

Table des figures

Liste des tableaux

Introduction générale :

L'une des tendances historiques qui a conduit à l'apparition des services web est l'utilisation de l'architecture par composants comme approche d'intégration d'applications. L'architecture de composants distribués a engendré un développement rapide et évolutif d'applications distribuées et complexes. Parmi les architectures qui ont été mise en place, on peut citer: CORBA, EJB et COM. La mise en œuvre de ces trois architectures soulève des difficultés dans le cadre d'une infrastructure ouverte telle qu'Internet.

En effet, ces architecture, bien qu'utilisant un modèle objet distribué, proposent chacune sa propre infrastructure. Ce qui impose une forte liaison entre les services offert par les composants et leurs clients. Ainsi on ne peut assembler que des objets CORBA (ou COM) entre eux. Les systèmes construits à bases de ces architectures sont monolithiques. Après l'avènement du B2C (Business To Customer), les entreprises mettent en ligne leurs services pour leurs consommateurs à travers des applications web. Le B2B repose sur l'échange de produits, d'informations et de services entre entreprises. Ceci implique l'utilisation de services et la collaboration avec des systèmes proposés par d'autres partenaires et par conséquent il exige une maitrise d'hétérogénéité et d'interopérabilité. C'est justement ce que les services Web apportent par rapport aux solutions dites monolithiques.

Les services Web sont un type d'architecture reposant sur les standards de l'Internet. Ils permettent à des applications de communiquer directement entre elles sans qu'elles se préoccupent des technologies d'implantation utilisées.

L'intégration de sources d'information est devenue un domaine de recherche très important à cause de l'explosion du nombre de sources disponibles via le Web. Il s'agit d'un processus par lequel plusieurs sources de données autonomes, réparties et sous forme hétérogène (où chaque source est associée à un schéma local) sont intégrées sous forme de source unique représentée par un schéma global.

La diversité des sources d'information distribuées et leur hétérogénéité est l'une des principales caractéristiques du web d'aujourd'hui. Dans des environnements comme l'eHealth, eGov… etc., l'accès à un nombre important de sources de données se fait par des services web qui permettent d'accéder et de rechercher des données élémentaires tenues par

des sources hétérogènes autonomes indépendamment des systèmes et des plates-formes utilisées.

Ce type services est appelé DaaS (Data-as-a-Service), dans des travaux de recherche, est appelé: IaaS Services (Information-as-a-Service) Services, DaaS (Data-as-a-Service) Services, Data-Providing Services, Stateless Services, Information-Providing Services, ou simplement Data Services.

Les Services Web DaaS sont maintenant utilisés dans de nombreux domaines d'application comme un moyen standard pour le partage des données: par exemple le partage des données scientifiques (la bio-informatique, traitement et partage de données géo-spatiales, etc.), le partage des données médicales (eHealth), le partage des données entre les organismes gouvernementaux (eGovernment), etc.).

Problématique

La composition de service Web permet de répondre aux besoins d'un utilisateur ne pouvant être satisfaits par un seul service web, alors qu'une intégration de plusieurs le permettrait. Les méthodes de composition, telles qu'elles sont appliquées aux services web traditionnels (i.e. *AaaS Application-as-a-Service* Web services), ne permettent pas de prendre en compte la relation sémantique entre les entrées/sorties d'un service web d'accès aux données, et en conséquence, elles ne sont pas adaptées pour composer les services Web DaaS. Certains chercheurs (Barhamgi et al. [BAR10]) proposent des architectures permettant de saisir cette sémantique

L'invocation d'un Service web DaaS a comme conséquence l'exécution d'une requête au-dessus des sources de données. Dans la plupart des cas, les requêtes des utilisateurs exigent la composition de plusieurs services.

Le défi est d'accélérer les requêtes de tels appels de services web.

Contribution

Des invocations parallèles de services web DaaS peuvent accélérer l'exécution des requêtes et optimiser le temps d'exécution.

Notre proposition vise l'optimisation de la composition de DaaS et la recherche d'informations en implémentant des appels parallèles de service web DaaS dans l'architecture de (Barhamgi et al. [BAR10]).

Ce mémoire est organisé en quatre chapitres :

- **Chapitre 01 -** Services Web: Ce chapitre est consacré à la description de l'architecture orienté services SOA et de la technologie des services web ainsi les langages qui présentent les différentes couches de l'architecture de référence des services web seront décrits, nous présentons aussi la technologie des services web sémantiques et ses composants. Une synthèse est faite sur les standards de cette infrastructure, enfin nous allons présentons quelques outils logiciels disponible permettant d'assister le développeur durant le cycle de vie du SWS.

- **Chapitre 2 -** Etat de l'art : Nous proposons dans ce chapitre une présentation du contexte dans lequel s'inscrit notre travail, à savoir la notion de composition de services Web, l'intégration des données et les services web DaaS.

- **Chapitre 3 -** Conception : Dans ce chapitre, nous présenterons notre approche permettant d'optimiser la composition de DaaS et la recherche d'information par implémentation des appels parallèles de DaaS.

- **Chapitre 04** - Implémentation: Dans ce chapitre, nous présenterons notre plateforme de test de la composition de services web d'accès aux données et l'architecture de notre application, à travers une étude des méthodes et des services web utilisés pour invoquer les DaaS en parallèle.

  Enfin, nous terminons ce mémoire par une conclusion générale dans laquelle nous discutons notre travail et nous proposons quelques perspectives de recherche que nous souhaitons les réaliser dans le futur.

Chapitre I

Les Services Web

I. 1. Introduction

Dans ce chapitre, nous présenterons l'architecture orientée services et sa principale réalisation: les services web. Nous aborderons le concept d'un service web, ses définitions et ses déférents standards; ensuite nous allons détailler le service web sémantique et quelques outils développés dans le cadre de cette technologie.

L'objectif de la notion de service est de promouvoir un accès simple et rapide aux fonctionnalités mises à disposition par les organisations [MRI07]. Il s'agit d'une technologie née à la fin des années quatre-vingt-dix, sous l'impulsion de géants de l'informatique comme Microsoft, IBM, Sun ou encore SAP, elle est en grande partie basée sur les technologies XML.

I. 2. Architecture orientée services

En parallèle à l'évolution des outils basés sur Internet et à l'interopérabilité des données et logiciels, l'architecture des applications a fortement évolué. L'architecture est passée de la programmation modulaire à la programmation orientée objet, puis à l'architecture orientée composant et enfin à l'architecture orientée service. L'architecture orientée service ou SOA (*Service-Oriented Architecture*) est aujourd'hui envisagée par de nombreuses entreprises dans le cadre de l'évolution de leur système d'information. Elle a été mise en avant afin de permettre des interactions entre applications distantes.

La définition suivante est proposée par OASIS (*Organization for the Advancement of Structured Information Standards*) [OAS09] :

Définition : L'architecture orienté service est un paradigme pour organiser et utiliser des capacités distribuées qui peuvent être sous contrôle de différents propriétaires de domaines. Elle fournit des moyens pour offrir, découvrir, interagir et utiliser les capacités pour produire les effets souhaités consistants aux attentes et pré–conditions prévisibles.

Elle s'articule autour des trois éléments : Le fournisseur de service : le client et L'annuaire de services.

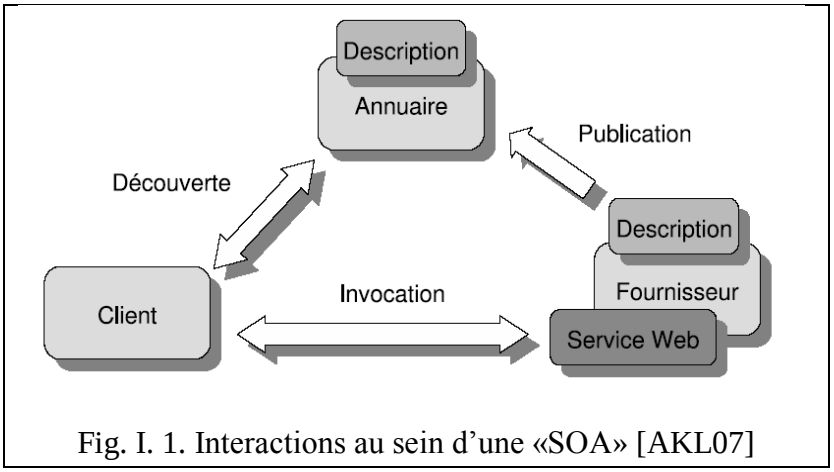

Fig. I. 1. Interactions au sein d'une «SOA» [AKL07]

Les architectures orientées service se construisent autour de plusieurs protocoles et langages, selon quatre couches de fonctionnalités, à savoir : [MRI07]

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

I. 3. Services Web

Le développement du Web a favorisé le dialogue entre systèmes informatiques et le déploiement d'applications distribuées. Il a également révélé des manques évidents en matière d'intégration et d'interopérabilité entre services. Le modèle des Services Web [HUB03] a été mis en place afin de pallier ces manques. Les Services Web peuvent être définis:

Définition 1: la définition proposée par le World Wide Web Consortium (W3C) fait figure de référence [W3C04b]: « Un service web est un système logiciel destiné à supporter l'interaction ordinateur – ordinateur sur le réseau. Il a une interface décrite en un format traitable par l'ordinateur (*e.g. WSDL*). Autres systèmes réagissent réciproquement avec le service web d'une façon prescrite par sa description en utilisant des messages SOAP,

typiquement transmis avec le protocole HTTP et une sérialisation XML, en conjonction avec d'autres standards relatifs au web ».

Définition 2 : IBM donne dans un tutoriel la définition suivante des services web [PON04]: Les services Web sont la nouvelle vague des applications Web. Ce sont des applications modulaires, auto-contenues et auto-descriptives qui peuvent être publiées, localisées et invoquées depuis le Web. Les services web effectuent des actions allant de simples requêtes à des processus métiers complexes. Une fois qu'un service web est déployé, d'autres applications (y compris des services web) peuvent le découvrir et l'invoquer.

I. 3. 1. Pourquoi les services web ?

Pour l'entreprise, disposer des services web c'est avant tout ouvrir son système d'information à d'autres usages, d'autres besoins et d'autres clients extérieurs à l'entreprise. La mise en œuvre des services web dans l'entreprise accélère ce processus : [HUB03]

- Les services web permettent d'intégrer, gérer et automatiser rapidement les processus métiers intra et interentreprises en échangeant des informations au format XML.
- Les services web facilitent l'interopérabilité entre systèmes et plateformes hétérogène.
- Les services web facilitent l'intégration d'application et de services.

I. 3. 2. Architecture des services web:

L'architecture des services web repose sur un mécanisme de transport d'une demande de service entre un client et un serveur [GIR05]. Les trois acteurs essentiels participant à une communication service web sont :

1. Le fournisseur de service : C'est le propriétaire du service Web. il se charge de la description de son interface d'accueil, de sa publication dans l'annuaire afin qu'il puisse être trouvé par le client.
2. Le client : est une application qui cherche, localise et invoque le service.
3. L'annuaire des services : fournit l'emplacement où sont conservés les SW classés par leurs fournisseurs. Ces annuaires fournissent l'infrastructure de base pour la publication et la découverte de ces services.

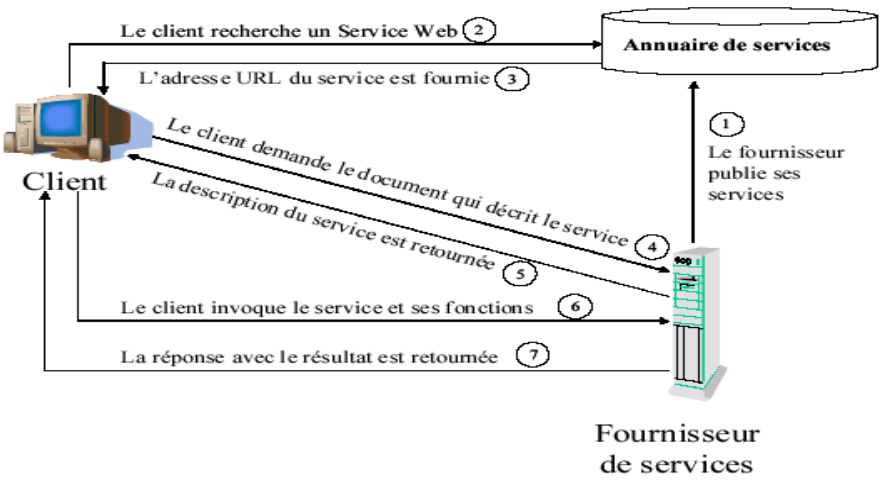

Fig. I. 2. Scénario d'utilisation des Services Web [GIR05].

La figure I.2 présente un scénario d'utilisation de ces différents composants : Le fournisseur d'un service définit sa description et la publie dans un annuaire (1). Le client accède à l'annuaire et utilise les outils de recherche et de découverte qu'il fournit pour localiser et sélectionner le service demandé (2,3). Ensuite, le client analyse la description du service qui contient les informations nécessaires à son invocation et le nombre et les types des paramètres utilisés (4,5).Finalement, le client peut invoquer le SW et ses fonctions (6). Ces dernières sont utilisées pour répondre aux requêtes envoyées par le client (7).

I. 3. 3. Les standards des services web

Dans ce qui suit nous détaillons les différents protocoles et standards utilisés dans les infrastructures des services web à savoir XML, SOAP, WSDL et UDDI.

I. 3. 3. 1. XML

XML (*Extensible Markup Language*) est une norme du W3C depuis février 1998 [MAE03]. Historiquement, il a profité du meilleur de SGML (*Standard Generelized Markup Language*) et de HTML [W3C08]. C'est un langage de description et d'échange de documents structurés ou semi structurés [GIR05] (Un document XML est structuré lorsque toute la structure est décrite en XML alors que dans un document semi-structuré, une partie de la structure échappe à XML). Il a la particularité de n'avoir aucune sémantique et peut donc s'appliquer à n'importe quelle application pour représenter l'organisation de tout type de données ou de document [HUB03].

Il est ouvert, extensible et peut être spécialisé pour décrire au mieux chaque type de données. Il est basé sur les balises, seulement en XML toutes les balises doivent être fermées. C'est un *métalangage* : il permet de décrire un vocabulaire et une grammaire associée selon un certain formalisme. L'intérêt premier de XML est de séparer le *fond* (le contenu) de la *forme* (le contenant ou la mise en page).

I. 3. 3. 2. SOAP (Simple Object Access Protocol)

SOAP Est un protocole léger basé sur XML permettant d'assurer l'échange d'informations dans un environnement distribué et décentralisé [HUB03].

Les messages SOAP sont transportés dans des requêtes HTTP traditionnelles. Une requête HTTP est un ensemble de lignes (texte) envoyé par le navigateur au serveur. Elle comporte un en-tête et un corps (optionnels). A l'intérieur de la partie corps de HTTP se trouve le message SOAP. Ce dernier se compose de trois éléments : une enveloppe, un en-tête (optionnel) et un corps [GIR05].

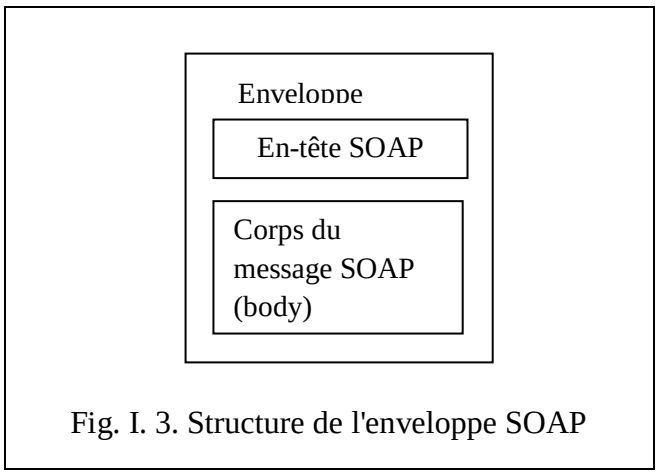

Fig. I. 3. Structure de l'enveloppe SOAP

L'enveloppe SOAP : L'élément enveloppe est obligatoire et sert de conteneur aux autres éléments du message SOAP [HUB03]. La spécification oblige que tous les éléments de l'enveloppe soient associés à un *namespace* pour éviter toute confusion et ambiguïté. L'enveloppe contient l'en-tête et le corps du message SOAP.

L'entête SOAP : L'en-tête est un élément optionnel. S'il est présent dans un message SOAP, il doit être un descendant direct de l'élément enveloppe et placé comme le premier de la séquence des descendants directs. L'en-tête fournit le mécanisme général et flexible

qui permet d'ajouter des traits nouveaux et spécialisés (des informations complémentaires) à un message SOAP [MAE03]. .

Le corps SOAP : l'élément corps (body) contient les données spécifiques à l'application. Il fournit un mécanisme simple pour échanger l'information avec le destinataire final du message [HUB03]. Il est utilisé pour effectuer des appels de procédure à distance (RPC) ou de transporter le report des erreurs.

I. 3. 3. 3. WSDL

Le protocole WSDL (Web Service Description Language) [CHR01] est un langage basé sur XML qui permet de faire une description précise de l'interface publique d'un service web [GIR05]. De façon plus précise, il permet de définir l'adresse du service, son identité, les méthodes qui peuvent être invoquées par les clients ainsi que leurs paramètres. Ces derniers seront décrits de façon détaillée par leurs types, valeurs de retour, etc.

Les opérations et les messages sont décrits d'une manière abstraite et lié à un protocole réseau concret et un format de message pour définir une destination. Les opérations dans le fichier WSDL peuvent être orientées *document* ou orientées *Remote Procedure Call (*RPC) comme défini par l'attribut style de l'élément <*soap:binding*> dans le fichier. Un ensemble d'opérations sont liées à un port et possèdent un type. Un port est un point terminal identifié de façon unique par la combinaison d'une adresse IP et d'un protocole d'accès. Finalement, un SW représente un ensemble de ports implémentant un ensemble d'opérations. Ces ports sont reliés par des protocoles de transport tels que (SOAP, HTTP, MIME) qui véhiculent des messages contenant les données à échanger entre les ports.

Les éléments définis à l'intérieur de WSDL sont représentés comme suit :

1. Type : WSDL utilise XML schéma pour définir les types des données utilisés dans les messages échangés.
2. Message : WSDL définit indépendamment de la couche transport des messages avec leurs paramètres.
3. PortType : collection d'opérations et de messages liés à un type de port.
4. Binding : liaison décrite par les messages concrets (réels) pour un port donné.
5. Port : adresse (assure l'unicité du rattachement).
6. Service : définit les informations d'accès et l'URL du service.

I. 3. 3. 4. UDDI

UDDI(Universal Description, Discovery and Integration) [HUB03] est un standard né de l'initiative d'un certain nombre d'entreprises, dont notamment Microsoft, IBM, SUN, Oracle, Compaq, Hewlett Packard, Intel, SAP et bien d'autres qui sont réunies pour développer une spécification basée sur des technologies standards afin de faciliter la collaboration entre partenaires dans le cadre des échanges commerciaux.

L'annuaire UDDI permet de publier et de découvrir des informations sur une entreprise et ses services web. Il contient des informations sur les entreprises et les services qu'ils publient. L'inscription d'une société à l'annuaire UDDI lui permet de se présenter et présenter ses services. L'adoption de cet annuaire par l'entreprise permettra d'accélérer les échanges commerciaux de types B2B. Il permet de rendre public et visible un service développé et fournit par une organisation en décrivant comment l'invoquer, l'intégrer et éventuellement comment l'utiliser.

L'annuaire UDDI facilite la localisation d'un service web et l'agrégation des services web émanant d'entreprises différentes. Par contre, ile ne permet pas de localiser des partenaires (seulement des services) et n'exonère pas les entreprises d'instaurer une relation de confiance préalable à la consommation réciproque de leurs services web

L'UDDI est subdivisé en deux parties principales : partie publication ou inscription, et partie découverte. La partie publication regroupe l'ensemble des informations relatives aux entreprises et à leurs services. Ces informations sont introduites via une API d'enregistrement. La partie découverte facilite la recherche d'information contenue dans UDDI grâce à l'API SOAP.

L'UDDI peut être vu comme un annuaire contenant :

- **Les pages blanches :** elles recensent les entreprises et contiennent des informations telles que : le nom de l'entreprise, ses coordonnées, des descriptions accessibles aux clients et des identifiants permettant de la retrouver par recherche.
- **Les pages jaunes :** contiennent au format WSDL la description des services Web déployés par les fournisseurs de services. Les services sont répertoriés par catégorie.

- **Les pages vertes :** contiennent des informations techniques du service offert, la manière d'interagir avec le service, une définition du « BusinessProcess » et aussi un pointeur vers le fichier WSDL et une clé unique identifiant le service.

La structure de données du registre UDDI contient cinq types de données :

- BusinessEntity: contient une description de l'entreprise offrant le service dans l'annuaire UDDI. Les quatre autres types de données sont référenciés via cette partie.
- BusinesService: contient le nom et une description du service Web proposé.
- BindingTemplete: information concernant le service web incluant l'adresse physique du fournisseur. Il est possible d'enregistrer de multiple « BindingTemplete » pour le même service, afin de définir différents points d'accès. Les différents points d'accès de «BindingTemplete » incluent les types d'URL suivant : HTTP, FTP,...etc.
- TModel : identifie d'une façon unique les spécifications relatives à l'interface d'un service Web. «tModel» est un mécanisme d'échange de méta données relatives au service, tel qu'une description du service Web ou un pointeur vers le fichier WSDL décrivant le service. «tModel» peut être considéré comme étant un point d'accès alternatif aux types de données contenues dans UDDI.
- PublishAssertion : met en correspondance deux ou plusieurs structures «BusinessEntity» selon le type de relation (filiale de, département de).

I. 4. Services Web Sémantiques

La section suivante est consacrée au service web sémantique, nous allons détailler quelques outils développés dans le cadre de la technologie des services web sémantiques.

L'idée consiste à faire converger le Web sémantique et les services Web afin de proposer des services et des procédés qui prennent en compte la connaissance que l'on peut avoir d'un service ou d'un procédé afin de pouvoir profiter des travaux sur le raisonnement à partir de connaissances qui ont été mis en avant par le Web sémantique.

Tim Berners-lee a proposé la définition suivante [CHA07]: «Les services web sémantiques sont des services web munis d'une sémantique ayant pour but de les rendre interopérables et compréhensibles par des moteurs ou des agents logiciels capables de raisonner ».

