N° d'ordre: 06/2013-D/INF

République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université des Sciences et de la Technologie « Houari Boumediène »

Faculté d'Electronique et d'Informatique

THESE

Présentée pour l'obtention du grade de **Docteur**

En : **INFORMATIQUE**

Spécialité : **Intelligence Artificielle et Base de données Avancées**

Par : **Mr. AMAROUCHE Idir Amine**

Sujet :

APPROCHE ORIENTÉE SERVICE POUR L'INTÉGRATION DES DONNÉES : CAS «E-SANTÉ»

Soutenue publiquement, le **12 Octobre 2013**, devant le jury composé de :

Mr. M. AHMED-NACER,	Professeur à l'USTHB,	Président
Mme. Z. ALIMAZIGHI,	Professeur à l'USTHB,	Directeur de thèse
Mr. D. BENSLIMANE,	Professeur, UCB Lyon 01- France,	Co-Directeur de Thèse
Mr. M. MIMOUN,	Professeur, UDL- Sidi Bel Abbes,	Examinateur
Mr. N. ZAROUR,	Professeur, UMC - Constantine,	Examinateur
Mr. K. BOUKHALFA,	Maitre de Conférences à l'USTHB,	Examinateur

A mes parents,
A ma femme,
A mes enfants Haroun, Farouk et Mehdi

Remerciements

Je tiens à remercier le Professeur ALIMAZIGHI Zaia, non seulement pour avoir encadré cette thèse, mais aussi de l'entière confiance qu'elle m'a témoignée pendant tout le cursus doctoral.

Je suis particulièrement reconnaissant envers le Professeur Djamal BENSLIMANE pour son entière disponibilité, pour ses conseils précieux et pertinents, pour son aide inestimable lesquels ont contribué à l'aboutissement de la présente thèse.

Mes remerciements vont à Mr le Directeur Central de la Sante Militaire qui a soutenu le projet de thèse par les moyens qui lui sied.

Aussi, mes remerciements vont au directeur adjoint du LIRIS, Mohand-Said HACID et notamment mes collègues de l'équipe SOC du LIRIS, Michael Mrissa, Mahmoud Brahamdji et Karim Benouaret pour avoir collaboré à l'effet de promouvoir les idées que je défendais et que je défends encore.

Je remercie également le President du Jury, le professeur Mohamed AHMED-NACER pour avoir accepté d'évaluer et de juger mon travail.

Je remercie Mr.le professeur MIMOUN Malki, le Professeur ZAROUR Nacereddine et le Maitre de conférences Kamel BOUKHALFA pour avoir accepté d'être membres du jury en tant qu'examinateurs.

Tous les autres membres du LSI/FEI/USTHB pour tous les conseils qui m'ont prodigué afin d'appréhender au mieux un projet de thèse.

Mes remerciements vont aussi à tout le personnel de la SousDirection informatique de la Direction Centrale de la Santé Militaire pour avoir contribué, de près ou de loin, dans le cadre de cette entreprise.

Enfin, je remercie ma chère famille et, surtout, mon épouse, car sans leur soutien inconditionnel et permanent, ce projet n'aurait pu ni voir le jour, ni évoluer vers cet accomplissement scientifique.

Je leur dédie cette thèse.

Résumé

En raison du développement important des systèmes d'information hospitaliers, et de l'aspect collégial et distribué de la pratique médicale, il existe aujourd'hui un fort besoin d'infrastructures logicielles qui intègrent de manière uniforme les collections distribuées et hétérogènes des données du patient. Dans le contexte e-santé, l'adoption des dossiers de santé électroniques (Electronic Health Record) et des services Web, comme infrastructure logicielle, sont devenus des choix prédominants pour l'intégration desdites données.

En outre, il est communément admis que les services Web se répartissent en deux catégories en fonction de leurs fonctionnalités, les services qui provoquent un changement sur l'état du monde (SaaS) et les services fournisseurs de données (DaaS). Plusieurs travaux ont été menés à l'effet d'automatiser la découverte et la composition des services Web notamment les services DaaS. À ce titre, la composition de services DaaS basée sur la réécriture de requêtes, proposée par la communié des bases de données, a été prônée par de nombreux travaux. À cet effet, ces travaux en question proposent un modèle de description de services DaaS basé sur les Vues RDF paramétrées (VRP) exprimées en termes d'une Ontologie de Domaine (OD). La vue en question permet d'expliciter la relation sémantique reliant les paramètres d'entrée et de sortie du service DaaS. Toutefois, une OD est incapable de saisir les différents perspectives ou points de vue de la connaissance du domaine médical. Cette limitation soulève des conflits sémantiques qui entravent l'exécution de la composition de services DaaS. En effet, les différences d'interprétations de données basées sur le contexte, non capturées par les VRP, surgiront lors des échanges de données entre services DaaS participants à une composition.

Pour cela, nous proposons une approche basée, à la fois, sur le contexte et sur les services de médiation pour détecter et résoudre automatiquement les conflits sémantiques dans une composition de services DaaS. La détection des conflits est rendue possible par l'enrichissement de modèle de vues par des informations sur le contexte, et ce, pour soutenir la médiation sémantique entre les services DaaS dans une composition. La résolution des conflits est basée sur les services de médiation à l'effet de bénéficier des mécanismes offerts par les architectures orientées services. Les services de médiation appropriés sont insérés automatiquement dans des compositions de services DaaS pour résoudre les conflits sémantiques dans leur flux de données. La mise en œuvre et les évaluations expérimentales réalisées dans ce cadre ont montré des résultats satisfaisants et prometteurs.

Mots-clés : Architecture Orientée Service, Intégration de données, Data as a Service (DaaS), réécriture de requêtes, Electronic Health Record (EHR), conflit sémantique, médiation sémantique, composition de services, Vue RDF Paramétrée, SPARQL, RDFS.

Abstract

Due to the significant development of hospital information systems, and distributed and collegial aspect of medical practice, there is now a strong need for software infrastructures that incorporate uniformly distributed and heterogeneous collections of data patient. In the context of e-health, the adoption of electronic health records (Electronic Health Record) and Web services, such as infrastructure software, have become predominant choice for the integration of the data.

As commonly agreed, Web services fall into two categories depending on their functionality world-altering services (SaaS) and Data as a Service (DaaS). Much work has been done on automatic DaaS discovery and composition, such as the query rewriting approaches proposed by the database community. In this context, DaaS service is described as Parameterized-RDF View over Domain Ontology (DO).This model specifies the semantic relationships between inputs and outputs DaaS parameters in a declarative way. However, the DO is unable to capture the different perspectives or viewpoints for the same domain knowledge. This limitation raises semantic conflicts between pieces of data exchanged during the DaaS composition process. Indeed, the differences of interpretations of data based on the context, not captured by the RDF view, arise when exchanging data between DaaS services participating in composition.

Therefore, we propose an approach based on both the context and the mediation services to detect and resolve automatically semantic conflicts in DaaS services composition. Conflict detection is based on context-driven approach that aims at supporting semantic mediation between composed services DaaS. Conflict resolution is ensured through mediation-as-a-service in order to benefit from SOA available mechanisms. The mediation services can seamlessly be integrated into existing DaaS services compositions to resolve semantic conflicts in their data flow.

The implementation and the experimental evaluations performed showed us satisfactory results.

Keywords : Service Oriented Architecture, Data integration, Electronic Health Record (EHR), Data as a service (DaaS), query rewriting, semantic annotation, semantic conflict composition, mediation, Data-as-a-Service (DaaS), DaaS composition, mediation service, semantic conflict, Parameterized RDF View, SPARQL query, query rewriting, RDFS.

Table des matières

Liste des abréviations

API	Application Programming Interface.
AOS	Architecture Orientée Service (Service Oriented Architecture).
BPEL	Business Process Execution Language.
DaaS	Data-as-a-Service.
DTD	Document Type Definition.
ebXML	Electronic Business using eXtensible Markup Language.
OD	Ontologie du Domaine.
OAC	Ontologie des Aspects Conflictuels.
EHR	Electronic Health Record.
ERP	Entreprise Ressource Planning.
IDE	Environnement de Développement Intégré.
HTTP	HyperText Transfer Protocol.
RDF	Resource Description Framework.
RDFS	Resource Description Framework Schema.
RR	Réécriture de Requêtes.
OWL-S	Ontology Web Language - Service Semantic.
SaaS	Software as a Service.
SAWSDL	Semantic Annotation - Web Service Description Language.
SCD	Systeme de Compoistion de services DaaS.
SWS	Service Web Sémantique.
WSDL	Web Service Description Language.
SOAP	Simple Object Access Protocol.
XML	Extensible Markup Language.
UDDI	Universal Description, Discovery and Integration language.
URI	Uniform Resource Identifiers.
WSMO	Web Service Modeling Ontology.
W3C	World Wide Web Consortium.

Table des figures

Liste des tableaux

Chapitre 1

Introduction Générale

Dans ce chapitre, nous présenterons les éléments de base délimitant le cadre de la présente thèse. À cet effet, la section 1.1 aborde le contexte dans lequel notre problématique est considérée ainsi que son domaine d'application lequel est exposé dans la section 1.2. Subséquemment, un exemple de motivation est présenté dans la section 1.3, qui vise la mise en évidence des défis auxquels nous faisons face et qui sont énumérés dans la section 1.4. À l'effet d'apporter une solution aux défis mentionnés, les contributions qui leur siéent sont énoncées dans la section 1.5.

1.1 Mise en contexte

Les dernières décennies ont été marquées par l'utilisation généralisée des sources de données et tout particulièrement par la démocratisation de l'accès à Internet. Donc, il est plus que normal que les entreprises et les organisations requièrent des données issues de sources multiples. Lesdites sources sont conçues d'une façon autonome; supportées par des systèmes hétérogènes; gérées par des applications locales et distribuées. Par conséquent, la localisation et l'identification des données pertinentes, le traitement de requêtes sont autant de difficultés qui rendent l'accessibilité et l'intégration des données un problème complexe.

L'accessibilité implique la définition des mécanismes permettant un accès transparent aux sources de données par rapport à l'utilisateur. L'intégration de données exige la possibilité de fournir une vue globale uniforme sur toutes les sources de données [81].

À ce titre, les services Web se sont vus exploités par plusieurs travaux afin de permettre l'accessibilité et l'intégration des données dans un environnement inter-organisationnel [65, 133, 31, 61]. Ces travaux visent à fournir une vue unifiée et dynamique des données à partir des sources autonomes, hétérogènes et distribuées.

Les services Web sont la déclinaison technologique la plus répondue du modèle des Architectures Orientées Service (AOS). L'idée maîtresse du paradigme architectural AOS est que tout élément du système d'information est un service identifiable, accessible, do-

cumenté, indépendant des autres services, indépendant des langages de programmation, indépendant des plates-formes et réalisant un ensemble de tâches définies [22, 15]. L'adoption de l'AOS a été grandement facilitée par l'émergence opportune de la technologie des services Web et de leurs standards basés sur XML (WSDL, UDDI, SOAP)[1].

Les services web sont caractérisés par des paramètres d'entrée et de sortie, par les effets qu'ils provoquent sur le monde et par les préconditions, ennoncées sous forme d'assertions, devons être vérifiées avant l'invocation des services. Il est communément admis que les services Web se répartissent en deux catégories en fonction de leurs fonctionnalités [31]. La première catégorie comprend les services qui provoquent un changement sur l'état du monde, nommés services à effet (SaaS: Software-as-a-service) tels que les services dédiés à la réservation d'hôtel ou l'achat en ligne. La deuxième catégorie comprend les services qui ne provoquent aucun changement sur le monde, nommés services fournisseurs de données (DaaS : Data-as-a-Service). À titre d'exemple, un service DaaS peut retourner les résultats des examens du laboratoire pour un patient donné ou retourner une séquence d'ADN alignée avec une séquence donnée. La fonction du service DaaS se réduit à fournir des données en sortie selon des paramètres en entrée sans provoquer d'effets et sans vérifier des préconditions.

Par ailleurs, l'usage des services DaaS a permis de fournir un moyen de virtualisation des sources de données, et ce, à travers l'encapsulation des données. Dans ce contexte, de plus en plus de systèmes d'information de santé (p.ex. hôpitaux) rendent des fragments de données du dossier électronique de santé (EHR) accessibles à travers des services DaaS. Ainsi, les données EHR peuvent être manipulées via les mécanismes offerts par les AOS (découverte, sélection, invocation, composition) au moment où un professionnel de santé les requiert en spécifiant seulement des paramètres d'entrée [31, 38, 111, 122, 39]. Cependant, face aux requêtes complexes des utilisateurs, un seul service DaaS ne pourrait pas forcément y'répondre. Pour cela, la composition de services DaaS offre la possibilité de combiner les résultats retournés par plusieurs services DaaS pour répondre aux dites requêtes.

Plusieurs approches liées à l'automatisation de la composition de service Web (SaaS et DaaS) ont été proposées. Parmi ces approches figurent celles basées sur les Services Web Sémantiques (SWS) [108]. Elles sont motivées par le fait que le fichier de description du service Web (p.ex. WSDL) est insuffisant pour automatiser ses activités (découverte, sélection, composition....). Les approches de compositions basées sur les technologies des SWS se scindent en deux catégories. La première propose des ontologies génériques de description de services (p.ex. OWLS, WSMO) et la deuxième propose des mécanismes d'extension des fichiers de description des services Web (p.ex. WSDL-S, SAWSDL). Toutefois, plusieurs travaux [14, 132, 124] imputent à ces approches leur incapacité d'exprimer la relation sémantique reliant les paramètres d'entrée et de sortie d'un service DaaS. Cette limite d'ordre expressive entrave l'automatisation de la composition des services DaaS.

1. Dans la section 2.2, les standards des services Web seront présentés.

A ce titre, [14, 132, 124] proposent de décrire le service DaaS sous forme de Vues RDF Paramétrées (VRP) exprimées en termes d'une Ontologie du Domaine (OD) qui fait office d'un schéma de médiation [127]. Les vues définies sont utilisées pour annoter les fichiers de description (p. ex. WSDL) des services DaaS et exploitées pour automatiser leur découverte et leur composition. De cette façon, le problème de composition de services DaaS est assimilé à celui de la réécriture de requêtes largement étudié par la communauté des bases de données [82].

Cependant, la construction d'une OD unifiant toutes les représentations existantes des entités du monde réel du domaine e-santé est une forte limitation de l'interopérabilité entre les services DaaS. Cette limitation soulève essentiellement des problèmes de conflits sémantiques survenant lors des échanges de données au cours de la composition de services DaaS. En effet, vu que les contextes des fournisseurs et des clients des services DaaS sont différents, le même concept ontologique peut être sujet à diverses interprétations (p.ex. unité de mesure, monnaie, classification, etc.). Pour cela, l'executabilité des compositions de services DaaS générées par la réécriture de requêtes est compromise. Par conséquent, il est crucial de considérer la détection et la résolution des conflits sémantiques dans les compositions de services DaaS.

1.2 Domaine d'intérêt

Le terme e-santé est largement utilisé pour se référer à l'utilisation des technologies de l'information et de la communication en rapport avec le domaine de la santé. L'e-santé vise à améliorer la dispensation des soins et la gestion des dossiers médicaux des individus à travers les systèmes d'information de santé et souvent via l'Internet [114]. Le passage à la gestion électronique du dossier de santé d'un patient (EHR) [2] est devenu la principale préoccupation pour beaucoup de projets de recherche.

Selon le *Healthcare Information and Management Systems Society*[3](HIMSS) "Electronic Health Record (EHR) is a longitudinal electronic record of patient health information generated by one or more encounters in any care delivery setting" qui pourrait se traduire par : "*Le Dossier Electronique de Santé (EHR) est un dossier de santé longitudinal partagé rassemblant les données produites suite aux interactions ayant eu lieu entre un patient précis et des prestataires ou des établissements de soins*".

Le but visé derrière ce concept est de permettre aux professionnels de santé une accessibilité transparente aux données pertinentes et actualisées des patients, réduire les erreurs médicales et les coûts des prestations de soins, faciliter le partage et la communication des données du patient entre les fournisseurs de soins et les organismes de santé [70].

Dans ce contexte, caractérisé par la distribution des données EHR, l'intégration et

2. Dossier Electronique de Santé est la traduction de " Eltronic Helath Record " dont l'acronyme "EHR" sera utilisé le long de la thèse

3. http://www.himss.org/ASP/index.asp

l'échange de ces données sont devenus des exigences du monde e-santé. Pour cette raison, de nombreux efforts ont été entrepris afin d'identifier les exigences fonctionnelles et non fonctionnelles des architectures requises pour assurer la gestion des données EHR [70, 41]. En ce sens, le modèle d'Architecture Orientée Service est largement adopté à l'effet d'assurer, à la fois, l'accessibilité aux données EHR et leur intégration [98, 63, 77]. Cependant, la construction des systèmes à base de services dans le domaine e-santé requiert l'usage d'un nombre significatif de standards [128]. En effet, en plus des standards des services Web, les standards de l'informatique médicale, tels que, "HL7" [12, 45], the Systematized Nomenclature of Medicine "SNOMED" [116], "RxNorm" [35] sont employés dans ces systèmes.

A cet effet, malgré la prolifération des services DaaS fournissant des données EHR [4], le problème d'automatisation de leurs activités se complexifie avec la nature des données EHR. En effet, les données EHR retournées par les services DaaS se conforment à différents standards (SNOMED, LOINC, ICD, etc.), terminologies, classifications ou modes de représentations [5]. En ce sens, notre exemple de motivation met en exergue les limites des cadres de référence [14, 132, 124] proposés pour la composition de services DaaS dans le contexte e-santé.

1.3 Exemple de motivation

Le présent exemple de motivation met en évidence un cas de composition des services DaaS dans le domaine e-santé. Dans ce contexte, les besoins en informations des professionnels de santé seront satisfaits via l'exploitation des fonctionnalités du Système de Composition de services DaaS (SCD). Le SCD en question s'inspire des cadres de références proposés par [14, 132, 124]. Nous supposons que la composition de services DaaS est générée via le SCD. Ce système expose une multitude de services DaaS permettant d'interroger les données EHR tels qu'ils sont présentés dans le tableau 1.1 où les symboles "$" et "?" désignent respectivement les paramètres d'entrée et de sortie des services DaaS.

Nous supposons qu'un médecin veut consulter les lectures des tensions artérielles (BPR) prélevées sur son patient nommé "Joe". Pour ce faire, le médecin soumet à travers l'interface fournie par le SCD, la requête suivante

Q: "Quels sont les états des valeurs indiquées par les récentes lectures de BPR de Joe ".

Il faut mentionner que l'état d'une valeur BPR est interprété selon une table de classification des valeurs BPR que chaque médecin reconnait et applique. Dans notre cas, le médecin reconnait l'ancienne classification des états exprimés comme suit : état sévère; état modéré; état faible.(ref???)

4. Les services DaaS fournisseurs des données EHR seront nommés services DaaS dans le reste du mémoire.

TABLE 1.1 – Description des services DaaS

Service	**Fonctionnalité**	**Contexte**
$S_1(\$x, ?y)$	Retourne la référence de l'examen "y" des signes vitaux effectué sur le patient "x"	
$S_2(\$x, ?y, ?z)$	Retourne le code "y" et la lecture "z" d'un examen de pression artériel BPR faisant partie d'un examen de signes vitaux "x"	"z.unit" est en "$mmHG$", "z.valeur" est représenté par deux valeurs concaténées "$BPR.Diastolic/BPR.Systolic$", "$y$" est exprimé en LOINC
$S_3(\$x, \$y, ?z)$	Retourne l'état "z" pour la lecture "y" de la Pression Artérielle Moyenne (MAP) "x"	"x" est exprimé en SNOMED, "$y.unit$" est en "$cmHG$" et "$z.state$" est exprimé selon la nouvelle table de classification des valeurs "BPR"

En réponse à cette requête, le SCD génère automatiquement les compositions de services DaaS dont l'union des résultats constitue la réponse à la requête. A ce titre, les services suivants : S_1, S_2 et S_3 participent dans une composition retournant les résultats escomptés comme indiqué dans la Figure 1.1.(a). Nous supposons pour des raisons de simplification que le SCD génère une seule composition au vu des services DaaS fournis dans l'exemple. A cet effet, le SCD invoque automatiquement :

- "S_1" qui retourne les examens récents des signes vitaux effectués sur "Joe";
- "S_2" qui retourne les lectures BPR[5] de "Joe";
- "S_3" qui retourne l'état de la lecture "MAP"[6] selon la valeur du "BPR" donné[7].

Cette composition est générée par l'algorithme de réécriture implémenté dans le SCD. Cet algorithme utilise la requête de l'utilisateur et les descriptions de services DaaS (VRP) publiés dans le registre à l'effet de générer les différentes compositions de services DaaS.

Cependant, le modèle de VRP n'explicite pas le contexte. Par le contexte, nous entendons la connaissance permettant de détecter les conflits sémantiques. Lesdits conflits peuvent survenir à la fois, entre les services DaaS subséquents dans une composition donnée ou entre le résultat requis par la requête et celui retourné par la composition. Ainsi, le SCD n'est pas en mesure de détecter les conflits qui compromettent l'exécution de la composition générée. Afin d'y remédier, le médecin aura à procéder manuellement, en

5. La lecture du BPR est représenté par deux valeurs concaténés, p.ex. "120/80 mmHG" où "120" est la valeur du BPR Diastolique (BPR.D), "80" est la valeur du BPR Systolique (BPR.S) et "mmHg" est l'unité de mesure.

6. L'état de la lecture "MAP" selon la nouvelle table de classification est exprimée en terme de phases comme suit : 1, 2, 3, 4

7. La tension artérielle moyenne est une mesure de "BPR" où $MAP = \frac{2}{3}(BPR.D) + \frac{1}{3}(BPR.S)$

premier lieu, à la détection des conflits existants dans la composition indiquée dans la figure 1.1.(a). Les conflits en question sont :

- "cos_1", le conflit provoqué par la différence des systèmes de codification pour représenter une lecture de la tension artérielle BPR;
- "cos_2", le conflit provoqué par la structure de la valeur du BPR (Diastolique/Systolique , MAP) et de l'unité de mesure utilisée (mmHG, cmHG);
- "cos_3", le conflit d'échelle provoqué par la différence des systèmes de classification des lectures BPR.

En deuxième lieu, le SCD devrait être muni par d'autres fonctionnalités ayant pour but la résolution des conflits précédemment détectés. Ces fonctionnalités seront assurées par des services de médiation. Le fait de considérer les fonctions de résolution de conflits comme des services sera explicité dans le chapitre 4.1. A cet effet, le SCD sera étendu par un registre de services de médiation. Le tableau 1.2 indique quelques services dont le médecin aura besoin (p.ex. MS_1,..,MS_4). Dans ce contexte, le médecin aura la possibilité

TABLE 1.2 – Description des services de médiation

Service	**Fonctionnalité**	**Contexte**
$MS_1(\$x, ?z)$	Retourne la valeur d'un code d'examen exprimé selon le standard "z" pour une valeur d'un code exprimé selon le standard "x"	"x" est exprimé en LOINC et "z" est exprimé en SNOMED
$MS_2(\$x, \$y, ?z)$	Retourne la valeur MAP "z" pour une lecture BP.Diastolic "x" et BP.Systolic "y"	
$MS_3(\$x, ?z)$	Retourne la valeur d'un BPR exprimé en unite de mesure "z" pour une valeur BPR exprimé en unité de mesure "x"	$x.unit$ est en "mm/HG" et "$z.unit$" est en "$cmHG$"
$MS_4(\$x, ?z)$	Retourne l'état de la valeur d'une BPR selon la classification "z" vers une autre classification de valeur BPR "x"	"$x.state$" est exprimé selon la nouvelle table de classification des valeurs et "$z.state$" exprimé selon l'ancienne table de classification des valeurs

d'accomplir manuellement ce qui suit :

- Invoquer "MS_1" pour faire correspondre le code BPR retourné par S_2(LOINC[8]) au code BPR accepté par S_3 (SNOMED[9]);
- Composer "MS_2" et "MS_3" où "MS_2" procède au calcul de la valeur MAP acceptée par S_3 à partir des deux valeurs exprimant la mesure "BPR" retournées par "S_2";

8. LOINC : Logical Observation Identifiers Names and Codes
9. SNOMED: Systematized Nomenclature of Medicine, Clinical Terms

"MS_3" procède à la conversion de la valeur "MAP" exprimée avec l'unité de mesure ("mm/Hg") retournée par MS_2 à la valeur "MAP" exprimée avec l'unité de mesure acceptée par s_3 ("cm/Hg");
– Invoquer "MS_4" pour faire correspondre l'état de la valeur BPR retournée par "S_3" représentée selon la nouvelle table de classification vers l'état accepté par le médecin représenté selon l'ancienne table de classification.

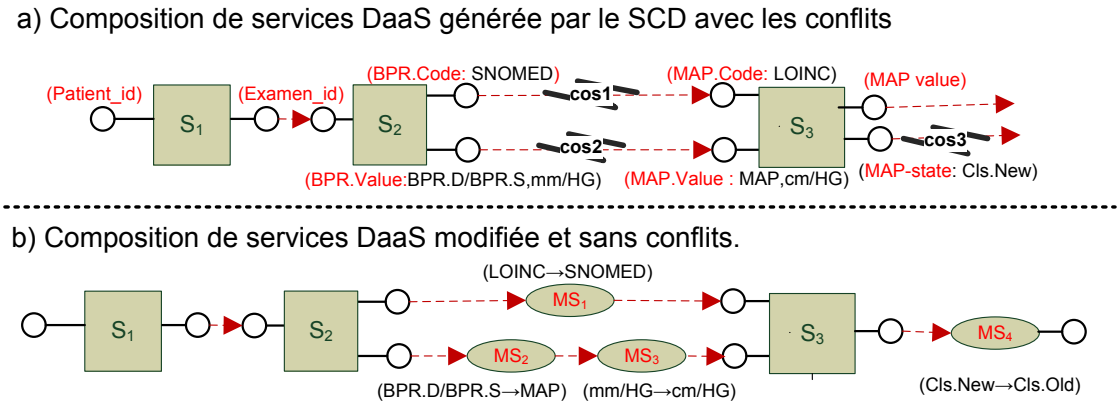

FIGURE 1.1 – Requête du scénario de motivation

A cet effet, l'exécution des tâches citées supra permet de modifier la composition générée par le SCD en une composition de services DaaS sans conflits et complétement executables (voir Figure 1.1.(b)). Cependant, le niveau de technicité requis pour procéder à la détection des conflits, à la découverte, à la sélection et à la composition des services de médiation appropriés rend ces tâches fastidieuses pour un utilisateur non expérimenté.

1.4 Problématique

Les défis soulevés dans l'exemple de motivation constituent les points fondamentaux de notre problématique. Par conséquent, la génération automatique des compositions de services DaaS, fournissant des données EHR, sans conflits dresse les défis ci-dessus.

1.4.1 Détection des conflits

Le modèle d'annotation des services DaaS basé sur la VRP est exprimé en terme d'une OD. Toutefois, une OD censée conceptualisée les connaissances du domaine médical, dans notre cas, ne peut se prétendre complète et concise. En effet, il est difficile d'exprimer les différents points de vue ou les représentations du même concept dans une seule ontologie [80].

Dans l'exemple de motivation, les paramètres de sortie et d'entrée des services S_2 et S_3 sont représentés différemment malgré qu'ils expriment le même concept selon l'OD. En effet, ces deux services retournent et consomment respectivement la même valeur d'une lecture BPR avec deux différentes unités de mesure et deux différentes codifications ou désignations. Les conflits se produisent chaque fois que deux services DaaS subséquents dans une composition n'adoptent pas la même interprétation de la donnée échangée [126]. À cet effet, l'exécution des compositions générées par l'algorithme de réécriture de requêtes se verra compromise. Pour cela, tout conflit doit être détecté avant l'invocation des services DaaS participants dans la composition pour en assurer l'exécution.

1.4.2 Résolution des conflits

Une fois les conflits détectés, la médiation de données devrait être effective. La médiation requise en de telles circonstances est similaire à la médiation de données dans d'autres domaines, comme celui des bases de données [26]. Cependant, elle est placée dans le contexte spécifique de la composition de services DaaS. Dans ce contexte, la médiation assurant la résolution des conflits devrait être supportée par des fonctions réutilisables et maintenables. Or, le fait d'opter pour une approche orientée service, pour bénéficier des mécanismes offerts par le modèle d'AOS, ne pourrait être considéré comme une solution en soit. En effet, il faut élucider les aspects liés à la découverte et à la sélection de services de médiation pour la résolution d'un conflit donné. En plus, dans le cas où il n'existe pas de services de médiation appropriés assurant la résolution d'un conflit donné, de quelle maniére la composition de services de médiation pourrait-elle y parvenir ?