I. 4. 1. Ontologie

La notion d'ontologie est apparue en informatique dans les années 90. Une ontologie vise à décrire de façon consensuelle et partagée par les experts d'un domaine d'applications assez large, l'ensemble des informations permettant de conceptualiser les connaissances de ce domaine. Plusieurs définitions ont été proposées pour cette notion; la plus citée [AKL07]. Est celle de Gruber qui définit une ontologie comme « an explicit spécification of a conceptualization ». Une ontologie contient:

- un ensemble de concepts (entités, attributs, ...) représentatif du domaine couvert par l'ontologie ; cet ensemble est généralement représenté par une taxonomie ;
- les définitions des concepts et les relations entre les concepts (conceptualisation) ;
- des assertions ou contraintes sur les concepts ;
- des instances qui sont des représentations des individus des concepts de l'ontologie.

I. 4. 2. Approches proposées pour la réalisation des services web sémantiques

L'approche Service Web Sémantique rejoint les préoccupations à l'origine du web sémantique, à savoir comment décrire formellement les connaissances de manière à les rendre exploitables par des machines. En conséquence, les technologies et les outils développés dans le contexte du web sémantique peuvent compléter la technologie des services Web en vue d'apporter des réponses crédibles au problème de l'automatisation.
Les initiatives les plus pertinentes [IZZ06] autour du concept de Service Web sémantique sont principalement : OWL-S [MAR08], WSMF [FEN02][W3C05], WSMO [BRU05], METEOR–S [MET05][CAR04] et IRS-II [MOR04].

I. 4. 2. 1. OWL-S

OWL-S (*Web Ontology Language for Services*), le successeur de DAML-S, est un langage qui définit une ontologie de services web, basé sur OWL [MAR08] : Les intérêts liés à l'utilisation d'OWL-S sont que ce langage inclut la sémantique et contient des fonctions indispensables à la mise en œuvre de services web: la description, la recherche et l'invocation de services. Il permet de réaliser les deux tâches suivantes:

1. Découverte automatique de services web: Cette tâche est possible parce qu'OWL-S permet d'exprimer et de résoudre des requêtes avec contenu sémantique. Par exemple une requête comme «Trouver des services web qui vendent des tickets d'avion entre deux villes spécifiques et qui permettent de payer avec une carte de crédit particulière».

2. Invocation automatique de services web : OWL-S fourni un ensemble d'APIs pour que l'invocation à un service web soit automatique.

OWL-S est un ensemble de trois ontologies en OWL [GAN08] pour décrire les services web:

1. *Service Profile* est utilisé pour décrire essentiellement ce que fait le service (nom, catégorie, qualité, etc.) ;
2. *Service Process* indique comment fonctionne le service (opérations, entrées et sorties) ;
3. *Service Grounding* précise comment accéder au service (interface et messages d'interaction)

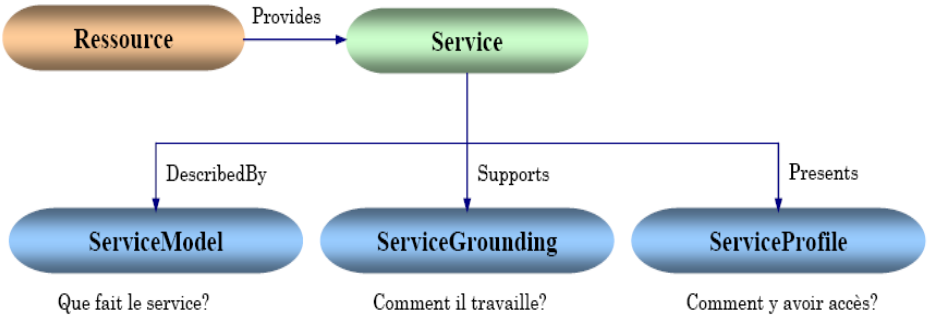

Fig. I. 4. Niveau supérieur de l'ontologie OWL-S [MAR08].

Dans l'approche OWL-S, la partie profil contient l'information la plus utile pour la découverte et la composition de services du fait qu'elle est utilisée à la fois par les fournisseurs de services pour la publication et par les clients pour l'interrogation [IZZ06].
Son défaut majeur est l'absence d'environnement d'exécution ce qui fait que la médiation des services reste encore à implémenter.

I. 4. 2. 2. WSMF

WSMF (*Web Services Management Framework*) [FEN02] est un cadre générique pour la réalisation des services sémantiques. Il constitue donc un Framework qui permet d'offrir beaucoup plus un cadre méthodologique qu'une infrastructure opérationnelle. Il est centré autour de deux principes complémentaires qui sont :

- Un découplage fort des différents composants servant à réaliser une application d'e–commerce

- Un service de médiation fort qui autorise quiconque à communiquer avec tout le monde, de manière évolutive.

L'architecture de WSMF est constituée de quatre éléments principaux:

1. *les objectifs* : permettent de définir les problèmes qui doivent être résolus par les services web. Ils incluent des pré-conditions et des post–conditions.
2. *les ontologies* : fournissent la terminologie et la sémantique utilisée par les autres éléments.
3. *les descriptions des services web* : définissent les différents aspects liés aux services web (input, output, opérations, etc.).
4. *les médiateurs* : Les médiateurs permettent de résoudre les problèmes liés à l'intégration entre les services web.

Le WSMF est un modèle conceptuel et ne définit pas de sémantique concrète particulière pour la représentation des services. Selon la spécification publique de WSMF, ceci peut être réalisé à travers l'utilisation de WSFL ou d'OWL-S, et de PSL [IZZ06].

I. 4. 2. 3. WSMO

WSMO (*Web Service Modeling Ontology*) [BRU05], est une ontologie de services basée sur WSMF. C'est un langage formel et une ontologie qui permet de décrire des aspects variés des services web sémantiques. La spécification de WSMO repose sur un modèle stratifié qui inclut la définition de trois différentes ontologies: WSMO Lite (ontologie de base), WSMO Standard (ontologie intermédiaire) et WSMO Full (qui contient tous les concepts définis dans WSMO). L'architecture de WSMO repose sur l'utilisation de médiateurs. On distingue deux classes de médiateurs:

1. des raffineurs (refiners) : sont utilisés pour générer de nouveaux composants comme des raffinements de ceux qui existent. On distingue
    - des raffineurs d'objectifs (ggMediator) (*goal refiners*) : qui sont utilisés pour raffiner un but source dans un but cible
    - des raffineurs d'ontologies (ooMediator). (*ontology refiners*) qui sont utilisés pour résoudre les éventuels problèmes lors de l'importation d'ontologies.
2. des ponts (bridges) : permettent de faire interopérer deux composants donnés. On distingue :
    - des ponts entre services et objectifs (wgMediator) (*service-to-goal bridges*) : résolvent les problèmes entre services et objectifs

- des ponts entre services (wwMediator) : (*service-to-service bridges*) effectuent le lien entre deux services Web.

Dans l'approche WSMO, on distingue deux composants principaux :

1. WSML (*Web service Modeling Language*) [BRU05] : fournit un langage formel pour la description des éléments définis dans l'architecture WSMO. WSML est basé sur différents langages logiques qui sont la logique de description, la logique de premier ordre et la logique de programmation. Ce langage est structuré en cinq sous-ensembles: WSML-Core, WSML-DL, WSML-Light, WSML-Rule et WSML-Full. Chacun de ces sous-ensembles fournit un niveau d'expressivité croissante.
2. WSMX (Web Service Execution Environment) [BUS05]. WSMX est un environnement d'exécution qui permet d'offrir un certain nombre de modules utilisables en run-time permettant de prendre en charge la découverte, la sélection, la médiation et l'invocation des services sémantiques.

I. 4. 2. 4. METEOR-S

METEOR-S (*METEOR-S (Managing End-To-End OpeRations for Semantic Web Services*) [ABH04] propose un système basé sur l'annotation sémantique de descriptions WSDL, de manière à enrichir des services existants avec des informations sémantiques [CARD04]. Le projet METEOR-S s'est développé selon trois phases principales qui ont permis d'introduire les trois concepts importants de cette initiative :

- mise en place de l'infrastructure : il consiste à installer une infrastructure de découverte sémantique définie au-dessus d'un registre UDDI et qui est appelée MWSDI (*METEOR-S Web Service Discovery Infrastructure*) [VER04];
- annotation sémantique : il définit l'outil MWSAF (*METEOR-S Web Service Annotation Framework*) [SIV04] et qui permet d'enrichir sémantiquement les services web en utilisant une extension enrichie de WSDL appelée WSDL-S (*WSDL Semantics*) [AKK05]. WSDL-S étend l'expressivité de WSDL avec des annotations, basées sur des ontologies externes, pour décrire les compétences et les besoins d'un service (entrées/sorties, pré-conditions, effets, opérations) [GAN08]. La composition et l'exécution de services : il a pour rôle d'assembler des services pour composer des services complexes en se basant sur BPEL4WS. L'outil se chargeant de cette tâche s'appelle MWSCF (*METEOR-S Web Service Composition Framework*) [AGG04].

METEOR-S comporte deux modules [IZZ06]:

- le module front-end : est utilisé pour créer des services web sémantiques en procédant à l'annotation du code source avec des ontologies. La source ainsi enrichie est convertie en descriptions sémantiques en utilisant soit des fichiers WSDL annotés soit des fichiers WSDL-S. Ces derniers sont ensuite publiés dans un registre UDDI enrichi.
- le module back-end : offre principalement des fonctionnalités liées à la découverte de services, à la composition de services et à l'exécution et l'orchestration des services composés.

Bien que cette approche repose sur des standards, il n'en demeure pas moins qu'elle manque d'abstraction dans le sens où elle constitue une approche plus proche du niveau d'implémentation que du niveau conceptuel.

I. 4. 2. 5. IRS-II

IRS-II (*Internet Reasoning Service*) [MOT03], est une structure et une implémentation pour les services web sémantiques, supportée par le projet AKT (*Advanced Knowledge Technologies*) [MOR04] dans le cadre d'une collaboration interdisciplinaire de recherche regroupant plusieurs universités européennes. Le but principal d'IRS-II est de supporter la découverte et la récupération de services de librairies distribuées sur Internet et de leur configuration semi-automatique, dans le but de réaliser des tâches spécifiques en fonction des exigences des utilisateurs [IZZ06].

IRS-II possède deux caractéristiques majeures, Premièrement, IRS-II est construit sur un modèle de connaissance, et deuxièmement IRS-II est basé sur le langage UPML (*Unified Problem-Solving Method-Development Language*) [CRU03] pour l'annotation des services.
Les principaux composants de l'architecture d'IRS-II et qui communiquent à travers le protocole SOAP sont :

- Le serveur IRS (IRS Server) : Le serveur ISR comporte les descriptions des services sémantiques à deux niveaux différents. Un niveau de description des connaissances permettant la description des tâches (Task models), du domaine (Domain models) et des méthodes de raisonnement (PSMs - Problem Solving Methods) et ce en se basant sur le framework UPML.
- le fournisseur IRS (IRS Publisher) : joue deux rôles principaux dans le framework IRS-II. Premièrement, il permet de lier les descriptions sémantiques aux services

web. Deuxièmement, il permet de générer des adaptateurs standards (Java Web Service) qui permettent ainsi d'encapsuler l'hétérogénéité due aux langages utilisés (Lisp, Java, etc.).

- le client IRS (IRS Cleint) : est toute application permettant d'interroger le Framework pour découvrir des tâches qui correspondent à un problème particulier et ensuite d'exécuter les services web associés à la résolution de ce problème.

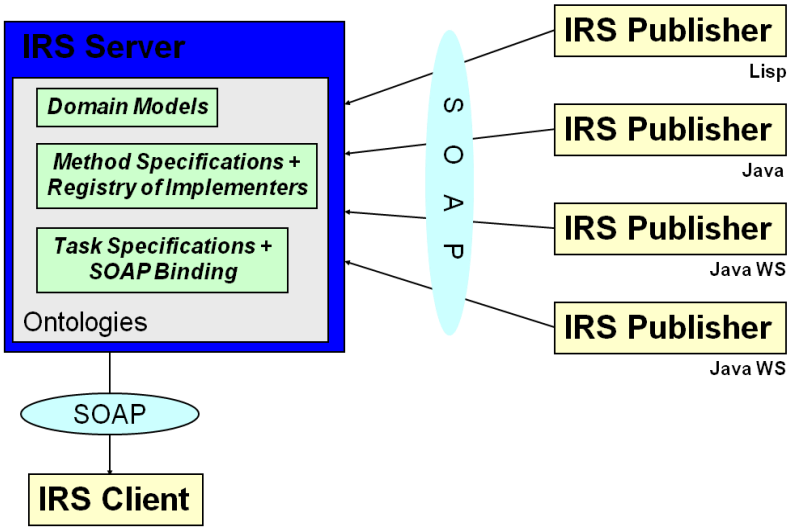

Fig. I. 5. Architecture d'IRS-II [MOT03].

I. 4. 2. 6. Étude comparative

Nous allons dans cette section, présenter une étude comparative entre les approches des services sémantiques précédemment décrits et qui sont: OWL-S, WSMO, METEOR-S et IRS-II**.**

Cette étude est basée plus particulièrement sur les études comparatives présentées dans [SHA07], [ALT 07] et [IZZ06].

I. 4. 2. 6. 1. Critères de Comparaison

Les critères de comparaison sont décrits comme suit :

- Le niveau d'abstraction (conceptuel, technique) qui est associé à l'approche considérée;
- La standardisation qui représente le degré de standardisation ;

- La description sémantique qui permet de préciser la manière avec laquelle les services sont décrits sémantiquement par l'approche étudiée;
- La médiation de services qui permet de préciser la prise en compte des mécanismes de médiation par l'approche étudiée;
- La découverte qui permet de connaître la manière de localiser les services;
- La composition et Invocation qui permettent de connaître la manière de composer et d'invoquer des services.

I. 4. 2. 6. 2. Tableau comparatif entre les approches

Approches / Critères	OWL-S	WSMO	METEOR-S	IRS-II
Niveau d'abstraction	Conceptuel (basée sur une Ontologie générique de services)	Conceptuel (basée sur le cadre WSMF)	Technique (basée sur l'enrichissement sémantique de WSDL)	conceptuel (basée sur des ontologies tâches et PSM[1])
Standardisation	Très forte (basée sur WSDL, OWL)	Faible (basée sur WSML)	Forte (basée sur WSDL et WSDL-S)	Faible (basée sur UPML et OCML[2])
Description Sémantique	Ontologie générique OWL-S	Ontologie WSMO	Annotation des fichiers WSDL	Ontologie PSM et ontologie de Tâche
Médiation de service	Non définie	définie mais en cours de développement	Non définie	Non définie
Découverte	Profile	Objectif + fonctionnalités	WSDL-S	Ontologie de domaine + PSM
Composition	Composite Process Control Construct	Interface Chorégraphie Orchestration	BPEL4WS [3]	PSM
Invocation	OWL-S / WSDL Grounding	WSMO /WSDL Grounding	WSDL	IRS-II Publisher

Tableau I. 1. Comparaison entre les approches SWS.

[1] Problem Solving Methods: décrit un modèle des processus pouvant être utilisés pour composer des services Web.

[2] OCML (Operational Conceptual Modelling Language) est développé en 1993 par KMI de Open University. Il a été conçu comme une extension du langage Ontolingua, dans la mesure où il comble les lacunes de ce dernier en prenant en charge les règles de production, ce qui permet d'améliorer les mécanismes de raisonnement d'Ontolingua. OCML est un langage adapté aux méthodes de résolution de problèmes.

[3] BPEL4WS est basé sur XML et propose un langage pour définir formellement les processus ainsi que les protocoles d'interactions métiers.

I. 4. 2. 6. 6. 3. Discussion

L'analyse du tableau permet de retenir un certain nombre de points importants qui sont:

- OWL-*S* constitue une ontologie générique de services permettant essentiellement de décrire les services Web dans un cadre général. Une des caractéristiques importante de OWL-S est sa compatibilité avec les standards industriels notamment WSDL, et ce grâce au ServiceGrounding permettant la définition des mappings entre OWL-S et WSDL. Beaucoup de travaux existent sur la découverte et la composition de services en utilisant OWL-S.

- WSMF est un cadre générique pour la réalisation des services sémantiques. Il constitue donc un Framework qui permet d'offrir beaucoup plus un cadre méthodologique qu'une infrastructure opérationnelle.

- WSMO constitue une approche plus orienté utilisateur puisqu'elle permet d'offrir un langage très proche des utilisateurs. Toutefois elle demeure immature du fait qu'il s'agit d'une initiative en cours de développement. De plus, cette approche ignore les standards industriels existants.

- METEOR-S constitue une approche basée sur l'extension de standards industriels. Le principe d'enrichissement sémantique est basé principalement sur l'utilisation des fichiers WSDL-S qui sont le résultat de l'annotation sémantique de fichiers WSDL. Bien que cette approche repose sur des standards, elle reste une approche plus proche du niveau d'implémentation que du niveau conceptuel.

- IRS-II est une approche qui permet de réaliser les services sémantiques à travers le Framework UPML. Elle négligeant des préoccupations importantes liées aux services sémantiques.

En résumé, nous pouvons remarquer la compatibilité entre OWL-S et METEOR-S qui peuvent ainsi être considérées comme des approches susceptibles d'être combinées, alors que ces deux approches sont plutôt contradictoires avec WSMO et avec IRS-II dans la mesure où elles utilisent des langages et des outils incompatibles avec les standards industriels.

I. 5. Conclusion

La technologie des services web est devenue une technologie de référence pour le développement des applications distribuées sur le web, l'utilisation des protocoles basés sur XML et standardisés par le W3C pour la description des services (WSDL) et l'échange des données et l'invocation des objet à distance (SOAP) formulé avec des messages XML et véhiculées au-dessus des protocole de web (http, SMTP,..) permet de surmonter les problème d'hétérogénéité et d'interopérabilité.

Les Services Web apportent des avantages indéniables dans les technologies d'intégration et en termes d'ouverture des entreprises vers ses partenaires en offrant des services directement issus de leur système d'information.

Les services Web sémantiques constituent une voie prometteuse permettant de mieux exploiter les services web en automatisant, autant que possible, les différentes tâches liées au cycle de vie d'un service. A l'état actuel, il existe des approches pour la description sémantique à savoir : OWL-S, WSMF, WSMO, METEOR-E et IRS-II.

Dans le chapitre suivant, nous allons définir le cadre dans lequel notre travail va se placer, à savoir la notion de composition de services web, l'intégration des données et les services web DaaS qui permettent l'accès aux données des organismes par l'intermédiaire des Services web.

Chapitre II

État de l'art : Composition de Services Web DaaS & Intégration de données.

II. 1. Introduction

La diversité des sources d'information distribuées et leur hétérogénéité est l'une des principales caractéristiques du web aujourd'hui et l'accès à un nombre important de sources de données se fait par des services web qui permettent d'accéder et de rechercher des données élémentaires tenues par des sources hétérogènes autonomes indépendamment des systèmes et des plates-formes utilisées. Ce type services est appelé DaaS (***D**ata-**a**s-**a**-**S**ervice*) [BAR10].

Les Services Web DaaS sont maintenant utilisées dans de nombreux domaines d'application comme un moyen standard pour la publication et le partage des données, par exemple, la bio-informatique, traitement et partage de données géo-spatiales, le partage des données médicales eHealth et le partage des données entre les organismes gouvernementaux (eGovernment), etc.).

Nous proposons dans ce chapitre une présentation du contexte dans lequel s'inscrit notre travail, à savoir la notion de composition de services Web, l'intégration des données et les services web DaaS. Nous commençons tout d'abord par introduire la composition des Services Web; ensuite nous présenterons l'intégration des données, puis nous aborderons quelques approches existantes de DaaS en discutant notre travail par rapport aux autres travaux.

II. 2. Composition des services

Les services Web, tels qu'ils sont présentés, sont conceptuellement limités à des fonctionnalités relativement simples qui sont modélisées par une collection d'opérations. Toutefois, pour certains types d'application, il est nécessaire de combiner un ensemble de services web basiques en services web plus complexes (services web agrégés ou composites) afin de répondre à des exigences plus complexes. La composition de services vise à faire inter-opérer, interagir et coordonner plusieurs services pour la réalisation d'un but. La composition des Services Web peut se faire de deux manières:

- ***Orchestration*** : décrit, du point de vue d'un service, les interactions de celui-ci ainsi que les étapes internes (ex. transformations de données, invocations à des modules internes) entre ses interactions [PEL04]. L'orchestration décrit l'interaction des services au niveau de messages, incluant la logique métier et l'ordre d'exécution des interactions. Les services web n'ont pas de connaissance (et n'ont pas besoin de l'avoir) d´être mêlées dans une composition et d'être partie d'un processus métier. Seulement le coordinateur de l'orchestration a besoin de cette connaissance.
- ***Chorégraphie*** : La chorégraphie décrit la collaboration entre une collection de services dont le but est d'atteindre un objectif donné. L'accomplissement de ce but commun se fait alors par des échanges ordonnés de messages [AUS04]. Contrairement à l'orchestration, la chorégraphie n'a pas un coordinateur central. Chaque service web mêlée dans la chorégraphie connaît exactement quand ses opérations doivent être exécutées et avec qui l'interaction doit avoir lieu. La chorégraphie est un effort de collaboration dans lequel chaque participant du processus décrit l'itération qui l'appartient. Elle trace la séquence des messages qui peut impliquer plusieurs services web

Selon les travaux effectués dans le champ des services web, on peut classifier la composition selon deux axes [BAR06] :

1. En fonction du degré de participation de l'utilisateur dans la définition du schéma de composition, ces propositions sont manuelles, semi-automatiques ou automatiques.

- Une composition est manuelle quand tous les services web qui font partie de la composition sont connus ainsi que ses ordres d'exécution.
- Les techniques de composition semi-automatiques sont un pas en avant en comparaison avec la composition manuelle, dans le sens qu'ils font des suggestions

sémantiques pour aider à la sélection des services web dans le processus de composition.

- Une composition est automatique quand les services web qui font partie de la composition sont connus progressivement, ainsi que l'ordre d'exécution nécessaire pour la composition.

2. Selon que la sélection des services et la gestion du flot soient faites ou non a priori, une approche sera dite statique ou dynamique. Dans le cas de la composition statique, le demandeur doit définir a priori le modèle abstrait de processus décrivant le plan de composition, par exemple, sous forme d'un graphe de tâches. Dans ce type d'approche, uniquement la sélection et la liaison aux services basiques se font de manière automatique, citons en l'occurrence le système eFlow [CAS00]. Par contre, la composition dynamique vise à créer le modèle de processus abstrait, sélectionner et lier les services atomiques aux tâches abstraites de manière automatique et à la demande**.**

Différentes approches ont été proposées pour composer automatiquement les services web. Selon [RAO04], la plupart des recherches dans ce domaine peuvent se regrouper selon deux axes : les workflows et la planification. Et il y a également des travaux relatifs au Calcul de Situation (Situation Calculus), et plus précisément aux réseaux de Pétri et au langage Golog, car de nombreuses contributions ont été effectuées dans ces domaines.

II. 2. 1. Workflow :

La définition de la composition de services correspond alors à un ensemble de services atomiques (ou composés) sur lesquels on effectue un contrôle de flux. D'autre part, un workflow a également besoin de définir un flux d'activités.

Un workflow est une abstraction d'un processus de type business [BAR06]. Il est composé d'un nombre d'échelons logiques (aussi appelés tâches ou activités), de dépendances entre tâches, de règles et de participants. Dans un workflow, une tâche peut représenter une activité humaine ou un système (logiciel). Il est nécessaire d'associer une tâche à un service quand le workflow est appliqué aux services web. Une composition d'un workflow implique la sélection de tâches appropriées à des fonctionnalités désirées, devant prendre en compte les connections entre ces tâches (flux de contrôle et de données). Les workflows qui gèrent les services web (aussi appelé de e-Services) sont appelés des e-workflows [CAR02].

Les compositions de services qui utilisent cette approche sont normalement basées sur les workflows manuels, aussi appelé statiques. Dans le workflow manuel, l'utilisateur doit définir l'ensemble des tâches et des dépendances parmi les données. Chaque tâche contient des requêtes qui permettent de chercher des services concrets pour la satisfaire, puis l'exécuter. Dans ce cas, seules la sélection et la liaison du service sont faites automatiquement. Du côté des compositions dynamiques, le modèle du processus ainsi que la sélection du service sont faits automatiquement [RAO04]. À cet égard, une composition automatique demande des workflows capables de reconnaître les services web correspondants à chaque tâche, mais aussi de trouver d'autres services au cas où ceux-ci soient indisponibles ou dévient de leur exécution normale.

II. 2. 2. Le Calcul de Situation.

Le Calcul de Situation [REI01] est un langage basé sur la logique du premier ordre permettant de représenter et raisonner sur un domaine dynamique. Ce formalisme a été introduit par John McCarthy et Patrick J. Hayes en 1969[BAR06]. Les principaux éléments sont les actions, les fluents et les situations. Le monde est conçu comme un arbre de situations, débutant par la situation initiale s_0 et évoluant jusqu' à la nouvelle situation par l'application d'une ou plusieurs actions. Une situation s donnée correspond toujours à un historique de l'ensemble d'actions réalisées sur s_0 .

II. 2. 3. Les réseaux de Petri

Les réseaux de Petri sont des graphes bipartis qui contiennent des places (représentées par des cercles), des transitions (représentées par des rectangles) et des arcs (représentés par des flèches). Les places peuvent contenir un certain nombre de jetons, représentant généralement des ressources disponibles. La distribution des jetons dans les places est appelée marquage d'un réseau de Petri. Les transitions sont des composants actifs. Quand toutes les places positionnées dans une transition contiennent un nombre adéquat de jetons, la transition peut s'effectuer : on enlève alors les jetons des places d'entrée et on ajoute un nouvel ensemble de jetons dans les places de sortie [NAR02]. Une des plus importantes caractéristiques des réseaux de Petri est ses capacités à modéliser les états et les événements dans un système distribué, et sa capacité à contrôler les séquences, les concurrences et les contrôles basés sur les événements. Le travail proposé dans [NAR02] utilise les réseaux de Petri pour composer automatiquement les services web, ainsi que les simuler et les vérifier. Ils commencent par transformer un sous-ensemble de termes de l'ontologie DAML-S en un

langage de premier ordre. Pour cela les auteurs utilisent le Calcul de Situations. Après avoir obtenu les descriptions sémantiques, ils les transforment dans un formalisme de réseau de Petri. Les auteurs ont implémenté une application où les entrées sont des descriptions de services en DAML-S. Ils génèrent automatiquement un réseau de Petri et effectuent son analyse.

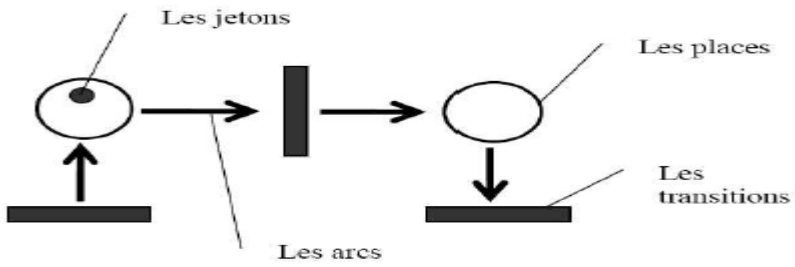

Fig II. 1. Les réseaux de Petri [BAR06]

II. 2. 4. Golog

Golog est un langage de haut niveau, conçu à l'Université de Toronto, pour la spécification et l'exécution d'actions complexes dans des domaines dynamiques [BAR06]. Il a été construit lorsque le Calcul de Situations était répandu dans le domaine. D'autres ajouts ont ensuite été effectués afin de reconnaître des actions complexes. La principale différence avec le Calcul de Situations proposé par McCarthy et Hayes et celui utilisé actuellement, est son interprétation en ce qui concerne les situations : elles correspondent actuellement à une séquence d'actions. De plus, pour la représentation de connaissances, le Calcul de Situations a été enrichi par un fluent *k(s', s)* qui permet de décrire des relations entre situations : un agent est dans une situation *s*, mais il croit qu'il peut être dans *s'*. Les travaux décrits dans [MCD02] proposent une adaptation et une extension du langage Golog, en ajoutant des procédures génériques qui sont valides pour une variété d'utilisateurs sous une variété de conditions. Pour exécuter leur programme, ils ont défini une approche hybride, où d'un côté ils obtiennent des informations nécessaires en ligne et d'un autre côté, récupèrent des simulations hors ligne pour les modifications du monde. Ces modifications ont été implémentées dans ConGolog, puis intégrées à son architecture pour la composition des services web. Le calcul de situation est considéré comme un des premiers travaux significatifs en planification.

II. 2. 5. La planification

La planification est un des domaines de l'Intelligence Artificielle (IA) qui permet de choisir et d'organiser des actions, en fonction d'un but donné. un grand nombre de travaux sont menés pour résoudre les problèmes de la composition automatique de services web. La plupart de ces travaux utilisent la planification. La planification et l'utilisation des ontologies, et plus précisément d'OWL-S, permettent d'accroître l'efficacité dans la composition de services.

Planification « classique » : De nombreux travaux ont été produits ces dernières années. [SHE03] proposent l'utilisation d'ontologies de domaines afin d'ajouter des contraintes pour optimiser la recherche dans un espace de plans. [MCD02] proposent d'utiliser la planification par régression estimée (*Estimated-Regression Planning*) qui permet d'utiliser des heuristiques pour guider la recherche dans un espace de situations. Le principe est le suivant : un agent qui désire interagir avec un service Web va construire un plan puis l'exécuter. Si l'exécution échoue, l'agent aura appris de nouvelles informations de cette interaction, qui seront utilisées dans le prochain cycle de planification et exécution. [SIR02] présentent une méthode semi-automatique pour la composition de services Web. À chaque fois qu'un utilisateur doit sélectionner un service Web, tous les services disponibles qui correspondent au service sélectionné, sont présentés à l'utilisateur qui fait alors le choix à la place du système. L'idée de la composition semi-automatique est assez intéressante car il est très difficile de capturer le comportement des services Web dans leurs moindres détails, puis de les composer automatiquement. Bien que cette méthode soit simple, elle permet de voir comment un planificateur automatique et un humain peuvent travailler ensemble pour répondre à la requête de l'utilisateur.

Planification basée sur les règles (*Rule-based planning*)**: [**MED03] présente une technique pour générer des services composites à partir de descriptions déclaratives de haut niveau. Cette méthode utilise des règles de composabilité pour déterminer dans quelle mesure deux services sont composables. L'approche proposée se déroule en quatre phases :

1. La phase de spécification. Elle offre une spécification de haut niveau de la composition désirée en utilisant le langage CSSL (*Composite Service Spécification Langage*).
2. La phase de correspondance. Elle utilise des règles de composabilité pour générer des plans conformes aux spécifications du service demandeur.

3. La phase de sélection. Si plus d'un plan est généré, la sélection est effectuée par rapport à des paramètres de qualité de la composition.
4. La phase de génération. Une description détaillée du service composite est automatiquement générée et présentée au demandeur.

Les règles de composabilité considèrent les propriétés syntaxiques et sémantiques des services Web.

Planification hiérarchique : [PAR03] préconisent l'utilisation du planificateur SHOP2 pour la composition automatique de services Web à partir de leur description sémantique. SHOP2 est un planificateur HTN (*Hierachical Task Network*). D'après les auteurs, le principe de décomposition d'une tâche en sous-tâches dans la planification hiérarchique est très similaire au concept de décomposition de processus composites dans OWL-S. Afin de rendre leur proposition réalisable, les auteurs font deux hypothèses concernant le modèle de processus :

- Les processus atomiques ont, soit des sorties, soit des effets, mais pas les deux en même temps.
- Les structures de contrôle Split et Split+Join ne sont pas utilisées dans les processus composites.

Discussion

Les approches actuelles ne tiennent pas en compte le rapport sémantique entre l'entrée et la sortie dans la composition des services web DaaS. Telle sémantique n'est pas facilement saisie avec des termes en ontologies prédéfinies et standards, il est nécessaire de définir une vue déclarative (en respectant une ontologie partagée dans le domaine d'application) pour saisir cette sémantique. [LIN04] [OUK08].

Le rapport sémantique entre l'entrée et la sortie doit être saisi dans le DaaS.

II. 3. L'intégration de sources d'information et les systèmes médiateurs

L'intégration de sources d'information est devenue un domaine de recherche très important à cause de l'explosion du nombre de sources disponibles via le Web. Il s'agit d'un processus par lequel plusieurs sources de données autonomes, réparties et sous forme hétérogène sont intégrées sous forme de source unique représentée par un schéma global [BEL04].. Les données intégrées peuvent être matérialisées (elles sont dupliquées dans un entrepôt de données), ou virtuelles (elles restent dans les sources d'origines et sont accédées via un médiateur).

II. 3. 1. Architecture matérialisée

Cette approche consiste à voir l'intégration comme la construction de bases de données réelles, appelées « entrepôts ». Ces entrepôts stockent physiquement les données des sources réparties, c'est pour cette raison que cette approche est souvent appelée « matérialisée ». Le processus de construction d'un système d'intégration matérialisé se compose en quatre étapes principales [HAC05] : (1) l'extraction des données des sources de données, (2) la transformation des données au niveau structurel, (3) l'intégration sémantique des données, et (4) le stockage des données intégrées dans le système cible. Ces étapes correspondent aux outils d'ETL (*Extract, Transform and Load*) [GIR05]. Les deux premières étapes sont habituellement prises en charge par le même composant logiciel, appelé adaptateur. L'objectif d'un adaptateur est d'aboutir à des données représentées dans un même format. Chaque adaptateur fournit ainsi une interface d'accès et de requêtes sur la source. Les données extraites sont ensuite intégrées en éliminant les conflits, puis stockées dans l'entrepôt.

Discussion

L'avantage principal d'une telle approche est que l'interrogation d'un entrepôt de données se fait directement sur les données de l'entrepôt et non sur les sources originales. Par contre cette approche exige un coût de stockage supplémentaire et surtout un coût de maintenance causé par les opérations de mises à jour au niveau de sources de données (toute modification dans les sources locales doit être répercutée sur l'entrepôt de données).

Les travaux récents sur l'intégration de données semi-structurées [MAH08], [CHR00], [FER00] , Xyleme[4], relèvent de l'approche entrepôt de données. Quelques projets spécifiques d'entrepôts de données servent actuellement de références, en particulier, le projet européen DWQ (*DataWarehouse Quality*) [JAR97] et le projet WHIPS à l'université de Stanford - USA [LAB97].

II. 3. 2. Architecture virtuelle

Cette architecture consiste à développer une application chargée de jouer le rôle d'interface entre les sources de données locales et les applications des utilisateurs. Ce type d'approche a été utilisé par un nombre important de projets [CHA97], [GEN97], [LEV95], [MEN96], [REY03], [LUM06]. Il repose sur deux composants essentiels : le médiateur et l'adaptateur.

[4] http://www.xyleme.com

- Le médiateur: chargé de la localisation des sources de données et des données pertinentes par rapport à une requête, il résout de manière transparente les conflits de données;
- L'adaptateur: comme dans l'approche entrepôt, c'est un outil permettant à un (ou plusieurs) médiateur(s) d'accéder au contenu des sources d'informations dans un langage uniforme. Il fait le lien entre la représentation locale des informations et leur représentation dans le modèle de médiation.

Dans cette approche, les données ne sont pas stockées au niveau du médiateur. Elles restent dans les sources de données et ne sont accessibles qu'à ce niveau. C'est pour cette raison que cette approche est appelée approche « virtuelle ». L'intégration d'information est fondée sur l'exploitation de vues abstraites décrivant de façon homogène et uniforme le contenu des sources d'information dans les termes du médiateur. Les sources d'information pertinentes, pour répondre à une requête, sont calculées en fonction de la définition de ces vues. Le problème consiste à trouver une requête qui est équivalente à la requête de l'utilisateur. Les réponses à la requête posée sont ensuite obtenues en évaluant cette requête sur les extensions des vues. Le problème d'hétérogénéité sémantique entre les différentes sources d'information ne se pose pas pour l'utilisateur puisqu'il est résolu par le médiateur.

De nombreux travaux ont porté sur l'architecture virtuelle. [HAC05] a identifié trois grands problèmes ayant fait l'objet d'études :

- Des études ont porté sur les *langages* pour modéliser le schéma global, pour représenter les vues sur les sources à intégrer et pour exprimer les requêtes provenant des utilisateurs humains ou d'entités informatiques ;
- d'autres travaux ont porté sur la conception et la mise en œuvre d'algorithmes de réécriture de requêtes en termes de vues décrivant les sources de données pertinentes ;
- certains travaux portent sur la conception d'interfaces intelligentes assistant l'utilisateur dans la formulation de requêtes, l'aidant à affiner une requête en cas d'absence de réponses ou en cas de réponses beaucoup trop nombreuses.

On distingue deux architectures. Une architecture dite « centralisée » qui correspond à l'architecture des premiers systèmes médiateurs implémentés et une architecture dite «décentralisée » basée sur le paradigme « pair à pair » (P2P)[GIR05].

- ***Les systèmes de médiation centralisée* :** Le médiateur communique avec les seules sources qu'il intègre afin de répondre à la requête de l'utilisateur. Dans une telle architecture, le médiateur est construit au-dessus d'un ensemble de sources de données pré-identifiées. La réponse à une requête est construite uniquement à partir d'éléments de ces sources de données. on trouve des adaptateurs ou « wrappers » spécifiques à chaque type de source. Le rôle de ces wrappers est de traduire les réécritures produites par le moteur en un langage propre à chaque source.
- ***Les systèmes de médiation décentralisée :*** Dans un processus décentralisé, la médiation met en jeu plusieurs serveurs intégrant chacun un ensemble de sources. Le traitement d'une requête utilisateur est effectué de manière distribuée, c'est-à-dire qu'il est pris en charge par un serveur, jouant le rôle de médiateur, mais son traitement peut nécessiter l'interrogation d'autres systèmes, jouant le rôle de serveurs de données (éventuellement d'autres systèmes médiateurs), avec lesquels celui-ci est en relation.

Discussion

L'approche « médiateur » présente l'intérêt de pouvoir construire un système d'interrogation de sources de données sans toucher aux données qui restent stockées dans leur source d'origine. Par contre le médiateur ne peut pas évaluer directement les requêtes qui lui sont posées car il ne contient pas de données, ces dernières étant stockées de façon distribuée dans des sources indépendantes. Dans cette approche, deux problèmes apparaissent : comment construire le schéma intégré et comment effectuer la mise en correspondance entre les termes utilisés dans ce schéma et les structures des schémas des sources qui contiennent les données.

II. 4. Ontologies et Intégration d'informations

Les données relatives à un même sujet peuvent être représentées différemment dans ces différentes sources. C'est le problème de l'hétérogénéité des données. [GOH97] a identifié trois principales causes à l'hétérogénéité sémantique des données.

- Les conflits de nom ont lieu lorsque des noms différents sont utilisés pour décrire le même concept (synonyme) ou lorsque le même nom est utilisé pour des concepts différents (homonyme).
- Les conflits de mesure de valeur ont lieu lorsque différents systèmes de référence sont utilisés pour évaluer une valeur. C'est le cas, par exemple, lorsque différentes unités de mesure sont utilisées par les différentes sources de données.

- Les conflits de contexte ont lieu lorsque des concepts semblent avoir la même signification mais diffèrent en réalité dû à différents contextes de définition ou d'évaluation.

Parce qu'une ontologie peut servir de pivot pour définir la sémantique des données des différentes sources, leur utilisation est une solution pour résoudre les problèmes d'hétérogénéité des données. Plusieurs approches d'intégration à base ontologique ont été développées [WAC01]. Ces dernières peuvent être divisées en trois catégories : approches avec une seule ontologie, approches avec ontologies multiples et approches hybrides. Dans l'approche avec une seule ontologie, chaque source référence la même ontologie globale de domaine. Les systèmes d'intégration SIMS et COIN sont des exemples de cette approche [AKL07] [GOH99]. En conséquence, une nouvelle source ne peut ajouter aucun nouveau concept sans exiger le changement de l'ontologie globale. Dans l'approche à multiples ontologies (exemple du projet OBSERVER [MEN96], chaque source a sa propre ontologie développée indépendamment des autres sources. Dans ce cas, les correspondances inter-ontologies sont difficiles à mettre en œuvre. L'intégration des ontologies est donc faite d'une façon manuelle ou semi-automatique [MEN96]. Pour surmonter l'inconvénient des approches simples ou multiples d'ontologies, l'approche hybride a été proposée. Dans cette dernière, chaque source a sa propre ontologie, mais toutes les ontologies utilisent un vocabulaire partagé commun (exemple du projet KRAFT [VIS99]).

II. 4. 1. Sens de mise en correspondance entre schéma global et schéma local

On distingue deux approches pour établir la correspondance entre les termes du schéma global ou « ontologie du domaine » et ceux des sources de données.

II. 4 .1. 1. Global As View (GaV)

L'approche GaV a été la première à être proposée pour intégrer des informations. Elle consiste à définir à la main (ou de façon semi-automatique) le schéma global en fonction des schémas des sources de données à intégrer puis à le connecter aux différentes sources. Pour cela, les prédicats du schéma global, aussi appelés relations globales, sont définis comme des vues sur les prédicats des schémas des sources à intégrer. Comme les requêtes d'un utilisateur s'expriment en termes des prédicats du schéma global, on obtient facilement une requête en termes de schéma des sources de données intégrées, en remplaçant les prédicats du schéma global par leurs définitions. Parmi les systèmes utilisant GaV, on peut citer TSIMMIS [CHA04] et MOMIS [BEN00]. En effet, dans TSIMMIS, les médiateurs sont

conceptuellement représentés par des vues définies sur une ou plusieurs sources. Dans le cas d'ajout d'une nouvelle source, TSIMMIS reconstruit les médiateurs.

II. 4 .1. 2. Local As View (LaV)

L'approche LaV est l'approche duale, elle suppose l'existence d'un schéma global et elle consiste à définir les schémas des sources de données à intégrer comme des vues du schéma global. Les principaux systèmes développés autour de cette approche sont : PICSEL [GIR05], Information Manifold [LEV95]. InfoMaster [GEN97], par exemple, est un entrepôt virtuel de catalogues développé par l'Université de Stanford . Il présente à l'utilisateur un catalogue virtuel, appelé *schéma référencé*, sur lequel il formule ses requêtes. Ce système utilise le langage KIF (*Knowledge Interchange Format* [5]) pour définir une liste de règles qui associe les schémas des sources intégrées aux vues du *schéma référencé*. Ces règles permettent aussi de convertir l'information afin de la mettre sous format adéquat et de l'adapter aux sources. Lors de l'ajout d'une nouvelle source, le système d'intégration ajoute des nouvelles règles représentant la vue de la nouvelle source.

II. 4 .1. 3. Bilan sur les approches GaV et LaV

On considère souvent que l'adjonction d'une nouvelle source est plus facile dans l'approche LaV que dans l'approche GaV [HAC05]. En fait, cela dépend de l'hypothèse faite concernant (1) les concepts contenus dans le schéma global, et (2) les concepts existants (auxquels le système d'intégration doit donner accès) dans la nouvelle source. Si l'on fait l'hypothèse qu'une nouvelle source ne doit pas modifier le schéma global (hypothèse faite dans le système LaV), soit qu'elle ne contienne aucun nouveau concept, que le schéma global ne représente que partiellement les différentes sources, l'effet d'adjonction est tout à fait similaire. Le coût de modification de l'implémentation de la vue globale, dans l'approche GaV, est contre balancée, dans l'approche LaV, par le coût de définition du schéma local comme vue du schéma global, puis par le coût de réécritures de requêtes en fonction des vues.

II. 4 . 2. Médiateurs, Systèmes d'intégration de données et services Web DaaS.

Les systèmes d'intégration de données, tels qu'InfoMaster [GEN97], Infosleuth [ROB97], Ariadne [CRA01] et Information Manifold [LEV95] fournissent une interface de requête uniforme à une multitude de sources de données autonomes. Les utilisateurs de systèmes d'intégration de données ne formulent pas leurs requêtes en termes de schémas dans lesquelles les données sont stockées, mais en termes de schéma de médiation.

[5] http://www-ksl.stanford.edu/knowledge-sharing/kif/

Notre travail présente de nombreuses similitudes et de différences avec les travaux dans ce domaine de recherche.

Pareil à ces travaux de recherche, notre système soulage les utilisateurs finaux autant que possible d'être impliqué dans le processus de résolution de leurs requêtes. Les DaaS qui sont pertinents à une requête donnée sont localisés, sélectionné et composés de façon totalement transparente. Nous partageons le même schéma pour atteindre cet objectif.