1.4.3 Modèle de description des services

L'utilisateur, à travers le SCD, devrait avoir les possibilités de découvrir, de sélectionner, de composer et surtout de discerner automatiquement les services DaaS des services de médiation. La sémantique des services DaaS, décrite par la relation sémantique entre les paramètres d'entrée et de sortie, ne pourrait pas être une sémantique appropriée aux services de médiation. C'est en termes de fonctions de transformation que les services de médiation devraient être identifiés. Aussi, étant donné que les services de médiation éxecutent une fonction de conversion entre paramètres, il devient évident que la représentation par le modèle de VRP n'est pas idoine.

1.5 Contributions

Au vu des défis mentionnés dans la section précédente, nos contributions peuvent être résumées comme suit :

1.5.1 AOS pour la gestion des données EHR

À l'effet d'assurer l'intégration des données EHR dans un environnement interorganisationnel, nous proposons l'extension du Système de Composition de services DaaS. En effet, au vu de la richesse des connaissances du domaine médical et la multitude des perspectives liées à l'interprétation des données EHR, notre extension vise à expliciter, sous forme de contextes, la sémantique enfouie dans les données EHR. Ledit contexte sera à la base de la description des services de médiation, de l'extension des requêtes et des descriptions des services DaaS, de l'annotation du registre des services de médiation.

1.5.2 Espace ontologique à deux niveaux

La conception d'une ontologie globale et intégrée, faisant office d'un schéma de médiation, est prévu pour décrire les services DaaS. Toutefois, il existe une impossibilité pratique pour maintenir cette ontologie dans un environnement très dynamique [80]. Pour cela, la conception de l'ontologie devrait prendre en considération le niveau générique du domaine et les niveaux spécifiques ayant trait à la contextualisation des concepts génériques. Ainsi, il devient possible de préciser la sémantique des paramètres des services DaaS. Cette précision vise à fournir la possibilité de détecter les conflits sémantiques survenant lors du passage des paramètres durant la composition de services DaaS [5]. A cet effet, une ontologie à deux niveaux est proposée pour scinder l'espace conceptuel en une Ontologie du Domaine (OD) et un ensemble d'Ontologies des Aspects Conflictuels (AOC). L'OD sera utilisée pour décrire les services DaaS permettant leur composition automatique [6]. Les OAC décrivent les aspects pouvant faire l'objet de médiation et caractérisent le contexte des paramètres des services DaaS.

1.5.3 Détection des conflits

Le modèle des services DaaS (VRP) sera étendue par le contexte exprimé en termes d'OAC. Le contexte vise l'enrichissement de la description sémantique des paramètres des services DaaS. Ces annotations explicitent les aspects pouvant faire l'objet de conflits entre les services DaaS. Aussi, la détection automatique des conflits dans une composition de services DaaS sera rendue possible. A cet effet, le SCD sera étendu par un module dédié à la détection des conflits sémantiques dans une composition de services DaaS [4].

1.5.4 Résolution des conflits

Nous proposons une approche basée sur les services de médiation pour la résolution des conflits sémantiques. Le choix des services pertinents pour la résolution des conflits doit être transparent à l'utilisateur. Pour cela, nous spécifions un modèle de services de médiation assurant la transformation de données entre paramètres de services DaaS contextualisés différemment. Ce modèle offre la possibilité d'automatiser la découverte, la

sélection et la composition des services de médiation. A cet effet, le SCD sera étendu par un module dédié à la résolution des conflits sémantiques [3].

1.5.5 Implémentation et expérimentations

Notre approche a été implémentée sur une plate-forme JAVA dédiée à la fois pour la gestion des services DaaS fournisseurs des données EHR et des services de médiation. Plusieurs expérimentations sont effectuées afin de montrer les performances de nos algorithmes. En plus, lesdites expériences permettent de mettre en évidence le comportement vis-à-vis du passage à l'échelle.

1.6 Organisation de la thèse

L'organisation de la thèse se décline comme suit :

Chapitre 2 "**Préliminaires**" : Dans ce chapitre les notions fondamentales ayant trait aux différents domaines entrant en ligne de compte dans notre problématique seront dressées. Dans un premier temps, les notions de bases ayant trait aux architectures orientées services et aux services Web seront explicitées. Ensuite nous le cycle de vie de la composition de services sera détaillé. En troisième lieu, étant donné que notre approche de composition est basée sur l'extension des fichiers de descriptions des services DaaS par des annotations sémantiques, les différentes approches de modélisation des connaissances seront présentées. En dernier lieu, un rappel des notions liées aux architectures de médiation de données sera exposé.

Chapitre 3 "**Etat de l'art**" : Dans ce chapitre une étude comparative des approches de découvertes et de compositions de services DaaS sera présentée. Les approches en question considèrent les services DaaS sous forme de vues. En deuxième lieu, les approches d'intégration de données EHR basées sur les approches orientées services seront revues. En dernier lieu, les approches de médiation sémantique dans une composition de services seront exposées.

Chapitre 4 " **Approche : architecture et modèle**" : Dans ce chapitre, notre approche orientée service et basée sur la notion du contexte pour rendre effective la médiation sémantique dans une composition de services DaaS sera présentée. A ce titre, nous détaillerons notre architecture ainsi que les fondements de notre approche. Nous formaliserons notre approche en terme de modèles liés au contexte, à l'ontologie à deux niveaux et au service de médiation.

Chapitre 5 " **Génération automatique des compostions de services DaaS**" : Dans ce chapitre, toutes les spécifications liées au traitement d'une requête seront explicitées. En premier lieu, l'algorithme de réécriture pour lequel des adaptations ont été apportées sera présenté. Ces adaptations s'inscrivent dans le cadre de l'extension proposée liée à la détection et à la résolution de conflits dans une composition de services DaaS.

Chapitre 6 "**Détection et de résolution des conflits**" :Dans ce chapitre, les algorithmes liés à la détection et à la résolution des conflits dans une compostions de services DaaS seront présentés. Aussi, les algorithmes proposés seront illustrés par des exemples. Par ailleurs, une représentation processuelle de cette nouvelle phase du processus de génération automatique de composition de services DaaS sera exposée.

Chapitre 7 " **Architecture et implémentation**" dans ce chapitre nous présentons les modules développés pour étendre les fonctionnalités du SCD par des fonctions de détection et de résolution des conflits ainsi que l'annotation et la publication des services (DaaS et médiation). Aussi les résultats des tests de performance des algorithmes seront présentés et commentés.

Chapitre 8 "**Conclusion et perspectives**" : Dans ce chapitre, une synthèse des contributions sera effectuée accompagnée par les différentes perspectives de recherches qui en découlent.

Chapitre 2

Préliminaires

2.1 Introduction

Dans ce chapitre, nous présenterons les notions fondamentales ayant trait à la problématique de détection et de résolution des conflits sémantiques dans une composition de services DaaS. La section 2.2 précisera le paradigme des Architectures Orientées Services, les concepts clés liés aux technologies à la fois des services Web et service DaaS. Dans la section 2.3, les étapes de la composition de services seront définies. Étant donné que l'automatisation de la composition, prônée dans cette thèse, est basée sur une approche sémantique, la section 2.4 exposera les notions de base ayant trait aux ontologies et au Web sémantique. La section 2.5, une synthèse des technologies des Services Web Sémantiques sera exposée afin d'expliciter leur emploi à l'effet d'automatiser les activités liées aux services Web. Au niveau de la section 2.6, les notions ayant trait à la médiation de données seront succinctement présentées.

2.2 Du service Web au service DaaS

Dans cette section, nous allons nous atteler à expliciter le paradigme des "Architectures Orientées Services", et ses déclinaisons technologiques notamment les services Web et DaaS.

2.2.1 Architecture Orientée service

Le consortium OASIS [10] a propose un modèle de référence pour les Architectures Orientées Services (AOS) [23] où il le definit comme suit "*Service Oriented Architecture (SOA) is a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains*". Cela pourrait se traduire par ceci : "l'Architecture Orientée Service (AOS) est un paradigme pour organiser et utiliser des capacités

10. http://www.oasis-open.org

ou fonctionnalités distribuées pouvant être contrôlées par différents propriétaires". Par capacités, il est entendu une fonctionnalité métier automatique invocable à distance.

Ce paradigme se base sur trois concepts clés, à savoir, la visibilité, les interactions et les effets. La visibilité permet aux fonctionnalités et aux besoins de se rencontrer. Les interactions consistent en l'utilisation des fonctionnalités à travers les échanges de messages. Enfin, le résultat d'une interaction est un effet lequel consiste en un changement d'état des entités informationnelles impliquées dans l'interaction en question. Ces concepts clés se retrouvent délégués au niveau d'un autre concept qui est le*service*, considéré comme la brique de base de l'AOS.

Selon [50], le service est un moyen d'accéder à des fonctionnalités offertes par un composant applicatif. Le service fournit une description de son interface indépendante de toutes plateformes (auto-descriptif). Le service est autonome, complet et cohérent, car il encapsule toute la logique d'exécution (self-contained). Le service possède une adresse ce qui permet de l'invoquer dynamiquement (localisable). Le service Web est l'une des déclinaisons technologiques de l'AOS.

2.2.2 Service Web

Plusieurs définitions du service Web ont été proposées dans la littérature. Cependant, celle proposée par le World Wide Web Consortium (W3C) fait figure de référence [59]: *« A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP-messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards»*. Cette définition met en évidence les caractéristiques d'un service Web, à savoir :

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

2.2.2.1 Modèle d'interaction

Le fonctionnement des services Web est basé sur un modèle d'interaction incluant trois types de rôles: le fournisseur de services (*service provider*), le client du service (*service requestor*) et le registre de services (*service registry*). Les trois opérations de bases permises entre ces rôles, telles que indiquée dans la figure 2.1, sont : publier (*publish*), rechercher (*find*) et interagir (*bind*).

Chaque service Web est défini par un fournisseur. Le fournisseur publie la description

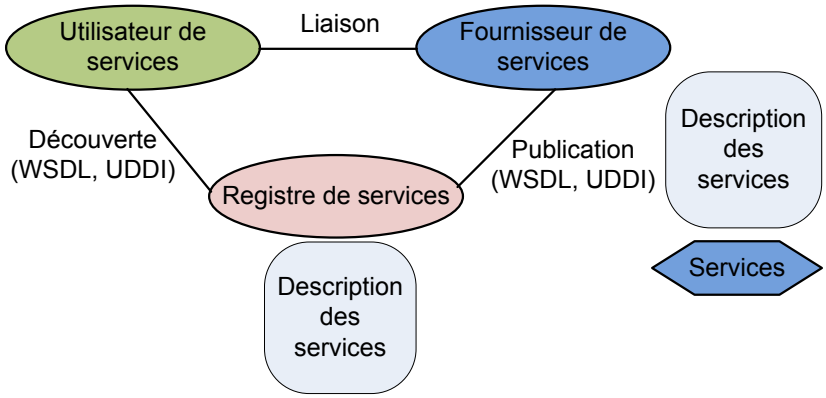

FIGURE 2.1 – Modèle d'interaction dans une Architecture Orientée Service (Service Web).

du service publié dans un ou des registres accessibles en consultation par des clients à travers l'Internet ou l'Intranet. Les services Web sont publiés avec des adresses URLs appropriées spécifiées par les fournisseurs. Les clients de services Web localisent leurs besoins en terme de services en opérant une recherche dans le registre de services. Une fois le service localisé, le client récupère la description du service sélectionné, à partir du registre, et l'intègre dans son application. Sur la base des informations fournies dans la description du service, le client interagit avec le service en vue de l'invoquer.

2.2.2.2 Protocoles de services Web

Une caractéristique majeure de la technologie des services Web est qu'elle se base sur des protocoles industriels standardisés, basés sur XML, défini par le W3C [11]. Lesdits standards sont déclinés succinctement comme suit:

- SOAP (Simple Object Access Control) [12] est un protocole de description et de structuration des messages échangés entre les Services Web basé sur XML. Le protocole SOAP permet d'invoquer un objet distant en communiquant les informations nécessaires à l'appel (Nom de la méthode et sa signature dans un message au format XML). La réponse à la requête est aussi renvoyée encapsulée au format XML. Un message SOAP est une structure qui comporte des parties obligatoires, l'enveloppe (*envelop*) et le corps du message, et une partie optionnelle, l'entête (*header*).
- WSDL (Web Service Description Language) est un langage de description des services basé sur XML [33]. Le WSDL spécifie le service Web sur deux niveaux, abstrait (réutilisable) et concret.

  1. Le niveau abstrait : Ce niveau défini les opérations que le service peut exécuter. Chaque opération est constituée d'un ensemble de messages d'entrée et de sortie. Chaque message contient une ou plusieurs données typées. Une liaison

11. World Wide Web Consortium
12. http://www.w3.org/TR/2000/NOTE-SOAP-20000508/

(binding) établit une correspondance entre le protocole utilisé pour communiquer, le port et l'opération.

2. Le niveau concret: Le document WSDL défini le service comme des collections de points finaux (*endpoints*) de réseaux aussi appelés "ports". Via ces ports le service communique par échange de messages. Chaque port est associé à une liaison (binding) laquelle détermine comment peut-on accéder à l'opération en utilisant un protocole de communication particulier (SOAP, HTTP,....).

– UDDI (Universal Description, Discovery and Integration) est un standard, publié en 2002 [17], dédié à la publication des services et les informations concernant le fournisseur de services. UDDI assure le stockage, le regroupement et la diffusion des descriptions de services Web.

2.2.3 Service fournisseur de données (DaaS)

Un service DaaS fournit un accès uniforme à une ou à plusieurs sources de données distribuées. L'invocation d'un service DaaS provoque l'exécution d'une requête sur le schéma des sources en question [30, 1]. Les services DaaS sont caractérisés par:

– **Les services DaaS sont des interfaces d'accès aux données**. Les services DaaS exploitent le paradigme des AOS afin d'exposer les sources de données hétérogènes d'un système d'information [122]. Les services DaaS épargnent aux développeurs d'applications d'avoir à interagir directement avec les sources de données permettant l'accès à des objets métier. Cela leur permet de se focaliser sur la logique métier seulement.
– **Les services DaaS sont sans état (stateless)**. Un service DaaS ne provoque pas d'effets, il est dit sans état [66]. Ainsi, le DaaS service fait abstraction de la syntaxe des données et de leur structure. L'exécution d'un service DaaS est déclenchée par la réception en entrée des données typées fournies par une entité (utilisateur ou service) et retourne en sortie les données sollicitées par la même entité.
– **Les services DaaS sont des applications faciles à utiliser**. Les services DaaS virtualisent les données afin de découpler leurs emplacements physiques de leurs logiques. Les services DaaS fournissent une vision simplifiée et intégrée des données liées à un objet métier tel que le patient ou le test de laboratoire.
– **Les services DaaS sont une norme appropriée pour la publication des données EHR**. Les services DaaS sont de plus en plus utilisés pour manipuler les différentes parties du Dossier Electronique de Santé (EHR). La granularité de ces parties est définie par les différents blocs de construction de données spécifiés par les différentes normes EHR telles que l'ISO TC215, HL7 et GEHR. Les services DaaS offrent un accès aux données EHR[13] pouvant être agrégées pour fournir la vue requise par l'utilisateur sur les données du patient.

13. c.-à-d. les attributs tels que: *allergies*, *médicaments prescrits*, *test de laboratoire*, etc.

2.3 Composition de services Web

La composition consiste à combiner les fonctionnalités offertes par plusieurs services Web à l'effet de répondre à des requêtes de sévices complexes qu'un seul service ne pourrait pas satisfaire [18]. Dans cette section, nous allons décrire les principales étapes d'une composition de services Web et les différentes approches de classification qui s'y réfère.

2.3.1 Étapes de la composition de services

Une composition de services consiste en l'identification des services requis et la définition de leur ordonnancement en alignement avec la requête du client. La composition de services est un processus de raffinement permettant de passer d'une spécification abstraite de la composition vers une description exécutable [130]. Une composition de services doit pouvoir être implantée et exécutée en invoquant et en exécutant des services existants. Considérées comme des artefacts, les compositions de services sont disponibles, réutilisables et adaptables pour répondre à de nouveaux besoins métier. Les étapes de création d'une composition de services ont été définies par [130] comme suit:

2.3.1.1 Spécification abstraite

Cette étape permet de spécifier d'une manière abstraite la composition de services. En effet, lors de cette étape il faut identifier la fonctionnalité devant être fournie par la composition ainsi que la fonctionnalité devant être apportée par les différents services participants. Finalement, les interactions entre les participants sont spécifiées.

2.3.1.2 Découverte et sélection

La découverte des services Web permettant de répondre aux besoins de la composition se fait par envoi de requêtes aux annuaires UDDI. Dans le cas où il pourrait y'avoir plusieurs services satisfaisant la même fonctionnalité, la sélection devrait être opérée afin de choisir le service le plus approprié.

2.3.1.3 Ordonnancement et interactions

Une fois les services Web requis sont découverts et sélectionnés, on distingue deux modes d'agencement des services sélectionnés lequel specifie le mode d'intercation entre services, à savoir, la chorégraphie et l'orchestration [104, 86].

- La chorégraphie ou la composition distribuée où chaque service Web gère les échanges de messages avec un autre service Web dans une composition. Dans ce contexte, les services ont la capacité d'interagir pour réaliser une tâche donnée.
- L'orchestration ou la composition centralisée où le nouveau service (composite) organise les interactions entre plusieurs services Web. L'orchestrateur, un composant

logiciel, prend en charge l'exécution d'un ensemble de tâches, prédéfinies ou non, afin d'obtenir le résultat attendu.

Une composition est associée à une description qui spécifie les échanges de messages et les structures de contrôle nécessaires (boucles itératives, opérateurs conditionnels et gestion des exceptions). Plusieurs langages de spécification ont été proposés dans la littérature (p.ex. pour la chorégraphie : WS-CDL [71], pour l'orchestration WS-BPEL [14][68],.etc). Cette étape fournie une composition prête à être exécuter à l'effet de répondre à la requête de services. Cependant, elle ne fait référence à aucun service mais elle décrit un processus avec ses activités, lesquelles devraient être réalisées avec les services Web disponibles.

2.3.1.4 Exécution de la composition

L'exécution de la composition est l'étape d'invocation effective des services Web participants à une composition. Une spécification de la composition est hébergée par un serveur. Un moteur de composition exécutera la composition par l'implémentation des liaisons avec les services participants et disponibles. La surveillance de l'exécution par le moteur de composition permet de gérer certaines erreurs dynamiquement.

2.3.2 Composition de services: Classification des approches

La composition de services Web vise à la fois de satisfaire les exigences complexes des clients et de réduire la difficulté liée au développement de services Web satisfaisant des applications complexes. En ce sens, la majorité des définitions suggérées dans la littérature s'accordent que la composition de services Web réside dans la capacité de fournir une nouvelle fonctionnalité obtenue à partir de la combinaison de plusieurs services offerts par plusieurs fournisseurs. Etant donné un ensemble de services Web publiés dans un registre et une requête de services formulée par un client, une composition de services peut être réalisée selon différentes approches. Les approches de compositions considèrent plusieurs critères [108]. En effet, quatre types de compositions peuvent être distingués selon [94]:

- Le moment durant lequel les services sont composés. Ainsi, une composition est dite statique si les services participant à la composition sont sélectionnés avant la phase d'exécution. Une composition est dite dynamique, si les services participant à la composition peuvent apparaître et (ou) disparaître à tout instant pendant l'exécution.
- La façon dont la composition est obtenue ou le niveau d'automatisation. Ainsi, une composition est dite : automatique s'il y' a intervention d'un agent logiciel qui implémente un algorithme de composition; semi-automatique dans le cas où l'utilisateur est assisté par un outil interactif pour choisir les services participant dans une composition lors de la phase d'ordonnancement des services ; manuelle

14. Business Process Execution Language (BPEL) [67].

dans le cas où un expert intervient, à la fois, pour sélectionner et ordonnancer dans une composition les services découverts.

Concernant les approches de compositions automatiques, elle peuvent être classées selon les techniques adoptées pour résoudre le problème de composition [108]. Ces techniques se basent sur la syntaxe ou sur la sémantique. L'approche syntaxique tente de définir une composition selon la syntaxe définie à travers différents formalismes (réseau de pétri, automate a état fini, les techniques de Workflow, Processus markovien,etc.). Ainsi, l'approche syntaxique assume la disponibilité d'une liste de services Web à composer. Cette liste sera exploitée par l'algorithme de composition qui détermine l'ordonnancement des services dans le composition. Les approches sémantiques se basent généralement sur l'émergence des standards du web sémantique. À ce titre, l'approche sémantique exploite les techniques suivantes ;la planification AI, L'algorithme de matching ou de chainage, le parcours des graphes, la réécriture de requêtes [14, 132, 84, 13, 119, 124]. Dans notre contexte on ne s'intéressera qu'aux approches sémantiques basées sur la réécriture de requêtes.

2.4 Ontologie et Web sémantique

2.4.1 Ontologie

Dans l'univers du traitement automatique de l'information le terme "Ontologie" a été introduit dans les années 1990 avec les travaux de Grüber et son équipe à Stanford [58, 57]. Selon Grüber, "*une ontologie est une spécification formelle, explicite d'une conceptualisation partagée*". Cette définition produite dans le contexte du partage et de la réutilisation des connaissances, explicite une ontologie à travers les concepts clés suivants:

- *Conceptualisation* qui fait référence à un modèle abstrait de certains phénomènes du monde et qui identifie les concepts pertinents de ces phénomènes. Une conceptualisation est une vue particulière, abstraite, au sujet des entités réelles, des événements et de leurs relations;
- *Explicite* qui signifie, à la fois, que le type des concepts utilisés et les contraintes liées à leur utilisation sont explicitement définis;
- *Formelle*, se réfère au fait que l'ontologie doit être compréhensible par les machines.
- *Partagée*, reflète la notion de connaissance consensuelle décrite par l'ontologie, c'est-à-dire qu'elle n'est pas restreinte au point de vue de certains individus seulement, mais reflète un point de vue plus général, partagé et accepté par un groupe.

Selon [36], une ontologie contient une description des types de concepts, la spécification de leurs attributs et des relations qui les relient (axiome ou relation). Le langage utilisé pour décrire les termes [15] d'une ontologie a un impact direct sur le niveau de formalisme de l'ontologie. On distingue les ontologies informelles, qui utilisent le langage naturel et qui

15. Terme, signifie les concepts ou attributs de concepts. Donc terme et concept seront utilisés à titre de synonyme

peuvent coïncider avec les terminologies; les ontologies semi-formelles, qui fournissent une faible axiomatisation [16], comme les taxonomies ; et les ontologies formelles, qui définissent la sémantique des termes par une axiomatisation complète et rigoureuse.

En fonction de leur usage, on distingue classiquement cinq catégories d'ontologies [125]:

- Les ontologies génériques décrivent des concepts généraux, indépendants d'un domaine ou d'un problème particulier. Il s'agit par exemple des concepts de temps, d'espace, d'événement. Les ontologies génériques sont aussi appelées modèles de haut niveau ou ontologies TOP (p.ex. Sowa [115] , CYC [79]).
- Les ontologies de domaine spécifient un point de vue sur un domaine particulier. Le vocabulaire est généralement lié à un domaine de connaissances générique (médecine ,loi). Les concepts d'une ontologie de domaine sont souvent définis comme une spécialisation des concepts des ontologies génériques.
- Une ontologie légère comprend des concepts, des types atomiques, une hiérarchie IS-A [17] entre concepts et des relations entre concepts.
- Une ontologie riche comprend des contraintes de cardinalité, une taxonomie de relations, des axiomes et suppose l'existence d'un système d'inférence.

2.4.2 Web sémantique

La manipulation des ressources du Web par des machines requiert l'expression ou la description de ces ressources. Le Web Sémantique (WS), qui est une extension du Web actuel, a comme objectif majeur d'apporter de la sémantique à l'information manipulée afin de faciliter la communication entre agents (hommes et machines). Ainsi, suite à un plan de route initié par Tim Berners-Lee [19], le W3C créa un groupe de travail lequel publie, en 2004, une recommandation nommée Resource Description Framework (RDF) [85, 76]. Selon ledit groupe, l'objectif du Web sémantique est de fournir aux machines un accès automatisé à l'information grâce à des métadonnées les décrivant. Ces métadonnées sont exprimées par le composant pivot du WS, à savoir, les ontologies. Plusieurs langages, dont la syntaxe est basée sur le langage XML, ont été conçus pour une utilisation dans le cadre du Web sémantique. Parmi les plus importants langages figurent le RDF/RDFS [40], OWL [91] et SPARQL[107]. Nous présenterons succinctement, dans ce qui suit, une revue de ces langages.

2.4.2.1 RDF

Resource Description Framework (RDF) est une recommandation du W3C pour représenter les métadonnées [76]. RDF est un langage formel qui permet d'affirmer des

16. L'axiomatisation est un procédé qui consiste à organiser une théorie en se basant sur des axiomes, et à en déduire rigoureusement des théorèmes, dans un cadre qui peut être purement logique, ou celui de la théorie des ensembles. (Source : Wikipedia)

17. La hiérarchie IS-A représente une relation de généralisation entre concepts d'une ontologie.

relations entre des "ressources". RDF est utilisé pour annoter des documents écrits dans des langages non structurés, ou comme une interface pour des documents écrits dans des langages ayant une sémantique équivalente (p.ex. bases de données) [11]. RDF décrit les ressources en exprimant des propriétés et en leur attribuant des valeurs. Il utilise pour cela le vocabulaire défini par RDFS [40].
L'élément de base d'un document RDF est la ressource, correspondant à la représentation conceptuelle d'une entité. Un document RDF est un ensemble de triplets ayant la forme de "<sujet, prédicat, objet>" appelée assertion ou triplet. Les éléments de ces triplets peuvent être des URIs (Universal Resource Identifiers), des littéraux ou des variables. Le document RDF sera codé en machine par un document RDF/XML, N3 ou Turtle [18]. L'ensemble de triplets peut être représenté de façon naturelle par un graphe (c.-à-d. multi-graphe orienté étiqueté), où les éléments apparaissant comme sujet ou objet sont les sommets et les arcs (prédicats) ayant comme origine le sujet et comme destination l'objet. **Example**: La figure 2.2 montre un graphe RDF représentant l'assertion suivante "le patient a effectué un examen de laboratoire" où ($<Patient_{id}>;<$ effectué $>;<$ examen de laboratoire$>$) est un triplet qui a pour ressource (sujet) $<Patient_{id}>$, pour prédicat <effectué> et pour valeur (objet) <examen de laboratoire>.

```
<rdf:RDF
  xmlns: rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:s="http://description.org/schema/">
  <rdf:Description about="http://www.w3.org/Home/Lassila">
    <s:Creator rdf:resource="http://www.w3.org/staffId/85740"/>
  </rdf:Description>

  <rdf:Description about="http://www.w3.org/staffId/85740">
    <rdf:type resource="http://description.org/schema/Person"/>
    <v:Name>Ora Lassila</v:Name>
    <v:Email>lassila@w3.org</v:Email>
  </rdf:Description>
</rdf:RDF>
```

FIGURE 2.2 – Exemple d'un document RDF [76].

2.4.2.2 RDFS

RDF Schéma [88] a pour but d'étendre le langage RDF en fournissant un ensemble de primitives simples. Lesdites primitives concernent la structuration de la connaissance d'un domaine en classes et sous-classes, propriétés et sous-propriétés, tout en ayant la possibilité de restreindre leur domaine d'origine (rdf:domain) et leur domaine d'arrivée

18. RDF Terse la Triple Langue (tortue) , une syntaxe textuelle pour RDF

(rdf :range). RDFS s'écrit toujours à l'aide de triplets RDF, en définissant la sémantique des nouveaux mots-clés.

Example: La figure 2.3 presente un morceau d'une ontologie RDFS dont la namesapce est "ex". 1- Le triplet <ex:Vital-exam rdf:type rdfs:Class> désigne la ressource <ex: Vital-exam> qui a pour type <rdfs:Class> 2- Le triplet <ex:BPR rdf:type ex:Vital-exam> désigne la ressource <ex:BPR> qui est une instance de la classe <ex:Vital-exam>; 3- <ex:Laboratory-test rdfs:subClassOf ex:Vital-exam> la classe <ex:Laboratory-test> est une sous-classe de la classe <ex:Vital-exam>, toutes les instances de la classe <ex:Laboratory-test> sont donc des instances de la classe <ex:Vital-exam>;

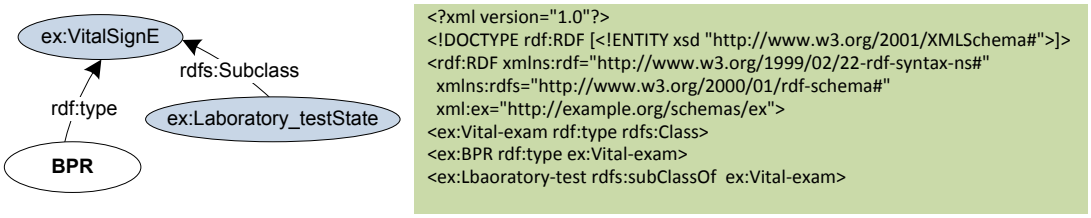

FIGURE 2.3 – Graphe RDFS et le code correspondant.

2.4.2.3 OWL

RDFS ne permet pas d'exprimer certaines notions que l'on voudrait décrire avec les ontologies. Dans le but d'étendre l'expressivité de RDFS, le groupe de travail sur les ontologies du W3C a proposé le Ontology Web Language (OWL), basé sur le langage DAML+OIL [90, 8]. OWL définit un vocabulaire permettant de décrire des ontologies, tout en offrant des possibilités de description plus riches. Parmi les notions importantes apportées par OWL, citons les propriétés d'équivalence, d'identité de deux ressources, de différences entre deux ressources, de contraire, de symétrie, de transitivité et de cardinalité. Ces propriétés permettent l'utilisation de raisonneurs afin de déterminer des relations d'équivalence ou de subsomption entre des concepts de domaines de connaissances [19] et d'évaluer automatiquement la compatibilité entre différentes représentations sémantiques.

2.4.2.4 SPARQL

SPARQL (Simple Protocol and RDF Query language) est une recommandation du W3C à la fois un langage et un protocole de requêtes pour l'interrogation de triplets RDF. Ce protocole va permettre à un client Web de consulter, en exécutant une requête SPARQL, un service ou point d'accès SPARQL (endpoint). Le point d'acces traitera la requête pour retourner la réponse sous différents formats (HTML, XML, RDF/XML,

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

N3, etc.). Une requête SPARQL spécifie le graphe à apparier (graph matching) sur un ou des graphs RDF à l'effet de retourner des réponses et ce par la liaison des variables (variable binding). SPARQL permet d'interroger des descriptions RDF en utilisant les clauses suivantes :

- PREFIX : Spécifie l'adresse exploitée dans la construction de la requête ;
- SELECT...[FROM]...WHERE : Requête interrogative qui permet d'extraire du graphe RDF un sous-graphe correspondant à un ensemble de ressources vérifiant les conditions définies dans une clause WHERE;
- CONSTRUCT : Requête constructive qui engendre un nouveau graphe qui complète le graphe interrogé;
- UNION, OPTIONNAL : Jointures et conditions optionnelles;
- FILTER : Conditions obligatoires;
- DESCRIBE, ASK : Respectivement ces commandes désignent la Description d'une ressource et l'évaluation d'une requête.

Example: la requête SPARQL présentée dans en bas stipule " sélectionner les villes ayant une population de plus de 100 000 habitants ". Le Prefix <http://example.org/> mentionne le namespace à la TRUTLE.

```
PREFIX ex:<http://example.org/>
SELECT ?city
WHERE {?city a ex:City.
?city ex:population ?population.
FILTER ( ? populat ion >= 100000)}
```

2.5 Services Web sémantique: Approches et langages

Les standards de publication et de description des services Web, basés sur XML, n'explicitent pas la sémantique des fonctionnalités fournies par le service, des paramètres d'entrée/sortie du service, la qualité des services, etc. La sémantique [20] faisant défaut à ces standards constitue un verrou technique à l'automatisation des activités liées aux services Web (découverte, sélection, composition).

Le besoin d'automatisation de ces activités joint les préoccupations à l'origine du Web sémantique, à savoir, comment décrire formellement les connaissances de manière à les rendre exploitables par des machines. A cet effet, 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 d'automatisation. En effet, la communauté du Web sémantique propose des solutions reposant sur des ontologies de référence pour expliciter la sémantique des services [92]. Pour cela, le domaine des Services Web

20. Le terme sémantique est utilisé pour décrire ce qui a trait à la signification véhiculée par un objet, généralement un mot.

Sémantiques (SWS) se situe au croisement du Web sémantique et des services Web. De nombreux langages et architectures ont été proposés afin de décrire le domaine du SWS [87, 73, 103, 74, 95, 10, 27].

Le SWS propose des langages permettant l'enrichissement sémantique ou l'annotation des standards du service Web (WSDL, UDDI,..). Ces annotations s'effectuent à travers l'exploitation des éléments d'extensibilité ou en modifiant les fonctionnalités initiales des standards. Ces annotations peuvent concerner : a) les processus métier (Semantic Service MArkup) permettant la description d'une composition de services par l'annotation du processus métier exprimé en WS-BPEL; b) Les registres UDDI à travers l'utilisation du langage SAWSDL ; c) Les fichiers de description des services (WSDL).

Deux approches d'annotation des descriptions de services (ex.WSDL) sont proposées, à savoir, l'approche Top-Down et l'approche Bottom-up.

2.5.1 Approche Top-Down

L'approche Top-Down se base sur l'utilisation des langages décrivant la sémantique des services Web tels qu'OWL-S et WSMO.

2.5.1.1 Web Ontology Language for Services Web (OWL-S)

OWL-S est un sous-ensemble du langage OWL dédié à la description sémantique de services Web [87]. Techniquement OWL-S est une ontologie OWL composée de trois sous ontologies où chacune décrit respectivement ce qui suit:

- "Que fait le service ": désigné par le "service profile", il décrit les fonctionnalités des services Web, il est utile lors de la découverte et la sélection des services. Le "service profile" consiste en les informations liées au fournisseur du service et à la description fonctionnelle du service (IOPE: entrée, sortie, précondition, effet).
- "Comment le service fonctionne ": désigné par le "service model", il détaille la sémantique des données impliquées dans les messages échangés entre services. Le "service model" décrit le processus d'exécution du service à travers le "ProcessModel" fournissant une description détaillée sur comment le service opère. Le "ProcessModel" spécifie la transformation de données et les transitions d'état.
- "Comment accéder au service ": désigné par le "service grounding", il spécifie l'encodage des données échangées, les protocoles de communication, ainsi que tous les détails concrets nécessaires à l'invocation du service.

2.5.1.2 Web Service Modeling Ontology (WSMO)

WSMO est une architecture conceptuelle proposée par le laboratoire DERI[21], pour expliciter la sémantique des services Web [10]. Le meta-model du WSMO définie quatre éléments principaux : Les services Web, les objectifs, les ontologies et les médiateurs.

21. http://www.deri.ie/

- Le WSMO associe une description pour chaque service Web, comprenant quatre parties, à savoir, 1) les propriétés non fonctionnelles telles que le fournisseur, le coût, la performance, la sécurité, etc. Ces propriétés sont utilisées pour la découverte et la sélection automatique des Web service; 2) Les médiateurs utilisés (usedMediators) par le service Web. 3) La fonctionnalité (capability) décrite par les préconditions, les postconditions et les effets; 4) L'interface est décrite selon deux points vus, à savoir, la chorégraphie (point de vue utilisateur) et l'orchestration (point de vue d'autres fournisseurs de services).
- Les objectifs décrivent les fonctionnalités requises par l'utilisateur que le service Web est censé satisfaire. Un objectif décrit la fonctionnalité, les entrées/sorties, les préconditions et postconditions d'un service Web et les flux de données et de contrôles.
- Les ontologies fournissent la terminologie de référence aux autres éléments de WSMO afin de spécifier le vocabulaire du domaine d'une manière interprétable par les machines.
- Les médiateurs sont utilisés pour résoudre de nombreux problèmes d'incompatibilité, à savoir, les incompatibilités de données, dans le cas où les services Web utilisent différentes terminologies, les incompatibilités de processus lors de la combinaison de services Web et les incompatibilités de protocoles lors de l'établissement des communications.

2.5.2 Approche Bottom-up

La deuxième approche (bottom-up) consiste à annoter les langages de description de services existants avec de l'information sémantique. En ce sens, des références vers des ontologies sont ajoutées aux fichiers de descriptions des services (p.ex. WSDL). L'avantage principal de ce genre de solutions réside dans la facilitation mise à la disposition des fournisseurs de services à l'effet d'adapter la description de leurs services aux annotations proposées. A ce titre, le " Semantic Annotations for WSDL and XML Schema " (SA-WSDL) [52] et le " Web service semantics " (WSDL-S)[2] figurent parmi les langages d'extension supportant l'approche Bottom-up.

2.5.2.1 SAWSDL

Le World Wide Web Consortium propose la recommandation technique désignée par SAWSDL à l'effet d'augmenter l'expressivité des descriptions WSDL par de l'information sémantique [74]. En ce sens, le SAWSDL fournit un ensemble d'attributs d'extension applicable sur les éléments du WSDL pour annoter ses opérations, ses messages d'entrée et de sortie. Trois attributs d'extensibilité sont définis par défaut : l'attribut <sawsdl:modelRefer> permet d'associer n'importe quel élément du WSDL à un ou plusieurs concepts d'ontologies ; les deux attributs <sawsdl:liftingSchemaMapping >

et <sawsdl: loweringSchemaMapping> sont ajoutés aux définitions des types (schema mapping) à l'effet de spécifier, respectivement, les correspondances entre les éléments du message du service Web (schéma des données en XML) et l'information sémantique de l'ontologie (lifting) et vice-versa (lowering).

2.5.2.2 WSDL-S

Le WSDL-S se base sur l'exploitation des éléments d'extensibilité du WSDL [2]. Cette approche augmente le langage WSDL avec plusieurs extensions liées aux opérations et aux messages d'entrée et de sortie des services Web par l'information sémantique [95]. Ces extensions consistent en des références aux concepts décrits dans les ontologies du domaine afin de spécifier la sémantique des messages, des préconditions et des effets. Le WSDL-S vise à fournir une approche "allégée" d'annotation sémantique de services Web.

2.6 Intégration de données basée sur la médiation

Le problème d'intégration de données a suscité beaucoup de projets de recherche. Le but de l'intégration de données est de fournir aux utilisateurs une vue unifiée et une accessibilité transparente sur les sources de données distribuées, hétérogènes et autonomes. Cette section étale un aperçu liés aux approches d'intégration de données ainsi que les bases théoriques y afférentes. Etant donné que l'approche de médiation de données s'inscrit comme une hypothèse de base, dans le cadre de notre travail, ses formalisations théoriques seront explicitées davantage.

2.6.1 Approches d'intégration de données

Selon la localisation des données, les approches d'intégration de données se scindent en deux catégories, à savoir, les entrepôts de données et la médiation de données.

2.6.1.1 Médiation de données

L'approche de médiation de données consiste à définir une interface d'interrogation centralisée entre l'agent (humain ou logiciel) qui pose une requête et l'ensemble des sources de données accessibles et pertinentes pour y répondre [60].

La première proposition d'architecture de médiation a été avancée dans [127]; celle-ci est déclinée en trois couches, comme nous le montre la figure 2.4; le médiateur, l'adaptateurs et les sources.

Au niveau du médiateur, le schéma global fournit un vocabulaire unique (p.ex. une ontologie) pour l'expression des requêtes des utilisateurs. Ce vocabulaire sera utilisé pour la description, de façon homogène et uniforme, des contenus des sources par un ensemble de vues abstraites.

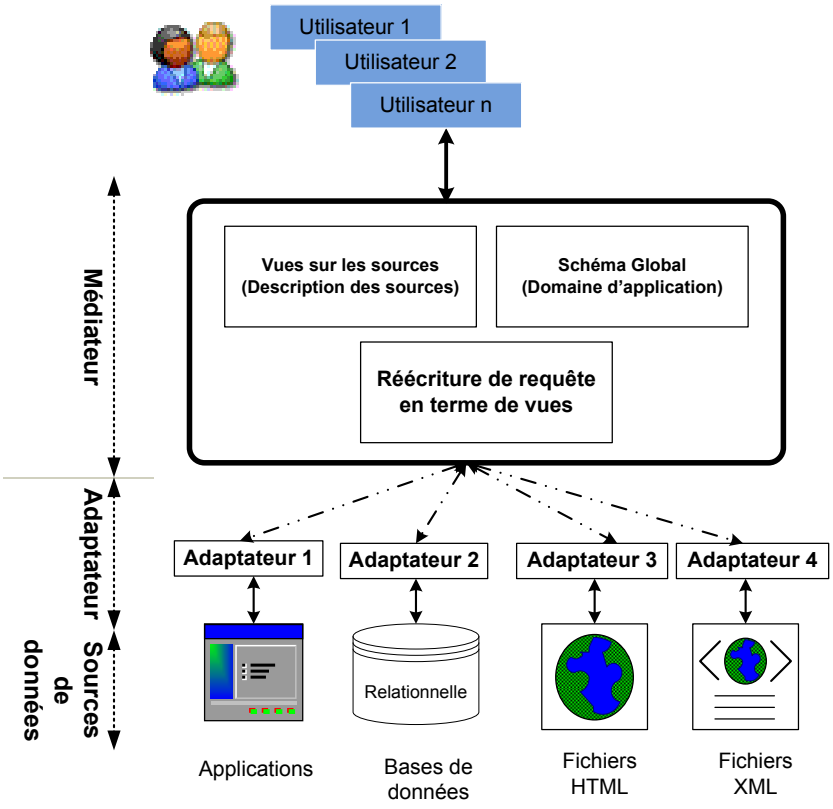

FIGURE 2.4 – Architecture de médiation de données [60].

Les adaptateurs, appelés " wrappers " en anglais, traduisent les requêtes exprimées en termes de vocabulaire du schéma global en des requêtes exprimées dans le langage de requêtes spécifique accepté par chaque source. Aussi, les adaptateurs transforment les réponses aux requêtes en des réponses conformes au schéma global du médiateur.

Pour répondre à la requête, le médiateur réécrit la requête en termes de vues abstraites. On parle alors d'interrogation basée sur les vues et de réécriture de requêtes [82]. Les réponses à la requête posée sont obtenues en évaluant les réécritures de cette requête sur les extensions des vues (les sources de données). Les différentes réponses sont combinées pour former le résultat final.

2.6.1.2 Entrepôt de données

L'approche entrepôts de données consiste à construire une base de données réelle, appelée entrepôt, regroupant les informations pertinentes provenant de différentes sources de données[60]. Les entrepôts de données sont mis en œuvre pour fournir un support pour le processus décisionnel. L'utilisateur pose directement ses requêtes ou lance un traitement sur les données stockées dans l'entrepôt pour effectuer de la fouille de données soit pour générer des rapports statistiques. Les problèmes posés par la construction d'un entrepôt concernent la définition de son schéma, de son peuplement et de sa mise à jour, et ce, en

fonction des différentes sources d'information à partir desquelles il est construit [72].

2.6.2 Approches de médiation de données

Un système de médiation de données peut être défini sous forme d'un triplet $<S,V,M>$ où S désigne le schéma global du système de médiation ; V représente l'ensemble des vues décrivant le contenu des sources de données, et M représente l'ensemble des mises en correspondance ou règles (mapping) définies entre les relations du schéma global S et les relations du schéma local V. En ce sens, on peut distinguer deux approches de médiation de données, et ce, sur la base des approches de définition des règles de mapping "M", à savoir, Global As View (GAV) et Local As View (LAV) [81, 48].

- Dans l'approche GAV, les relations du schéma global sont exprimées en fonction des relations du schéma local (source). L'approche GAV est connue pour être peu flexible dans la mesure où l'ajout de nouvelles sources de données implique la modification du schéma global et la conception de nouveaux adaptateurs ;
- Dans l'approche LAV, les relations du schéma local sont exprimées en fonction des relations du schéma global. L'ajout et la suppression de sources de données n'affectent pas le schéma global. Toutefois, l'approche LAV souffre d'un problème de performance due à la complexité de la réécriture de requêtes en termes de vues définissant les sources. L'approche LAV est conseillée dans le cas où les sources sont nombreuses et mises à jour fréquemment.

On constate sciemment que le contexte de notre thèse assoit l'approche LAV laquelle s'inscrit plus souvent dans le contexte e-santé où l'autonomie des sources de données, la confidentialité des données et l'imprévisibilité des requêtes sont les maitres mots.

2.6.3 La réécriture de requêtes

Dans le contexte d'un système de médiation de type LAV, la réécriture d'une requête consiste à déterminer les sources contributives (les vues) pour l'exécution d'une requête et à utiliser leurs définitions pour reformuler cette requête [28]. Ainsi, la réécriture de requêtes consiste à trouver une requête qui est équivalente ou implique logiquement, la requête de l'utilisateur, mais n'utilise que des vues [60].

Le problème de la réécriture a été étudié pour différentes classes de langages utilisés pour décrire les requêtes et les vues. À cet effet, nous nous situons dans le cas où le problème du calcul des réécritures considère les requêtes et les vues sont formulées sous forme de règles en Datalog [22]. Ce choix est inspiré par la nature des vues et des requêtes manipulées dans le cadre de la présente thèse, à savoir, des requêtes conjonctives à base de règles en présence de contraintes d'intervalle simples (des requêtes de type select-project-join [123]). Les notions de base ayant trait à cette problématique ainsi que les algorithmes

22. Datalog est un langage de requête et de règles pour les bases de données déductives. Il correspond à un sous-ensemble de Prolog.

de réécritures seront présentés subséquemment.

2.6.3.1 Définitions de base

On suppose que les différentes sources de données forment une seule base de données, appelée base de données globale, notée D. Le médiateur chargé du traitement de la requête Q, ne connaît pas le contenu réel de D. Il n'a en réalité qu'une vision partielle du contenu de D à travers les extensions des vues.

Définition 2.1 *La requête conjonctive: Etant donné un schéma global S, une requête conjonctive $Q(x)$ est de la forme $Q(x) \longleftarrow R_1(x_1), ..., R_n(x_n)$ où :*

- *$Q(x)$ est la tête de la règle ou l'entête de la requête et spécifie le schéma de la relation représentant le résultat de la requête;*
- *$R_1(x_1), ..., R_n(x_n)$ est le corps de la règle;*
- *$R(x_i)$ est un atome ou un sous-but de la requête;*
- *R_i est un nom de prédicat de S;*
- *Chaque x_i est une expression de la forme $e_1, ..., em_i$;*
- *Chaque e_j est soit une variable, soit une constante de domaine;*
- *Les variables qui apparaissent dans l'entête de la requête, sont dites libres (distinguished variables) tandis que celles qui n'y apparaissent pas sont dites liées (existential variables).*

Définition 2.2 *Inclusion de requêtes (Query containment) : Le problème qui consiste à trouver une requête Q_2 incluse dans la requête de l'utilisateur Q_1 passe par la vérification de la relation d'inclusion. Ainsi, on ne peut dire que Q_2 est une réécriture de Q_1 que si Q_1 est dite incluse ou contenue dans une requête Q_2. L'inclusion est vérifiée si et seulement si pour chaque base de données D, l'ensemble de résultats de Q_1 est un sous ensemble des résultats de Q_2, qui est dénotée par $Q_1 \subseteq Q_2$.*

Définition 2.3 *Equivalence de requêtes (Query equivalence): Le problème qui consiste à trouver une requête Q_2 équivalente à la requête de l'utilisateur Q_1 passe par la vérification de la relation d'équivalence. L'équivalence entre deux requêtes Q_1 et Q_2 est vérifiée si et seulement si $Q_1 \subseteq Q_2$ et $Q_2 \subseteq Q_1$.*

Définition 2.4 *Inclusion des correspondances (Mapping containment): L'inclusion des correspondances fournit les conditions nécessaires et suffisantes pour tester l'inclusion et l'équivalence de requêtes. Une correspondance φ des variables de Q_1 vers les variables de Q_2 est une inclusion de correspondances si :*

- *Pour chaque sous but dans le corps de Q_2, il existe une correspondance vers un sous but dans le corps de Q_1 ;*
- *Il existe une correspondance entre l'entête de Q_1 et l'entête de Q_2.*

Définition 2.5 *Réécriture de requêtes (Query rewriting): Le problème de réécriture de requêtes peut être énoncé comme suit : une requête Q' est une réécriture d'une requête Q si et seulement si:*

- *Q' réfère uniquement les vues de V tel que V est l'ensemble des vues $V = \{V1, .., Vn\}$;*
- *Q' est incluse dans Q ou Q' équivalente à Q.*

Ce type de réécriture est cependant trop restrictif dans le contexte de la médiation, car dans un environnement inter-organisationnel (p.ex. e-santé), il n'est pas toujours possible de trouver une réécriture équivalente à une requête donnée. En effet, la réécriture équivalente ne peut être vérifiée dans l'hypothèse du monde ouvert (OWA)[23] où les sources de données sont supposées incomplètes et peuvent être indisponibles [62]. Dans un tel contexte, les réécritures équivalentes ne peuvent pas être obtenues puisque les vues sont incomplètes. C'est pourquoi la notion de réécritures maximalement incluses est généralement utilisée à la place des réécritures équivalentes.

Définition 2.6 *Réécriture maximalement incluse (Maximally contained rewriting): Une réécriture Q' exprimée dans le langage L est dite maximalement incluse dans Q :*

- *S'il n'existe pas de réécriture Q'' dans L, tel que $Q' \subseteq Q'' \subseteq Q$ et Q'' n'est pas équivalente à Q' et*
- *$Q' \subseteq Q$*

Les réécritures maximalement incluses sont intéressantes, car elles permettent de trouver le maximum de réponses, à une requête donnée, selon le langage de réécriture considéré. Dans le cas des requêtes conjonctives exprimées en Datalog, la réponse à une requête est l'union de toutes les réécritures maximalement incluses pouvant être trouvées.

2.6.3.2 Algorithmes de réécritures

Un algorithme de réécriture de requêtes requiert en entrée une requête et un ensemble de vues et produit en sortie un ensemble de requêtes sémantiquement équivalentes à celle de l'utilisateur. Il existe deux catégories d'algorithmes de réécritures de requêtes: les algorithmes à base de Tas (bucket) tels que le Bucket et le Minicon [82] et l'algorithme à base de règles inverses [46].

Algorithme Bucket: L'idée principale de l'algorithme Bucket est de trouver les réécritures possibles en considérant chaque sous-but de la requête, en isolation, afin de déterminer les vues contributives pour chaque sous-but [62]. Un sous-but correspond à une relation virtuelle ou prédicat qui apparaît dans l'expression de la requête. Une vue v est dite contributive pour un sous-but g d'une requête Q si les conditions suivantes sont vérifiées :

23. Open World Assumption : Dans cette hypothèse, les sources de données sont supposées incomplètes.

- v contient un sous-but g_1 tel que g peut être unifié (substitué) avec g_1. L'unification est établie sur des sous-buts ayant le même nom en tenant compte de la position des variables. Elle consiste à construire les mappings entre les variables des deux sous-buts;
- Toutes les variables distinguées de Q sont mappées à des variables distinguées de v lors de l'unification de g et g_1.

A cet effet, l'algorithme Bucket procède en deux étapes :

- Construction d'un tas (bucket) pour chaque sous-but de la requête Q. Ce tas contient la définition des sources contributives pour ce sous-but.
- Construction de toutes les requêtes conjonctives possibles, incluses dans la requête de l'utilisateur, en faisant le produit cartésien entres les éléments des tas. Chaque réécriture candidate est une requête conjonctive obtenue en prenant un élément de chaque tas.

Algorithme Minicon: L'algorithme Minicon est un algorithme à base de tas [106]. Son avantage consiste à réduire le nombre de vues non contributives lors de la phase de constructions des tas, appelés MiniCon Descriptor (MCD). A cet effet, les deux étapes de cet algorithme se déclinent comme suit :

- La première étape consiste à chercher l'inclusion des correspondances entre les sous-buts de la requête et ceux des vues. Lorsque l'algorithme MiniCon détecte une correspondance entre un sous-but g de la requête Q et un sous-but g_1 d'une vue v, il examine les variables de jointure de la requête afin de déterminer l'ensemble minimal de sous-buts de V qui doivent être unifiés avec des sous-buts de Q . Autrement dit, l'algorithme vérifie que toutes les jointures de Q sont soit vérifiées par la définition de la vue, soit exprimées sur des variables distinguées de la vue afin d'être satisfaites ultérieurement. Pour chaque vue, l'algorithme note les sous-buts de la requête qui peuvent être couverts par cette vue.
- Une fois les MCD construits, l'algorithme les combine afin d'obtenir les réécritures de la requête. La combinaison de MCD forme une réécriture qui couvre tous les sous but de la requête. En plus, chaque paire de $MCDs$ doivent couvrir des sous-buts disjoints.

Algorithme de règles inversées: L'algorithme de règles inversées (*inverse-rule*), a été mis en œuvre dans le système Infomaster [47]. Le fonctionnement de cet algorithme est basé sur la construction d'un ensemble de règles qui montrent comment construire les tuples du schéma global à partir des tuples des vues [47]. L'avantage de cette approche est la construction des règles laquelle est exécutée en un temps polynomial. Par contre le calcul des résultats de la requête peut être répétitif lesquels ont été accompli pour le calcul des vues [75].

2.7 Conclusion

Dans ce chapitre, nous avons passé en revue de plusieurs notions liées à notre problématique. Cette revue, bien que n'étant pas exhaustive, introduit les éléments nécessaires à l'effet d'aborder les prochains chapitres. En ce sens, ce chapitre vise l'explicitation des hypothèses de base de notre cadre de travail. En premier lieu, les concepts clés liés à la technologie des services Web notamment la notion du service DaaS et les principales motivations derrière son émergence dans le domaine d'e-santé on été exposés. Au même titre, la composition automatique de services basée sur l'usage des technologies des SWS a été présentée. En deuxième lieu, la réécriture de requêtes dans une architecture de médiation de données et les algorithmes de réécritures ont été présentés. Les fondements théoriques présentés jusqu'ici visent à introduire, ensuite, l'assimilation du problème de compositions de services DaaS avec celui de la réécriture de requêtes. Le prochain chapitre s'attèlera à présenter une étude comparative des approches de compositions de services DaaS, notamment celles basées sur le modèle de vues, et de médiation de données lesquelles opèrent dans un environnement e-santé.

Chapitre 3

Etat de l'art

3.1 Introduction

Ce chapitre est dédié à l'examen des travaux ayant trait à notre problématique. Cette revue de l'état de l'art est scindée en trois sections. La section 3.2 exposera les caractéristiques et les standards des données EHR. Aussi, les différentes approches d'intégration de données, de type EHR, basées sur les services seront examinées. La section 3.3, présentera un aperçu sur les travaux liés à l'automatisation des activités liées aux services DaaS décrits par des vues. La section 3.4 dressera un aperçu sur les travaux liés à la médiation de données dans une composition de services.

3.2 Intégration des données EHR: Approche orientée service

Beaucoup de travaux de recherche se sont penchés sur l'intégration des données EHR [7, 37, 56]. Dans ce contexte, les systèmes Synapses [56], ARCHIMED [120], Synex[56] et Pangea-LE [7] figurent parmi les travaux qui proposent des cadres de référence pour l'intégration de données EHR basée sur une architecture de médiation. Ces projets proposent la définition d'un schéma de médiation global (XML) et les vues représentant des fragments de données EHR (XML).

En outre, la synthèse effectuée par [37] sur plus de trois milles papiers, liés à l'intégration des données EHR, publiés entre 1995-2005 met en évidence le fait que les architectures basées sur la technologie des services Web ont vu leur importance s'accroitre. En effet, parmi les 92% des systèmes étudiés, utilisant les navigateurs Web pour déployer leur application, 88% utilisent les services Web pour rendre disponible les données EHR, sur Internet ou Intranet, pour les professionnels de santé. Il ressort de cela que les architectures de médiation et les technologies des services Web sont de plus en plus adoptées à l'effet d'accéder et d'intégrer des données de type EHR.

3.2.1 EHR: Caractéristiques et standards

Les données relatives à l'identification du patient, les médicaments prescrits, les examens de signes vitaux, les rapports d'examens de laboratoire et de radiologie sont autant de données enregistrées dans l'EHR. Telle que spécifiée par la définition citée dans la section 1.2, chaque donnée est générée électroniquement et intégrée directement dans le Système d'information de l'entité santé où l'épisode de soins s'est tenue. L'EHR a une structure complexe pouvant inclure des données issues de 100 à 200 paramètres, tels que la température, la tension artérielle et l'index du poids corporel. Chaque paramètre comporte d'autres spécifications qui offrent une connaissance complète sur "le contexte clinique" (les attributs des données), "l'état" (le contexte d'interprétation) et le protocole suivi (information sur le recueil des données). La figure 3.1 présente les spécifications pouvant être capturées pour la tension artérielle. Un état de l'art et une analyse approfondie des données EHR pourraient être consulté dans [49].