Aucun des précédents systèmes d'intégration de données n'a été conçu pour fonctionner avec des requêtes paramétrées et ils n'ont pas de techniques d'optimisation pour optimiser les plans de composition d'une requête paramétrée. En plus notre objectif est différent de celui des systèmes d'intégration de données, car nous ne sont pas intéressés par l'intégration de données en soi, plutôt nous nous intéressons à fournir une solution intégrée pour composer automatiquement des services DaaS.

II. 5. Services web d'accès aux données :

Avec la démocratisation toujours croissante du Web, des sources de données de plus en plus nombreuses sont à la disposition des utilisateurs. Cependant, ces sources de données sont hétérogènes; on distingue généralement trois niveaux d'hétérogénéité :

- **l'hétérogénéité technique** concerne les modalités d'accès aux données : protocoles propriétaires nécessitant une application spécifique, protocoles ouverts (RPC, REST...), formulaires et tables HTML dans un navigateur Web.
- **l'hétérogénéité syntaxique** concerne la forme des données : formats propriétaires, données relationnelles, données semi-structurées (XML).
- **l'hétérogénéité sémantique** concerne la signification des schémas et vocabulaires utilisés pour décrire ces données : schémas relationnels, DTDs et schémas XML.

Une solution au problème posé par l'hétérogénéité technique réside dans l'utilisation de la technologie des services web.

II. 5. 1. Services web et intégration de données

Dans ce qui suit, nous allons présenter deux projets qui utilisent les services web dans la médiation des données : Active XML et le projet Picsel, puis nous allons présenter quelques travaux relatifs aux services web d'accès de données.

II. 5. 1. 1. Active XML

Active XML (AXML)[6] est un modèle déclaratif pour la gestion de données distribuées sur le Web basé sur XML et les services Web (SOAP, WSDL). Un document AXML est un document XML pouvant contenir des appels à des services Web [ABI08].

L'approche Active XML propose une architecture *pair à pair* (P2P) [BEN04], fondée sur l'échange de documents intensionnels entre différents pairs Active XML. Un pair Active XML est d'abord un entrepôt de documents AXML. Il est à la fois client (consommateur de services Web) et fournisseur de services déclaratifs définis par exemple par des requêtes (XQuery) ou des mises à jour sur les documents qu'il contient. L'utilisation d'appels de services Web inclus dans des documents permet de mettre à jour dynamiquement les données et de contrôler finement la périodicité de ces mises à jour et la pérennité des informations. La possibilité de mélanger des données et des appels de service dans un tel document permet à chaque pair de *matérialiser* (c'est-à-dire remplacer l'appel par son résultat) tous ou seulement une partie des appels de service avant l'échange du document. Les critères de ce choix sont multiples et dépendent des contraintes physiques et logiques du système et de l'application :

1. Par exemple, si la bande passante entre les deux pairs est faible, il est préférable de transmettre un appel de service à la place du résultat de cet appel qui a généralement une taille plus importante.
2. Un des deux pairs est incapable d'appeler le service pour des raisons de droits d'accès insuffisants.
3. Un des deux pairs ne veut pas appeler des services "inconnus" pour des raisons de sécurité.

Active XML possède deux composants fondamentaux :

- Les documents Active XML qui sont simplement basés sur le fait de faire des appels de services dans des documents XML.
- Les services Active XML qui permet de faire des appels à d'autres services.

[6] http://www.activexml.net

Les documents Active XML sont stockés dans les différents pairs. Ces derniers ont le rôle d'automatiser les appels de services et de mettre à jour les documents Active XML [BEN04].

Active XML permet de faire des appels explicites aux services web, ce qui pose des problèmes lors des appels de service par adresse physique (sans annotation sémantique) qui peuvent intervenir lors d'un changement de contexte de l'appel ou disponibilité de service ou lors d'une composition dynamique de services. Ce qui implique qu'il faut adapter «manuellement» les appels en cas de changement de services disponibles.

Prenons l'exemple de l'appel explicite suivant :

```
<Inventory> Inventory of the books of city libraries
<city name="GuangZhou">
<sc>zhongshan.com\getBooks()</sc>
        <sc>GuangZhou.com\Books()</sc>
</city>
<Inventory>
```

Fig. II. 2. Exemple I. appel de service web AXML

Ce qui a fortement motivé les chercheurs à définir des appels de services en terme d'ontologie indépendamment d'une adresse physique [ABI08], c'est-à-dire faire des appels implicites où on définit le domaine du service (catégorie), des paramètres d'entrée et de sortie et le système se charge de choisir *automatiquement* les services appropriés. Ce qui permettrait la découverte et la composition dynamique du service (source de données).

L'exemple précédent devient:

```
<Inventory> Inventory of the books of city libraries
<city name="GuangZhou">
<sc serviceCat="hierarchicalProfile.owl#book>
<output param_data_type="Concepts.owl#booklist" />
</sc>
</city>
</Inventory>
```

Fig. II. 3. Exemple 2. appel de service web AXML avec annotation

II. 5. 1. 2. Le projet Picsel

Picsel [GIR02] propose une structure « médiateur » qui permet d'interroger des sources d'information multiples, hétérogènes et éventuellement réparties. Les systèmes médiateurs auxquels PICSEL s'intéresse regroupent un ensemble important de sources d'information XML relatives à un même domaine d'application. PICSEL comporte:

1. le moteur de requêtes est conçu d'une façon générique pour être utilisable quel que soit le domaine d'application.

2. la base de connaissances est spécifique au domaine appliqué. Elle se compose d'une ontologie du domaine et des ontologies locales.

PICSEL utilise CARIN [LEV98] comme le langage de représentation de connaissances. Il possède également un langage de vues [GIR05] et un langage de requêtes permettant d'exprimer, en termes de l'ontologie du domaine, respectivement, le contenu des sources et les requêtes des utilisateurs. L'ontologie globale (l'Ontologie du domaine) dans PICSEL est créée à travers l'ONTOMEDIA (Ontologie pour un MEDIAteur)[GIR05].

II. 5. 1. 2. 1. L'intégration de services en PICSEL

Un service correspond, dans sa forme la plus simple, à une recherche d'informations. Un service peut aussi correspondre à l'exécution d'une fonctionnalité qui va modifier les données de l'application du fournisseur. Répondre à une demande d'un utilisateur peut nécessiter l'exécution d'un ou des plusieurs services. La communication entre un demandeur et un fournisseur d'un service est assurée par un mécanisme de transport de messages «demande/réponse ». Ces messages sont écrits sous forme de documents XML. Pour établir la communication, le demandeur émet un message vers le serveur du fournisseur du service. Ce message contient d'une part, le nom de la procédure qui déclenche le service et d'autre part, la valeur des paramètres nécessaires pour l'exécuter. Le protocole de transport véhicule le message. En réponse à l'appel effectué par le demandeur, le fournisseur exécute le traitement et retourne la valeur du résultat sous forme d'un document XML indiquant si l'exécution a réussi et en précisant la (ou les) valeur(s) de retour.

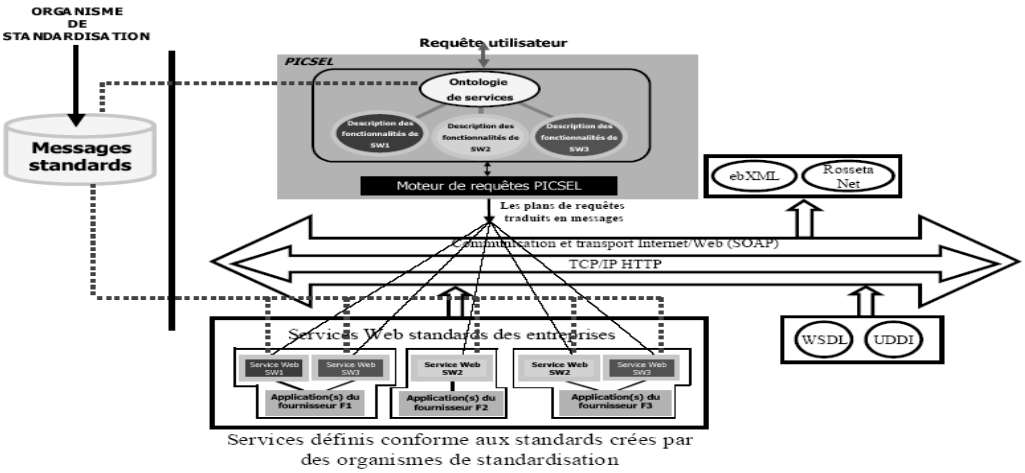

Fig. II. 4. Architecture de PicselWeb [GIR05].

Les entrées : Le moteur de PICSEL prend en entrée l'ontologie des services, la description des fonctionnalités des services et la requête de l'utilisateur qui correspond à la demande d'un service.

La sortie : Les réponses à une requête posée au médiateur PICSEL intégrant des services, comme d'ailleurs dans le contexte de l'intégration de sources d'information, peuvent être calculées par réécriture en termes de vues de cette requête. Dans le contexte de l'intégration de services, les plans de requêtes fournis par le moteur de requêtes PICSEL en réponse à la demande d'un service (pouvant être complexe) permettent d'identifier les fournisseurs des services demandés, d'indiquer la structure des messages à envoyer à chacun en permettant de les combiner pour obtenir des réponses complètes.

les applications de Picsel ont été développées dans le domaine du tourisme en exploitant les données standards de l'OTA (*Open Travel Alliance* [7]).

Discussion

AXML n'aborde pas la résolution de requête par la composition de service Web. Les services Web dans un document restent inchangés tout au long de la durée de vie le document et sont sélectionnés à l'heure de création du document. Ainsi, AXML est utilisé quand il y a une nécessité de gérer un ensemble prédéfini fixe des sources de données.

L'inconvénient de Picsel est qu'une fois l'ontologie partagée définie, chaque source doit utiliser le vocabulaire commun, ce qui limite l'autonomie des sources de données locales.

II. 5. 2. Autres travaux de recherche

WSMS dans [UTK06] porte sur l'optimisation des requêtes via les services Web DaaS. Un service DaaS est modélisée comme une relation WS (a_1, .., a_n), où a_i ($1 \leq i \leq n$) sont les attributs de la relation WS, et ils représentent Les entrées/sorties du service. Les utilisateurs expriment leurs requêtes directement en termes de relations de services DaaS conformément à la syntaxe suivante:

SELECT A_S

FROM $I(A_I) \bowtie WS_1(X_1, Y_1) \bowtie \ldots \bowtie WS_n(X_n, Y_n)$

WHERE $P_1(A_1) \wedge P_2(A_2) \wedge \ldots \wedge P_m(A_m)$

Where A_S is the set of projected attributes, I is the input relation, P_I is a predicate applied to the attribute A_I

Fig. II. 5. Exemple De WSMS

[7] http://www.opentravel.org/ Il s'agit d'un consortium qui regroupe plus de 150 organisations relatives à l'industrie du voyage

Les utilisateurs de ce système sont implicitement supposés avoir une bonne compréhension de la sémantique des services DaaS qui sont à leur disposition, et ils formulent leurs requêtes à travers les paramètres de projection / jointure entrées / sorties des services disponibles.

Dans notre architecture, les utilisateurs ne sont pas tenus d'avoir une connaissance implicite sur la sémantique des services DaaS disponibles et leurs entrées/sorties. Ils expriment leurs requetés à travers une ontologie de domaine et les requêtes sont réécrites en termes de services Web disponibles.

WSMED [SEB09] permet de définir des vues relationnelles sur des services web DaaS (vue WSMED). Le système donne la possibilité aux utilisateurs d'importer et de stocker les fichiers de description de services (WSDL) à l'intérieur du système et de définir des vues sur les services web importés. Les vues peuvent être définies directement sur les opération du service» ou sur d'autres vues, où ils sont appelés comme des vues multi-niveaux. Les utilisateurs peuvent interroger les données par les services web importés par la formulation de leurs requêtes (requêtes SQL) sur les vues définies. Les utilisateurs peuvent également améliorer les vues définies avec des contraintes de clé primaire afin d'optimiser le plan d'exécution d'une requête donnée.

L'un des principaux problèmes du système WSMED, c'est qu'il ne peut interroger et d'exécuter que les services importés de manière statique par les utilisateurs du système. L'inconvénient majeur du système WSMED est peut-être sa forte dépendance aux utilisateurs, les utilisateurs sont censés de savoir les services adaptés à leurs besoins; définir des vues au-dessus des services importés et d'améliorer les vue avec contraintes de clé primaire. En outre, cette dernière tâche exige que utilisateurs doivent avoir une bonne compréhension de la sémantique des services » chose qui ne peut pas être inférée à partir des fichiers de description de services.

Dans notre architecture, les utilisateurs sont soulagés de ces tâches manuelles, ils formulent leur requête à travers une ontologie de médiation et le système s'occupe du reste. En outre, dans notre système, les services Web DaaS sont modélisés comme des vues RDF sur les ontologies de domaine dans lesquelles les contraintes de clé primaire sont définies explicitement par des fonctions de Skolem. Donc les clés primaires sont déjà incluses dans le modèle de traitement de la requête.

Ouzzani et Bouguettaya dans [OUZ04] ont proposé un modèle de requête intéressant pour les services Web traditionnels (services SaaS). Le modèle proposé se compose de trois niveaux: le niveau requête, niveau virtuel et le niveau concret. Au niveau requête, les utilisateurs formulent leurs requêtes déclaratives sur un ensemble de relations de domaine. Par exemple *Rooms (City, Start, End, Rate)* and *Cars(City, Start, End, Model, Rate)* peut servir d'interface de requête dans le domaine d'application du tourisme pour interroger la disponibilité des hôtels chambres et des voitures dans une ville donnée sur une période de temps donnée.

Les relations de domaine sont mises en correspondance avec un ensemble d'opérations de services web au niveau virtuel. Ces opérations ont été appelés des «opérations virtuelles » soi-disant parce qu'ils n'appartiennent à aucun service web actuel et ils représentent les différents services qui peuvent offrir des fonctionnalités dans un domaine particulier. Au niveau concret ces opérations virtuelles sont utilisées pour représenter leurs fonctionnalités. Ces opérations virtuelles sont adaptés ensuite aux diverses opérations de niveau concret.

Les règles de correspondance «mappage» entre les relations de domaine et les opérations virtuelles ont la forme de requêtes, à savoir les relations de domaine sont définies comme des requêtes sur des opérations virtuelles. Ce qui suit est un exemple d'une règle de mappage:

Airlines(DepartueCity, ArrivalCity, AirelineNames, WebSites):-
InquireAirelines(DepartueCity, ArrivalCity, AirlineNames),
GetWebSites(AirlineNames, WebSites)

Notre architecture suit un schéma similaire de requête pour interroger les services DaaS, les requêtes sont formulées à travers une ontologie de médiation.

Les services dans le cadre [OUZ04] sont liés à l'opération virtuelle dont ils adhèrent, alors qu'ils sont liés à une vue RDF dans notre cadre à l'aide d'un algorithme de réécriture de requête.

II. 5. 3. Plateformes industrielles pour les services web fournisseurs de données.

Les services de données ont gagné une attention considérable de leaders de l'industrie des logiciels SOA au cours de ces dernières années. Beaucoup de produits sont actuellement proposés ou en cours d'élaboration pour la création de services de données(DaaS) plus facile que jamais, nous citons : AquaLogic de BEA Systems [CAR06], Astoria de Microsoft [MIC07], MetaMatrix de RedHat [RED07], Composite Software [COM08], Xcalia [INC09],

et IBM [WIL06]. Les produits proposés ici intègrent des sources de données de l'entreprise et fournissent un accès uniforme aux données grâce à des services de données. En plus, la plus parts des produits commerciaux de bases de données incorporent des mécanismes d'exportations des fonctionnalités des bases de données comme des DaaS [KHO09]. par exemple «IBM Document Acces Definition Extention (DADX)», technologie (Db2XMLextender [8]) et le Native XML Web Services for Microsoft SQL Server 2005 [9]. DADX fait partie du XML Extender IBM DB2, un couche de mapping XML/relational , et facilite le développement de services Web sur une base de données relationnelle qui peut, entre autres choses, exécuter des requêtes SQL et récupérer des données relationnelles au format XML.

BEA AquaLogic Data Services Platform : BEA AquaLogic Data Services Platform [CAR06] a été conçu dès le départ pour fournir un appui pour le concept de services de données. Puisqu'elle vise le monde de SOA, elle prend une vue orientée services de données. BEA AquaLogic Data Services Platform est une collection de fonctions qui ont toutes un schéma de sortie commun, acceptent différents ensembles de paramètres et sont implémentées via des expressions XQuery individuelles. Dans un exemple simplifié, un service de données exporte un ensemble de fonctions retournant des objets d'un client, où une fonction prend comme entrée le nom du client, une autre son adresse et pays, etc. AquaLogic exporte ces services de données aux développeurs des applications SOA comme un service web de données, où les fonctions deviennent des opérations.

Le projet « Astoria » : En 2007, Microsoft propose le projet « Astoria » [MIC07], connu aujourd'hui sous le nom d'ADO.NET Data Services. C'est un Framework permettant d'effectuer des requêtes sur des services de données disponibles sur le web ou en intranet. Les données sont transmises sous la forme de requêtes REST (Representational State Transfer), identifiées par des URL. Les applications clientes utilisent bien souvent des requêtes HTTP (GET, POST, PUT et DELETE) afin de procéder aux différentes transactions. Les formats utilisés pour représenter et transporter ces données sont en général ceux que l'on retrouve dans des formats d'échange de données tels que JSON, AtomPub et surtout XML.

MetaMatrix : MetaMatrix [RED07], fournit des outils déclaratifs pour créer une gamme importante de services de données, un référentiel pour stocker les définitions des services de données avec leurs métadonnées associées et enfin un environnement d'exécution robuste qui

[8] http://www.306.ibm.com/software/data/db2/extenders/xmlext/
[9] http://msdn2.microsoft.com/en-us/library/ms345123.aspx

assure des performances d'entreprise, l'intégrité et la sécurité des données. MetaMatrix Enterprise offre une solution plus performante et plus rapide pour répondre aux défis posés par les données SOA. La plateforme MetaMatrix Data Services supporte différentes sources de données. Les applications accèdent aux services de données au travers de SQL ou des interfaces Web services.

Xcalia Intermediation Core (XIC) [INC09] est une plate-forme d'intermédiation permettant à une entreprise d'accéder à l'ensemble de ses données, de déployer des *Applications Métier*, à partir de briques composites. XIC réalise du Mapping de Données mais aussi du Mapping de services (incluant les nouveaux Services Web, les services Mainframe, Legacy, etc.). XIC est basé sur les standards SDO (Service Data Objects), JDO (Java Data Objects), EJB (Entreprise Java Bean). XIC permet de faire coexister les *applications métier* avec des environnements totalement hétérogènes.

Composite Software : La virtualisation de données avec Composite Software [COM08] permet aux entreprises d'intégrer de manière très flexible leurs données internes. Parmi les fonctionnalités clés de Composite Software, on pourra en particulier citer le fait que :

- Composite apporte aux développeurs bases de données et Java, un environnement de développement relationnel appelé 'Composite Studio'. Pour les développeurs XML / SOA utilisant des outils de développement, orientés modélisation relationnelle pour construire des vues fédérées et des services de données, on utilisera le module 'Composite Designer'.
- Les Services d'intégration de données écrits et exécutés par le 'Composite Information Server' sont parfaitement adaptés pour être opérés à travers l'Internet.
- Parce que Composite accède, fédère, résume et délivre les données à la demande, il n'est pas nécessaire de prévoir de zone de stockage de ces données sur le réseau étendu.

En outre, Composite supporte de multiples options de sécurité pour garantir un accès et une délivrance sécurisée des données idoines aux utilisateurs authentifiés et approuvés en interne ou en externe. Composite démystifie la complexité de la modélisation de données, au profit de leur utilisation dans le 'Cloud'.

Discussion

Dans ces plates-formes, la sémantique d'un service web DaaS est connue, tant que le développeur d'applications SOA, reste dans la plate-forme (par exemple AquaLogic). Une fois il sort de cette plateforme, par exemple lorsque les services web DaaS sont publiées dans un registre de service en dehors des frontières de l’entreprise, il devient difficile de distinguer entre les services tant que leur sémantique n’est pas définie. Il est nécessaire de compléter ces efforts industriels, en offrant un cadre permettant de décrire la sémantique des dervices web DaaS et ces entrées/sorties.

II. 6. Conclusion

L’objectif de ce chapitre était de définir le cadre dans lequel notre travail va se placer, à savoir la notion de composition de services Web, l’intégration des données et les services web DaaS qui permettent l'accès aux données des organismes par l'intermédiaire des Services web.

L'invocation d'un Service web fournisseurs des données (DaaS) a comme conséquence l'exécution d'une requête au-dessus des sources de données. La composition de service Web permet de répondre aux besoins d'un utilisateur ne pouvant être satisfaits par un seul Web service, alors qu'une intégration de plusieurs le permettrait. Les méthodes de composition, telles qu'elles sont appliquées aux services Web traditionnels (i.e. *AaaS Application-as-a-Service* Web services), ne permettent pas de prendre en compte la relation sémantique entre les entrées/sorties d'un service Web d’accès aux données, et en conséquence, elles ne sont pas adaptées pour composer les services Web d’accès aux données.

Dans le chapitre suivant nous présenterons notre proposition qui consiste à étendre le modèle de Barhamgi et al. en introduisant la notion de parallélisme pour optimiser la composition et la recherche d’informations.

Chapitre III

Conception : Optimisation de la composition des DaaS par invocations Parallèles de services web

III. 1. Introduction

La composition permet de combiner des services pour former un nouveau service dit composé ou composite. L'exécution d'un service composé implique des interactions avec des services partenaires en faisant appel à leurs fonctionnalités. Le but de la composition est avant tout la réutilisation de services (simples ou composés) et de préférence sans aucune modification de ces derniers.

Les méthodes de composition, telles qu'elles sont appliquées aux services web traditionnels, i.e. AaaS (***A****pplication-**a**s-**a**-**S**ervice Web services*), ne permettent pas de prendre en compte la relation sémantique entre les entrées/sorties d'un DaaS (***D****ata-**a**s-**a**-**S**ervice*), et en conséquence, elles ne sont pas adaptées pour composer ce type de service.