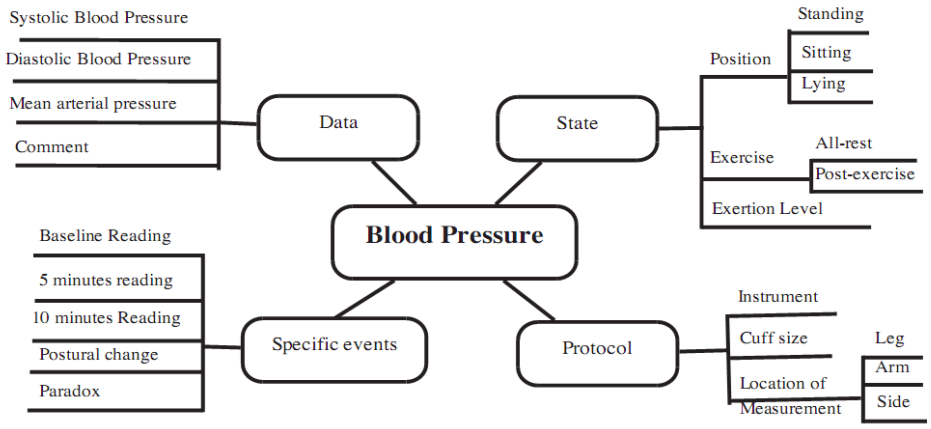

FIGURE 3.1 – Spécification de la tension artérielle [110]

Aussi, les données EHR sont stockées dans différents systèmes sous des formats spécifiques. Cette situation entraine des problèmes d'interopérabilité (hétérogénéités structurelles, sémantiques, etc..), car le sens de ces données pourrait être mésinterprété par les différents systèmes lorsqu'ils sont censés interagir entre eux ou partager des données. À cet effet, plusieurs normes ont été développées dans le but de structurer et de sémantiser les données EHR afin d'assurer des échanges significatifs entre les systèmes dans un environnement e-santé. A ce titre, plusieurs organismes de standardisation ont proposé différents standards de communication et de description des données EHR. Leurs efforts se sont focalisés autour de trois axes principaux, à savoir :

– La structuration des données EHR. Cet aspect implique la spécification des règles de structuration des éléments de données EHR. À titre d'exemple, des documents XML basés sur des documents de validation DTD ou XML Schema sont définis. Ainsi, Clinical Document Architecture (CDA) de Health Level 7 (HL7) [44], openEHR [118]

sont des standards de structuration des données EHR.

- La structuration des flux d'interaction entre systèmes d'e-santé. Cet aspect précise, dans un profil d'intégrations IHE[24], les messages, les acteurs et les événements déclencheurs pour chaque processus e-santé (Radiologie, Laboratoire, etc.). Ainsi, le profil " Cross-Enterprise Document Sharing (XDS) " d'IHE propose comment échangés des documents comprenant des données EHR.
- Le codage du contenu des données EHR. Cet aspect est lié à l'usage des systèmes de terminologies tels que les classifications et les vocabulaires contrôlés (ontologies médicales) pour assigner une sémantique aux données EHR. Ainsi, ICD[25], LOINC[26], SNOMED-CT[27], UMLS[28] figurant parmi les ontologies médicales les plus référencées. Aussi, d'autres aspects liés au codage du contenu se basent sur la structure (lettre de sortie, compte rendu médical, etc.).

Ces standards proposent une normalisation des échanges de données EHR entre systèmes dans un environnement e-santé. Ils se focalisent principalement sur la façon d'accéder et de rendre interopérable les données EHR plutôt que de les standardiser elles-mêmes. En effet, les données EHR sont modélisées et exprimés indépendamment de la façon dont elles sont décrites et stockées dans la source de données sous-jacente. En outre, au vu de la richesse du domaine médicale, les standards de données EHR proposent une nouvelle approche de modélisation de données nommée " modélisation à deux niveaux " [49, 16]. Les deux niveaux correspondent à "un modèle d'information générique" (orienté objet) et "un modèle de connaissance du domaine". Le modèle de connaissance exprime un ensemble de contraintes ((type simple et complexe), la cohérence interne (type, intervalle de valeurs, échelle, unité de mesure, plage), des données de référence (format XML, Ontologie médicale)) sur les instances des classes du modèle générique. A cet effet, pour rendre interopérables les données EHR issues de plusieurs systèmes d'e-santé, cette modélisation à deux niveaux devrait être prise en compte.

3.2.2 Service DaaS EHR

L'introduction de la technologie des services Web est motivée par le besoin d'encapsuler les données EHR lors des échanges entre systèmes e-santé. En effet, cette technologie a été adoptée pour fournir l'accessibilité aux données EHR, tout en préservant l'autonomie. Dans cette section, les services web fournissant des données EHR seront désignés par service DaaS EHR.

En ce sens, avant que le service DaaS retourne les données EHR, plusieurs opérations sont exécutées sur ces données (voir figure 3.2), à savoir, :

24. IHE, "Integrating the Healthcare Enterprise" `http://www.ihe.net`.
25. International Statistical Classification of Diseases and Related Health Problems, http://www.who.int/classifications/icd/en/index.html
26. Logical Observation Identifiers Names and Codes, http://loinc.org/
27. Systematized Nomenclature of Medicine- Clinical Terms, http://www.ihtsdo.org/snomed-ct/
28. Unified Medical Language System, http://www.nlm.nih.gov/research/umls/

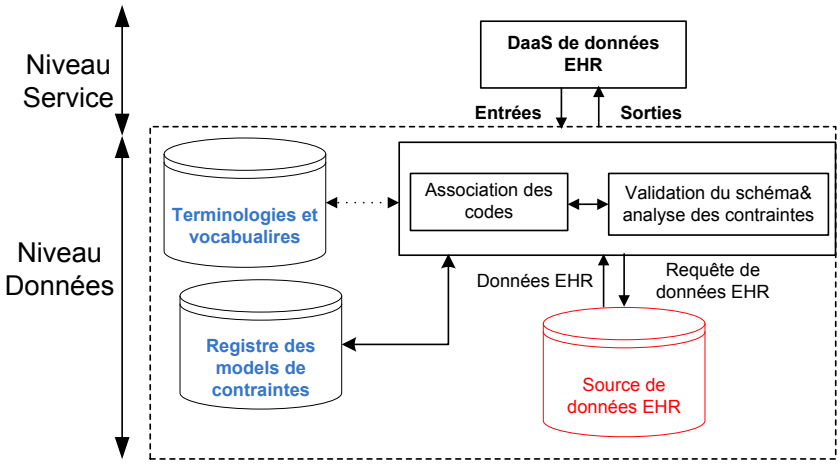

FIGURE 3.2 – Encapsulation des données EHR par un service DaaS

- La récupération des données EHR que le service DaaS est censé fournir à partir du système de gestion des données EHR, par exemple: "Liste des problèmes", "Histoire des maladies", "Administration des médicaments", "Pathologie actuelle"', "Antécédents familiaux", "Antécédents chirurgicaux", "allergies", etc.
- Le codage des données EHR par les standards du contenu assurant le contrôle au niveau type et instance. Ceci est effectué par l'attachement des définitions de données EHR aux concepts d'ontologies médicales [80]. Les données codifiées seront structurées selon un gabarit (Template) de documents médicaux (p.ex. HL7/CDA, etc.);
- Envoi du document crée dans un message SOAP qui encode à la destination appropriée l'entité (application, utilisateur) émettrice de la requête.

Dans ce cadre, plusieurs projets ont proposé l'accessibilité aux données EHR à travers des services Web. En ce sens, le projet ARTEMIS [43, 20] figure parmi les cadres de référence les plus référencés visant l'interopérabilité des données EHR par l'emploi de la technologie de services Web. Les auteurs en question présentent un mécanisme pour la publication, la découverte et l'invocation des services Web, sémantiquement annotés, dans un environnement de partage de données EHR de type Point-a-Point. Cependant, le projet ARTEMIS ne fournit pas de moyens pour composer automatiquement les services Web afin de répondre aux requêtes complexes des utilisateurs. En outre, les hétérogénéités au niveau des valeurs de données n'ont pas été abordées dans ce travail.

3.2.3 Composition de services DaaS (Données EHR)

Une fois la publication des données EHR est rendue effective à travers les services DaaS EHR, un grand nombre de travaux de recherches se sont penchés sur le problème de la composition de service DaaS EHR [9, 99, 129, 69]. Ces travaux exploitent le concept

de services Web sans pour autant spécifier sa nature, à savoir, sans effet ou avec effet.

3.2.3.1 Composition dirigée par les Workflow

Les auteurs [9] proposent un processus de modélisation des services Web appliqués dans un environnement hospitalier. Ils modélisent la composition de services Web en utilisant BPEL. Cette composition semi-automatique se base sur la définition de profils d'intégration IHE. Les interactions entre services Web EHR sont orchestrées par un médiateur. Aussi, dans [64], une mise en œuvre d'un système de composition semi-automatique et dynamique de services Web EHR pour une unité de soins intensifs est présentée. La composition se base sur la description sémantique des services (OWL-S). Ces deux approches proposent des solutions de composition de service Web EHR semi-automatique, car elle se base sur la définition des gabarits de compositions (Template) et l'intervention de l'utilisateur pour la sélection de services. En outre, ces travaux sont portés uniquement sur des compositions de services orientées sur le modèle de workflow c.-à-d. une composition consacrée à l'exécution des processus métiers relavant du domaine hospitalier (p.ex admission d'un patient, etc.). En plus, ces approches sont limitées au système hospitalier où l'hétérogénéité des données n'a pas la même ampleur par rapport à celle qui subsiste dans le système e-santé.

3.2.3.2 Composition dirigée par les requêtes

Les auteurs dans [25] ont proposé un médiateur pour l'intégration des données dans les systèmes e-santé (IBHIS : Integration Broker for Healthcare Systems). Le médiateur proposé est basé sur une AOS pour interroger les données issues des systèmes autonomes. À cet effet, le médiateur utilise les descriptions sémantiques des services stockés dans un registre de services. Ce registre inclut une fonctionnalité de mise en correspondance (matchmaker) ente la description sémantique des services Web "OWL-S" et les requêtes des utilisateurs. Le médiateur décompose les requêtes des utilisateurs, interagit avec le module de mise en correspondance pour y répondre et retourne les résultats aux utilisateurs. Toutefois, cette approche souffre de quelques limites, en premier lieu, la composition est basée sur la spécification des requêtes les plus fréquentes. En deuxième lieu, aucune indication quant au traitement des hétérogénéités sémantiques provoquées par l'usage de différents standards EHR n'a été fournie. Aussi, l'utilisation d'OWL-S comme langage pour annoter les services Web n'offre pas la possibilité de spécifier explicitement la relation sémantique entre les entrées et les sorties d'un service Web.

3.2.4 Synthèse

Il ressort de cette succincte revue que les données EHR se conforment à plusieurs standards liés aux aspects de structuration et de contenu. Le besoin de standardisation est né de la nécessité de partager, d'échanger et d'intégrer ces données entre les différents

systèmes d'e-santé. Cependant, l'adoption des technologies des services web ne garantit pas à elles seules l'intégration effective des données pour plusieurs raisons. En premier lieu, la manipulation des données EHR prônée dans ces approches est " centrée document ". Par contre, un usage "centré données" est plus prépondérant dans le contexte e-santé. A ce titre, l'extraction sélective, basée sur des critères, de données EHR est une pratique primordiale dans le domaine e-santé.

En deuxième lieu, selon [80], des problèmes d'interopérabilité surgiront lors de l'intégration de données EHR malgré qu'elles sont rendues accessibles par des services Web. En effet, faire correspondre les données EHR aux concepts des ontologies médicales, qui souvent se chevauchent sémantiquement, entraîne toujours des problèmes d'interopérabilité. Aussi, au vu de la volatilité et l'évolution des concepts médicaux, jusqu'à présent aucune ontologie unique et complète du domaine médical n'est proposée [7, 101]. Ainsi, étant donné que chaque standard dispose de son propre modèle d'information et du domaine de connaissance qui lui correspond, nous nous focaliserons seulement, dans le cadre de cette thèse, sur les caractéristiques du domaine de connaissance. En effet, les données EHR retournées par les services DaaS EHR se conforment à différentes contraintes définies dans le modèle de connaissances. Ces différentes contraintes seront regroupées sous le nom de "contexte". Chaque contexte spécifie les terminologies, les codifications et les présentations de données retournées par le service DaaS. En ce sens, les données EHR seront caractérisées par l'association fréquente avec des concepts ontologiques, définis dans l'une des ontologies médicales, et des contraintes sémantiques et structurelles décrivant ce que fourni le service DaaS EHR.

3.3 Services DaaS et modèles de description a base de vues

Plusieurs travaux ont proposé des approches pour automatiser les activités liées aux services DaaS (sélection, découverte, composition) [14, 132, 124, 13]. L'élément commun entre ces travaux est l'usage du modèle de vues pour annoter la description des services DaaS. Dans ce contexte, l'architecture de médiation de données de type LAV est adoptée où le schéma de médiation est une ontologie de domaine [29]. L'ontologie en question est utilisée, à la fois, pour décrire les vues et exprimer les requêtes. Cependant, ces approches diffèrent sur plusieurs aspects qui peuvent être résumés comme suit :

- Les modalités d'annotation et de publication des services DaaS;
- L'étendu de la solution d'automatisation des activités liées aux services DaaS;
- Les algorithmes proposés pour l'automatisation de la découverte et la composition ;

Ces différents aspects sont détaillés dans les prochaines sous-sections.

29. Les expressions "ontologie de domaine" et "ontologie de médiation" seront utilisées pour désigner la même notion

3.3.1 Annotation et publication des services DaaS

L'approche de [14] modélise les services DaaS sous forme d'une VRP exprimée en terme d'une ontologie de médiation. Ces vues sont incorporées dans les fichiers WSDL des services DaaS selon les spécifications "WSDL-S". Lors de la publication de chaque service DaaS, le fournisseur aura à définir et à insérer la VRP capturant la sémantique de chaque opération du service dans le fichier de description correspondant. Les services DaaS sont ordonnés, dans le registre, selon les concepts de l'ontologie de médiation mentionnés dans leurs vues correspondantes. Ce mécanisme permettant de faire correspondre les services DaaS à leur sémantique est basé sur le *tModel keys* et les catégories des entrées du registre. En outre, cette approche propose d'opérer un prétraitement sur les VRP qui consiste en l'ajout des contraintes sémantiques décrites dans l'ontologie de médiation et la spécification des fonctions de Skolem [30] sur les propriétés de chaque concept (classe). La requête de services est exprimée en SPARQL.

Selon [132] les services DaaS sont modélisés sous forme d'une règle de mapping (requête SPARQL) entre le schéma de la source de données (à publier) et l'ontologie de médiation. Cette règle est insérée dans la partie "Process Model" selon les spécifications "OWL-S". Les fichiers de descriptions " OWL-S" sont inversement indexés dans le registre. L'indexation en question est basée sur les concepts de l'ontologie de médiation où chaque concept est associé aux descriptions des services DaaS dans lesquelles il est mentionné.

D'après [124], les services DaaS désignés par DPS (Data Providing Service) sont modélisés par des vues RDF où le schéma local de chaque source de données est associé à des concepts d'une ontologie de médiation exprimée en "OWL". La vue en question est assimilée au "Service Profile" selon les spécifications "OWL-S". Ainsi, chaque service DaaS est décrit par un fichier OWL-S où la vue remplace les spécifications IOPE du "Service Profile ". Toute requête R est de la forme d'un quadruplet $(I_R, O_R, C_{I_R}, C_{O_R})$ où I_R et O_R représentent respectivement les variables d'entrée et de sortie et C_{I_R} et C_{O_R} représenter un ensemble de triplet RDF décrivant les contraintes définies sur les paramètres d'entrée et de sortie de la requête. Lors de la publication du fichier de description du service, un prétraitement lui est opéré afin d'alimenter la table des services dans le registre. Cette table renseigne sur le degré de correspondance (matching) calculé pour chaque concept de la vue par rapport à celui de l'ontologie de médiation. Le prétraitement procède à l'extraction de chaque vue RDF : 1) la classe $Types_{P_{DPS}}$, appartenant à l'ensemble des classes citées dans les vues RDF du service $ClassView$; 2) le profile P_{DPS} : l'ensemble des vues RDF d'un service DaaS. Subséquemment, ces informations seront exploitées pour calculer la matrice de distance sémantique D entre les concepts représentant les entrées et les sorties du service DaaS par rapport aux concepts de l'ontologie de médiation. En résultat, l'ensemble $< P_{DPS}, Types_{P_{DPS}}, D >$ représente la table des services.

Les services DaaS dans l'approche proposée par [131], sont représentés par des Qua-

30. La fonction Skolem permet de spécifier pour chaque classe d'une vue RDF la propriété de donnée qui peut être utilisée pour identifier une instance de la classe (clé primaire)

druplets $< I, O, P(I, L), E(I, L, O) >$ où I et O sont respectivement les ensembles de variables d'entrée et de sortie des services; P et E représentent des formules bien formées de la logique du premier ordre qui décrivent les préconditions et les effets des services; L est l'ensemble des variables existentielles. P pourrait être vide mais pas E. La notion de l'effet considérée dans cette approche est interprétée différemment, car elle correspond à celle de la vue et non à l'effet au sens compris dans les services SaaS.

3.3.2 Découverte et sélection des services DaaS

L'approche proposée par [14] opère la découverte d'un service DaaS lors de la première étape de composition de services. La découverte en question est assimilée à la résolution d'un problème de recouvrement de graphes (graphe de la requête et celui de la vue). Par conséquent, un service DaaS est jugé pertinent, pour répondre partiellement ou complètement à une requête, s'il inclut dans la vue qui le décrit un ou plusieurs concepts ontologiques (classe, propriété d'objet) utilisés dans la requête.

L'algorithme de découverte proposé par [132] se base sur la recherche de la correspondance sémantique entre les paramètres de sortie des services DaaS avec ceux que la requête requiert. Le médiateur retourne l'ensemble des services DaaS pouvant satisfaire la requête. L'approche traite séparément le cas où le système trouve un seul service et le cas où une composition de services DaaS peut répondre à la requête.

L'algorithme de découverte proposé par [124] exploite la table des services DaaS stockée dans le registre. La découverte se base sur la définition de la requête R et sur l'ensemble $< P_{DPS}, Types_{P_{DPS}}, D >$ pour retourner les services pouvant correspondre aux sorties demandés ($T_{O_R} \in O_R$). Pour chaque service, l'algorithme calcule le degré de couverture et il retourne comme résultat l'ensemble de services découverts. Les services retournés n'exprimant aucun lien entre les entrées et les sorties (par rapport à la requête) et ceux qui ne tiennent pas compte des contraintes (de la requête) ne seront pas pris en considération.

3.3.3 Composition des services DaaS

L'approche de composition de services DaaS proposée par [14] est basée sur l'algorithme de réécriture des requêtes (Bucket). Lors de la première étape, l'algorithme détermine si les services DaaS découverts peuvent être sélectionnés pour répondre à la requête ou pas. L'algorithme construit individuellement pour chaque concepts de la requête Q une liste L_i laquelle contiendra les vues (services DaaS) ayant le même concept dans leur définition. Lors de la deuxième phase, l'algorithme procède à la génération de toutes les combinaisons qui résulte du produit cartésien des divers vues des listes L_i pour couvrir la requête. Les combinaisons générées seront vérifiées afin de statuer sur leur executabilité laquelle est prononcée sur la base des correspondances des patterns d'accès entre services participants à une composition.

L'algorithme proposé par [132] pour générer la composition de services DaaS est inspiré des approches de planification des graphes avec une recherche à rebours (planning graph).

Les auteurs [131] proposent une approche de composition automatique des services DaaS basée sur l'algorithme de réécriture de requêtes MINICON [106]. L'algorithme proposé consiste en deux phases.

La première phase consiste en deux étapes. La première étape consiste à former les MCDs (MiniCon Descriptor). Chaque MCD représente un mapping partiel F des sous but ordinaires de Q vers des sous buts ordinaire de V. Les vues V représentent les services DaaS et Q représente la requête de service. Dans la deuxième étape, l'algorithme de réécriture considère seulement les sous but E nommés " sous but ordinaire " et ignore ceux de P nommés " sous but de comparaison " et procède à la génération des combinaisons entre les différents membres des MCD. La deuxième phase consiste en deux étapes, procède à la vérification de la validité des combinaisons précédemment générées pour produire un plan de composition $p(G, \varphi, CMP)$ où G est un graphe orienté acyclique (DAG), φ est une fonction décrivant les mapping des flux de données dans G et CMP est un ensemble d'expressions arithmétiques de comparaison. Ainsi, la première étape vérifie les entrées des services composants les réécritures, par la création du graphe G. Dans le graphe G les sous buts ordinaires de Q sont considérés comme des nœuds et les variables partagées entre deux nœuds reliés par un arc représentant le mapping entre les entrées du service cibles et les sorties du service source par φ. La seconde étape vérifie si la précondition de chaque service composant, après insertion de sous but de comparaison, est satisfaite. Les sous buts de comparaison dans la réécriture comportent CMP directement.

L'approche proposée par [13] pour la composition de services DaaS est basée sur un algorithme de réécriture sémantique de requêtes. La réécriture sémantique est une réécriture de requêtes qui s'opère sur des entités exprimées en langage $CARIN$ [31][54]. Cette approche se focalise uniquement sur l'étape d'ordonnancement des services DaaS et ne traite pas leur découverte. La composition en question se base sur la correspondance (matching) entre les services Web décrits sous forme de vues et la requête de l'utilisateur. La composition des services DaaS consiste en deux étapes.

La première étape procède à la conversion des services Web en une conjonction de vues exprimées en terme d'une ontologie de médiation. Ainsi, l'algorithme retourne les vues qui contiennent une conjonction T_{ij} qui est une sous classe ou super classe de la conjonction T_{qk}. Chaque conjonction T_{qk} dénote la K^ieme conjonction du corps de la requête Q. Les vues T_{ij} seront stockées dans l'ensemble Set_K.

La deuxième étape consiste à appliquer la réécriture sémantique pour trouver toutes les réécritures possibles. Les réécritures initiales sont générées par la combinaison de vues sélectionnées dans les ensembles Set_K. La réécriture sémantique implique l'application des restrictions internes et externes définies par l'ontologie de médiation. Finalement,

31. Le langage $CARIN$ est un langage, de la famille des langages logiques hybrides, qui combine un langage de règles logiques avec une logique de description.

l'algorithme proposé procède à la génération des compositions des services web à partir des réécritures trouvées en filtrant toutes les solutions dépourvues de sens.

3.3.4 Synthèse

L'état de l'art présenté dans cette section a étalé les approches d'automatisation des activités liées aux services DaaS basées sur le modèle de vues. Dans ce contexte, notre choix s'est porté sur les travaux les plus référencés suivant [14, 132, 84, 13, 119, 124].

L'approche de [84] propose un cadre de référence pour répondre aux requêtes via la génération d'un plan conjonctif incluant les entrées et les sorties des services Web annotés par des expressions Datalog. Dans [119], la réécriture de requêtes basée sur un algorithme de combinaison de règles inversées et un filtrage des tuples sont utilisés pour générer une composition de services. Cependant, ces deux travaux ne prennent pas en considération la sémantique des services durant la phase de mise en correspondance (matching entre services et requête). En effet, la mise en correspondance employée dans ces approches se limite à la correspondance de type des paramètres uniquement. La raison pour laquelle ces deux approches n'ont pas été développées dans le présent état de l'art.

L'approche de composition proposée par [13] se base sur un algorithme de réécriture sémantique de requêtes lequel applique un raisonnement basé sur un langage de Logique de description. Cependant, les mises en correspondances entre les descriptions des services et la requête sont de type un à un. Cela est dû au fait que la définition de la vue ne comprend que des prédicats de classes et non de propriétés. Cette méthode simplifie l'algorithme de réécriture au détriment de l'expressivité. Pour cela, la description des services est jugée pauvre et manque de précision.

L'approche de [132] propose une cadre de référence basée sur l'ontologie de médiation pour la publication et la composition des services DaaS. Ils proposent l'annotation de la description des services DaaS, par une requête SPARQL, selon les spécifications "OWL-S". La composition de services DaaS est générée par un algorithme de planification de graphes. Cependant, leur algorithme n'indique aucune information quant aux performances qu'il peut atteindre. En plus, aucune indication n'a été fournie quant à la prise en charge des hétérogénéités des données lors de l'exécution de la composition.

L'approche de [124], dédiée à la découverte principalement, fournit une caractérisation des conditions de mise en correspondance entre les services DaaS et la requête. Cette approche propose un algorithme de mise en correspondance basé sur le calcule du degré de couverture. Cependant, elle n'explicite pas les relations entre les entrées et les sorties des services DaaS ce qui entraine des erreurs liées aux résultats des correspondances.

L'approche de [14] se base sur l'extension de la description du fichier de description du DaaS (WSDL) par la VRP. Chaque VRP est prétraitée afin d'optimiser le temps de calcul requis par le raisonnement lors de la mise en correspondance entre les VRP et la requête. L'algorithme de réécriture proposé permet à la fois de découvrir les services pertinents et de les combiner pour en faire des compositions valides et exécutables. Cependant,

le problème d'hétérogénéité des données surgissant lors de l'exécution des compositions générées n'a pas été évoqué. Aussi, des problèmes d'optimisation ont été soulevés au niveau de la deuxième phase de réécriture de requêtes.

En conclusion, les différentes approches présentées font usage d'un modèle de vues pour annoter les services DaaS à l'effet d'automatiser leur découverte, leur composition ou les deux à la fois. Ces approches ne couvrent pas complètement le cycle de vie des compositions (de la découverte jusqu'à la composition) sauf pour celle de [14]. Aussi, aucune approche ne considère les conflits sémantiques surgissant lors de l'ordonnancement des services dans une composition. Cette limite est due au fait que ces approches considèrent que l'ontologie de médiation complète et minimale. Ces aspects sont difficiles à réunir dans un environnement interorganisationnel notamment celui d'e-santé.

3.4 Médiation dans une composition de services Web

Dans le contexte des interactions entre services Web des incompatibilités peuvent surgir. Elles peuvent concerner, à la fois, les propriétés fonctionnelles, non fonctionnelles et les données échangées entre services Web [78]. En ce sens, la médiation intervient pour résoudre les conflits à l'effet d'assurer l'interopérabilité entre les services Web. Les termes incompatibilités, conflit et hétérogénéité peuvent être utilisés de façon interchangeable. Eu égard aux différents types de conflit, plusieurs niveaux de médiation sont proposés [96, 78]. Par exemple, [26] proposent trois niveaux de médiation, à savoir : la médiation de données, la médiation de fonctionnalités et la médiation de processus. Dans le contexte de la présente thèse, la médiation de données est le problème auquel nous faisons face plus qu'aux deux autres niveaux. En effet, la médiation fonctionnelle assure la correspondance entre la fonctionnalité requise par l'utilisateur (requête de services) et les services Web fournis par le système. Cet aspect est pris en charge dans le cadre des propositions suscitées [14, 132, 124]. Aussi, la médiation de processus se focalise sur la résolution des problèmes surgissant au niveau du comportement des interactions entre services Web selon un protocole métier (ordre d'invocation des services). Cependant, les compositions de services DaaS comprennent des services qui déclinent un comportement en deux étapes, recevoir un certain nombre de paramètres en entrée et retournes en résultat des données en sortie. Pour cela, l'ordre des interactions dans le cas des services DaaS est sans intérêt particulier à l'exception que les entrées doivent précéder les sorties.

3.4.1 Médiation de données

Le problème de médiation de données à une longue histoire initiée par la communié des bases de données et reprise par la communauté des services Web [34]. Dans le contexte de composition de services, la médiation de données peut être envisagée selon trois niveaux, syntaxique, structurelle et sémantique [96]. Nous assumons que les conflits (hétérogénéités)

au niveau syntaxique sont résolus dans le contexte des services Web (faisant usage d'une syntaxe commune, XML).

Le niveau structurel se réfère à la médiation entre conflits liés à l'implémentation des services Web eux-mêmes, c'est-à-dire, lié à la structure, la valeur ou le format des messages d'entrée/sortie. Ainsi, les différences au niveau du schéma empêchent les échanges de données du moment que les données devraient être fusionnées ou scindées en différentes parties. Donc la médiation de à ce niveau n'est effective que lors de l'exécution de la composition des services et lors de leur invocation. Le niveau sémantique se réfère à la résolution des conflits entre les représentations sémantiques des services dans une composition. La sémantique en question est mise en évidence à travers l'attachement des paramètres de services aux concepts pouvant être équivalents ou similaires lesquels sont issus d'une ontologie de médiation et l'interprétation correcte des données devrait être effective et manipulable par la machine. La médiation au niveau sémantique vise principalement le support de la découverte et de la sélection des services.

La médiation de données dans le cadre de la composition automatique de services pourrait être effectuée en combinant à la fois les deux niveaux, à savoir, sémantique et structurelle. En effet, il est possible de combiner un médiateur structurel au médiateur sémantique si les annotations sémantiques assertent que deux paramètres d'un message type, d'un service, sont équivalents et similaires s'ils pointent vers le même concept d'une ontologie de médiation. Pour cela, l'examen fait sur la médiation de données dans une composition de services DaaS, est lié à la médiation structurelle et sémantique de données, tout en évoquant les fonctions de conversion ou de transformation intervenant dans la résolution des conflits.

3.4.2 Médiation structurelle de données

L'une des sources de conflits entre services est la structure des données échangées. La médiation structurelle consiste en l'adaptation des schémas des paramètres de sortie et d'entrée des services subséquents dans une composition. Les opérations de médiation structurelle sont la fusion (merge) et le fractionnement (split). A titre d'exemple, la figure 3.3 mentionne un cas de conflit structurel où deux paramètres de services DaaS $S_2(I_2, O_2)$ et $S_3(I_3, O_3)$ sont structurés différemment. À ce titre, une fois la correspondance entre les paramètres au niveau de l'ontologie de médiation est statuée, et ce, à travers l'exécution d'un algorithme de composition, (par exemple, $S_2.O_2$ est équivalent au $S_3.I_3$), les deux services sont jugés composables. Cependant, il est nécessaire d'affirmer, au niveau instance, que les valeurs échangées correspondent aussi, c.-à-d. la valeur o_2 du tuple s_2 retournée par S_2 est identique à la valeur i_3 du tuple s_3 retournée par S_3.

En effet, tel que présenté dans la figure 3.3, le paramètre en question (BPR value), structuré différemment, est fourni par une interface et consommé par une autre. Ainsi, lorsque le service S_2 fournit le paramètre en sortie $S_2.O_2$, le service S_3 n'accepte pas la valeurs qui lui a été fournit car ça ne correspond pas à la définition de son paramètre

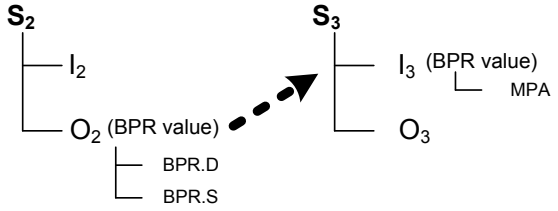

FIGURE 3.3 – Médiation structurelle de données

d'entrée.

Dans ces cas, les conflits pourront être résolus par une série de transformations du paramètre original jusqu'à l'interface requise par le service. Cette opération garantit que le message résultat est valide structurellement, mais ce n'est pas pour autant que ce dernier n'a pas omis des informations obligatoires. Un des inconvénients de ce mécanisme est le problème d'optimisation du nombre et de l'ordre des transformations à opérer en vue de transformer la structure d'un paramètre en une autre. Ce problème est connu sous le nom du " calcul de la distance d'édition entre arbres " [21].

3.4.3 Médiation sémantique de données

Plusieurs approches ont été proposées pour remédier aux conflits sémantiques surgissant lors de la composition des services Web. Les travaux liés à la médiation sémantique des services web se basent sur une ou plusieurs ontologies de domaine pour vérifier qu'un paramètre de sortie d'un service est subsumé par un autre paramètre d'entrée d'un autre service. La relation de subsomption est conçue ou inférée à partir des connaissances contenues dans les ontologies de médiation. Pour cela, [117, 53, 109, 100] ont proposé des solutions pour remédier aux problèmes d'hétérogénéités des données entre les services Web, et ce, à travers la spécification des relations de correspondances directes entre les messages de deux services. Plus précisément, ces approches se basent, à la fois, sur l'annotation sémantique des paramètres d'entrée/sortie des services avec des classes de l'ontologie de médiation et l'implémentation des scripts de lifting et de lowering entre les messages des services (types de données) et les concepts de l'ontologie de médiation.

Aussi, [34] proposent des modèles d'annotation sous forme de règle de mapping exprimé en WSML (langage WSMO) et décrivant des mapping entre ontologies. Ces médiateurs sont en mesure de créer et d'exécuter des règles de mapping pour la découverte, l'invocation et la composition de services Web.

Dans notre contexte, ces solutions sont jugées rigides et limitées en terme d'expressivité, car leurs modèles d'annotation ne permettent pas d'exprimer des concepts avec des variables pour décrire les contraintes liées aux entrées et aux sorties des services DaaS.

Aussi, plusieurs travaux se sont focalisés sur l'interprétation des données échangées par les services en interaction. Initié par [55], une structure arborescente des objets sémantiques est proposée pour fixer l'interprétation des données, nommée "contexte".

Ainsi, les données seront représentées par leur contexte. Les approches présentées dans [83] et [97] se basent sur l'utilisation de la notion du contexte pour assurer la médiation sémantique dans la composition des services Web. Précisément, ils proposent une extension de l'ontologie du domaine par une ontologie légère incluant un petit ensemble de concepts génériques pour capturer le contexte.

Cependant, la représentation du contexte sous forme d'arborescence d'objets sémantiques est intéressante pour la médiation des données néanmoins les objets sémantiques requièrent des outils spécifiques pour leurs manipulations. En plus, ces approches sont limitées par leur modèle de contexte qui ne permet d'assurer que des correspondances simples entre paramètres sémantiquement équivalents (prix, unité, etc.). Aussi, les fonctions de transformation proposées, généralement codées en XSLT, assurant la conversion d'un contexte à l'autre sont difficiles à écrire, à maintenir et limitées en termes de réutilisabilité dans une solution d'intégration de données à grande échelle. En ce sens, pour conférer plus de maintenabilité et de réutilisabilité aux fonctions de transformations, [96] considère lesdites fonctions comme des services. Cependant, aucune indication quant à la manipulation automatique de ces services n'a été évoquée.

Dans le même sillage, l'approche de [42] introduit une nouvelle notion à savoir, l'espace de médiation. L'espace de médiation décrit le contexte des données comme un espace multidimensionnel où chaque aspect du contexte est représenté par un point dans un vecteur. Les espaces de médiation offrent une propriété intéressante telle que la conversion des données à travers l'usage des opérateurs de la géométrie euclidienne. Cependant, cette approche montre rapidement ses limites quand il s'agit de décrire les données non numériques.

3.4.4 Synthèse

En résumé, la majorité des approches revues liées à la médiation de données dans une composition de service ne traite pas cet aspect en sa totalité. Pour cela, la médiation requiert une intervention des utilisateurs avertis soit pour détecter les conflits, soit pour programmer, sélectionner ou composer les fonctions de résolution des conflits. Condition faite, des approches orientées service pour la résolution des conflits proposée par [96, 34]. Toutefois, la première approche [96] ne développe pas assez l'automatisation des activités liées aux services de médiation tandis que la deuxième approche est assez rigide en terme d'expressivité hormis le fait qu'elle utilise un langage spécifique de WSMO (WSML). Pour cela, l'automatisation de détection et de la résolution des conflits dans une composition de DaaS ne seraient rendues possibles qu'à travers l'apport d'une sémantique plus précise. Cette sémantique permet de spécifier à la fois la nature du conflit, son emplacement (dans la composition de services DaaS) et le service de médiation, simple ou composite, approprié pour le résoudre.

3.5 Conclusion

Dans ce chapitre, nous avons présenté les différentes approches faisant usage d'un modèle de vues, pour annoter les services DaaS, à l'effet d'automatiser leurs activités (découverte, sélection, composition). Les contraintes liées à la manipulation des données EHR encapsulées par des services DaaS sont évoquées. Ces contraintes sont imposées par l'usage des différents standards ayant pour but de rendre les données EHR interopérables.

En outre, la notion du contexte a été abordée, car elle permet de fournir la sémantique nécessaire pour preciser l'interprétation à assigner aux données échangées entre services dans une composition. En ce sens, l'enrichissement du modèle DaaS, basé sur le model de vues, par la notion du contexte permet de fournir un moyen pour détecter les conflits lors de la composition de services DaaS. Aussi, enrichir la description des services assurant la transformation des données d'un contexte à l'autre permettra l'automatisation de la sélection et de la composition des services de médiation.

Cette étape de médiation devra être insérée dans un cadre de référence de composition de services DaaS, basé sur la réécriture de requêtes, tel que celui proposé par [14]. Ceci permettra la mise en œuvre d'un cadre de référence couvrant le cycle de vie complet d'une composition de service DaaS.

Chapitre 4

Approche proposée : Architectures et modèles

Le présent chapitre sera consacré à l'explicitation de notre approche de détection et de résolution des conflits sémantiques dans une composition de services DaaS. En premier lieu, la section 4.1 étalera les fondements de notre approche en terme de formalismes adoptés et de processus de traitement des requêtes. En deuxième lieu, l'architecture de référence liée à notre approche sera présentée dans la section 4.2. En troisième lieu, les modèles de base ayant trait aux notions manipulées dès le début de la présente thèse seront définis dans la section 4.3. En quatrième lieu, la section 4.4 dressera les modèles proposés à l'effet d'automatiser la détection et à la résolution des conflits dans les compositions de services DaaS.

4.1 Présentation générale de l'approche

Notre contribution vise à fournir un cadre de référence pour détecter et résoudre les conflits sémantiques dans les compositions de services DaaS générées selon l'approche de réécriture de requêtes. En ce sens, nous jugeons qu'il serait opportun de nous inspirer des systèmes de composition de services DaaS proposés dans ce cadre et cités dans la section 3.3.3. Plus précisément, nous allons nous baser sur le Système de Composition de services DaaS (SCD) proposé dans [14, 5]. Le SCD, en question, fait office d'un orchestrateur de services DaaS. La figure 4.1 illustre notre approche qui vise principalement à offrir la possibilité aux professionnels de santé de générer automatiquement des compositions de services DaaS sans conflits satisfaisant leurs requêtes.

Pour ce faire, on considère l'enrichissement, à la fois, des descriptions des services DaaS et des requêtes. Cet enrichissement, consiste en la notion du contexte, permet d'expliciter les conflits potentiels requérant une médiation de données lors d'une composition de services DaaS. Aussi, cet enrichissement est à l'origine de la spécification des services de médiation appropriés pour la résolution des conflits. Avant d'expliciter notre approche, ses

éléments fondamentaux et son intégration dans le processus de traitement de la requête seront exposés.

4.1.1 Eléments fondamentaux

L'enrichissement des descriptions des services DaaS (VRP) par le contexte offre un moyen pour détecter les conflits. À cet effet, les VRP exprimées en termes de l'Ontologie de Domaine (OD), seront étendues par la notion du contexte exprimée en termes d'"Ontologies des Aspects Conflictuels (OAC)". Ces ontologies étendent les concepts de l'OD et de cette façon le schéma de médiation sera constitué de deux niveaux ontologiques. La nouvelle description des services DaaS sera nommée "Vue RDF Paramétrée Contextualisée (VRPC) ".

Faisant suite à la spécification des aspects conflictuels, les services de médiation assurant la résolution des conflits seront décrits sous forme de règles de transformations. Dans ce qui suit, les choix fondamentaux liées aux formalismes de modélisation des services et à l'assimilation de la médiation au servies seront exposés.

4.1.1.1 Architecture d'intégration de données basée sur les services DaaS

Les services DaaS et de médiation seront respectivement annotés par les VRPC et les règles de transformations appropriées. Les deux catégories de services sont stockées dans deux registres distincts (voir Figure 4.1).

Lors de la publication des différents services, leurs fournisseurs, du domaine e-santé, auront comme tâche d'annoter les fichiers de description WSDL des services en question. Concernant les aspects conflictuels, ils sont identifiés par des experts du domaine et décrits en un ensemble d'OAC.

4.1.1.2 Modèle de services de médiation

On considère les fonctions de résolution des conflits assurant la médiation de données comme des services. Ainsi, leur manipulation sera basée sur les principes des Architectures Orientées Services. Ce choix vise à la fois , de maximiser la maintenabilité et la réutilisation de ces fonctions et de garder l'aspect de médiation orthogonal à celui du métier assuré par les services DaaS.

Le model de règles proposé pour décrire les services de médiation exprime, en mode déclaratif, les fonctions de transformation entre deux concepts appartenant au même aspect. Chaque aspect conflictuel consiste en les interprétations de données susceptibles de faire objet de médiation. La notion de l'aspect est inspirée de la Programmation Orientée Aspect (AOP) [102], ainsi nous considérons les conflits comme des aspects orthogonaux aux données manipulées par les différents services constituant une composition.

4.1.1.3 Annotations à base de graphes RDF

Nous avons adopté le modèle d'annotation des services DaaS basé sur les graphes RDF. Ce choix est dicté par la capacité expressive conférée aux vues RDF. Cette expressivité offre la possibilité d'expliciter la relation sémantique entre les paramètres d'entrée et de sortie d'un service DaaS. En effet, selon [66] OWL-S est décrit avec OWL lequel impose des restrictions syntaxiques, ce qui rend complexe l'expression de la relation liant les paramètres d'entrée et de sortie d'un service DaaS. Aussi, selon [121] OWL n'exprime pas de variables qui sont requises pour spécifier des contraintes et des caractéristiques des concepts. Du point de vue d'OWL, de telles expressions sont traitées comme des littéraux du langage SWRL (Semantic Web Rule Language) ou du langage KIF (Knowledge Interchange Format).

Dans le même sillage, WSMO fourni une classe "capability" pour décrire la fonctionnalité du service ainsi que la relation liant les paramètres d'entrée et de sortie du service, et ce, par le partage de variables au même titre que les vues RDF. Cependant, aucun résultat n'a été publié quant à la décidabilité [32] de la mise en correspondance (matching) entre services ou entre service et requête de services décrits en WSMO.

Aussi, le modèle de description des services de médiation se base sur les graphes RDF. Toutefois, ce modèle ne décrit par une vue, mais une règle de transformations. En effet, la seule relation qui lie les paramètres d'entrée et de sortie d'un service de médiation est une relation de transformation de contextes semblable à une relation de "convertir-à". La description des services de médiation comme une correspondance entre deux graphes RDF permet de décrire au mieux la transformation que le service de médiation exécute. Aussi, ce choix est motivé par le fait qu'il est difficile, avec OWL-S ou WSMO, de décrire les instances des concepts utilisés pour caractériser le contexte.

4.1.2 Traitement de requêtes

Le processus de composition des services DaaS est entamé une fois que l'utilisateur spécifie la requête SPARQL [33] en terme, à la fois, de l'Ontologie de Domaine et des Ontologies des Aspects Conflictuels (voir l'étape 1 de la Figure 4.1).

Une fois la requête formulée via l'interface du Système de Composition des services DaaS (SCD), la découverte des services DaaS pouvant être combinés pour répondre à la requête sera entamée (voir l'étape 2 de la Figure 4.1). Cette tâche est assurée par le localisateur de services du SCD qui exploite la description des services DaaS publiés dans le registre correspondant. Il s'ensuit que les services découverts seront combinés pour former des compositions valides. Le SCD utilise le même principe de l'algorithme de

32. Un problème est dit décidable s'il existe un algorithme, pour sa résolution, qui termine en un nombre fini d'étapes

33. Nous adoptons SPARQL: http://www.w3.org/TR/rdf-sparql- comme langage de requête de facto le langage de requêtes pour le Web sémantique.

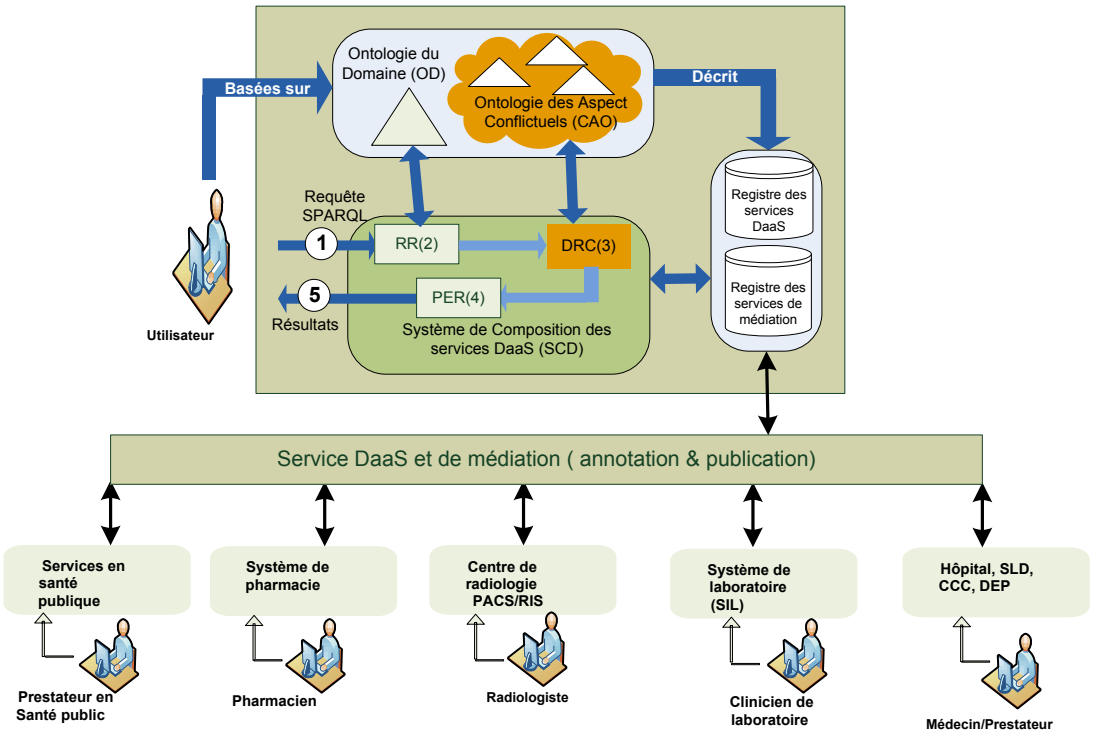

FIGURE 4.1 – Présentation générale de l'approche proposée

Réécriture de Requêtes (RR) proposé par [14]. Cet algorithme sera utilisé moyennant des améliorations et des adaptations qui seront dressées dans le chapitre 5.

À ce niveau, l'algorithme de détection et de résolution de conflits, que nous proposons, exploite le contexte afin de vérifier l'existence des conflits dans chaque composition générée (voir l'étape 3 de la Figure 4.1). Le contexte concerne à la fois les services DaaS sélectionnés et la requête. En effet, les conflits peuvent survenir à plusieurs emplacements dans une composition de services DaaS, à savoir, entre :

- Les services successifs dans une composition de services DaaS;
- Les résultats retournés par la composition et ceux requis par la requête;
- Les paramètres d'entrée de la composition et les contraintes spécifiées par la requête;

Selon l'existence ou l'absence de conflits dans chaque composition, l'algorithme de DRC insère automatiquement les services ou les compositions de services de médiation appropriés pour résoudre les conflits sémantiques.

L'insertion des services de médiation dans les compositions de services DaaS permettra de fournir des compositions sans conflits au module d'Exécution du Plan de la Requête (EPR) du SCD. L'EPR procède à la description des flux de données et de contrôles (voir l'étape 4 de la Figure 4.1). Le plan sera exécuté et les données seront retournées à l'utilisateur (voir l'étape 5 de la Figure 4.1). Il faut noter que le processus du traitement de la requête est entièrement automatisé.

4.2 Architecture de référence

Dans cette section, nous allons présenter l'architecture de référence sur laquelle se base notre approche. Cette architecture est structurée en quatre niveaux, comme indiqué dans la figure 4.2.

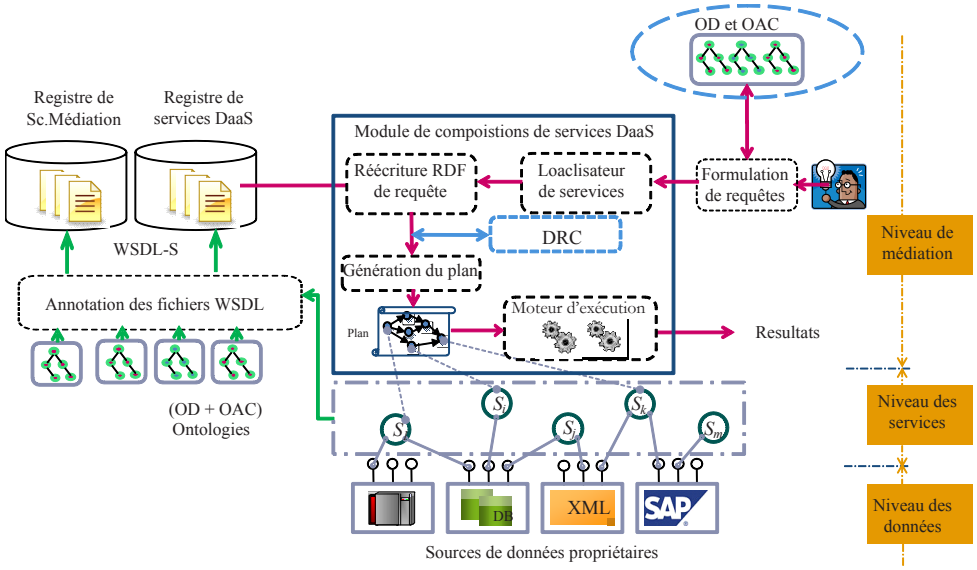

FIGURE 4.2 – Architecture de référence

4.2.1 Niveau des données

Le niveau le plus bas de l'architecture contient les données stockées dans différentes sources. Les sources peuvent être des bases de données ou des documents XML incluant, les deux, des données EHR. Ces sources préservent, pour chaque patient, la trace des actes de soins effectués au cours d'une ou de plusieurs épisodes de soins. Aussi, ce niveau peut inclure des ressources documentaires et terminologiques fournissant des moyens pour la recherche et le partage des informations.

Une source de données est caractérisée par sa localisation, le type de données qu'elle gère, le format des résultats qu'elle retourne et les possibilités d'interrogation qu'elle met à la disposition des utilisateurs. Dans le cadre de la présente thèse, les données EHR sont rendues accessibles à travers les services DaaS.

4.2.2 Niveau des services

Ce niveau fournit deux catégories de services. La première catégorie comprend les services DaaS fournissant des données EHR. La deuxième catégorie comprend les services assurant des fonctionnalités de résolution de conflits dans une composition de services DaaS.

- *Services DaaS* : Deux types de services DaaS peuvent être publiés. Ils sont classés selon le format des résultats qu'ils retournent. Les services DaaS fournissant des fragments de données EHR (maladies, symptômes, médicament, historique et autres) et les services DaaS fournissant des documents cliniques conforment à des modèles standardisés (Template de résumé de sortie, de compte rendu médical, etc.);
- *Services de médiation*: Les services de médiations assurent une fonctionnalité orthogonale à celle assurée par les services DaaS. Ils assurent des fonctions de transformations lesquelles peuvent être scindées en deux types. Le premier type concerne les transformations opérées sur les instances des paramètres des services DaaS, telle que la transformation d'un paramètre exprimé en terme une ontologie médicale vers un autre paramètre exprimé en terme d'une autre ontologie médicale (p.ex. UMLS Terminology Services [34]). Le deuxième type consiste en les fonctions de transformations structurelles, telle que l'extraction des données d'un paramètre d'un service DaaS complexe pour le passer à un autre paramètre d'un service subséquent dans une composition donnée.

Les fichiers de description (WSDL) des services DaaS et de médiation sont publiés dans deux registres séparés. Le registre des services DaaS inclut un ensemble de descriptions de services annotées par des VRPC exprimées en termes de l'OD et de l'OAC. Cependant, le registre des services de médiation inclut un ensemble de descriptions de services annotées par des règles de transformation exprimées en termes d'OAC.

4.2.3 Niveau de médiation

Le niveau de médiation inclut l'ontologie de médiation et le Système de Composition des Services DaaS.

4.2.3.1 Ontologie de médiation

L'ontologie de médiation comprend tous les concepts et les relations définis dans le domaine médical notamment le dossier électronique de santé (EHR). L'ontologie de médiation sera utilisée pour annoter les différents services (DaaS et médiation) et constitue la base sur laquelle les requêtes des utilisateurs sont exprimées. Ainsi, afin de concevoir une ontologie de médiation, à la fois, complète et minimale, nous proposons de scinder l'espace conceptuel en deux niveaux, à savoir, l'Ontologie du Domaine (OD) et l'Ontologie des Aspects Conflictuels (OAC).

- *Ontologie du Domaine*: L'OD définit les concepts génériques et leurs relations couvrant le domaine médical. Les concepts définis, à ce niveau, ne doivent pas faire l'objet de conflits entre les différents fournisseurs de services DaaS. En effet, ils doivent être basiques et partagés tels que les concepts " patient ", " examen de signe vital ", " lecture de tension artérielle (BPR) ", etc. À titre d'exemple, l'OD

34. https://uts.nlm.nih.gov/home.html

exprime que la lecture BPR est désignée par un code et une mesure. Toutefois, aucune connaissance n'est spécifiée quant à la terminologie employée pour spécifier le code en question ou l'unité de mesure pratiquée (ces propriétés seront spécifiées en utilisant l'OAC par la suite). L'OD fournit une base pour décrire la liaison sémantique entre les paramètres d'entrée et de sortie des services DaaS. La description du service DaaS sera exploitée au cours de la phase de découverte et de composition des services DaaS.

- *Ontologie des Aspects Conflictuels*: L'OAC est un ensemble d'ontologies légères crée principalement pour enrichir la description des concepts génériques décrits au niveau de l'OD. Cet enrichissement à pour but de mettre en évidence la connaissance nécessaire à la détection d'éventuels conflits sémantiques survenant lors de toute interaction entre services DaaS. En effet, les OAC permettent de décrire les différents aspects conflictuels caractérisant le contexte de chaque paramètre d'un service DaaS. Par exemple, l'OAC stipule que le code exprimant la désignation de la "tension artérielle (BPR)" soit exprimé en "LOINC" ou en "SNOMED".

4.2.3.2 Système de Composition de services DaaS (SCD)

Le Système de Composition de services DaaS (SCD) est le deuxième composant du niveau de médiation qui consiste en quatre (04) modules, à savoir, le module de sélection de services DaaS; le module de Réécriture de Requêtes (RR); le module de Détection et de Résolution des Conflits (DRC) et le module d'Exécution du Plan de la Requête (EPR).

La fonction principale du SCD est d'assurer la composition des services DaaS pour satisfaire la requête de l'utilisateur. Il faut mentionner que notre approche vise la concrétisation du module (DRC) lequel fait défaut aux approches de compositions de services DaaS basées sur la réécriture de requêtes (voir la section 3.3.3).

- *Module de sélection de services DaaS*: Ce module reçoit le résultat de l'analyse de la requête de l'utilisateur via l'interface fournie par le SCD. L'analyse de la requête à pour but d'extraire les concepts de l'OD mentionnés dans la requête. Ces concepts seront utilisés par ledit module afin de découvrir le ou les services pouvant répondre à la requête de l'utilisateur.
- *Module de réécriture de requêtes*: Ce module reçoit les VRP (sans spécification de contextes) des services découverts jugés pertinents pour contribuer à la réponse de la requête. L'algorithme de réécriture procède en deux étapes à la génération des compositions répondant à la requête de l'utilisateur. Lors de la première étape, un classement des services sera opéré sur la base des correspondances vérifiées entre les concepts ontologiques (classe, propriété d'objets) mentionnés dans les VRP, des services DaaS découverts, et ceux de la requête. Lors de la deuxième étape, l'algorithme procède à la génération des combinaisons de services DaaS sur la base de la table de correspondance. Ces combinaisons seront testées pour en garder que celles qui sont valides et qui seront nommées par la suite les compositions de services DaaS.

- *Module de détection et de résolution des conflits*: Ce module reçoit toutes les compositions générées lors de la RR pour vérifier l'existence des conflits. Une fois le ou les conflits détectés, ils seront enregistrés dans l'ensemble des objets conflictuels. Sur la base des informations fournies par cet ensemble et des descriptions des services de médiation, l'algorithme de DRC procède à la sélection d'un service ou à la composition de services de médiation appropriés pour résoudre les conflits. Subséquemment, il les insère dans toutes les compositions là où le conflit a été détecté. À l'issue, l'ensemble des compositions sera subdivisé en trois sous ensembles selon le nombre de conflits résolus, à savoir les compositions complètement exécutables, les compositions partiellement exécutables et celles qui ne le sont pas.
- *Module d'exécution*: Ce module reçoit les compositions de services DaaS complètement exécutables afin de constituer le plan de composition de la requête. L'exécution du plan retournera les résultats de la requête à l'utilisateur.

4.2.4 Niveau Interface

Le but de ce niveau est de fournir une interface pour l'utilisateur lui permettant d'effectuer des requêtes et recevoir leurs résultats. Les requêtes des utilisateurs sont exprimées en langage SPARQL sur les ontologies OD et OAC.

4.3 Modèles de base

Dans cette section nous présenterons la formalisation des concepts manipulés le long des chapitres précédents, notamment l'Ontologie du Domaine (OD), les Vues RDF Paramétrées (VRP) et le modèle de requêtes conjonctives (Q). Ces modèles constituent la base sur laquelle les modèles proposés se reposent.

4.3.1 Ontologie du Domaine (OD)

L'Ontologie du Domaine (OD) est une conceptualisation spécifique d'un domaine donné, dans notre cas c'est celui du dossier électronique de santé (EHR). L'ontologie spécifie des contraintes sur la structure et le contenu de la connaissance du domaine. Dans notre contexte, l'OD est exprimée en RDFS. Les concepts de l'ontologie du domaine seront préfixes par "OD" comme nom d'espace (namespace).