Certains chercheurs (BARHAMGI et al.[BAR10]) proposent d'exploiter les principes de base des systèmes d'intégration des données pour composer les DaaS. Plus précisément, ils proposent la modélisation de DaaS comme des vues sur une ontologie de domaine. Cela permet de représenter la sémantique d'un service d'une manière déclarative en se basant sur des concepts et des relations dont les sémantiques sont formellement définies. Ensuite, ils utilisent les techniques de réécriture des requêtes pour sélectionner et composer automatiquement les services pour répondre aux requêtes des utilisateurs.

Dans ce chapitre nous présenterons notre approche permettant d'optimiser la composition de DaaS et la recherche d'informations en introduisant la notion de parallélisme au modèle de (BARHAMGI et al. [BAR10]).

III 2. Problèmes avec les approches traditionnelles :

Considérons le cas d'un scénario e-prescription[10] dans lequel un médecin cherche à vérifier si le médicament qu'il veut prescrire peut interagir avec d'autres médicaments déjà prescrits à son patient. Supposons que ce médecin a l'ensemble de services dans le tableau suivant:

Service	La sémantique du service	Contraintes
S_1(?a, $b)	Renvoie des médicaments (*code b*) pris par un patient donné (*identificateur a*)	a ≥ x888
S_2(?a, $b)	Renvoie des médicaments (*codes b*) pris par un patient donné (*identification a*)	a ≤ x888
S_3(?a, $b)	Renvoie les interactions de médicaments (b) d'un médicament donné (a)	
S_4(?a, $b)	Renvoie divers informations sur un médicament donné	a ≥ p660
S_5(?a, $b)	Renvoie divers informations sur un médicament donné	a ≤ x8999
S_6(?a, $b)	Renvoie les médicaments équivalents (b) à un médicament donné (a)	

Tableau III. 1. Exemple de services web.

Les problèmes qui peuvent être rencontrés :

- procédure manuelle de sélection et d'invocation de services : le médecin doit effectuer plusieurs tâches pour exécuter sa requête. Ces tâches peuvent être pénibles dans des domaines d'application tels que la bio-informatique vu le nombre important de services web. Tout d'abord, le médecin doit passer par un nombre impressionnant de services DaaS, comprendre la sémantique des entrées et de sorties de chaque service. Par exemple, 'S_3' et 'S_6' ont les mêmes paramètres d'entrée et de sortie (c.-à-d. médicament). Cependant, 'S_3' renvoie les interactions des médicaments avec un médicament donné « a » tandis que 'S_6' renvoie les médicaments qui sont équivalents à « a ». Deuxièmement, il a besoin de sélectionner manuellement les services qui sont pertinents pour sa requête et déterminer leur ordre d'exécution. Troisièmement, après que les services sont choisis, il a besoin de les invoquer manuellement l'un après l'autre et calculer la jointure et filtrer les données non pertinentes.

[10] http://en.wikipedia.org/wiki/E-prescribing

- description imprécise des DaaS : les DaaS ont des requêtes paramétrées. La sémantique d'un DaaS réside non seulement dans ses types d'entrée et de sortie mais également dans la façon dont l'entrée et la sortie sont interconnectées en relation (sémantique entrée-sortie). par exemple, les deux services ont les mêmes types d'entrée et de sortie (c.-à-d. médicament) mais ils ont complètement deux sémantiques différentes, un service renvoie les interactions des médicaments et l'autre renvoie les équivalents. De tels rapports sémantiques d'entrée-sortie peuvent être représentés de manière déclarative à l'aide des vues sur une ontologie de domaine. Notons que, parfois, ces relations sémantiques peuvent être très complexes. Considérons par exemple le service où les médicaments retournés sont «pris» par le patient et ils comprennent à la fois «prescris», ainsi que les médicaments en vente libre. Supposons maintenant un second service qui donne les médicaments prescrits (à savoir, exclut les médicaments en vente libre), le rapport sémantique (entre l'entrée-sortie) doit ici non seulement exprimer le fait que le médicament retourné « est pris » par le patient, mais également le médicament retourné « est prescrit par un médecin » D qui « soigne » le patient.

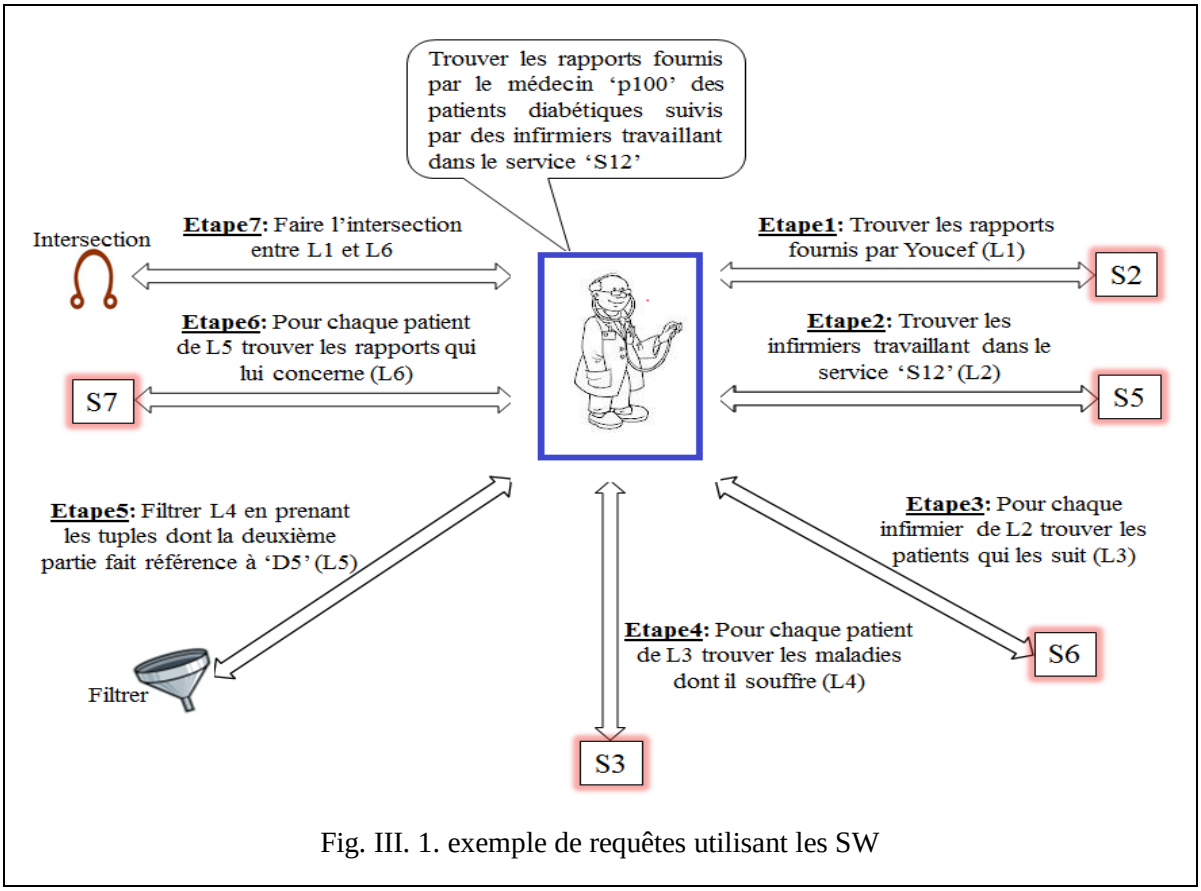

Fig. III. 1. exemple de requêtes utilisant les SW

Le OWL-S service profil permet la modélisation de l'entrée du service, de la sortie, de ses conditions préalables exigées et des effets produits pour invoquer le service. Il permet également la catégorisation des services selon leur fonctionnalité dans un domaine utilisant la classe de catégorie de service en même temps que quelques arrangements de catégorisation disponibles dans un domaine d'application [**BOU08**]. Les DaaS sont simplement concernés la recherche des données de sortie appropriées en en donnant une entrée spécifique. Ils ne fournissent aucune fonctionnalité, au-delà de la recherche, et n'ont aucun effet. Le souci est capturer correctement la manipulation de la relation sémantique entre leurs entrées et sorties. [BAR09]. Dans divers DaaS, il peut y exister un ensemble de services qui ont les mêmes entrées et sorties mais différents seulement dans la sémantique joignant les deux. Par conséquent l'OWL-S ne peut pas être le meilleur choix pour décrire les DaaS [WEN08].

III. 3. L'architecture du système proposée

Pour résoudre le problème de rapport sémantique entre les entrées et sorties, on peut suivre les étapes suivantes :

1. Décrire sémantiquement les classe DaaS. En effet, les langages sémantiques actuels pour les services web comme OWL-S n'offrent pas la possibilité d'exprimer des relations sémantiques d'entrée/sortie [WEN08]. Modéliser les DaaS comme des vues RDF paramétrées sur une ontologie de domaine [VAC08].
2. Résolution de requête par la composition d'un ensemble minimum DaaS par l'utilisation d'un algorithme de réécriture de requête-orienté RDF pour des requêtes non-paramétrées et paramétrées respectivement. Le traitement des requêtes paramétrées traite le cas où plusieurs services fournissent la même sémantique de sortie. En conséquence, il y a un besoin de déterminer l'ensemble minimum de services qui couvrent une requête donnée [BAR09].
3. Optimisation du service composé pour accélérer son exécution. Ceci est fait en déclenchant un mécanisme de sélection de service pendant l'exécution basée sur les valeurs réelles des paramètres d'entrée [OUK08].
4. Introduire la notion de parallélisme dans l'invocation des services web.

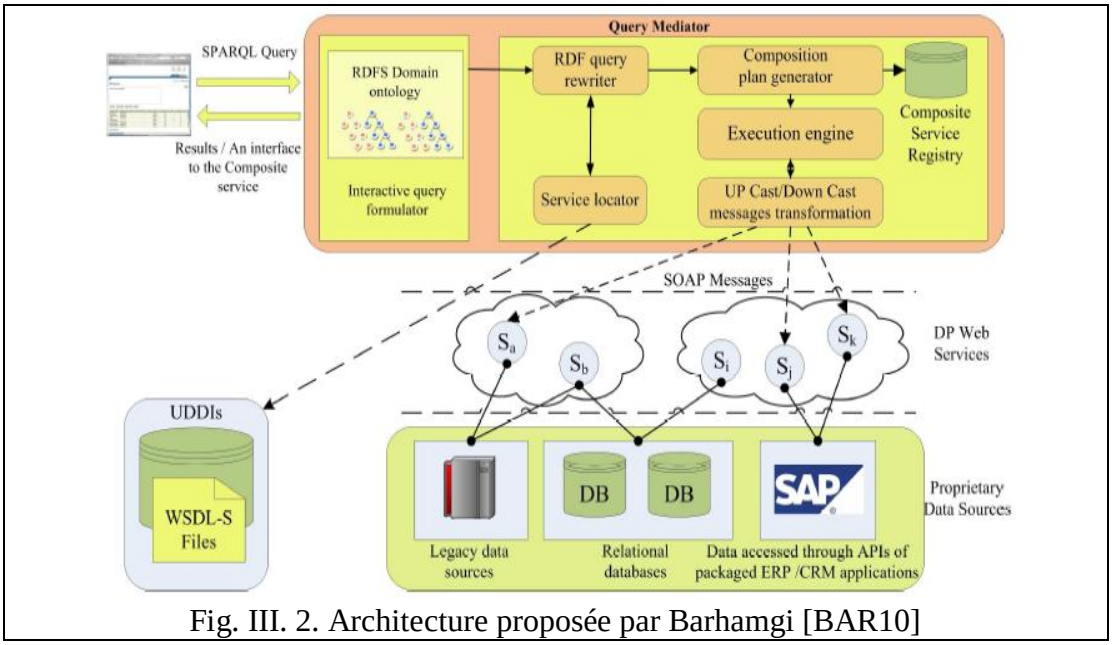

Fig. III. 2. Architecture proposée par Barhamgi [BAR10]

III. 3. 1. L'Ontologie de médiation

Dans l'approche proposée, les utilisateurs formulent leurs requêtes à travers d'une ontologie de médiation. Cette dernière est décrite en RDF/RDFS. Formellement, une ontologie de médiation est définie par 6-tuple (C, L, TP, OP, SC, SP**) [BAR09]**: où :

- C : est un ensemble de classe.
- L : est un ensemble de types de données (*datatypes*).
- TP : est un ensemble de propriétés de type de données (*datatype properties*) chaque propriété de type de données à un domaine en C et un intervalle en L.
- OP est l'ensemble de propriétés des objets : chaque propriété d'objet a son domaine et l'intervalle en C.
- SC : est une relation de CxC représentant la relation de sous-classe entre les classes.
- SP : est une relation de OPxOP U DPxDP représente les relations sous propriétés entre les propriétés homogènes, par exemple $p_2 Sp_1$ veut dire que p_2 est une sous propriété de p_1.

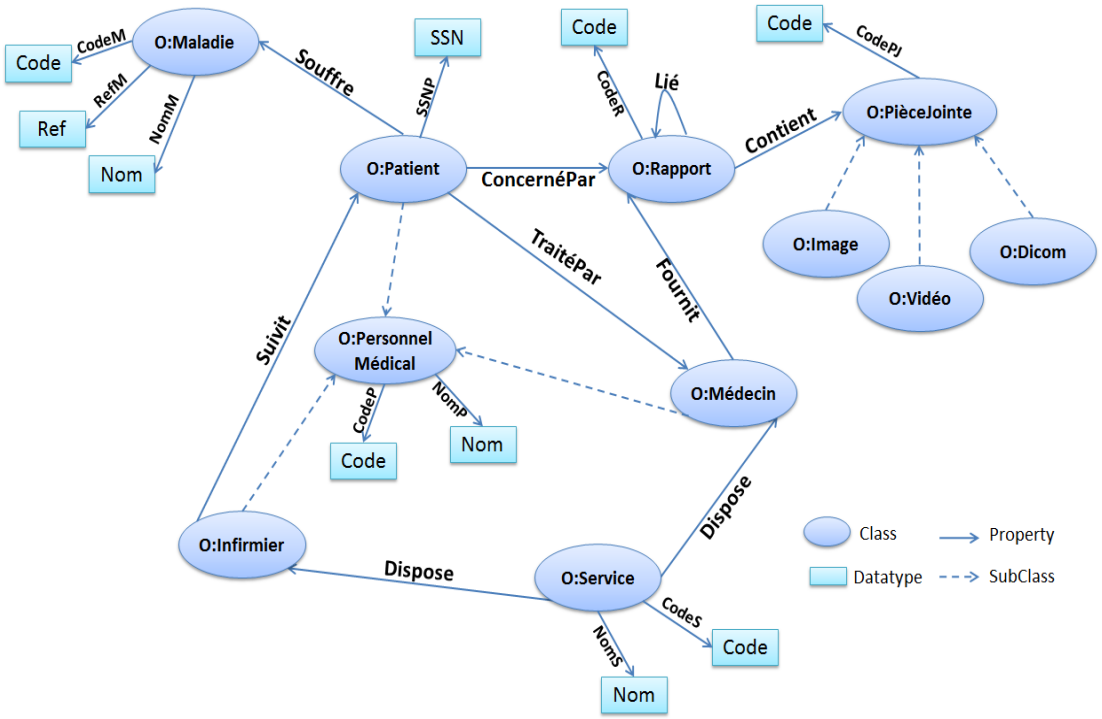

Fig. III. 3. Exemple d'e-prescription

Prenons les trois concepts Patient, Médecin et Service, cette ontologie spécifie que le patient est traité par un médecin et ce dernier travaille dans un service. Un Service a deux caractéristiques (*DataTypes*) qui sont le Code et le Nom. Les concepts sont reliés entre eux par des Object *Properties* et aux *DataTypes* via les *DataType Properties*.

III. 3. 2. Requête :

Les requêtes sont exprimées en utilisant SPARQL. Formellement une requête Q a la forme suivante : $Q(\overline{X}) :- G(\overline{X},\overline{Y})$ **[VAC08] [ZHA05].** Où

$Q(\overline{X})$: est l'entête de la requête et $G(\overline{X},\overline{Y})$ est le corps de la vue, c'est un ensemble de triple RDF où chaque triple est de la forme : <*subject.property.object* >). Dans une certaine mesure, G définit la sémantique de l'attribut apparenté de la perspective de l'ontologie RDF. Les variables qui apparaissent dans l'entête de requête $\overline{X}$ s'appellent les variables distinguées, tandis que ceux qui apparaissent dans le corps, tel que $\overline{Y}$, s'appellent les variables existentielles.

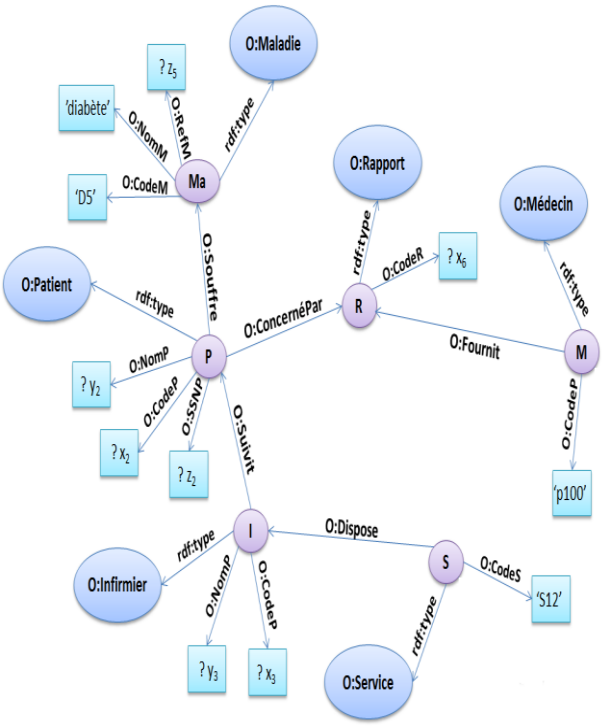

Fig. III. 4. Représentation graphique de la requête Q

Q(y_1,x_2,y_2,z_2,x_3,y_3,y_4,z_5,x_6):-

?M	.rdf:type	.O:Médecin
?M	.O:CodeP	.'p100'
?M	.O:Fournit	.?R
?P	.rdf:type	.O:Patient
?P	.O:CodeP	.?x_2
?P	.O:NomP	.?y_2
?P	.O:SSNP	.?z_2
?P	.O:ConcernéPar	.?R
?P	.O:Souffre	.?Ma
?I	.rdf:type	.O:Infirmier
?I	.O:CodeP	.?x_3
?I	.O:NomP	.?y_3
?I	.O:Suivit	.?P
?S	.rdf:type	.O:Service
?S	.O:CodeS	.'S12'
?S	.O:Dispose	.?I
?Ma	.rdf:type	.O:Maladie
?Ma	.O:CodeM	.'D5'
?Ma	.O:NomM	.'diabète'
?Ma	.O:RefM	.?z_5
?R	.rdf:type	.O:Rapport
?R	.O:CodeR	.?x_6

Fig. III. 5. Représentation SPARQL de la requête Q

Exemple : requête Q = Donnez Les rapports fournis par le médecin 'p100' des patients souffrant de la maladie du diabète identifiée par le code 'D_5' et qui sont suivis par des infirmiers travaillant dans le service 'S_{12}'.

Une requête Q peut être vue comme un graph avec deux types de nœuds : [**BAR09**]

1) **les nœuds de classe :** sont des nœuds dont les types sont des classes en ontologie. Les nœuds de classe sont liés par des propriétés d'objet.
2) **nœuds littéraux :** nœuds dont les types sont des *datatypes*. Des nœuds littéraux sont joints avec des nœuds de classe par l'intermédiaire des propriétés de *datatype*.

III 4. Modélisation de DaaS comme Vues.

Les Vues RDF sont indépendants du schéma physique et de la technologie particulière utilisée dans la source de données. La vue RDF nous dit quelles variables sont directement fournies par la source de données et comment ces éléments de données se rapportent à OWL, cas dans le schéma de médiation. Les vues RDF fournissent

suffisamment d'informations sur la structure des données stockées dans les DaaS aux fins de découverte, ainsi que pour l'identification de requête et de réponse.

Une Vues RDF paramétrée peut être vue comme une requête SPARQL paramétrée. Une Vues RDF paramétrée sur une ontologie de domaine est définie comme suit [BAR09]: $S_i(\overline{X_i},\overline{Y_i}) :\prec \phi(\overline{X_i},\overline{Y_i},\overline{Z_i}), Ct_i >$ Où Xi sont des variables d'entrée. Yi sont des variables de sortie. Zi sont des variables existentielles et Ct_i sont les variables de contrainte sur X ou Y. (Il existe deux types de variables pour les vues RDF. Les variables figurant dans l'entête est souvent appelées variables distinguées. Les variables qui figurent seulement dans le corps de vue, mais pas dans l'entête sont appelés des variables existentielles [HUA06].

S_1,S_2($a,?b):-
(?M .rdf:type .O:Médecin)
(?M .O:CodeP .$a)
(?R .rdf:type .O:Rapport)
(?R .O:CodeR .?b)
(?M .O:Fournit .?R)

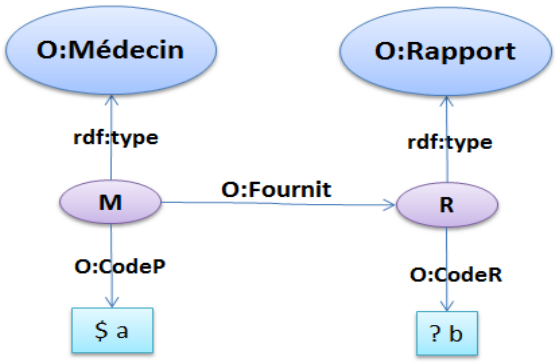

Fig. III. 6. Graphe RDF des services S_1 et S_2

Les services S_1 et S_2 donnent les rapports (b) fournit par un médecin (a).

III. 5. Pré-traitement des vues RDF

Les vues de RDF sont prétraitées avant la résolution de requête, ils peuvent subir l'une des étapes suivantes:[**BAR08**]

1. Application des contraintes sémantiques RDFS

Dans cette étape, les vues RDF sont étendues pour prendre en compte les contraintes sémantiques RDFS de l'ontologie de médiation. Les contraintes sémantiques RDFS inclus : "*rdfs:subClassOf*", "*rdfs:subPropertyOf*", "*rdfs:domain*" et "*rdfs:range*".