Définition 4.1 *Ontologie du Domaine est un sextuple* $< C, D, OP, DP, SC, SP >$ *où:*

- C *est un ensemble de classes;*
- D *est un ensemble de types de données;*
- OP *est un ensemble de propriétés d'objet. Chaque propriété objet a un ensemble d'origine (domaine) et un ensemble d'arrivée (range) dans* C*; les classes sont reliées par des relations non taxonomiques.*

- *DP est un ensemble de propriétés de données. Chaque propriété de données a un ensemble d'origine (domain) dans C et un ensemble d'arrivée (range) dans D;*
- *SC est une relation dans $C \times C$, représentant une relation de sous-classe;*
- *SP est une relation dans $(OP \times OP) \cup (DP \times DP)$ représentant une relation de sous-propriété entre des propriétés homogènes.*

La figure 4.3 illustre une représentation d'une partie de l'OD ayant trait à l'examen des signes vitaux capturé par l'EHR. Cette ontologie décrit le fait que chaque patient a effectué des examens de signes vitaux comprenant un examen de tension artérielle, de pulsion cardiaque, etc. La lecture de la tension artérielle (classe "BPReading") est liée au type de données " BPRcode " par la propriété "DO:HasBPRCode" caractérisant le code employé pour designer l'examen en question. Chaque patient peut avoir plusieurs examens effectués où la lecture de tension artérielle fait partie. Dans la figure 4.3, les concepts du domaine (p.ex. Patient, BPRreading, etc.) sont représentés sous une forme ovale et les propriétés de données sous forme de rectangle. Ces concepts sont reliés par des propriétés d'objet. Les concepts sont liés aux types de données par les propriétés de données.

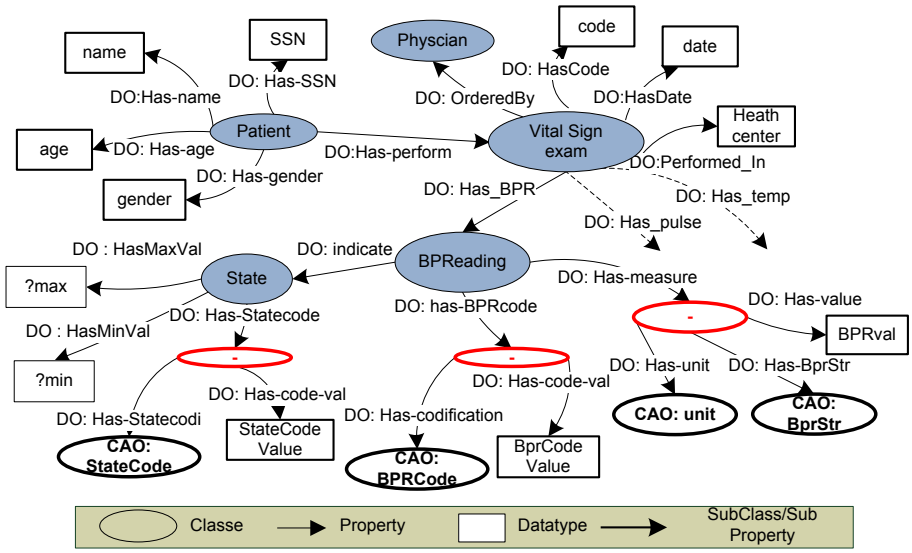

FIGURE 4.3 – Ontologie du Domaine (OD)

L'OD offre la possibilité de greffer des extensions en vue spécifier les aspects conflictuels d'une propriété de données ou d'une classe via les Ontologies des Aspects Conflictuels (OAC). Par exemple, la classe "BPReading" est liée au type de données " BPRcode " exprimé dans une ontologie médicale spécifiée par l'usage de la classe "CAO:BPRCode". Ainsi, notre conception permet d'enrichir l'OD sans pour autant y'modifier la structure. Les OAC seront détaillées dans la sous-section 4.4.1.

4.3.2 Vue RDF Paramétrée (VRP)

Les travaux de [14, 132] ont proposé un modèle de services DaaS sous forme d'une Vue RDF paramétrée exprimée en terme d'une OD. A ce titre, leur formalisation sera adoptée dans le cadre de la présente thèse.

Définition 4.2 *Vue RDF Paramétrée modélise le DaaS "S_j" sous la forme d'un un prédicat exprimé sur OD ayant la forme suivante, en notation Datalog,* $S_j(\$X_j, ?Y_j) : - < G_j(X_j, Y_j, Z_j), Co_j >$ *où*

- X_j *est l'ensemble des variables dénotant les paramètres d'entrée nécessaires pour l'invocation de* S_j*. Ces variables sont nommées variables d'entrée.*
- Y_j *est l'ensemble de variables dénotant les littéraux retournés par l'invocation de "S_j". Ces variables sont nommées variables de sortie. Les variables d'entrée et de sortie constituent les variables distinguées ou libres.*
- $G_j(X_j, Y_j, Z_j)$ *représente la relation sémantique reliant les variables d'entrée et de sortie.* Z_j *est un ensemble de variables existentielles ou liées reliant les ensembles de variables* X_j *et* Y_j*.* G_j *correspond à un ensemble de triplets RDF où chaque triplet RDF a la forme (sujet.propriété.object);*
- $Co_j = \{Co_{j_1}, ..., Co_{j_n}\}$ *est un ensemble de contraintes imposé sur les variables* X_j*,* Y_j *ou* Z_j*. Les contraintes ont la forme « θconstant » où* $\theta \in \{>, \geq, \leq, <\}$

La figure 4.4.(a) expose un aperçu sur la VRP du service DaaS "S_2" présenté dans l'exemple de motivation (voir la section 1.3). En outre, la VRP peut être vue comme une requête SPARQL où les paramètres d'entrée et de sortie sont précédés respectivement par "$" et par "?".

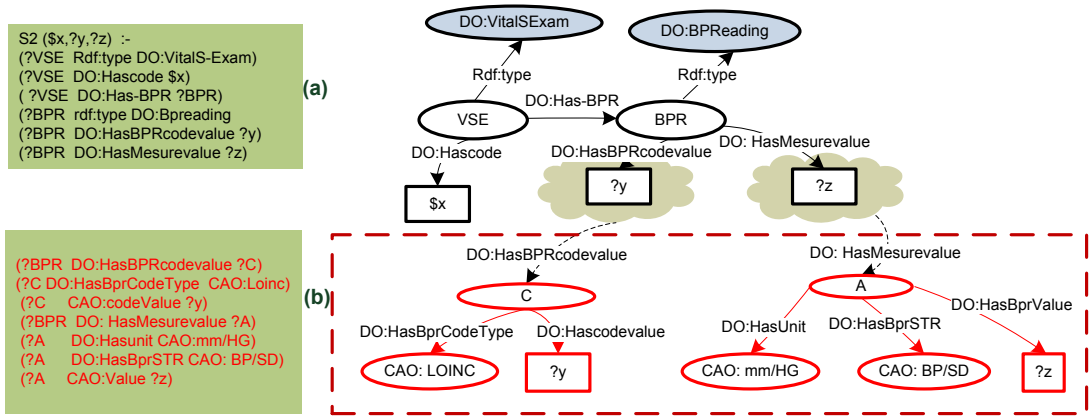

FIGURE 4.4 – (a)VRP du service DaaS "S_2",(b)Extension de la VRP en rectangle pointillé.

Chaque VRP est caractérisée par un patron d'accès. Ce patron définit pour chaque vue les paramètres faisant office d'entrée et ceux faisant office de sortie. Par exemple, le patron d'accès du service "S_2" est $(\$x, ?y, ?z)$. D'autres services peuvent avoir la même signature (paramètres, relation entre paramètres) que "S_2" mais différents patrons d'accès.

4.3.3 Requêtes conjonctives

Nous considérons les requêtes conjonctives comme une formalisation pour abstraire les requêtes SPARQL sur les données du Web sémantique [24]. La requête en question est exprimée sur l'OD en utilisant SPARQL.

Définition 4.3 *Requête conjonctive "Q" a la forme : $Q(X)$:-$< G(X,Y), Co_q >$ où :*

- *$Q(X)$ est la tête de Q représentant le résultat de la requête;*
- *$G(X,Y)$ est le corps de Q, il contient un ensemble de triplets RDF. Chaque triplet RDF a la forme (sujet.propriété.object); X et Y sont respectivement des variables distinguées (libres) et existentielles (liées), X et Y sont les sujets et les objets dans les triplets RDF;*
- *$Co_q = \{Co_{q_1}, ..., Co_{q_n}\}$ est un ensemble de contraintes exprimé sur les variables X et Y.*

Étant donné l'ontologie du domaine de la figure 4.3, la figure 4.5.(a) présente la représentation graphiques de la requête "Q_1" mentionnée dans l'exemple de motivation (voir la section 1.3). La même requête est formulée concrètement en SPARQL comme indiqué sur la figure 4.5.(a).

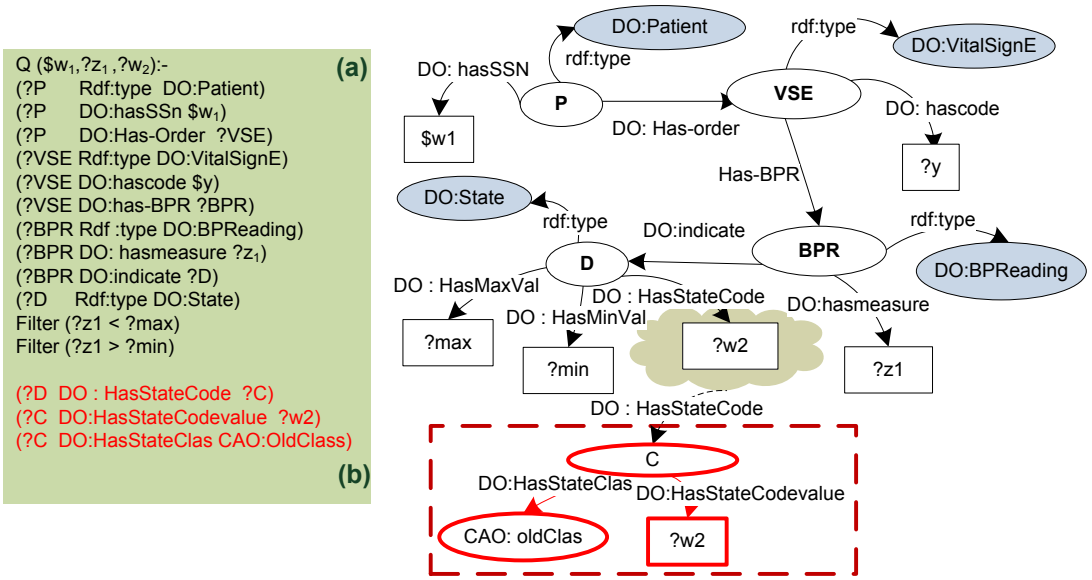

FIGURE 4.5 – (a)Requête "Q",(b) Requête "Q" étendue par le contexte présenté en pintoillé.

La requête est représentée par deux types de nœuds. Les nœuds classes correspondent aux classes de l'OD, lesquelles sont liées par des propriétés d'objet. Dans la figure 4.5.(a) , les nœuds "P", "VSE" et "BPR" représentent à la fois des nœuds de classes et des variables existentielles dans la requête. Les nœuds littéraux sont des types de données. Les nœuds "w_1","y","z_1","w_2","min" et "max" représentent des nœuds littéraux qui correspondent aux variables existentielles et distinguées dans la requête.

4.3.4 Composition de services DaaS

Au sens de la présente thèse, toute composition de services DaaS "cs" est générée par la réécriture de requêtes. La composition "cs" est modélisée sous forme d'un graphe de dépendances. Nous définissons le graphe de dépendances "cs" par l'ensemble des services participant dans "cs" et de leurs dépendances. Dans ce graphe, les nœuds représentent les services et les arcs représentent les dépendances d'exécution.

First(cs) représente la racine du graphe ou le premier service de "cs", $Last(cs)$ représente le dernier service dans "cs". Il y'a lieu de mentionner que chaque composition ne peut avoir qu'un seul First(cs) mais peut avoir plusieurs $Last(cs)$. Cette limitation est due au fait que les algorithmes de compositions, basés sur la réécriture de requêtes, procèdent à la construction d'une composition à partir d'un seul service sélectionné dans l'un des tas (bucket).

Etant donné que les services DaaS ne provoquent pas d'effet (stateless) donc la composition "cs" requiert uniquement un ensemble de paramètres d'entrée, fourni par la requête Q, pour être invoquée et retourne les résultats requis par la requête. Nous entendons par "CS" l'ensemble des compositions générées par l'algorithme de réécriture de requêtes du SCD. Ces compositions requirent des vérifications quant à la présence de conflits sémantiques, auquel cas leur résolution s'impose à l'effet d'assurer l'executabilité des dites compositions.

Définition 4.4 *Composition de services DaaS est un graphe acyclique orienté* $G = (S; O/I)$ *où:*

- S *est un ensemble de nœuds* $S_1, S_2, ..., S_n$. *Chaque nœud représente un service DaaS défini par le couple* $S_K = (E_K; S_K)$ *où* E_K *et* S_K *sont respectivement les variables d'entrée et de sortie. Une variable est un paramètre ou une instance d'un paramètre.*
- $O/I \leftarrow S \times S$ *est un ensemble ordonné de couples de services ou arc* (s_i, s_j). *Un arc* (s_i, s_j) *entre deux services DaaSs* s_i *et* s_j *représentent une dépendance d'exécution (c-à-d* s_i *doit être exécuté avant* s_j*).*

Dans notre contexte chaque arc est associé à une opération de Sortie/Entrée (O/I). Cette notion sera détaillée dans la sous-section 4.4.3.

4.4 Modèles proposés

Nous présentons dans cette section les modèles proposés dans le cadre de notre approche laquelle se base principalement sur la notion du contexte et des services de médiation. Cette approche vise la détection et la résolution des conflits sémantiques dans une composition de services DaaS. En ce sens, l'OAC permettant de décrire le contexte, les conflits sémantiques et le modèle de description des services de médiation seront présentés.

4.4.1 Ontologie des Aspects Conflictuels

L'Ontologie des Aspects conflictuels (OAC) est une famille d'ontologies légères spécifiées en RDFS. Les OAC étendent les concepts de l'OD avec une structure taxonomique décrivant les différents conflits sémantiques survenant entre services DaaS dans une composition "cs". Cette structuration de conflits s'est inspirée de la classification des différents problèmes d'incompatibilité entre services Web proposée par [100]. L'OAC varie en étendue, du conflit générique au conflit spécifique. Les concepts de l'ontologie des aspects conflictuels sont préfixés par "OAC" comme nom d'espace (namespace).

Définition 4.5 *Ontologie des Aspects Conflictuels est un 3-tuple* $< AC_g, AC_i, \tau >$, *où :*

- *"*AC_g*" est un ensemble de classes représentant différents aspects conflictuels liés aux concepts de l'OD. Chaque classe "*ac_g*", appartenant à "*AC_g*", a une superclasse et un ensemble de sous-classes. Chaque classe "*ac_g*" porte une désignation ou un nom représentant un aspect conflictuel;*
- *"*AC_i*" est un ensemble distinct de classes instanceables ayant une seule superclasse dans "*AC_g*". Par définition, "*ac_i*" n'a pas de sous-classes.*
- *"*τ*" représente les relations entre concepts similaires ou de fratrie (sibling relationship) entre concepts "*AC_i*" ou "*AC_g*". Les relations entre les classes appartenant à "*AC_g*" sont de type disjoint. Par contre, les relations entre les classes de "*AC_i*", d'un "*ac_g*" donné, sont des relations entre pairs (Peer relationship) qui indiquent des conflits entre des données ayant la même sémantique.*

Comme indiqué sur la figure 4.6, la classe "`CAO:Measurement-Unit`" représente un aspect conflictuel lié à l'unité de mesure employée par une valeur d'un examen. Cette valeur est représentée comme un type de données lié à un concept de L'OD (p. ex. la classe BPReading). Aussi, les unités de mesure "$mmHg$ " et "$cmHg$" sont deux classes instanceables appartenant à l'aspect conflictuel unité de mesure des lectures de la tension artérielle représentée par la classe "`CAO:BPR-Unit`". Cette classe est une sous-classe de l'aspect générique "`CAO:Measurement-Unit`" qui comprend d'autres aspects liés à la mesure des liquides, des solides, etc.

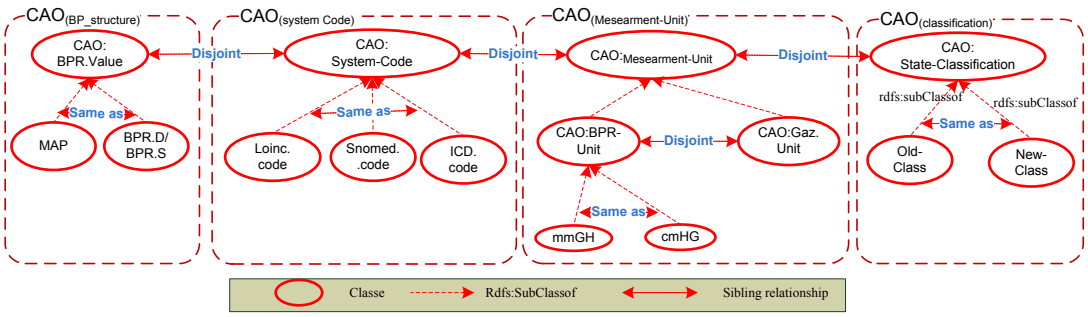

FIGURE 4.6 – Ontologie des Aspects Conflictuels (OAC)

4.4.2 Modèle du contexte

Le contexte comme connaissance permettant la détection des conflits sémantiques a été adopté dans le cadre de plusieurs travaux [55, 112]. Ces travaux proposent un modèle du contexte sous forme d'un ensemble de meta-attributs valorisés. En ce sens, la notion du contexte sera exploitée pour enrichir la description de la requête et des services DaaS (VRP). En effet, la VRP exprimée en terme d'OD ne fournit pas une sémantique explicite sur les paramètres d'entrée et de sortie d'un service DaaS.

Pour cela, nous proposons l'extension de la VRP par le contexte afin d'expliciter les concepts de l'OD en terme d'OAC. Précisément, le contexte sera utilisé pour spécifier les variables distinguées et les contraintes, à la fois, de la requête et de la VRP. A cet effet, dans le cas d'un service DaaS, il sera introduit, dans la VRP, comme une annotation sur les variables apparaissant au niveau des paramètres d'entrée et de sortie du service. Ce mécanisme d'extension appelé le "couronnement de la requête" (*adornment query*) a été proposé dans [29]. De cette manière, il est possible d'avoir des variantes du même service DaaS où chacun est publié sous différents contextes.

Définition 4.6 *Le contexte "Ct" est un ensemble de couples dimension-valeur identifié de manière unique ayant la forme suivante :* $\{(D_i, V_i)|i \in [1, m]\}$ *où* $D_i \in AC_g$ *et* $V_i \in AC_i$.

Définition 4.7 *La relation entre deux contextes "Ct" et "Ct'" où* $Ct = \{(D_i, V_i)\}$ *et* $Ct' = \{(D_i, V'_i)\}$, *existe si et seulement si pour chaque* $i \in \{1..n\}$ *il y'a* $v_i R_{Di} v'_i$ *où* $R_{Di} \in \tau_{AC_i}$.

A titre d'exemple, le contexte $Ct_{MU} = \{(CAO : Unit, mm/HG)\}$ indique que le l'unité de mesure employée est "mmHg", laquelle indication sera exploitée pour annoter les variables distinguées d'un service DaaS ou d'une requête. Ainsi, assigner le contexte "Ct_{MU}" au type de données "BPRvalue" du concept "$OD : BPR$" permet de préciser l'unité de mesure que le type de données "BPRvalue" emploie. La relation entre contextes permet de mettre en évidence la nature, à la fois, du conflit entre deux services DaaS et la fonction de résolution du conflit requise.

4.4.2.1 Service DaaS contextualisé

Le service DaaS contextualisé est de la forme $S_j(\$X_j, ?Y_j) : - < V_{DO} > | < X :: Ct_{X_j}, Y :: Ct_{Y_j} >$ où

- V_{DO} est une VPR de S_j exprimée uniquement en terme d'OD;
- Ct_{X_j} et Ct_{Y_j} sont respectivement les contextes des paramètres d'entrée et de sortie du service DaaS exprimés sur l'OAC. Ct_{X_j} et Ct_{Y_j} sont décrits par un ensemble de triplets RDF sur l'OAC et mentionnant chacun le tuple $< AC_g, AC_i >$.

Par exemple, la figure 4.4.(b) illustre la graphe RDF ainsi que les triplets RDF ajoutés à la PRV du service DaaS "S_2" lequel est mentionné dans l'exemple de motivation (voir la section 1.3).

4.4.2.2 Requête contextualisée

Une requête contextualisée CQ à la forme suivante : $CQ(X) : - < Q(X)|X :: Ct_{qx}, Co_q :: Ct_{qc} >$ où :

- $Q(X)$ est une requête exprimée en terme d'OD;
- Ct_{qx} est le contexte de la variable distinguée X exprimé en terme d'OAC ;
- Ct_{qc} est le contexte des contraintes de la requête Co_q exprimé en terme d'OAC .

A titre d'illustration, la figure 4.5 expose la requête mentionnée dans l'exemple de motivation (voir section 1.3). Cette requête est entendue par le contexte $Ct_q = ((X :: (CAO : OldClassification)), (Co_q :: \emptyset))$. Le contexte "$Ct_q$" exprime le fait que l'état des valeurs "BPR" qui devrait être retourné à l'utilisateur doit être présenté selon l'ancienne classification. Aussi, aucune spécification quant aux contraintes exprimées par la requête (ensemble vide).

4.4.3 Conflit sémantique

En général, les conflits entre les services Web peuvent être scindés en deux types, à savoir, les conflits dépendants du contexte et ceux qui sont indépendants du contexte. Les conflits dépendant du contexte consistent en les disparités résultantes d'hypothèses contradictoires ou d'interprétations dans différents systèmes. Dans la majorité des cas, ces conflits sont résolus par l'application des règles de transformations. Par contre, les conflits indépendants du contexte sont les conséquences d'événements aléatoires, erreurs humaines ou des instruments imparfaits. Pour cela, il n'existe pas de procédures automatiques pour les résoudre. Par souci de simplicité, nous allons nous atteler à examiner les conflits dépendants du contexte dans le cadre de cette thèse.

En outre, selon la proposition de [100], les conflits structurels et sémantiques, pouvant surgir lors des interactions entre services Web, peuvent être de trois niveaux. Les conflits liés au niveau des attributs des services, les conflits liés au niveau des entités décrivant le service et les conflits liés à l'abstraction des attributs et des entités. D'après cette classification, les conflits traités dans le cadre d'une composition de services DaaS sont liés au niveau "attribut" et de nature sémantique. En effet, les services DaaS participant à une composition ne présentent pas de conflits de niveau entité du moment qu'ils sont, au préalable, résolus par l'OD. Plus précisément, nous considérons les conflits sémantiques surgissants au cours des opérations "O/I".

Définition 4.8 *Opération de Sortie/Entrée (O/I) représente un flux de données qui se produit entre deux paramètres (sortie et entrée) appartenant respectivement à :*

- *deux service DaaS "s_i" et "s_j" successifs dans une composition "cs";*
- *"First(cs)" et "CQ_{Co_q}";*
- *"Last(cs)" and "CQ_X".*

A cet effet, dans une composition "cs" toute opération "O/I" relie deux paramètres conceptualisés par la même entité de l'OD. Le conflit est identifié quand lesdits paramètres

adoptent, au niveau implémentation, différentes désignations, structurations ou échelle de données. Précisément, dans le cas où deux contextes annotent deux paramètres similaires selon l'OD, un conflit sémantique de niveau attribut est spécifié. A cet effet, la notion du conflit sémantique sera explicitée en ce sens dans la définition suivante.

Définition 4.9 *Conflit sémantique : Etant donné deux services DaaS, s_i et s_j dans une composition "cs" ayant respectivement "O_k" et "I_k" comme paramètre d'entrée et de sortie, décrits respectivement par les contextes "Ct_{O_k}" et "Ct_{I_k}". Si ces contextes référencent deux différentes classes "ac_i" ayant le même "ac_g" on dit que l'opération "O_k/I_k" provoque un conflit sémantique de type "ac_g". Un conflit lié à un seul aspect "ac_g" est nommé conflit simple. Par contre, un conflit lié à plusieurs aspects est nommé conflit complexe.*

A titre d'illustration, les conflits sémantiques présentés dans l'exemple de motivation (voir section 1.3) et surgissant dans deux opérations " O/I" entre " s_2" et " s_3" dans "cs" sont identifiés comme suit :

- Le conflit simple cos_1 désigne un conflit de système de code entre "s_2" et "s_3" dans "cs". Le code de BPR est exprimé selon "SNOMED" ou "LOINC";
- Le conflit complexe cos_2 désigne un conflit d'unité de mesure et de structure entre "s_2" et " s_3" dans "cs". En premier lieu, la valeur BPR a comme unité de mesure "mmHg" ou "cmHg". En deuxième lieu, la valeur BPR est représentée en une valeur "MAP" ou en deux valeurs $BPR.D$ et $BPR.S$;
- Le conflit simple cos_2 désigne un conflit de précision entre "s_2" et " s_3" dans "cs". L'état de la valeur BPR est exprimée selon la nouvelle ou l'ancienne classification;

4.4.4 Services de médiation

Pour résoudre les conflits sémantiques dans les compositions de services DaaS, notre approche se base sur les services de médiation. Le service de médiation est un service sans état (stateless) au même titre que le service DaaS. Cependant, à la différence du modèle de service DaaS, le modèle de service de médiation ne nécessite pas d'exprimer la relation sémantique reliant les paramètres d'entrée et de sortie. En effet, la relation reliant les paramètres d'entrée et sortie d'un service de médiation est une relation de transformation. Ladite transformation peut être une conversion ou un mapping, et ce, selon la nature des données impliquées dans la transformation.

Les transformations en question manipulent la représentation physique des correspondances, établies lors de la première étape de réécriture, et des règles y relatives pour transformer les éléments d'un schéma à l'autre [126]. A cet effet, les services de médiation sont modélisés sous forme de règles assurant des fonctions de transformation d'un contexte à l'autre dans le cas où une opération "O/I" provoque un conflit. En ce sens, nous jugeons approprié d'adopter le modèle proposé par [51, 105] pour représenter les règles de transformation comme des requêtes SPARQL de type "construct". À cet effet, nous

allons définir le modèle des services de médiation, leurs classifications et leurs processus de publications dans ce qui suit.

4.4.4.1 Modèle du service de médiation

Le modèle du service de médiation a la forme d'une requête SPARQL paramétrée de type "construct" [35].

Définition 4.10 *Le modèle du service de médiation à la forme suivante* $MS_j(\$I_J, ?O_J) : G_I \rightarrow G_O$, *où :*

- *"*$\$I_J$*" et "*$?O_J$*" sont respectivement un ensemble de variables d'entrée et de sortie de* MS_J*;*
- G_I *et* G_O *sont une ensemble de triplets RDF représentant les paramètres d'entrée et de sortie contextualisés.*

Le service MS_J retourne un graphe RDF "G_O" formé à partir de l'instanciation du Template du graphe "G_I" avec la séquence des solutions de la requête [51].

A titre illustratif, la description du service de médiation "MS_1", cité dans notre exemple de motivation, assurant la transformation du code BPR exprimé en "LOINC" vers le code BPR exprimé en "SNOMED" est présenté dans la figure 4.7.

```
MS1 ($x,?y) :
CONSTRUCT
{(BPR DO:HasBprCode ?A).
(?A  rdf:type   CAO:BPC).
(?A  CO:HasBprCodetype CAO:SNOMED).
(?A  CAO:Codevalue  ?y)}
WHERE
{(BPR DO:HasBprCode ?C).
(?C  rdf:type  CAO:BPC).
(?C  CO:HasBprCodetype CAO:LOINC).
(?C  CAO:Codevalue  $x) }
```

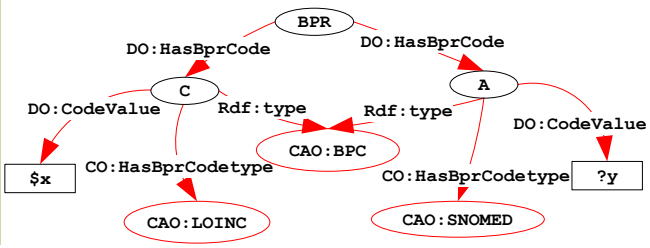

FIGURE 4.7 – Modèle du service de médiation "MS_1"

4.4.4.2 Types des services de médiation

Pour chaque aspect conflictuel, nous définissons un ensemble de règles atomiques pour décrire la transformation entre deux paramètres contextualisés. Nous identifions trois types de services de médiation de base, et ce, selon les concepts ontologiques impliqués par la transformation.