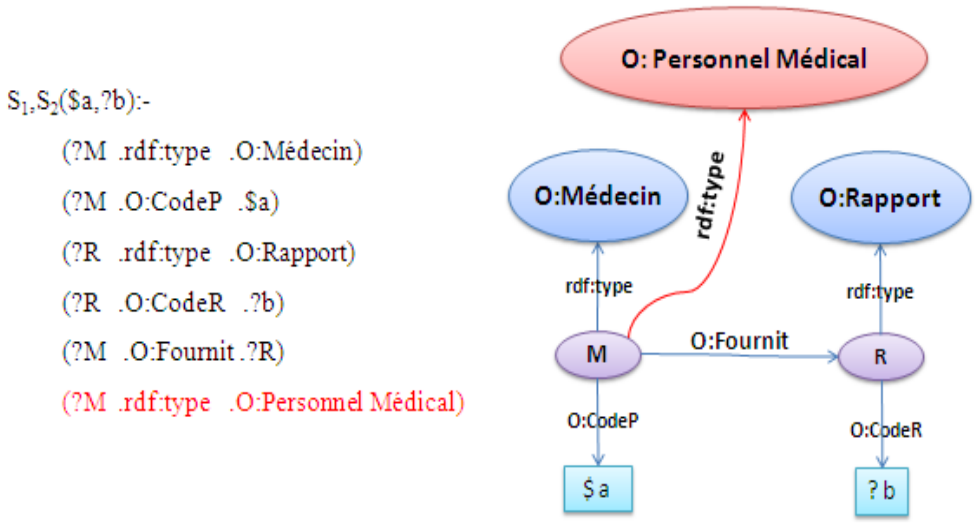

Fig. III. 7. Graphe RDFS des services S_1 et S_2

2 . Association des fonctions Skolem aux nœuds classe.

« La skolémisation est une transformation syntaxique utilisée de manière routinière dans les systèmes d'inférence automatiques, dans lesquels les variables existentielles sont remplacées par des fonctions « nouvelles » — des noms de fonction non utilisés ailleurs — appliquées aux variables universelles englobantes. En RDF, la skolémisation équivaut à remplacer chaque nœud anonyme par un « nouveau » nom, c'est-à-dire une référence URI qui est garantie ne pas apparaître ailleurs.»[W3C04a].

chaque variable dénotant une classe est conceptuellement associé à une fonction skolem qui aide à fusionner des exemples provenant des différents services par exemple la variable *?P* (de type patient) est associée à une fonction *F1* (hasName), c'est-à-dire si deux exemples ont le même nom puis ils sont considérés en tant que dénotation de la même entité et peuvent être fusionnés ainsi. Les propriétés d'une fonction skolem pour une classe particulière sont choisies par l'expert en matière de domaine.

3. Inverser les vues définies

L'approche LAV (Local As View) consiste à définir les schémas des sources de données à intégrer en fonction du schéma global. L'approche LAV est très flexible par rapport à l'ajout ou la suppression de sources de données. Le schéma global n'est pas impliqué dans

la mise à jour du nombre de sources à intégrer. Pour cela, il suffit d'ajouter ou de supprimer les vues sur les sources. C'est un gros avantage de l'approche LAV mais l'inconvénient de cette approche est la complexité de la construction des réponses à une requête utilisateur. Pour les construire, il est nécessaire de réécrire la requête en termes des vues [**GIR05**].

La première étape dans ce processus est inverser les vues locaux pour obtenir les définitions des relations de schéma de médiation comme vues sur des relations du schéma local. Les vues locales sont mises sous une forme appropriée pour reformuler les requêtes globales. Spécifiquement, des propriétés d'objet sont groupées ensemble dans les vues et des triples sont groupés selon leurs sujets.

III. 6. Algorithme de réécriture de requête

Soient une requête Q et un ensemble de services représentés par leurs vues correspondantes $V = v_1, v_2 \ldots v_m$. une réécriture Q utilisant les services est construite en composant des services tels que l'union de leurs graphes couvre le graphe RDF de la requête. Le terme «couverture" signifie :

i. toutes les propriétés d'objet qui se tiennent entre les nœuds de classe dans Q sont couvertes par le graphe de composition ;
ii. il y a un mapping β entre les nœuds de classe dans Q avec ceux dans le graphe de composition, c.-à-d. ils ont les mêmes types (classe), β mappe également les nœuds littéraux dans Q à quelques nœuds littéraux dans le graphe de composition ; et
iii. des variables distinguées dans Q (c.-à-d. les coquilles demandées) sont fournies par la composition.

Les services individuels (vues) peuvent couvrir seulement des parties du graphe de la requête. Pour cette fin, la réponse de la requête peut seulement être obtenue en examinant les divers services qui remplissent les conditions ci-dessus.

La construction de compositions est un processus en deux phases [BEN08] :

a. l'algorithme de réécriture compare Q à chaque vue v dans V, détermine les nœuds couverts et les propriétés de l'objet (de Q) dans chaque v, et documente alors cette information comme inclusion partielle dans une table de mapping et de connectivité. Cette table inclut les différentes manières d'utilisation d'une une vue pour couvrir une partie de la requête.
b. l'algorithme combine des vues basées sur les informations extraites dans la première phase de telle façon que le graph résultant de la composition couvre le graphe de la requête et la composition soit exécutable.

III. 7. Optimalisation de la composition

Chacun des services constitutifs dans une composition a ses contraintes sur les valeurs de paramètres d'entrée. L'invocation d'un service avec une valeur de données violant ses contraintes renverra un résultat vide réglé ou jettera un message d'erreur. Pour réduire ce temps système inutile, insérer sur l'entrée de chaque service constitutif un filtre pour éliminer des invocations fausses. Un filtre inséré vérifie si une valeur d'entrée satisfait les contraintes du service avant la marche à suivre avec l'invocation.

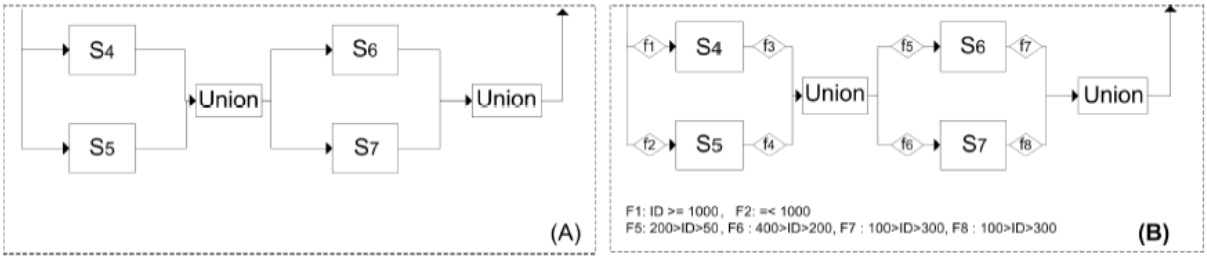

Fig. III. 8. Le service composé. (a). avant d'ajouter les filtres, (b). après avoir ajouté les filtres

III. 8. Invocations Parallèles de services web

Dans cette section nous présenterons notre proposition pour l'optimisation de la composition des services web DaaS.

Le défi est d'accélérer les requêtes de tels appels de services web. Des invocations parallèles peuvent accélérer l'exécution des requêtes avec plusieurs appels de service web, nous présentons une approche pour paralléliser les appels de service web tout en gardant les dépendances parmi elles.

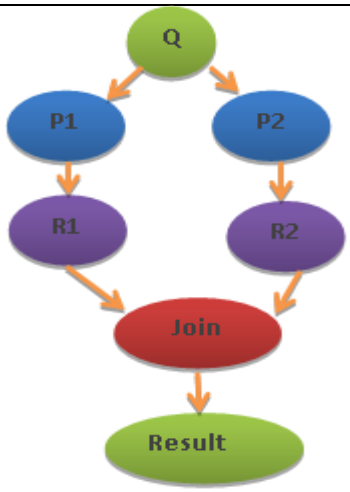

Fig. III. 9. Processus multi niveaux

Notre approche consiste à paralléliser les appels des services web par des processus de requêtes séparés. Pour une requête donnée, les processus de requête sont automatiquement arrangés dans un arbre de processus multi –niveaux où les services web sont appelés en parallèle.

Un service web sera traité comme un processus séparé, lorsque l'utilisateur invoque des services web indépendants, ces services web seront traités en parallèles en tant que processus séparés. Les plans d'exécution multi-niveaux sont générés avec plusieurs couches de parallélisme dans différents processus de requête.

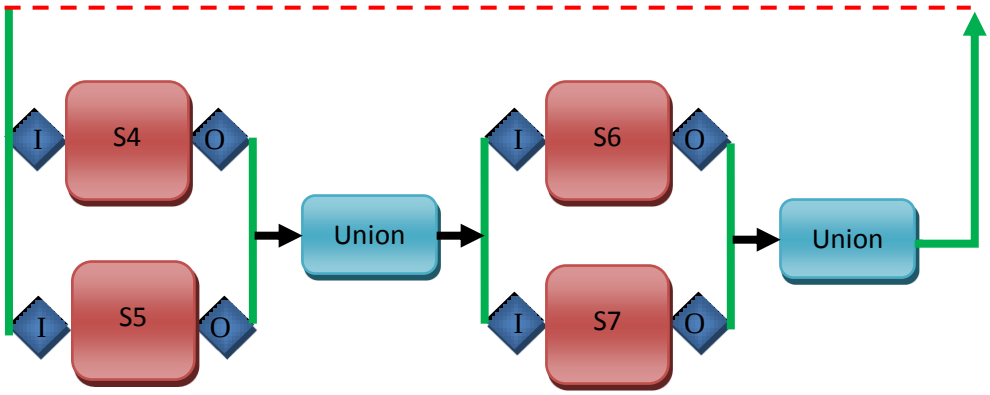

Fig. III. 10. Exemple d'exécution parallèle

Cela constitue à mettre en arbre les processus pour la requête. Chaque processus de requête enfant fournit le résultat de retour à son processus parent de façon asynchrone. Les résultats des enfants sont fournis au parent en parallèle comme flots. Ce qui est l'objet de notre approche.

III. 9. Conclusion

Nous avons présenté une nouvelle approche permettant d'optimiser la recherché d'information dans le modèle proposé par Barhamgi et al. Qui propose de modéliser DaaS comme des vues RDF sur une ontologie de domaine.

Chaque vue RDF contient des concepts et des relations de l'ontologie de domaine pour saisir les rapports sémantiques entre l'entrée et les paramètres de sortie. En second lieu, il propose des algorithmes de réécriture de requête pour traiter des requêtes sur des Services web DaaS. Le médiateur de requête transforme automatiquement une requête d'utilisateur (pendant l'étape de réécriture de requête) en composition DaaS.

Nous avons proposé d'étendre ce modèle par des appels parallèles de services web. Ces services web seront traités en parallèles en tant que processus séparés.

Chapitre IV

Réalisation du prototype.

IV. 1. Introduction

Le DaaS permet l'accès aux données des organismes par l'intermédiaire des Services web. L'invocation d'un service web fournisseurs des données a comme conséquence l'exécution d'une requête au-dessus des sources de données. Dans la plupart des cas, les requêtes des utilisateurs exigent la composition de plusieurs services.

Dans ce chapitre, nous présenterons notre plateforme de test de la composition de services web d'accès aux données. Notre approche consiste à invoquer en parallèles ces services web.

Le principal domaine d'application de notre plateforme est le domaine d'e-santé, où les services Web d'accès aux données sont utilisés pour partager les dossiers médicaux des patients.

Dans la première section de ce chapitre, nous présentons l'architecture de notre application, à travers une étude des méthodes et des services web utilisés pour invoquer les DaaS en parallèle.

Ensuite, la deuxième section est réservée pour exposer les détails d'implémentation appliqués pour une étude de cas d'e-santé.

Enfin, la dernière section est consacrée pour présenter les résultats de l'évaluation des mécanismes précédemment décrits.

IV. 2. Architecture du prototype

Dans cette section, nous allons décrire avec détails l'architecture de notre application ainsi que les méthodes et les services web utilisés.

Notre prototype est développé sous l'environnement Java (NetBeans IDE 7.2 [11] et JDK 7u7 [12]). Nous avons utilisé la Librairie FreeChart[13] 1.0.13 pour dessiner les graphes

Nous avons utilisé le kit de déploiement fourni avec le GlassFish Web server[14] 3.1.2 pour déployer nos services Web DaaS.

L'architecture proposée est organisée en quatre couches. La première couche contient un ensemble de bases de données qui stockent les données médicales (c'est à dire le contenu du dossier de soins médicaux). La seconde couche comprend une application qui accède à aux bases de données de la première couche (c'est à dire qu'il exécute des requêtes paramétrées sur les bases de données).

Ces applications sont exportées sous forme de services Web DaaS pour le système et constituent la troisième couche.

La couche supérieure comprend une interface graphique permettant à l'utilisateur d'exécuter ces requêtes.

Les services web DaaS déployés peuvent être invoqués en parallèle ou de manière séquentielle selon le choix de l'utilisateur, et voici quelques services utilisés dans notre prototype.

- «PatientBySSNWS » qui peut être testé en tapant l'adresse : http://localhost:8080/MedicalServices/MedicationInfoByIDService?Tester
- «MedicationsInfoByID» qui peut être testé en tapant l'adresse : http://localhost:8080/MedicalServices/MedicationInfoByIDService?Tester
- « InteractingMedications» qui peut être testé en tapant l'adresse : http://localhost:8080/MedicalServices/InteractingMedicationsService?Tester
- «InversePatientBySSNWS» qui peut être testé en tapant l'adresse : http://localhost:8080/MedicalServices/InversePatientBySSNWSService?Tester
- «MedicationByPatient» qui peut être testé en tapant l'adresse : http://localhost:8080/MedicalServices/MedicationByPatientWSService?Tester

[11] http://netbeans.org/
[12] http://www.oracle.com/technetwork/java/javase/downloads/index.html
[13] http://sourceforge.net/projects/jfreechart/files/1.%20JFreeChart/1.0.13/jfreechart-1.0.13.zip/download
[14] https://glassfish.dev.java.net/

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!-- Generated by JAX-WS RI at http://jax-ws.dev.java.net. RI's version is JAX-WS RI 2.1.3.3-hudson-757-SNAPSHOT. -->
<definitions targetNamespace="http://org.me/" name="PatientBySSNWSService"
xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:tns="http://org.me/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
 <types>
  <xsd:schema>
   <xsd:import namespace="http://org.me/" schemaLocation="PatientBySSNWSService_schema1.xsd"/>
  </xsd:schema>
 </types>
 <message name="PatientBySSNWS">
  <part name="parameters" element="tns:PatientBySSNWS"/>
 </message>
 <message name="PatientBySSNWSResponse">
  <part name="parameters" element="tns:PatientBySSNWSResponse"/>
 </message>
 <portType name="PatientBySSNWS">
  <operation name="PatientBySSNWS">
   <input message="tns:PatientBySSNWS"/>
   <output message="tns:PatientBySSNWSResponse"/>
  </operation>
 </portType>
 <binding name="PatientBySSNWSPortBinding" type="tns:PatientBySSNWS">
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
  <operation name="PatientBySSNWS">
   <soap:operation soapAction=""/>
   <input>
    <soap:body use="literal"/>
   </input>
   <output>
    <soap:body use="literal"/>
   </output>
  </operation>
 </binding>
 <service name="PatientBySSNWSService">
  <port name="PatientBySSNWSPort" binding="tns:PatientBySSNWSPortBinding">
   <soap:address location="REPLACE_WITH_ACTUAL_URL"/>
  </port>
```

Fig. IV. 1. Exemple –"PatientBySSNWSService.WSDL"

Nous avons développé une classe java permettant d'invoquer un ensemble de services web en parallèle. L'invocation d'un service web donnée peut être effectué grâce à une fonction invokWebService().

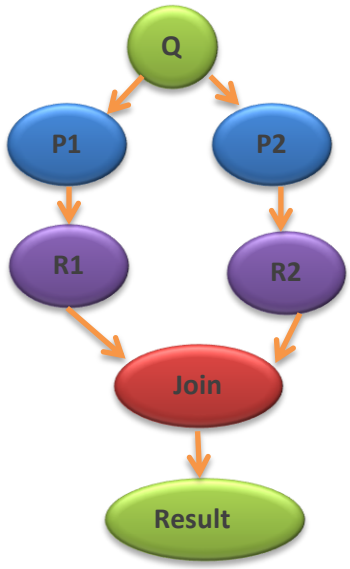

Fig. IV. 2. Exécution de SW en parallèle

Une fonction appelée « *run_In_Parallel* ()» permet de créer et d'exécuter des processus séparés. Chaque processus invoque un service web particulier selon la requête paramétrée de l'utilisateur.

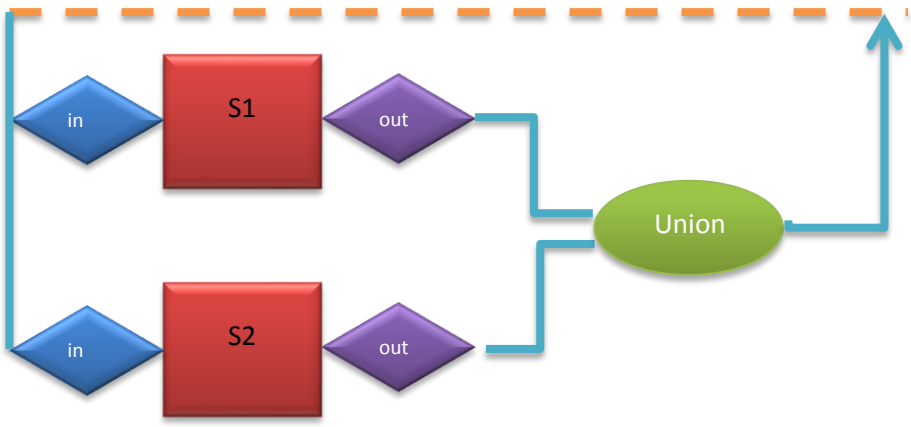

Fig. IV. 3. Exemple d'exécution de SW en parallèle

IV. 3. Fonctionnement détaillé de l'interface graphique :

Nous avons donné la possibilité à L'utilisateur de choisir la méthode de l'appel du DaaS, il a le choix entre une invocation parallèle, séquentielle comme il peut effectuer une comparaison entre les deux méthodes.

1) Invocation parallèle de DaaS

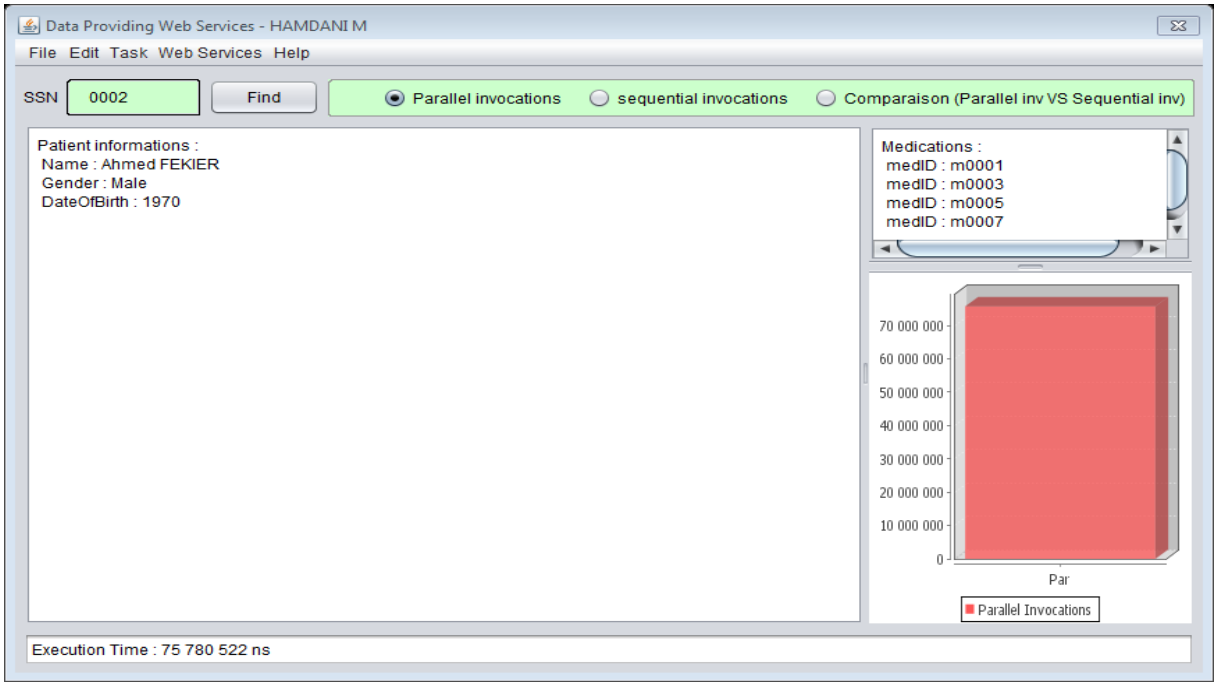

Fig. IV. 4. Invocation parallèle de DaaS

2) Invocation séquentielle de DaaS

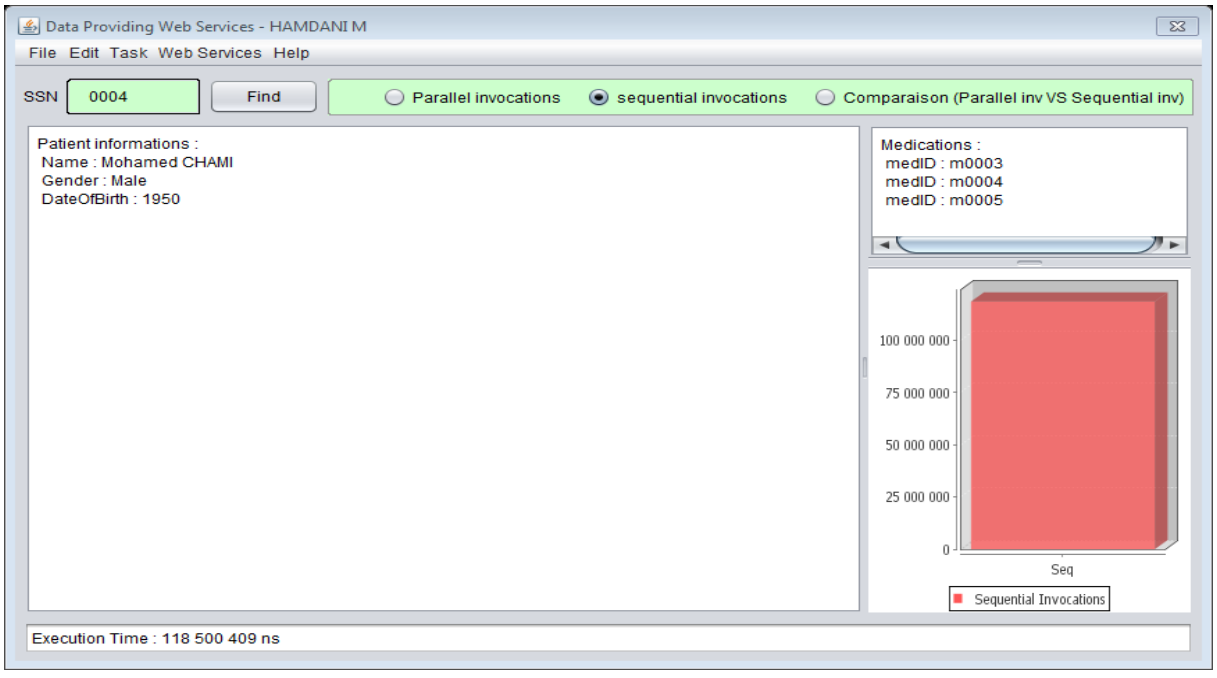

Fig. IV. 5. Invocation séquentielle de DaaS

3) Comparaison entre les deux méthodes

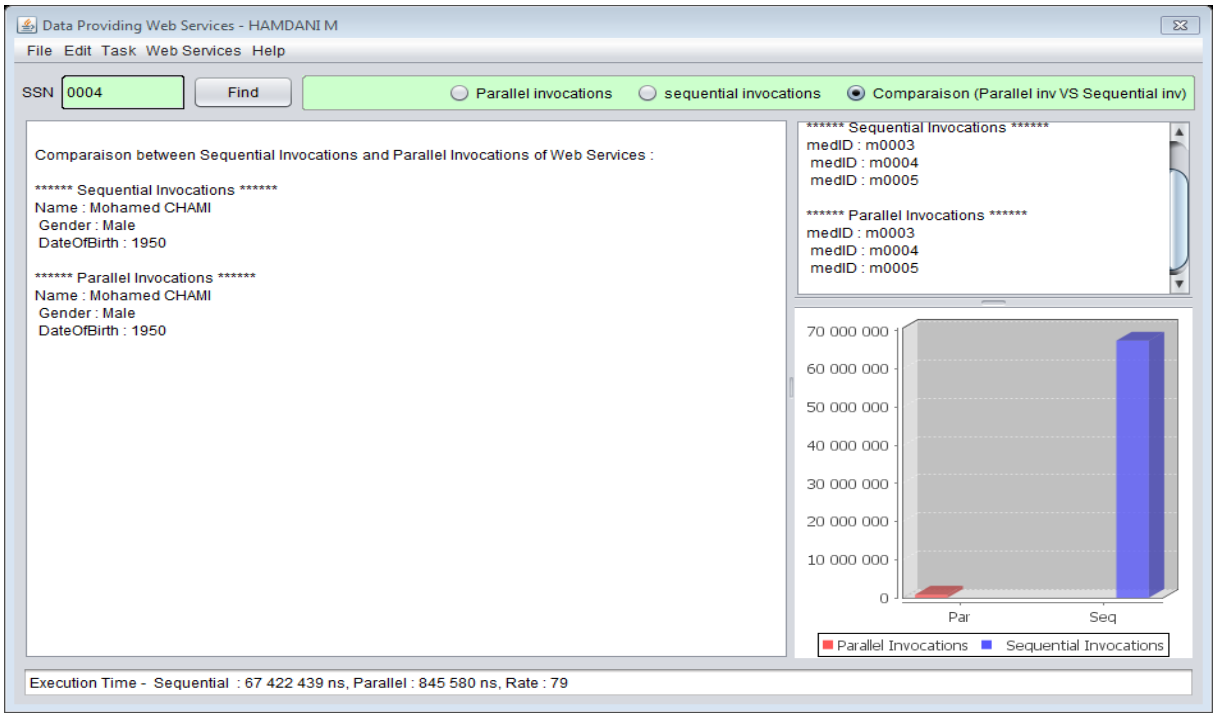

Fig. IV. 6. Comparaison entre les deux méthodes (1)

4) Comparaison entre les deux méthodes : effectuer plusieurs tests

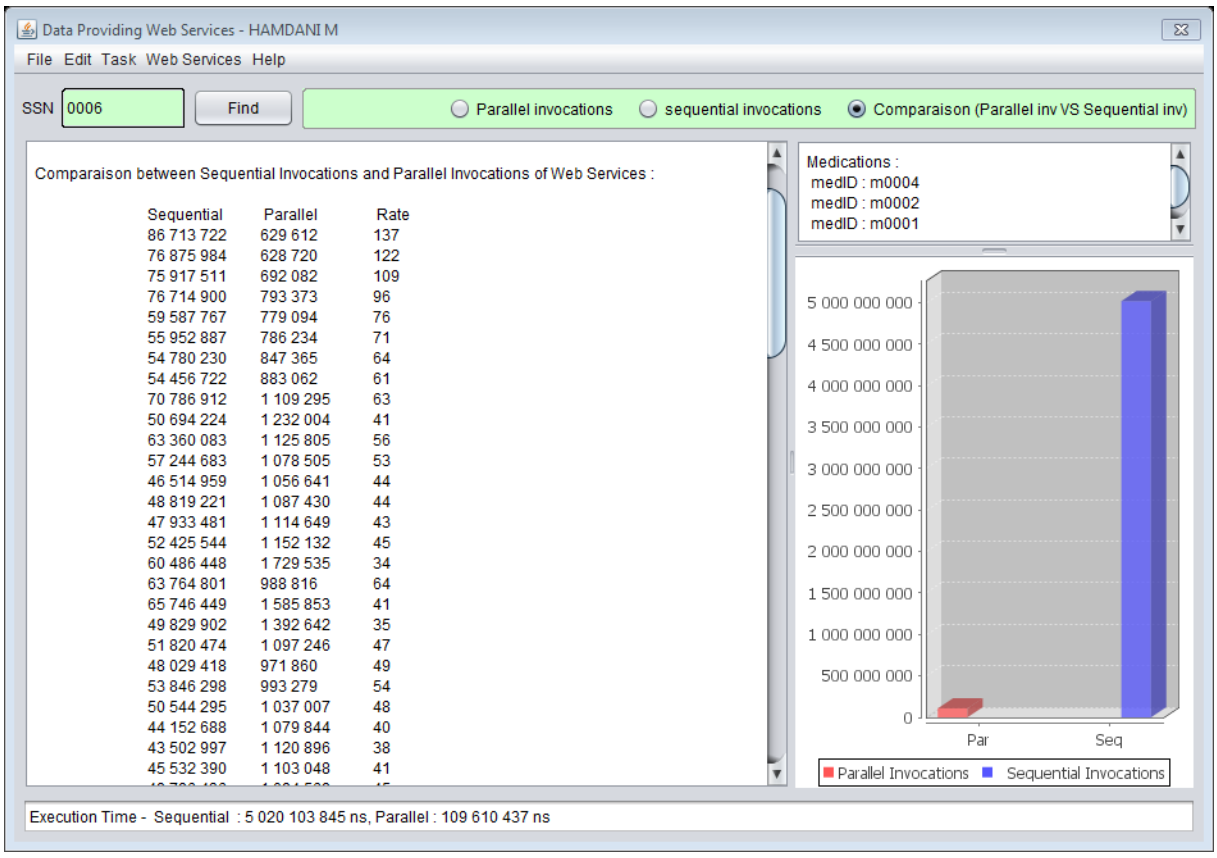

Fig. IV. 7. Comparaison entre les deux méthodes (2)

5) L'utilisateur peut également voir la liste des différents services web

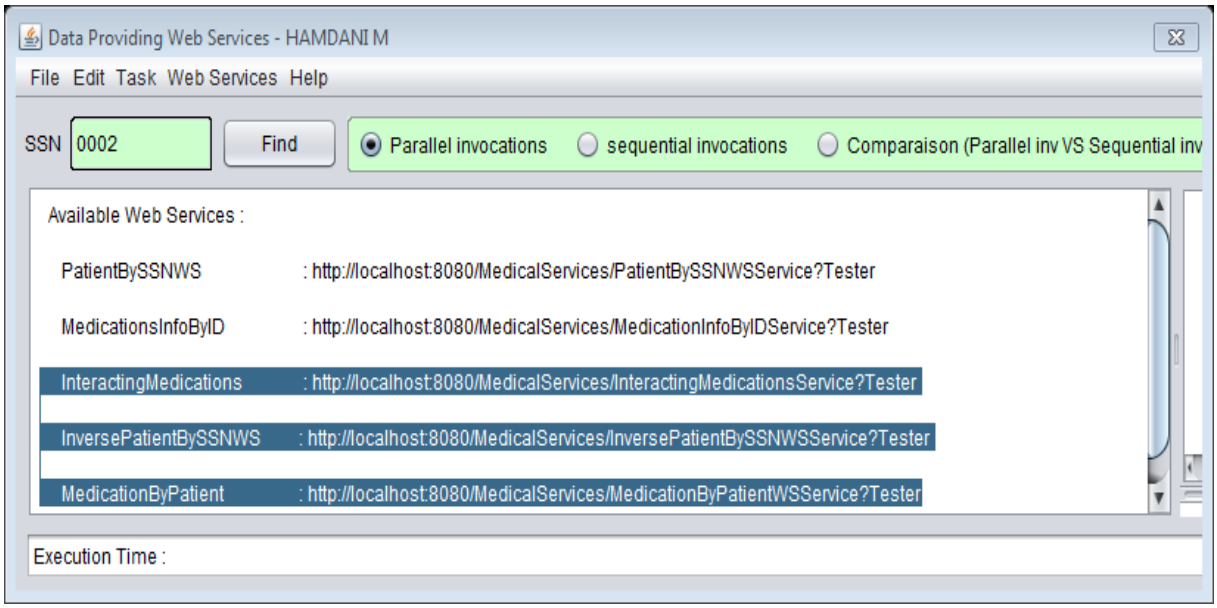

Fig. IV. 8. 8. Liste des services web.

6) Il peut aussi consulter la liste des patients, des médicaments ou les interactions entre médicaments.

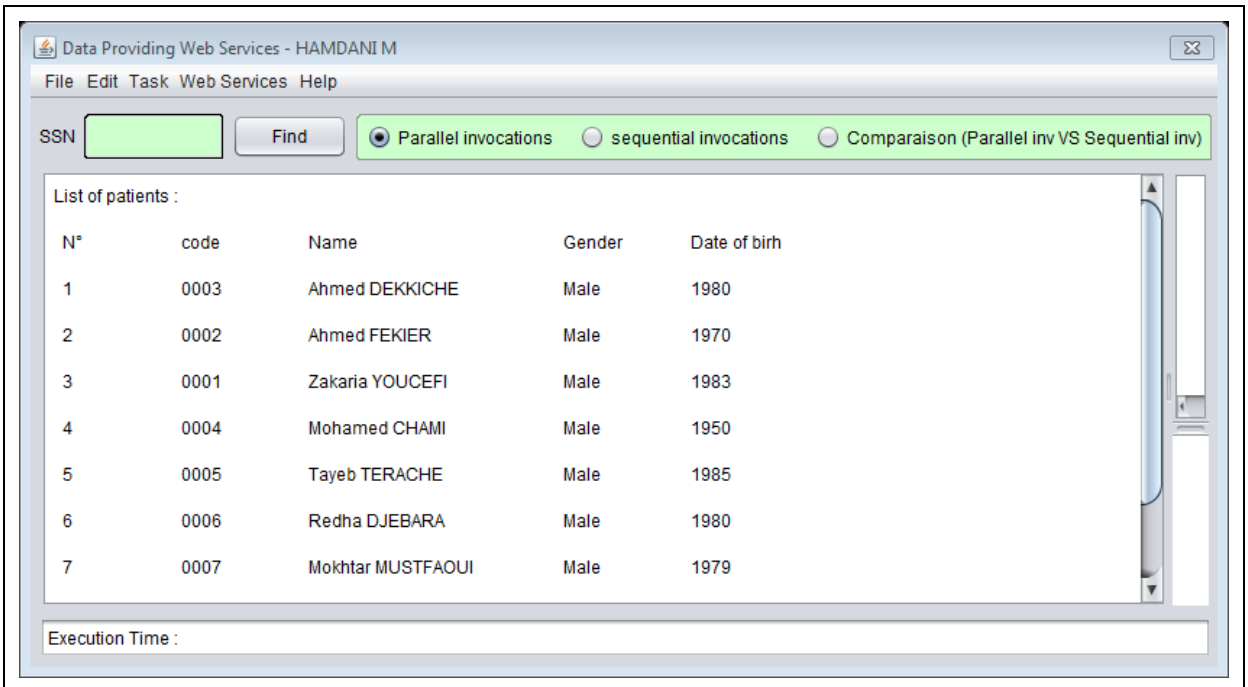

Fig. IV. 8. liste des patients.

IV. 4. Analyse de performance :

Dans cette section nous allons présenter une analyse de performance de notre approche en étudiant la comparaison entre les deux méthodes d'invocation de DaaS : l'invocation parallèle et l'invocation séquentielle.

Le tableau suivant contient un ensemble de tests effectués dont nous avons calculé le temps d'exécution pour chaque méthode de composition. Le temps execution a été calculé comme suit :