- Transformation de type 1-1 : Ce cas de transformation d'un seul attribut vers un seul attribut peut intervenir dans les différents cas suivants :

  1. Conversion : Selon notre exemple de motivation, les conflits des unités de mesure sont associés à des règles de conversion de type un-à-un. A titre indicatif,

35. http://www.w3.org/TR/rdf-sparql-query/construct

ce cas se présente lors de la composition d'un service DaaS qui retourne une valeur de tension artérielle moyenne "MAP" mesurée en "mmHg" à un service DaaS requérant comme entrée la valeur "MAP" mesurée en "cmHG". Par conséquent, la transformation requise, dans ce cas, pour convertir la valeur du "MAP" du "mmHg" au "cmHg" peut être assurée par le service de médiation MS_3.

2. Extraction : Lors de la composition d'un service DaaS requérant comme entrée le nom d'un patient à un service DaaS retournant l'affiliation complète du patient (p.ex Nom-Prénom), la transformation requise, dans ce cas, devrait assurer l'extraction du nom uniquement de l'affiliation complète. Aussi, l'extraction est envisagée dans le cas où un service DaaS requière uniquement la valeur diastolique de la tension artérielle "BPR.D" de la valeur globale "BPR.D/BPR.S" retournée par un autre service.

3. Identité : lors de la composition de deux services DaaS ayant des paramètres décrits en terme de différentes ontologies médicales ou terminologies (Tension artérielle systolique selon SNOMED-CT: " 271649006 " et selon LOINC: " 8480-6 ") ayant le même type de données (un <string> par exemple), une transformation de type identité est requise. Selon l'exemple de motivation, les conflits des systèmes de codes sont associés à des règles de transformation de ce type. À titre illustratif, le service MS_1 permet de composer deux services DaaS où leurs paramètres en liaison sont sémantiquement similaires, mais décrits selon différentes ontologies médicales.

4. Échelle : pour composer deux services DaaS où leurs paramètres en liaison sont sémantiquement similaires, mais décrits selon différentes échelles, différent domaine de précision ou différentes granularités ("BPRstate" selon la nouvelle ou l'ancienne classification). Le service MS_4 s'inscrit dans les services opérant ce type de transformations.

- Transformation de type 1-N: C'est le cas d'une transformation entre un attribut simple et un attribut complexe. Par exemple, une transformation 1-N peut être la division (split). En utilisant l'opération "split", le "nom-prénom" du patient peut être représenté par le nom et le prénom séparément. Les conflits de structures sont associés à des règles de transformation de type un-à-plusieurs.
- Transformation de type N-1 : Ce cas de transformation correspond à la transformation d'un attribut complexe en un seul attribut. Par exemple, la transformation N-1 peut être opérée par la fusion (merge). A cet effet, par l'application de la fusion, l'affiliation du patient représentée par le couple <nom, prenom> peut être représentée comme une chaine de caractères <nom-prénom> avant d'être passé au service DaaS subséquent dans une composition donnée. Les conflits de structures sont associés à des règles de transformation de type plusieurs-à-un.

En ce qui concerne les transformations de type N-N, elles peuvent être déduites à partir des trois catégories de transformations de bases suscitées.

4.4.4.3 Création des services de médiation

Un service de médiation peut être créé par un fournisseur de services DaaS ou par un tiers fournisseur. La création de ce genre de services peut être intuitive pendant la publication d'un service DaaS. Par exemple, lors de la publication d'un service DaaS dédié à la consultation des lectures de tensions artérielles, le fournisseur peut en même temps, publier le service de médiation opérant la conversion des unités de mesure employées par les lectures en question.

En outre, la création du service de médiation peut être basée sur les relations existantes entre les classes instaceables d'un aspect conflictuel dans l'OAC. En effet, si on considère à la fois notre exemple de motivation et que les concepts MAP et BPR.D/BPR.S sont décrits dans l'OAC, nous pouvons déduire le ou les services de médiations requis. Ainsi, sur la base des concepts des OAC, le fournisseur du service "S_2" peut fournir deux services de médiation de type 1-1 pour opérer des transformations respectivement du MAP (resp.BPR.D/BPR.S) vers le BPR.D/BPR.S (resp. MAP).

4.4.4.4 Modèle d'annotation du registre des services de médiation

Le modèle du registre de services de médiation ($\mathcal{R}MS$) regroupe l'ensemble des services de médiation publiés pour chaque aspect conflictuel "AC_g". L'annotation du registre de services de médiation offre la possibilité de dériver des compositions de services de médiation ou de services de médiations alternatifs à partir de services de médiation atomiques.

Définition 4.11 *Le modèle d'annotation du registre de services de médiation est modélisé par un graphe $G_{AC_g} = (V, E)$ où :*

- *"V" est un ensemble de nœuds représentant l'ensemble des classes "ac_i";*
- *"E" est un ensemble d'arcs représentant les services de médiation publiés dans le registre.*

Le modèle d'annotation du registre des services de médiation est prôné à l'effet de bénéficier des possibilités fournies par les algorithmes de recherche dans les graphes, et ce, pour découvrir le service de médiation alternatif dans le cas où celui requis n'est pas publié.

Par exemple, le graphe présenté dans la figure 4.8.(b) expose le RMS de l'aspect conflictuel "Codification System". Le graphe en question représente les services de médiation publiés et assurant la transformation entre paramètres de services DaaS exprimés en $LOINC$ vers $SNOMED$ et de $SNOMED$ vers ICD.

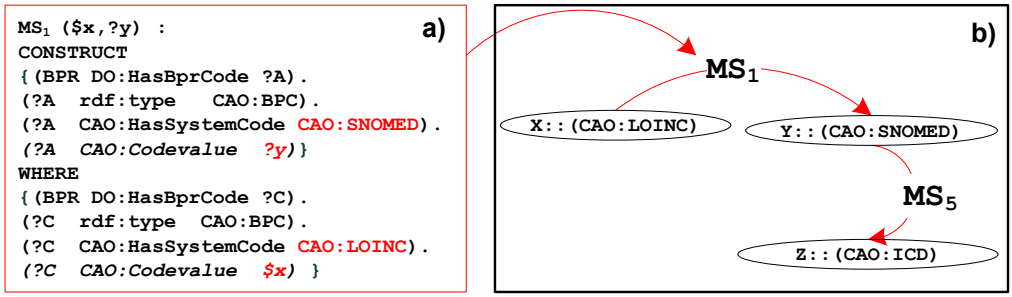

FIGURE 4.8 – Modèle d'annotation du registre des services de médiation

4.5 Conclusion

Dans ce chapitre nous avons mis en évidence les éléments fondamentaux de notre approche de détection et de résolution des conflits dans une composition de services DaaS. Ainsi, l'architecture de références et les modèles de bases sont explicités. Cela a permis aussi d'argumenter quant à l'adoption de l'approche orientée service pour assurer la médiation de données. Aussi, la notion du contexte permettant de fournir la connaissance nécessaire à la détection des conflits et la description du service de médiation approprié pour les résoudre a été définie. À cet effet, le prochain chapitre aura à exploiter l'ensemble des modèles proposés à l'effet d'assurer la génération automatique des compositions de services DaaS complètement exécutables.

Chapitre 5

Génération automatique des compositions de services DaaS

5.1 Introduction

Le présent chapitre dressera la génération automatique des compositions de services DaaS basée sur la réécriture de requêtes. Ce chapitre va mettre en évidence le cadre processuelle dans lequel la nouvelle phase de détection et de résolutions de conflit sera intégrée. En premier lieu, le prétraitement des descriptions de services DaaS contextualisées (VRPC) sera exposé dans la section 5.2. En deuxième lieu, les améliorations et les adaptations apportées au cadre de référence de [14] pour les besoins de notre approche, seront exposées dans la section 5.3. En dernier lieu, le plan d'exécution de la requête et la restitution de ses résultats seront présentés dans la section 5.4.

5.2 Prétraitement des VRP

En amont de la génération automatique des compositions de services, la phase de prétraitement vise à entendre les VRP des services DaaS par les contraintes sémantiques décrites dans l'OD et par la définition des fonctions de Skolem. Ces prétraitements s'inspirent des travaux de [32] inscrits dans la cadre de l'intégration des bases de données hétérogènes par une approche de médiation de type LAV. La nécessité de ces prétraitements est expliquée dans les deux sous-sections suivantes.

5.2.1 Application des contraintes RDFS

Le premier prétraitement consiste en l'extension des VRP par les contraintes sémantiques RDFS mentionnées dans l'OD. Pour cela, chaque concept (classe, propriété objet) du VRP sera étendu par les relations de type "rdfs:subClassof", "rdfs:subPropertyof", "rdfs:domain" et "rdfs:range" décrites dans l'OD. L'extension des VRP, par ces contraintes,

vise à réduire le temps requis par un raisonneur pour opérer la mise en correspondance (matching) entre les concepts ontologiques mentionnés dans la requête et ceux mentionnés dans la VRP. En plus, ceci permet au médiateur, via le sélecteur de services, de renvoyer plus de services DaaS pouvant contribuer à la réponse d'une requête donnée.

Exemple: Comme indiqué sur la figure 5.1.(a), la VRP représentant la description du service "S_1" est augmentée par un triplet indiquant que la classe "$DO : P$" de type "$DO : Patient$" est liée à la classe "$DO : Person$". Ceci est rendu possible, car la classe "$DO : Patient$" est liée à la classe "$DO : Person$" par une relation de spécialisation "$rdfs : SubClassof$".

5.2.2 Skolemisation

L'association des fonctions de Skolem aux instances des classes de chaque VRP peut être considérée comme la définition d'une contrainte de type clé primaire afin de rendre possible la jointure des instances des différentes sources de données. En effet, la valeur d'une variable de Skolem est unique, c.-à-d. l'instance de la classe qui lui correspond renvoie toujours la même valeur, de cette variable, chaque fois qu'elle est invoquée. La définition d'une fonction de Skolem est conditionnée par les contraintes que l'utilisateur veut imposer sur le schéma des données publiées par le service DaaS correspondant.

Exemple: Comme indiqué sur la figure 5.1.(b), la VRP représentant le service "S_1" où les instances "$DO : P$" et "$DO : SVE$" sont associées respectivement avec les fonctions de Skolem "$SF_1(?ssn)$ " et "$SF_2(?VSEref)$". La fonction de Skolem "$SF_1(?ssn)$" impose une contrainte indiquant que "si deux instances de la classe "$DO : Patient$" ont le même numéro de sécurité sociale, pour la propriété de données "$DO : SSN$", alors celles-ci représentent la même instance et peuvent être jointes ".

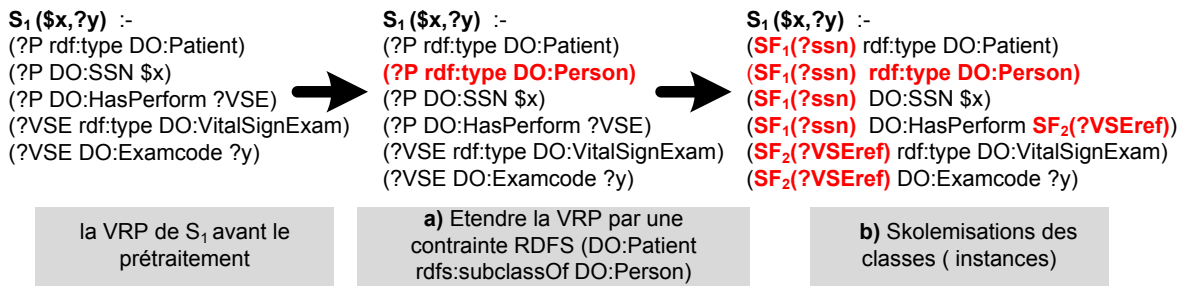

FIGURE 5.1 – Phase de prétraitement d'une VRP.

5.3 Réécriture de requêtes

Le processus de génération automatique des compositions de services DaaS est représenté dans la figure 5.2. Ledit processus, scindé en plusieurs étapes, se base principalement

sur la réécriture de requêtes.

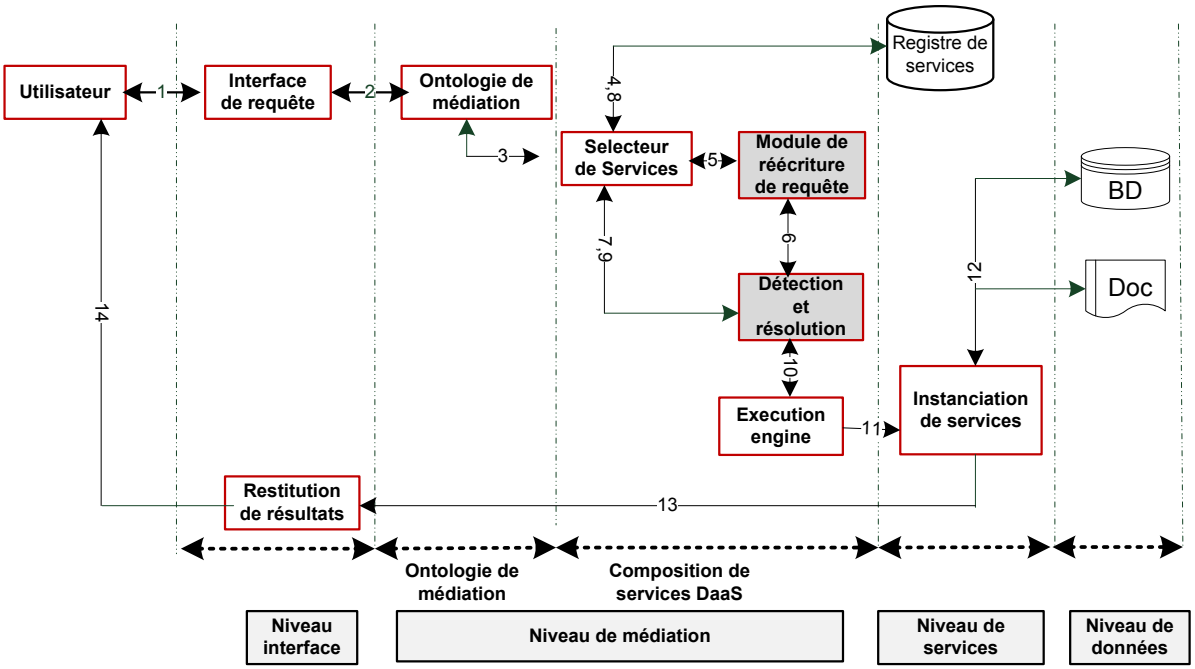

FIGURE 5.2 – Processus de traitement de la requête.

L'utilisateur formule la requête SPARQL en termes de l'ontologie de médiation (voir l'étape 1 dans la Figure 5.2) via l'interface du SCD (voir l'étape 2 dans la Figure 5.2) comme expliqué dans la section 4.1. Faisant suite à la formulation de la requête, le médiateur procède à l'analyse de la requête. Cette analyse procède à l'extraction des informations nécessaires, pour le sélecteur de services (voir étape 3 dans la Figure 5.2), à la découverte des services DaaS pouvant être combinés pour répondre à la requête. Les services DaaS découverts correspondent complètement ou partiellement aux concepts de l'OD (nœuds classes et propriétés objet) évoqués dans la requête.

L'algorithme de réécriture reformule la requête en une nouvelle requête qui se réfère à l'ensemble de services DaaS découverts. L'algorithme en question s'inspire des algorithmes de réécriture de requêtes à base de Tas (*Bucket*) [62]. L'algorithme de réécriture consiste en deux étapes, à savoir, trouvez-les sous graphes de la requête couverts par les services DaaS, retournés par le sélecteur de services, et la génération des compositions. A titre d'exemple, sur la base de la requête et les VRP des services DaaS mentionnés dans l'exemple de motivation et représentés dans la Figure 5.3, nous allons illustrer les différents algorithmes liés aux deux étapes suscitées et proposés dans cette section.

Notons que l'algorithme de réécriture ne considère pas la détection et la résolution des conflits. Cette étape n'est considérée qu'après la génération des compositions et avant la génération du plan d'exécution de la requête. En effet, les concepts caractérisant les contextes dans les descriptions de services DaaS, à base de VRPC, sont extraits. Donc,

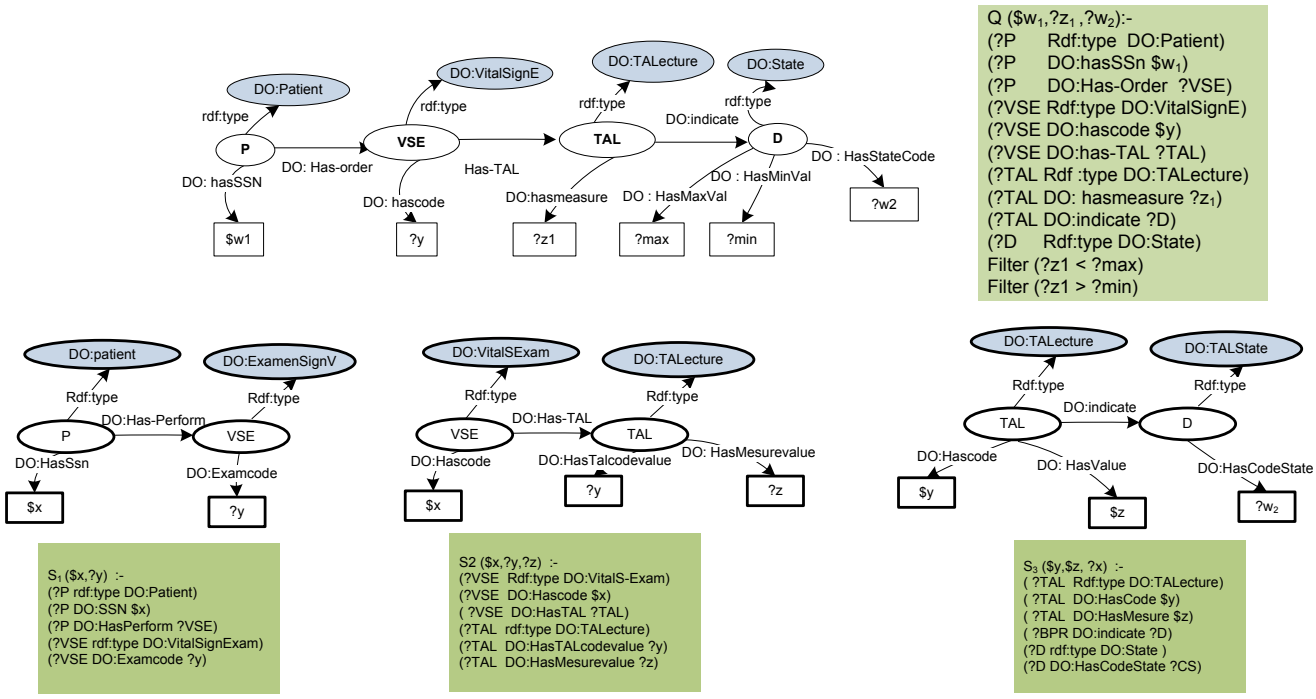

FIGURE 5.3 – Requête et VRPs de l'exemple de motivation sans contextualisation.

nous allons considérer que les variables utilisées dans les VRP sont définies dans le même contexte. Pour cela, l'algorithme de réécriture n'exploite que les VRP et la requête sans faire allusion aux extensions basées sur le contexte. Cette adaptation est requise à ce niveau, car les contextes des différents paramètres ne seront exploités que lors de la phase de détection et de résolution de conflits.

5.3.1 Couverture des sous-graphes de la requête

La première étape de l'algorithme de réécriture exploite les VRP décrivant les services DaaS retournés par le sélecteur de services. A partir de ces vues, l'algorithme sélectionne les services pouvant être combinés pour répondre à la requête. La réécriture de la requête "Q" par les services DaaS est construite par la composition des services DaaS dont l'union des graphes couvre le graphe de la requête.

En effet, étant donné que les VRP ne peuvent couvrir, généralement, qu'un sous graphe de la requête, la réponse à une requête peut être obtenue en examinant les diverses combinaisons des sous graphes des services DaaS couvrant le graphe de la requête. Dans ce cadre, le problème de mise en correspondance entre la requête et les vues s'assimile à un problème de recouvrement de graphes.

Notations: Avant d'entamer la présentation de la notion du recouvrement de graphes, des notations devrait être adoptées lesquelles sont basées sur les notions évoquées dans la

section 2.6.3 liées à la réécriture de requêtes.

- la requête Q et les VRP sont représentées par des graphes respectivement notés "G_Q" et "G_v". Le graphe de la composition de services DaaS est représenté par "G_C".
- Les VRP des services découverts constituent un ensemble de vues $V = \{v_1, v_2, ..., v_i\}$;
- Les prédicats de la requête et des vues sont appelés respectivement les sous-buts de la requête, notés par "sb_Q", et sous-buts de la vue notés par "sb_{Vi}". Les sous-buts correspondent à des concepts de l'ontologie représentant des nœuds classes ou des propriétés objets.

Couverture de graphes: La couverture du graphe de la requête G_Q par le graphe de la composition G_C n'est effective que si:

1. Tous les nœuds classes dans G_Q existent dans le graphe de la composition G_C ;
2. Toutes les propriétés objets reliant des nœuds classes dans G_Q, relient aussi les nœuds classes correspondants dans G_C.
3. Toutes les contraintes appliquées sur les classes sb_Q elles sont satisfaites aussi dans sb_v.
4. L'inclusion de correspondance $\beta : G_V \rightarrow G_Q$ est vérifiée, tel que :
   (a) β fait correspondre les nœuds classes de G_Q à des nœuds classes dans G_V (i.e. ils ont le même type de classe et les mêmes variables);
   (b) β fait correspondre chaque nœud littéral dans G_Q à un nœud littéral dans G_V ;
   (c) Les variables distinguées dans Q (c.-à-d. littéraux requis) sont fournis par la composition de services.

5.3.1.1 Principe de l'algorithme de découverte des sous graphes

L'algorithme de découverte des sous graphes procède à la comparaison de chaque sous-but de la requête aux sous-buts de chaque PRV (v_i) dans V. Cette comparaison vise à déterminer les sous-buts de la requête qui sont couverts par les sous-buts des VRP et ceux qui ne le sont pas. Ladite comparaison consiste en l'unification de sous-buts qui vérifie que si deux nœuds classes ou propriétés objets peuvent être rendues identiques par une assignation de variables.

Le résultat de cette étape est un ensemble de correspondances qui seront enregistrées dans la table de correspondance. À partir de cette table, l'algorithme de découverte des sous graphes procède à la création des Tas (*Bucket*) pour chaque sous-but de la requête "Q". Chaque sous-but de la requête "sb_Q" lui sera assigné les VRP correspondants aux services contributifs à la réponse de la requête.

5.3.1.2 Améliorations apportées

L'examen de l'algorithme de réécriture de [14] au niveau de l'étape de découverte des sous graphes de la requête pouvant être couvert par les VRP fait ressortir quelques limites qui se déclinent comme suit:

1. L'algorithme de [14] fait appel à deux fonctions "CoveringClassNodes" et "CoveringObjectNodes" pour vérifier respectivement la couverture des classes et des propriétés objets. En effet, lors de la vérification de la couverture d'une propriété objet de la requête "O_Q" par une propriété objet d'une VRP "O_v" (via la fonction "CoveringObjectProperty"), si les classes "C_i" et "C_j" liées à "O_v" fournissent les variables de Skolem, "O_v" sera insérée dans la table des correspondances même si les contraintes définies sur "C_i" et "C_j" ne vérifient pas les contraintes définies sur les classes liées à "O_Q".
2. L'algorithme de [14] expose un risque de boucle infinie qui pourrait être précisé comme suit. Si les classes "C_i" et "C_j" liées par la propriété objet "O_v" ne fournissent pas les variables de Skolem, la fonction "CoveringObjectProperty" appelle la fonction "CoveringClassNode", qui à son tour appelle la fonction "CoveringObjectProperty" parce que les classes "C_i" et "C_j" sont liées à une propriété objet dans la requête (O_Q) ce qui va causer une boucle infinie.

Ces limites provoquent la sélection de services non contributifs à la génération de compositions de services DaaS. Ceci est dû à la prise en considération des sous buts contenant uniquement une classe ou une propriété objet sans pour autant vérifier le recouvrement maximal entre les sous-buts de la requête et ceux des services. En s'inspirant de l'algorithme du MINICON [106], le fait de relier les propriétés objets avec leurs classes dans le même sous-but apporte une solution intéressante. En effet, considérer un tas contenant des sous-buts constitués de deux classes et de la propriété objet qui les relie permet de réduire considérablement le nombre de services sélectionnés. Ainsi, lors de la génération des compositions, le nombre de celles qui sont non contributives à la réponse à la requête sera considérablement réduit.

5.3.1.3 Algorithme de découverte des sous graphes

Le déroulement de cette étape est décrit par l'algorithme 1. Cet algorithme procède à la création pour chaque vue "vi", un ensemble "E_s" contenant tout les sous buts de la requête (Algorithme 1 ; lignes 1-2). Les éléments de cet ensemble seront comparés avec les sous-but de "v_i". Ainsi, si "v_i"comprend un sous but "sb_v" qui correspond à un sous but "sb_Q" dans "E_s", alors l'algorithme crée un sous graphe "sb_g" contenant le sous but "sb_v" . Le sous graphe "sb_g" représente une ligne candidate dans la table des correspondances (Algorithme 1 ; lignes 3-6). Si "sb_g" est pertinent il sera inséré dans la table des correspondances et les sous but couvert de "E_s" seront supprimés (Algorithme 1; lignes 7-13).

Algorithme 1 Découverte des sous graphes (services pertinents)

Entrées: Requête Q, ensemble de vues V ;
Sorties: Table des correspondances T

```
1: pour Chaque vue v ∈ V faire
2:   E_s → SB_Q {L'ensemble des sous buts de Q }
3:   pour chaque sous but sb_g ∈ S faire
4:     pour chaque sous but ġ ∈ V faire
5:       Soit φ un mapping de g vers ġ tel que φ(g) = g'
6:       Créer un sous graphe sbg {une ligne candidate dans T}
7:       si (estValide(sb_g,φ)) alors
8:         sb_g → T
9:         supprimer sb_g.SousButCouverts de E_s,
10:      finsi
11:    fin pour
12:  fin pour
13: fin pour
14: return Table des correspondances T
```

Un sous graphe "sb_g" contenant le sous but "sb_v" est dit pertinent si "sb_v" couvre le sous but "sb_Q" de la requête "Q" et "sb_g" est valide.

1. Couverture du sous but: "sb_v" couvre "sb_Q", si les classes $< c_i, c_j >$ reliées par "sb_v" couvrent respectivement les classes $< c_k, c_l >$ reliées par "sb_Q". Par couverture de classes, on entend qu'une classe "c_v" appartenant à une vue "v_i" couvre une classe "c_Q" dans "sb_Q", si elle est de même type avec la classe "c_Q" et elle vérifie les conditions suivantes :
   (a) Si la classes couverte "c_Q" possède une variable distinguée x (c.à.d. une des ses propriétés de données correspond à une variables distinguée dans Q), alors une de ces conditions doit être vérifiée :
   - La variable x correspond à une variable distinguée dans v_i ;
   - La propriété qui correspond à x peut être couverte parce que les propriétés utilisées par la fonction du Skolem de la classe "c_Q" sont projetées dans la classe "c_v" et elles peuvent être utilisées pour couvrir les propriétés de données de "c_v" qui correspondent à x ;

   (b) Si la classe couverte "c_Q" possède une constante laquelle doit correspondre à une variable distinguée dans v_i ou elle peut être couverte par la fonction du Skolem ;

   (c) Si la classe "c_Q" possède une contrainte sur une propriété de donnée x, alors la classe "c_v" doit avoir une contrainte sur la propriété de données correspondante à x et les valeurs acceptées par cette contrainte sont incluses dans les valeurs acceptées par la contrainte définie dans "c_q" .

2. Validité du sous graphe "sb_g" consiste, après avoir trouver un sous but "sb_Q" couvert par la vue v_i, à vérifier pour chaque classe "c_Q" partagée (c.à.d. liée à d'autres sous but dans la requête) du "sb_Q", les conditions suivantes :
   - La classe "c_v" correspondant à la classe "c_Q" peut avoir des variables existentielles dans sa définition, dans ce cas, tous les sous-buts liés à "c_Q" doivent être couverts par la vue v, ils seront ensuite mis dans le même sous graphe couvert "sb_g". Dans le cas contraire, aucun sous graphe ne sera inséré dans la table des correspondances "T".
   - Si toutes les variables de "c_v" correspondent à des variables distinguées dans Q, alors l'algorithme récupère tous les sous-buts liés par "c_Q" et couverts par "c_v" et les met dans le même sous graphe dans la table des correspondances "T".