```
long t  = 0 ;
long start ;
start = System.nanoTime();
instructions ……
t  = System.nanoTime() – start ;
```

Fig. IV. 9. Temps d'exécution.

Sequential inv (ns)	Parallel inv (ns)	Rapport de Sequential / Parallel
90 195 379	3 269 471	27
71 012 724	1 203 908	58
67 314 879	1 255 224	53
63 165 903	1 862 086	33
72 951 560	1 187 844	61
64 458 610	1 126 265	57
100 226 011	1 262 364	79
69 031 496	1 182 489	58
64 949 454	1 194 537	54
62 230 622	1 148 576	54
61 920 496	1 398 014	44
75 220 155	1 399 800	53
80 049 174	1 141 883	70
59 266 365	1 077 180	55
55 444 470	948 669	58
69 712 879	963 840	72
57 628 283	894 675	64
67 013 679	919 218	72
70 613 355	954 470	73
66 635 729	981 689	67

Tableau IV. 1. Comparaison entre les deux méthodes (parallèle vs séq.)

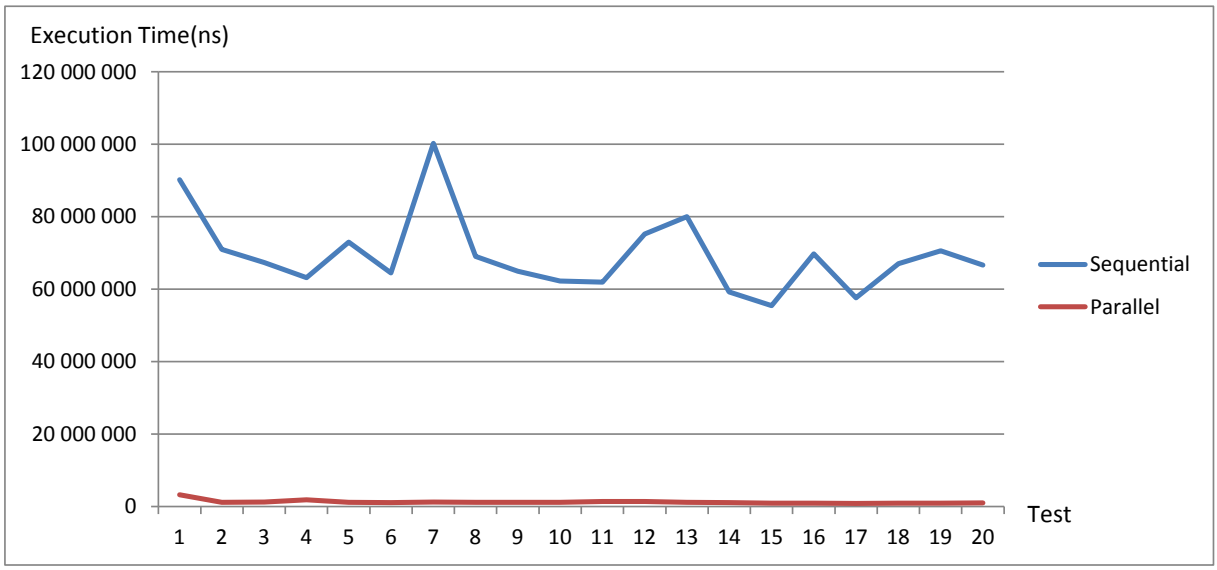

Fig. IV. 10. Comparaison entre les deux méthodes (parallèle vs séq.)

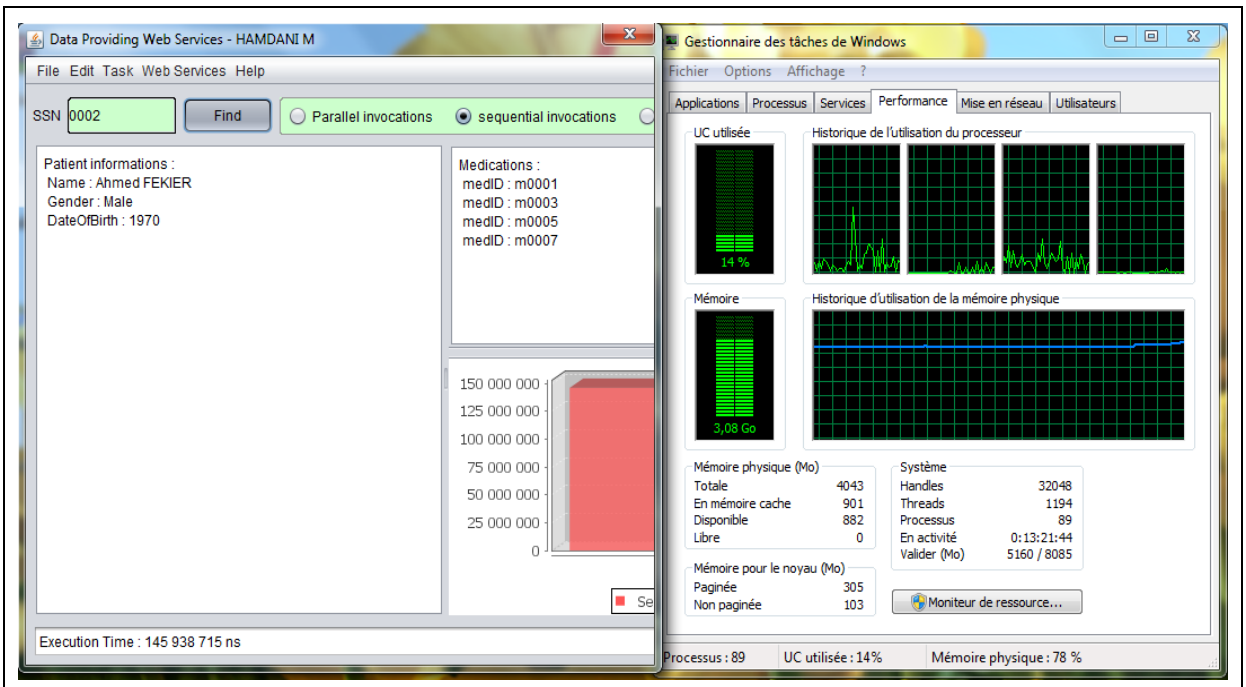

Fig. IV. 11. Analyse de la performance (Méthode séquentielle)

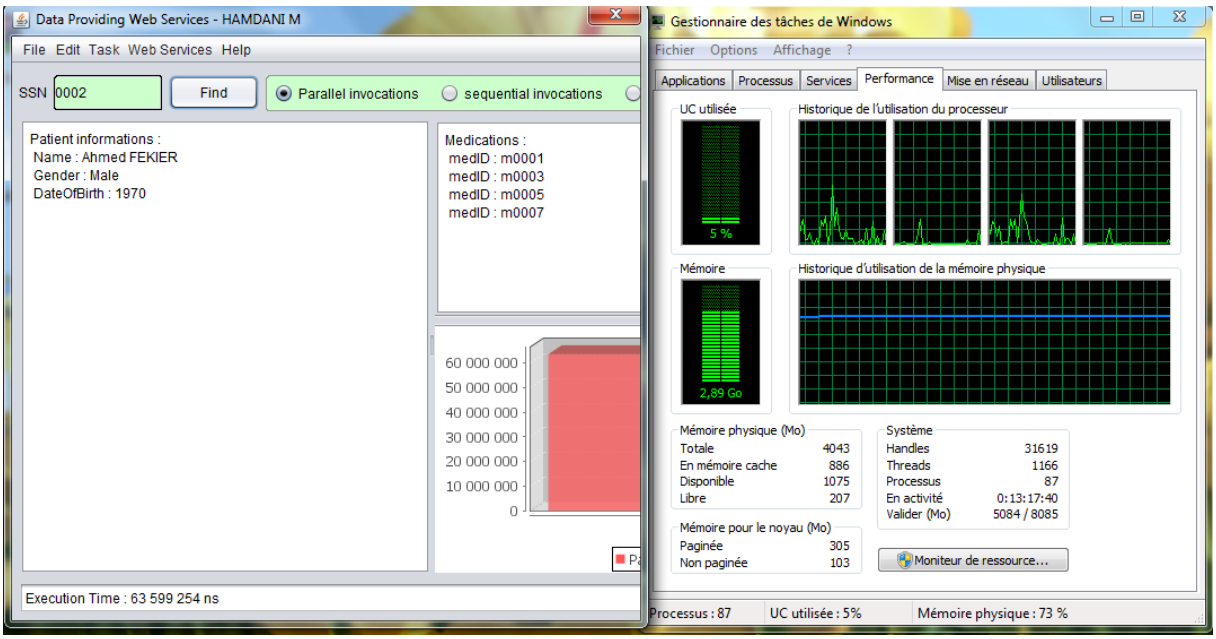

Fig. IV. 12. Analyse de la performance (Méthode Parallèle)

IV. 5. Analyse des résultats obtenus :

L'analyse du tableau permet de retenir que l'invocation d'un service web DaaS en utilisant la méthode classique (i .e, une suite d'appels successifs aux services web d'accès aux données) est plus lente que celle où ces services sont appelés en parallèle. Le temps d'exécution d'un appel de service web dépend aussi du processeur, de sa vitesse et de son occupation.

Les résultats obtenus montrent que l'introduction de la notion du parallélisme permet d'optimiser le processus de la composition des DaaS et améliore considérablement le temps d'exécution de la recherche d'informations via les services web.

IV. 6. Conclusion

Ce chapitre représente un cadre pratique pour l'architecture globale de notre proposition. La composition de services permet de répondre aux besoins de plus en plus complexes des utilisateurs, par la combinaison de plusieurs services Web au sein d'un même processus métier.

Nous avons présenté une approche permettant d'optimiser la recherché d'information et la composition de services DaaS.

Notre approche consiste à paralléliser les appels des services web par des processus de requêtes séparés, chacun gère une sous requête paramétrée pour les différents tuples. Chaque processus invoque un service web DaaS. Pour une requête donnée, les processus de requête sont exécutés en parallèle.

.

V. Conclusion Générale

La technologie des services web est devenue une technologie de référence pour le développement des applications distribuées sur le web, 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'un même processus métier.

Les travaux liés à l'interopérabilité dans le cadre de la composition de services Web sont particulièrement orientés vers le niveau sémantique pour permettre aux machines d'interpréter les données traitées et de saisir leur signification de manière automatique. Les méthodes de composition, traditionnelles, ne permettent pas de prendre en compte la relation sémantique entre les entrées/sorties d'un service web d'accès aux données, et en conséquence, elles ne sont pas adaptées pour composer les services Web DaaS.

Nous avons adopté l'architecture qui modélise les services Web d'accès aux données comme des vues sur des ontologies de domaine. Cela permet de représenter la sémantique d'un service d'une manière déclarative en se basant sur des concepts et des relations dont les sémantiques sont formellement définies dans l'ontologie de domaine. Ensuite, utilise les techniques de réécriture des requêtes pour sélectionner et composer automatiquement les services pour répondre aux requêtes des utilisateurs [BAR10].

Notre proposition enrichit ce système par l'introduction de la notion de parallélisme où plusieurs services web d'accès aux données peuvent être invoqués en parallèle pour optimiser la composition des DaaS.

Les perspectives identifiées concernent l'introduction du parallélisme dans les appels des services web dépendants (le résultat du premier service web est l'entrée du deuxième service) et l'inclusion de la qualité des données DQ et les aspects de qualité de service dans la composition.

BIBLIOGRAPHIE

[ABI08] ABITEBOUL S., BENJELLOUN O., MILO T., "The Active XML project: an overvie**w ",** VLDB Journal, Volume 17 , Issue 5, Pages: 1019 – 1040, ISSN:1066-8888, 2008.

[ABH04] ABHIJIT A. Patil, SWAPNA A. OUNDHAKAR, AMIT P. SHETH, and KUNAL Verma. Meteor-s web service annotation framework. In Proceedings of the 13th conference on World Wide Web, ACM Press, pages 553–562., 2004.

[AGG04] AGGARWAL Rohit , VERMA Kunal, MILLER John, MILNOR Willie ,Dynamic Web Service Composition in METEOR-S , Technical Report, LSDIS Lab, Computer Science Dept., UGA, http://lsdis.cs.uga.edu/proj/meteor/mwscf/mwscf.html, May 2004.

[**AKK05**] AKKIRAJU Rama , FARRELL Joel, MILLER John, NAGARAJAN Meenakshi, "Web Service Semantics–WSDL-S, [On Line] http://www.w3.org/Submission/WSDL-S/, 2005.

[AKL07] AKLOUF Youcef, Intégration du modèle d'ontologies PLIB et des Services Web dans les échanges inter-entreprises, Thèse Présentée pour l'obtention du diplôme de Doctorat en informatique, Université des Sciences et de la Technologie Houari Boumediene, 2007.

[**AUS04**] AUSTIN D., BARBIR A., PETERS E., and ROSS-TALBOT S. Web Services Choreography Requirements W3C Working Draft. Technical report, (W3C), 2004.

[BAR06] BARREIRO CLARO Daniela, SPOC - Un canevas pour la composition automatique de services web dédiés `a la réalisation de devis, Thèse de doctorat en Informatique, Université d'angers, 6 Octobre 2006.

[**BAR08**] BARHAMGI Mahmoud, Djamal BENSLIMANE, Composing Data-Providing Web Services in P2P-Based Collaboration Environments, 19th International Conference on Advanced Information Systems Engineering (CAiSE'07), Springer ed. Trondheim, Norway. pp. 513-545. Springer. ISBN 978-3-540-72987, 2007.

[BAR09] BARHAMGI M., BENSLIMANE D., Composing Data-Providing Web Services, VLDB09 (PhD Workshop), Lyon, France. 2009.

[BAR10] BARHAMGI M., BENSLIMANE D. and MEDJAHED B. "A Query Rewriting Approach for Web Service Composition", IEEE Transactions on Services Computing (TSC), Feb. 2010.

[BEL04] BELLATRECHE Ladjel, PIERRA Guy, XUAN Dung Nguyen and HONDJACK Dehainsala, Intégration de sources de données autonomes par articulation a priori d'ontologies, XXII-ème congrès INFORSID, edited by Inforsid, Biarritz, France, IUT de Bayonne, pp. 283-298 ,Mai, 2004

[BEN00] BENEVENTANO D., BERGAMASCHI S., CASTANO S., CORNI A., GUIDETTI R., MALVEZZI G., MELCHIORI M., and VINCINI M.. Information integration: The MOMIS project demonstration. In The VLDB Journal, pages 611–614, 2000.

[BEN04] BENJELLOUN, O , Active XML: A data centric perspective on Web services, PhD thesis, Universtity of Paris XI, 2004

[BOU08] QI YU , XUMIN liu , BOUGUETTAYA Athman, MEDJAHED Brahim, Deploying and managing Web services: issues, solutions, and directions, The International Journal on Very Large Data Bases, Pages: 537 - 572 , Springer-Verlag New York, 2008.

[**BRU05**] BRUIJN J., LAUSEN H., KRUMMENACHER R., POLLERES A., KIFER M. and FENSEL D., "Deliverable D16.1v0.2, The Web Service Modeling Language WSML". Final Draft, WSMO project, [On Line] http://www.wsmo.org/ TR/d16/d16.1/v0.2/20050320/, 2005.

[**BUS05**] BUSSLER Christoph, CIMPIAN Emilia, FENSEL Dieter, Web Service Execution Environment (WSMX), http://www.w3.org/Submission/WSMX/ , 3 June 2005.

[CAR02] CARDOSO J. and SHETH A.. Semantic e-workflow composition. Technical Report 02-004, LSDIS Lab., Computer Science Department, Univesity of Georgia, Athens, USA, 2002.

[CAR04] CARDOSO J., SHETH A., "Semantic Web services and web processes composition". Springer-Verlag, New York Inc, 2004.

[**CAR06**] Carey, M.: Data delivery in a service-oriented world: the BEA aquaLogic data services platform. In: Proc. The 2006 ACM SIGMOD international conference on Management of data (2006).

[**CAS00**] CASATI F., ILNICKI S., and JIN L., "Adaptive and dynamic service composition in EFlow". In Proceedings of 12th International Conference on Advanced Information Systems Engineering (CAiSE), Stockholm, Sweden, Springer Verlag, 2000.

[CHA01] CHAKRABORTY D., PERICH F., AVANCHA S., JOSHI A., "DReggie: Semantic Service Discovery for M-Commerce Applications". In Workshop on Reliable and Secure Applications in Mobile Environment, 20th Symposium on Reliable Distributed Systems, pp. 28–31, 2001.

[CHA07] CHARIF Yasmine, Chorégraphie dynamique de services basée sur la coordination d'agents introspectifs, Thèse présentée pour obtenir le titre de Docteur en Informatique, Université Pierre et Marie Curie - Paris VI, 10 décembre 2007.

[CHA97] CHAWATHE S., GARCIA-MOLINA H., HAMMER J., IRELAND K., PAPAKONSTANTINOU Y., ULLMAN J. D., and J. WIDOM. The TSIMMIS Project: Integration of heterogeneous information sources. Proceedings of the 10th Meeting of the Information Processing Society of Japan, pages 7–18, Mars 1994.

[CHR00] CHRISTOPHIDES V., CLUET S., and SIMÉON Jérôme. On Wrapping Query Languages and Efficent XML Integration. In SIGMOD, Dallas, Texas, Mai, 2000.

[CHR01] CHRISTENSEN E., CURBERA F., MEREDITH G. and WEERAWARANA S.. Web Service Description Language 1.1. W3C Note., http ://www.w3.org/TR/wsdl., 2001.

[COM08] Composite Software, SOA Data Services Solutions, technical report http://compositesoftware.com/solutions/soa.html, (2008),

[CRA01] CRAIG A. KNOBLOCK et al., "The Ariadne Approach to Web-Based Information Integration," Int. J. Cooperative Inf. Syst., vol. 10, no. 1-2, pp. 145-169, 2001.

[CRU03] CRUBEZY M., FENSEL D., BENJAMINS R., WIELINGA B., MOTTA E., MUSEN M. and OMELAYENKO B., «UPML: The language and tool support For making the semantic Web alive», Technical report, MIT, 2003.

[**FEN02**] FENSEL D., and BUSSLER C., "The Web Service Modeling Framework WSMF". In Electronic Commerce Research and Applications, 1(2), Elsevier, Scinces B. V., 2002.

[**FER00**] Fernandez M.F, FLORESCU D., LEVY A., and SUCIU D.. Declarative Specification of Web Sites with Strudel. VLDB journal, 9(1) p. 38-55, 2000.

[GAN08] GANDON Fabien, Graphes RDF et leur Manipulation pour la Gestion de Connaissances, Mémoire d'Habilitation à Diriger les Recherches, Université de Nice – Sophia Antipolis, 5 novembre 2008.

[GEN97] GENESERETH M. R., KELLER A. M., and DUSCHKA O. M.. INFOMASTER : an information integration system. In ACM SIGMOD International Conference on Management of Data, pages 539–542, 1997.

[GIR02] GIRALDO G. & REYNAUD Ch.. Construction semi-automatique d'ontologies à partir de DTDs relatifs à un même domaine. 13èmes journées francophones d'Ingénierie des Connaissances. Rouen. 2002

[GIR05] Gloria-Lucia GIRALDO GÓMEZ, Construction automatisée de l'ontologie de systèmes médiateurs - Application à des systèmes intégrant des services standards accessibles via le Web, THÈSE présentée pour obtenir le grade de Docteur en Sciences de l'Université Paris XI Orsay, Spécialité : Informatique, 2005.

[GOH97] GOH, C. H. Representing and reasoning about semantic conflicts in heterogeneous nformation systems. PhD thesis, MITSloan School of Management. (1997).

[GOH99] SIEGEL M. D.. Context interchange : New features and formalisms for the intelligent integration of information. ACM Transactions on Information Systems, 17(3) :270–293, 1999.

[**HAC05**] HACID M.-S. and REYNAUD C.. L'intégration de sources de données. La revue I3 : Information - Interaction - Intelligence, Vol. 5 n˚1, 2005.

[**HAY04**] HAYES P., HORROCKS I., and PATEL-SCHNEIDER P.F.. OWL web ontology language semantics and abstract syntax. Technical report, W3C, http://www.w3.org/TR/owl-semantics/, 2004.

[**HUA06**] CHEN Huajun, WU Zhaohui, WANG Heng, and MAO Yuxin. RDF/RDFS-based relational database integration. 2006.

[**HUB03**] HUBERT Kadima, Valérie MONFORT, Les web services, Techniques, démarches et outils, DUNOD, ISBN 2 10 006558 0, 2003.

[INC09] X. Inc, Xcalia Data Access Services (2009), http://www.xcalia.com/products/xcalia-xdasdata-access-service-SDO-DAS-data-integration-through-web-services.jsp

[**IZZ06**] IZZA Saïd, Intégration des systèmes d'information industriels - Une approche flexible basée sur les services sémantiques, Thèse présentée pour obtenir le grade de Docteur en Informatique, École Nationale Supérieure des Mines de Saint-Etienne, 20 novembre 2006.

[**JAR97**] JARKE M. and Vassiliou Y.. Data warehouse quality design : A review of the DWQ project. In Invited Paper, 2nd Conference on Information Quality. Massachusetts Institute of Technology, Cambridge, 1997.

[**KHO09**] KHOULOUD Boukadi1, CHIRINE Ghedira , ZAKARIA Maamar , BENSLIMANE Djamal, Transactions on Large-Scale Data- and Knowledge-Centered Systems I, Volume 5740/2009 Springer, ISBN 978-3-642-03721-4, Pages 91-115, aug 2009

[LAB97] LABIO W. J., ZHUGE Y., WIENER J. L., GUPTA H., GARCIA-MOLINA H., and WIDOM.J. The WHIPS prototype for data warehouse creation and maintenance. In Proceedings of SIGMOD, pages 557–559, 1997.

[**LEV95**] ALON Y. LEVY, ALBERTO O. MENDELZON, SAGIV Yehoshua and SRIVASTAVA Divesh. Answering queries using views. In Proceedings of the Fourteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, San Jose, California, pages 95-104, 1995.

[LEV98] LEVY A., ROUSSET, M.C. "Combining Horn Rules and Description Logics in CARIN", In Artificial Intelligence Journal, September 1998, Vol. 14, p. 165-209.

[**LIN04**] LIN Chenxi, YU Yong, YANG Yin, ZHANG Lei, ZHOU Jian**,** Towards Composing Information Providing Services, IEEE International Conference on Services Computing, Pages: 117 – 122, ISBN:0-7695-2225-4 2004.

[**LUM06**] N. Lumineau, A. Doucet, and S. Gan-¸carski. Thematic schema building for mediation-based peer-to-peer architecture. Electronic Notes in Theoretical Computer Science, 150(2) :21–36, 2006.

[**MAE03**] Libero MAESANO, Christian BERNARD, Services Web avec J2EE et .NET Conception et implémentations, ISBN : 2-212-11067-7, Eyrolles, 2003.

[**MAH08**] MAHBOUBI Hadj, Optimisation de la performance des entrepôts de données XML par fragmentation et répartition, Thèse Présentée pour l'obtention du diplôme de Doctorat en informatique, Université Lumière Lyon 2, 2008.

[**MAR08**] MARTIN David, BURSTEIN Mark, HOBBS Jerry, LASSILA Ora,, "OWL-S: Semantic Markup for Web Services". [On Line] http://www.ai.sri.com/daml/services/owl-s/1.2/overview/, 2008.

[**MCD02**] MCDERMOTT D., Estimated-regression planning for interactions with web services. In Proceedings of the 6th International Conference on AI Planning and Scheduling (Toulouse, France, 2002), AAAI Press.

[MCI02] MCILRAITH S. and SON S . Adapting Golog for Composition of Semantic Web Services. In Proc. KR'02, 2002.

[**MED03**] MEDJAHED, B., BOUGUETTAYA, A., and ELMAGARMID, A. Semantic web enabled composition of web services. The VLDB Journal 12, 4 (November 2003), 333–351.

[**MEN96**] MENA E., KASHYAP V., SHETH A. P., and ILLARRAMENDI A.. OBSERVER : An approach for query processing in global information systems based on interoperation across pre-existing ontologies. In Conference on Cooperative Information Systems, pages 14–25, 1996.

[**MET05**] METEOR-S, METEOR-S project deliverables. [On Line] http://lsdis.cs.uga.edu/Projects/METEOR-S/, 2005.

[**MIC07**] Microsoft, ADO.NET Data Services (also known as Project Astoria) (2007), http://astoria.mslivelabs.com/

[**MOR04**] Morris E., Levine L., Meyers C., Place P., and Plakosh D., "System of Systems Interoperability - Final Report". Techincal Report, CMU/SEI-2004-TR-004, ESC-TR-2004-004, 2004.

[**MOT03**] Motta Enrico, Domingue John, Cabral Liliana, and Gaspari Mauro, IRS-II: A Framework and Infrastructure for Semantic Web Services, In Proceedings of the 2nd International Semantic Web Conference (ISWC), vol. 2870, **ISSN** 0302-9743, pp. 306-318 2003, Springer, 2003.

[**MRI07**] Michaêl MRISSA, Médiation Sémantique Orientée Contexte pour la Composition de Services Web, Thèse présentée pour l'obtention du diplôme de Docteur en Informatique, Université Claude Bernard Lyon I, 15 Novembre 2007.

[NAR02] NARAYANAN S. and MCILRAITH S., Simulation, Verification and Automated Composition of Web Services. In Proc. WWW'02, 2002.

[**OAS09**] OASIS, Reference Architecture Foundation for Service Oriented Architecture, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=soa-rm, 14 October 2009.

[**OUK06]** Aris M. OUKSEL, Oana JURCA, Ivana PODNAR, and Karl ABERER. Effcient probabilistic sub-sumption checking for content-based publish/subscribe systems. In Middleware 2006, the 7th International Middleware Conference, Melbourne, Australia, Proceedings, pages 121-140, 2006,

[**OUK08**] OUKSEL Aris M , BARHAMGI Mahmoud and BENSLIMANE Djamal**,** Composing and optimizing data providing web services, 17th international conference on World Wide Web, Pages 1141-1142, ACM 2008.

[OUZ04] Mourad Ouzzani and Athman Bouguettaya, "Efficient Access to Web Services," IEEE Internet Computing, vol. 8, no. 2, pp. 34-44, 2004.

[**PAR03**] Parsia, B., Wu D., Sirin, E., Hendler, J., and Nau, D. Automating DAML-S web services composition using SHOP2. In ISWC'03 (2003).

[**PON04**] Julien PONGE, Compatibilité et substitution dynamique des web services, Prise en compte de contraintes temporelles et de propriétés d'activation, 2004.

[**PRU08**] Prud'hommeaux E. and Seaborne A., SPARQL Query Language for RDF, W3C Recommendation, http://www.w3.org/TR/rdf-sparql-query/, January 2008.

[RAO04] Rao J. and X. Su. A survey of automated web service composition methods. In Semantic Web Services and Web Process Composition, First International Workshop, SWSWPC 2004, volume 3387 of Lecture Notes in Computer Science, pages 43–54, San Diego, USA, Springer 2004.

[**RED07**] REDHAT.: MetaMatrix Enterprise Data Services Platform (2007), http://www.redhat.com/jboss/platforms/dataservices/

[**REI01**] R. Reiter. Knowledge in Action : Logical Foundations for Specifying and Implementing Dynamical Systems. The MIT Press, 2001.

[REY03] C. Reynaud and G. Giraldo. An application of the mediator approach to services over the web. Special track "Data Integration in Engineering, Concurrent Engineering (CE'2003), pages 209–216, July 2003.

[SAB10] Manivasakan Sabesan, Tore Risch, and Feng Luan, Automated Web Service Query Service, International Journal of Web and Grid Services (IJWGS), Inderscience, Volume 6, Number 4, 2010.

[**SHA07**] O. Shafiq, M. Moran, E. Cimpian, A. Mocan, M. Zaremba, D. Fensel, Investigating Semantic Web Service Execution Environments: A comparison between WSMX and OWL-S tools, 2nd International Conference on Internet and Web Applications and Services (ICIW2007), IEEE, page 31, 2007.

[**SHE03**] Sheshagiri, M., Desjardins, M., and Finin, T. A planner for composing services described in DAML-S. In Proceedings of the AAMAS Workshop on Web Services and Agent-based Engineering , (Jun. 2003).

[**SIR02**] Sirin, E., Hendler, J., and Parsia, B. Semi-automatic composition of web services using semantic descriptions. In Proceedings Web Services : Modeling, Architecture and Infrastructure. Workshop in Conjunction with ICEIS2003 (Angers,France,), ICEIS Press, pp. 17–24. 2002

[**SIV04**] Sivashanmugam K., Verma K., and Sheth A. P.. Discovery of web services in a federated registry environment. In ICWS, IEEE, pages 270–278, 2004.

[UTK06] Utkarsh Srivastava, Kamesh Munagala, Jennifer Widom, and Rajeev Motwani, "Query Optimization over Web Services," in VLDB, Seoul, Korea, 2006, pp. 355-366.

[**VAC08**] VACULIN Roman,, CHEN NERUDA Huajun, Roman, SYCARA Katia, "Modeling and Discovery of Data Providing Services," ICWS (International Conference on Web Services), IEEE, pp.54-61, , 2008.

[**VER04**] Verma K., Sivashanmugam K., Sheth A., Patil A., Oundhakar S., and Miller J., METEOR-S WSDI: A Scalable Infrastructure of Registries for Semantic Publication and Discovery of Web Service, Journal of Information Technology and Management, 2004; http://lsdis.cs.uga.edu/lib/download/MWSDI.pdf

[**VIS99**] P. R. S. Visser, M. Beer, T. Bench-Capon, B. M. Diaz, and M. J. R. Shave. Resolving ontological heterogeneity in the kraft project. 10th International Conference on Database and Expert Systems Applications (DEXA'99), pages 668–677, September 1999.

[**WAC01**] H. Wache, T. V¨ogele, U. Visser, H. Stuckenschmidt, G. Schuster, H. Neumann, and S. H¨ubner. Ontology-based integration of information - a survey of existing approaches. Proceedings of the International Workshop on Ontologies and Information Sharing, pages 108–117, August 2001.

[**W3C04a**] W3C, RDF Semantics**,** W3C Recommandation, http://www.w3.org/TR/rdf-mt/ , 10 February 2004.

[**W3C04b**] W3C,. W3C working group note,. [On Line] http://www.w3.org/TR/ws-gloss/. February 2004.

[W3C05] W3C Member Submission, WSMF, "Semantic Web Services Framework (SWSF) Overview", [On Line] http://www.w3.org/Submission/SWSF/, 2005.

[W3C08] W3C, Extensible Markup Language (XML) 1.0 (Fifth Edition), W3C Recommendation, [On Line] http://www.w3.org/TR/xml/.2, 6 November 2008.

[WEN08] Wenfeng ZHAO, Xiangwu MENG, Junliang CHEN, Chuanchang LIU , Integrating Information-Providing Web Services into the Data Integration System, IEEE International Conference on Web Services, Pages 801-802, ISBN:978-0-7695-3310-0, 2008.

[**WIL06**] Williams, K., Daniel, B.: SOA Web Services - Data Access Service. Java Developer's Journal (2006).

[ZHA05] Huajun CHEN, ZHAOHUI Wu , Yuxin MAO, Rewriting Queries using Views for RDF-based Relational Data Integration, Proceedings of the 17th IEEE International Conference on Tools with Artificial Intelligence, Pages: 260 – 264, ISBN: 1082-3409, 0-7695-2488-5, 2005.