5.3.1.4 Exemple

Nous allons dérouler cette phase de l'algorithme sur la requête Q citée dans l'exemple de motivation (voir section1.3). Le tableau 5.1 présente la table des correspondances générée après l'analyse des correspondances entre les services DaaS et la requête illustrée dans la figure 5.3.

VRP	Correspondances des variables et des classes	Les relations et les classes couvertes
$S_1(\$w_1, ?y)$	$P_q \rightarrow Ps_1$, $VSE_q \rightarrow VSE_{S1}$, $x(\text{"}p222\text{"}) \rightarrow w_1$, $y \rightarrow P_q(x)$	$VSE_q(y)$, $Has - perform(P_q, VSE_q)$, $P_q(w_1)$
$S_2(\$w_4, ?w_2, ?w_3)$	$y \rightarrow w_4$, $z_1 \rightarrow w_2$, $VSE_q \rightarrow VSE_{S2}$, $BPR_q \rightarrow BPR_{S2}$,	$hasBPR(VSE_q, BPR_q)$, $VSE_q(y)$, $BPR_q(z_1)$
$S_3(\$w_5, ?w_6, ?w_7)$	$z_1 \rightarrow w_5$, $z_1 \rightarrow w_6$, $z_1 \rightarrow w_7$, $BPR_q \rightarrow BPR_{S3}$, $D_q \rightarrow D_{S3}$	$Indicate(BPR_q, D_q)$, $D_q(z_2)$, $BPR_q(z_1)$

TABLE 5.1 – Table des correspondances générées.

Après avoir créée la table des correspondances, l'ensemble des Tas (*Bucket*) crée sont représentés dans le tableau suivant :

$Has - perform(P_q, VSE_q)$	$hasBPR(VSE_q, BPR_q)$	$Indicate(BPR_q, D_q)$
$S_1(\$x, ?y)$	$S_2(\$y, ?z, ?k)$	$S_3(\$z, \$k, ?l)$

TABLE 5.2 – Tas (*Bucket*) générés à partir de la table des correspondances.

5.3.2 Génération des compositions de services DaaS

Lors de la deuxième étape, l'algorithme de réécriture procède à la génération des différentes combinaisons de services DaaS à partir des Tas (*Buckets*) construits lors de la précédente étape. Les appellations "combinaison" et "composition" sont utilisées de manière interchangeable le long de cette section. Chaque Tas comprend l'ensemble des services qui couvre un sous but de la requête. A partir de ces Tas qui sont au nombre des sous-buts de la requête, les compositions sont construites en opérant un produit cartésien entre les membres des Tas.

5.3.2.1 Principe de génération des compositions

Cette étape consiste à combiner les différents éléments des Tas pour générer des compositions valides et invocables. La combinaison consiste en le produit cartésien entre les différents membres des Tas créés pour chaque sous but de la requête. La génération des compositions passe par deux étapes, à savoir, la génération des compositions valides de services DaaS et le teste d'invocabilité. Ces deux étapes sont détaillées respectivement dans l'algorithme 2 et l'algorithme 3.

Algorithme 2 Combinaison des sous graphes

Entrées: Ensemble de Tas T

Sorties: Ensemble de combinaisons valides de services DaaS

```
1: sbgs=TrouverSousButdeT()
2: combSub=CombineSousButs(sbgs)
3: pour Chaque combinaison c de combSub faire
4:   si c[1].Sb_v ∪ c[2].Sb_v ∪ c[i].Sb_v = Q.Sb and c[1].Sb_v ∩ c[2].Sb_v ∩ c[i].Sb_v = ∅ alors
5:     pour Chaque service element i from C faire
6:       Créer un sous ensemble Set_i
7:       pour Chaque service S dans T faire
8:         Mettre S dans Set_i qui le couvre.
9:       fin pour
10:    fin pour
11:    Faire le produit cartésien P entre les différents éléments de Set
12:    Ajouter P à l'ensemble des compositions candidates
13:  finsi
14: fin pour
15: return Combinaisons valides
```

5.3.2.2 Algorithme de génération des compositions valides

L'algorithme 2 explore les différentes compositions pour couvrir le graphe de la requête. Les compositions générées par l'algorithme 2 doivent être testées en terme de validité. Une

composition est dite valide si elle contient, à la fois, un ensemble disjoint de sous-buts et la totalité des sous-buts de la requête. Autrement dit, une composition est valide si elle couvre l'ensemble des nœuds classes et des propriétés objets dans Q.

A cet effet, l'algorithme 2 procède à la génération des combinaisons couvrant le graphe de la requête "Q". Il requiert comme entrée les Tas où chaque nœud classe ou propriété objet de la requête "Q" lui est assigné un sous ensemble "Set_i" contenant les services couvrant les nœuds ou les propriétés objets de la requête. A partir de chaque sous-ensemble $\{Set_i ... Set_n\}$, l'algorithme 2 combine les services. Pour chaque combinaison "c_i", il élimine la redondance des services et il vérifie la couverture de la requête par cette combinaison et la disjonction entre les services appartenant à la combinaison.

5.3.2.3 Algorithme de vérification d'invocabilité

L'algorithme 3 vérifie les compostions valides en termes d'invocabilité. Une composition valide est dite invocable si tous les paramètres nécessaires pour l'invocation des services qui la constituent sont fournis. L'algorithme 3 vérifie l'invocabilité de chaque composition "c_i" dans l'ensemble des compositions candidates "C". Il reçoit comme entrée l'ensemble de compositions candidates générées par l'algorithme 2 et retourne un ensemble de compositions valides et invocables.

A cet effet, il initialise la liste des services invocables (liste A) avec un ensemble vide et la liste des variables fournies (liste B) liées à des constantes (contraintes) de la requête "Q" (Algorithme 3, lignes 1-2) et il initialise la variable booléenne "invoque" a "False".

Algorithme 3 Vérification d'invocabilité

ENTRÉES: Une combinaison valide c_i de services DaaS

SORTIES: Une valeur booléene pour déterminer si la combinaison est invocable ou non.

```
1:  A= ∅
2:  B = {Les variables qui correspondent à des constantes dans Q}
3:  répéter
4:     Fait =true
5:     pour i =1 à k  faire
6:        A=A ∧ S_i
7:        B=B ∪ outputVariables (S_i)
8:        Fait=false
9:     fin pour
10: jusqu'à FAIT
11: si  c_i ≡ A  alors
12:    return  true
13: sinon
14:    return  false
15: finsi
```

Ainsi, il parcoure la composition "c_i" service par service, si le service concerné n'appartient pas à l'ensemble des services invoqués "A" et que ses entrées sont incluses dans l'ensemble "B" donc il l'ajoute à l'ensemble des services invoqués "A" et ses sorties à l'ensemble "B" et il affecte la valeur "True" a la variable "invoque" (Algorithme 3, lignes 6-13). Ces étapes sont répètes autant de fois jusqu'à ce que la variable "invoque" devient "False", ce qui signifie que tous les services de la composition "c_i" sont invoqués.

Ensuite, il rajoute les services invocables ayant comme paramètres d'entrée les variables courantes de la liste "B", les paramètres d'entrée de chaque service ajouté à la liste "A" seront insérés dans la liste "B" (Algorithme 3, lignes 3-10).

L'algorithme 3 vérifie encore les services qui sont devenus invocables après la modification de la liste "B". Si aucune variable n'est rajoutée à la liste B, l'algorithme 3 vérifie si la liste "A" contient tous les services de la composition (Algorithme 3 , lignes 11-13), dans ce cas il retourne vrai et l'ordre de l'exécution des services de la composition sera celui de la liste "A".

5.3.2.4 Exemple

Considérons la combinaison $c_1 = \{S_1(\$x, ?y), S_2(\$y, ?z, ?k), S_3(\$z, \$k, ?l)\}$, comme l'indique le tableau 5.2, la variable "x" est la seule variable fournie au début de l'exécution (c.-à-d. la requête fournit uniquement l'identifiant du patient Joe x="p222", selon l'exemple de motivation), par conséquent, le service "$S_1(\$x, ?y)$" est le seul service invocable.

Après l'invocation de "$S_1(\$x, ?y)$", les variables "x" et "y" seront fournies et le service "$S_2(\$y, ?z, ?k)$" devient invocable. L'invocation de "$S_2(\$y, ?z, ?k)$" fournie les variables nécessaires pour l'invocation du "$S_3(\$z, \$k, ?l)$". Après l'invocation du service "$S_3(\$z, \$k, ?l)$" tous les paramètres nécessaires pour l'invocation des services de la combinaison seront fournis. Par conséquent, la combinaison "c_1" est dite invocable.

Variables fournies	Services invocables
x	$S_1(\$x, ?y)$
x,y	$S_1(\$x, ?y)$, $S_2(\$y, ?z, ?k)$
x,y,z,k	$S_1(\$x, ?y)$, $S_2(\$y, ?z, ?k)$, $S_3(\$z, \$k, ?l)$

TABLE 5.3 – Variables fournies et services invocables dans chaque étape de l'algorithme 3.

5.4 Plan d'exécution de la requête

Sur la base des compositions valides et invocables, le plan d'exécution de la requête décrivant les flux de données et de contrôles sera constitué. L'exécution dudit plan, par le

module de restitution des résultats, fournit à l'utilisateur les résultats de sa requête via l'interface.

Avant d'entamer la génération du plan d'exécution, les services DaaS participants dans chaque composition doivent être exécutés selon un certain ordre lequel dépend de leur patron d'accès. Si le service S_j a comme entrée x laquelle est fournie par la sortie y de S_i donc S_j doit être précédé par S_i dans le plan d'exécution. Nous dirons qu'une contrainte de dépendance d'exécution existe entre S_i and S_j (S_j dépend de S_i).

Nous définissons un graphe de dépendances comme un graphe acyclique direct G où les nœuds correspondent aux services et les arcs correspondent aux contraintes de dépendances entre les services participants dans la composition.

La figure 5.4 illustre le graphe de dépendance pour la composition c_1. Une dépendance entre $S_1(\$x, ?y)$ et $S_2(\$y, ?z, ?k)$, et entre $S_2(\$y, ?z, ?k)$ et $S_3(\$z, \$k, ?l)$. A cet effet, $S_1(\$x, ?y)$ devrait être invoqué en premier, ensuite $S_2(\$y, ?z, ?k)$ peut être invoqué, idem pour $S_3(\$z, \$k, ?l)$. Le services $S_1(\$x, ?y)$ est appelé $First(c_1)$ ou le parent du service $S_2(\$y, ?z, ?k)$. Au même titre que pour les services $S_2(\$y, ?z, ?k)$ et $S_3(\$z, \$k, ?l)$ où le premier est le parent du second.

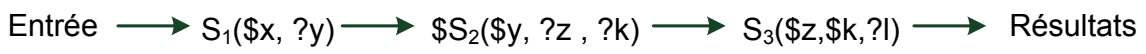

FIGURE 5.4 – Exemple du graphe de dépendances

L'algorithme 4 de cette phase explicite la façon dont les services composites sont exécutés. L'algorithme 4 requiert comme entrée l'ensemble de compositions et produit comme sortie, le résultat de la requête Q.

Ainsi, pour chaque composition, l'algorithme 4 créé un thread T_i pour chaque service S_i dans une composition c_i. Le thread T_i prend ses tuples d'entrées à partir d'un autre thread de jointure J_i qui joint les sorties des parents du S_i.

A cet effet, Si S_i n'a pas de parents dans c_i, alors T_i prend ses entrées à partir de la relation d'entrée I qui contient toutes les valeurs spécifiques dans Q (c.-à-d. les constantes de la requête).

Le thread T_i invoque S_i pour chaque tuple d'entrée, filtre les tuples retournées, les joins avec les tuples d'entrée et les retourne à sa sortie. Le résultat de la requête est obtenu à partir de la sortie du thread J_{out} qui relie les sorties de tous les services qui sont des feuilles dans le graphe de dépendance de C_i.

Algorithme 4 Plan d'exécution de la requete

Entrées: Ensemble de compositions executables
Sorties: Résultat de la requête Q

pour Chaque s_i dans c_i **faire**
  Créer un Thread T_i
  si s_i n'a pas de parents dans c_i **alors**
    T_i prend son entrée depuis la relation d'entrée contenant toutes les valeurs fournies par Q
  sinon
    si s_i a un seul parent s_P dans c_i **alors**
      T_i prend son entrée depuis la sortie de T_P
    sinon
      Créer un thread joignant J_i
      T_i prend son entrée depuis la sortie de J_i
    finsi
  finsi
fin pour
Créer un thread joignant J_{out}
return Résultat de la requête

5.5 Conclusion

Dans ce chapitre, nous avons mis en évidence le processus de génération automatique des compositions de services DaaS. En premier lieu, les prétraitements opérés sur les vues RDF contextualisées pour la réécriture de requêtes ont été exposés. Aussi, les adaptations et les améliorations apportées sur l'algorithme de [14] ont été introduites. Faisant suite à la phase de réécriture on a abordé la phase de création du plan d'exécution de la requête. La phase d'exécution du plan de la requête requiert uniquement les compositions valides, invocables et complètement exécutables. L'executabilité en question est assurée par la nouvelle phase de détection et de résolution de conflits. Laquelle phase sera abordée dans le prochain chapitre et insistera sur le traitement d'une seule composition valide et invocable. Par la suite, le même traitement sera généralisé sur toutes les compositions générées lors de la réécriture. Ainsi, les spécifications quant à l'intégration de cette phase dans le processus de génération automatique des compositions seront mises en évidence.

Chapitre 6

Détection et résolution des conflits

6.1 Introduction

Les compositions générées, lors de la phase de réécriture, doivent être vérifiées en termes de présence de conflits. Pour chaque conflit détecté, un service de médiation, simple ou composite, sera sélectionné et intégré dans la composition concernée à l'effet de le résoudre. La phase de détection et de résolution de conflits sera abordée en trois parties. En premier lieu, le principe et l'algorithme de l'étape de détection des conflits au sein d'une composition seront dressés dans la section 6.2. En deuxième lieu, la résolution des conflits dans une seule composition sera détaillée dans la section 6.3. En dernier lieu, l'intégration de la phase de détection et de résolution dans le processus de génération automatique de compositions de services DaaS exécutables sera détaillée dans la section 6.4.

6.2 Détection des conflits

Les conflits surgissent lors des échanges des données entre les services participants à une composition "cs", valide et invocable, ou entre une composition "cs" et les paramètres de la requête. En effet, les paramètres des services impliqués dans les opérations "O/I", sont décrits en terme de concepts similaires selon l'OD et de différents contextes exprimés selon l'OAC. Cette différence sera à la base de la détection de conflits. Dans cette section, l'algorithme 5 illustre le déroulement de la détection des conflits dans une seule composition de services DaaS.

À ce stade, les contextes mentionnés dans la requête et dans les descriptions des services DaaS participants dans une composition "cs", non exploités lors de la phase de réécriture, seront utilisés. Pour les besoins d'illustration, la requête et les VRP mentionnées dans l'exemple de motivation avec les extensions décrivant le contexte sont présentées dans les figures 6.2 et 6.1.

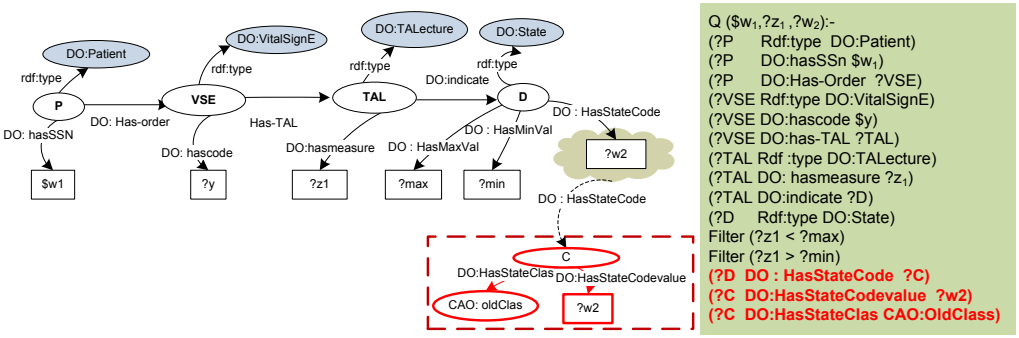

FIGURE 6.1 – Requête "Q" contextualisée.

6.2.1 Principe de la détection des conflits

L'algorithme 5 illustre le déroulement de l'étape de détection des conflits dans une composition "cs" donnée. Le même traitement sera opéré sur toutes les compositions générées par l'algorithme de réécriture.

L'algorithme 5 scrute chaque opération "O/I" dans la composition de services DaaS "cs". Dans le cas où aucun conflit n'est détecté dans la composition "cs", celle-ci sera insérée dans l'ensemble des compositions sans conflits "R". Sinon, dans le cas où un ou des conflits sont détectés dans "cs", la composition en question sera insérée dans l'ensemble des compositions "CS" comportant des conflits (Plus de détails dans la section 6.4). Les informations liées aux conflits détectés seront insérées dans l'ensemble des Objets Conflictuels "COS".

Les conflits dans "cs" peuvent être détectés dans trois endroits, à savoir :

- Entre les services DaaS de la composition "cs" ;
- Entre les paramètres spécifiés par la requête et le premier service ($First(cs)$) de la composition "cs" ;
- Entre le dernier service de la composition ($Last(cs)$) et les résultats requis par la requête.

En effet, les paramètres requis par le premier service de la composition "cs" doivent correspondre à ceux fournis par la requête. Aussi, les données que le ou les derniers services retournent doivent correspondent à ce que la requête requiert. Pour cela, nous proposons d'augmenter chaque composition de services DaaS "cs", au début et à la fin, par deux services virtuels (voir figure 6.3), à savoir :

- Un service au début de chaque composition ayant comme pattern d'accès $S_{Qin}(\ ?O_Q)$. Ce service fournit les variables d'entrée spécifiées par la requête.
- Un service à la fin de chaque composition ayant comme pattern $S_{Qout}(\$I_Q)$. Ce service consomme les variables de sortie de la composition de services DaaS.

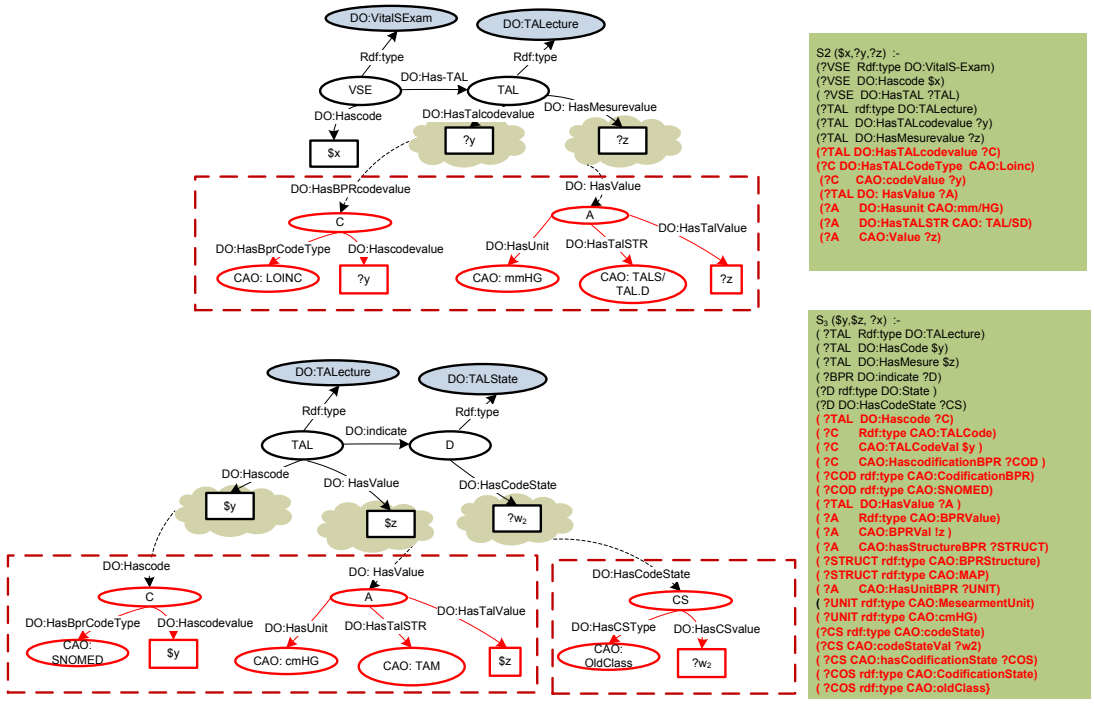

FIGURE 6.2 – Vues RDF Paramétrées et contextualisées.

6.2.2 Algorithme de détection des conflits

L'algorithme 5 examine chaque opération "O_k/I_k" [36] dans "cs" et procède à la vérification de l'existence des conflits. Notons que la composition "cs" est représentée sous forme d'un graphe acyclique orienté $G = (S, OP)$ où l'arc "OP_k" correspondent à l'opération "O_k/I_k" (Voir section 4.4). L'algorithme 5 requiert comme entrée, la composition "cs", les opérations "O/I" qui y sont incluses où chaque paramètre "O" et "I" est associé à son contexte. Les contextes sont extraits lors de l'étape de traitement, à la fois, de la requête et des vues décrivant les services DaaS, et ce, avant l'étape de réécriture.

L'existence d'un conflit est basée sur la vérification de la similarité entre les contextes de "O_k" et de "I_k". A cet effet, pour chaque arc $(s_i, s_j) \in OP$, le contexte "Ct_O" annotant les variables de sortie du "s_i" et le contexte "Ct_I" annotant les variables d'entrée du" s_j" sont récupérés (Algorithme 5 ligne 1).

La paire de contextes $(O.Ct_O, I.Ct_I)$ correspondant aux contextes des paramètres d'une opération "O/I" est exploitée pour la détection d'un conflit. Dans le cas où celui-ci est avéré, la paire de contextes en question sera aussi exploitée pour la sélection d'un service de médiation à l'effet d'assurer la résolution du conflit entre les services "s_i" et "s_j" d'une composition "cs".

Les informations relatives aux conflits détectés sont stockées dans l'ensemble des objets conflictuels "$\mathcal{COS}$". L'algorithme 5 retourne l'ensemble "$\mathcal{COS}$" ayant la forme d'un 6-tuple

36. K représente la $K^{\text{éme}}$ opération de "O/I" pour une composition "cs" donnée.

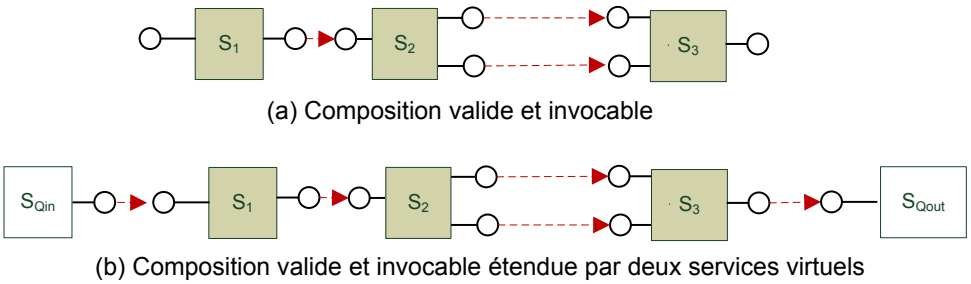

FIGURE 6.3 – Extension de la composition "cs" par deux services DaaS virtuels.

$< id - conflict, ac_g, Ct_O.ac_i, Ct_I.ac_{i'}, Position, MService >$ où :

- "$id - conflict$" est l'identificateur du conflit détecté;
- "ac_g" représente l'aspect conflictuel lié aux deux paramètres O_i and $I_{i'}$;
- "ac_i" est une classe instanceable appartenant au contexte Ct_O;
- "$ac_{i'}$" est une classe instanceable appartenant au contexte Ct_I;
- "$position$" indique les positions de l'opération "O/I" dans "cs" où le conflit est détecté;
- "$MService$" est l'ensemble de services de médiation apte à résoudre le conflit identifié par "$id - conflict$". Cet ensemble est alimenté durant l'étape de résolution par l'algorithme 6 (voir section 6.3.2).

6.2.2.1 Absence de conflits

Sur la base de la comparaison effectuée entre les contextes $O.Ct_O$ et $I.Ct_I$, l'algorithme 5 retourne le résultat de la similarité des contextes. Dans le cas où la similarité est vérifiée, l'algorithme en question indique une compatibilité entre la sortie fournie par le service "s_i" et l'entrée requise par le service subséquent "s_j". Cela implique qu'aucun conflit n'est détecté et qu'aucun service de médiation n'est requis entre les services "s_i" et "s_j". L'algorithme 5 renvoi l'information stipulant que l'opération "O_k/I_k" ne cause pas de conflits dans la composition "cs" (Algorithme 5 ligne 9).

6.2.2.2 Présence de conflits

Si un conflit est détecté dans l'opération "O_k/I_k", l'algorithme 5 vérifie son existence dans l'ensemble des objets conflictuels "$\mathcal{COS}$" (Algorithme 5, ligne 3).

Si le conflit en question n'existe pas dans l'ensemble des objets conflictuels "$\mathcal{COS}$" alors l'algorithme 5 procède à l'insertion de ses informations dans l'ensemble "$\mathcal{COS}$" (Algorithme 5, ligne 4). Cette insertion s'accompagne aussi par la spécification de la position dans laquelle le conflit a été détecté dans "cs".

Or, si le conflit détecté existe, au préalable, dans l'ensemble des objets conflictuels "$\mathcal{COS}$", la liste des positions de ce conflit, "$cos.position$", sera mise à jour ou augmentée

par la nouvelle position (Algorithme 5, ligne 6). Cela indique que le même conflit peut être détecté à plusieurs endroits et dans plusieurs compositions.

En résumé, chaque fois qu'une comparaison entre contextes $O.Ct_O$ et $I.Ct_I$ d'une opération "O/I" liée à une composition "cs", relève une différence, l'ensemble des objets conflictuels se verra augmenté par un nouveau conflit où la "position" du conflit correspondant sera mise à jour.

Algorithme 5 Détection des conflits

ENTRÉES: cs composition de services DaaS; Ensemble d'opérations O/I dans cs.
SORTIES: COS Ensemble des objets Conflictuels,

1: **pour** Chaque O_k/I_k dans O/I **faire**
2: **si** les contextes de O_k et I_k appartiennent à la même classe ac_g et ont différents ac_i **alors**
3: **si** $(O_k/I_k) \notin \{(cos.Ct_O.ac_i, cos.Ct_I.ac_{i'})\}$ **alors**
4: cos_t.add (ac_g, $Ct_O.ac_i$, $Ct_I.ac_{i'}$, Position)
5: **sinon**
6: Ajouter la nouvelle position dans $cos_t.position$;
7: **finsi**
8: **sinon**
9: Pas de conflits détectés dans O_k/I_k de cs;
10: **finsi**
11: **fin pour**
12: **return** COS

L'ensemble des objets conflictuels "$\mathcal{C}OS$" sera exploité par l'algorithme 6 proposé pour la résolution des conflits. Le ou les services requis pour la médiation seront découverts à travers les spécifications fournies par l'ensemble "$\mathcal{C}OS$".

6.2.3 Exemple

A titre illustratif, l'ensemble "$\mathcal{C}OS$" généré sur la base de la requête et des vues étendues issues de l'exemple de motivation, respectivement illustrées sur les figures 6.1 et 6.2, est représenté dans le tableau 6.1. Les conflits "cos_1" et "cos_3" mentionnés dans ledit tableau représentent des conflits simples détectés entre "S_2" et "S_3". Le conflit "cos_2" est un conflit complexe détecté aussi entre "S_2" et "S_3".

La partie coloriée du tableau indique la partie non renseignée pendant l'étape de détection. Plus précisément, cette colonne recevra les désignations des services de médiation sélectionnés pour la résolution des conflits. Cette colonne sera renseignée durant l'étape de résolution.

TABLE 6.1 – Ensemble des objets conflictuels ("$\mathcal{COS}$")

Conflit-id	**position**	ac_g	$Ct_O.ac_i$	$Ct_I.ac_{i'}$	$MService$
cos_1	$\{(s_2, s_3)\}$	$SystemCode$	$LOINC$	$SNOMED$	MS_1
cos_2	$\{(s_2, s_3)\}$	$Meas-Unit$	mm/HG	cm/HG	MS_2-MS_3
		$Structure$	BPD, BPS	MPA	
cos_3	$\{(s_3, S_{qout}\}$	$Classificat$	$NewCla$	$OldClasA$	MS_4

6.3 Résolution des conflits

L'étape de résolution des conflits est décrite dans l'algorithme 6. Cette étape consiste à fournir les services de médiation appropriés pour résoudre les conflits révélés lors de l'étape de détection. A ce stade, les mécanismes liés à la découverte, la sélection, la composition et à l'insertion automatique des services de médiation dans la composition "cs" comprenant un ou des conflits seront explicités.

A l'issue, la composition "cs" comprenant des conflits sera augmentée par des services de médiation. Cette composition modifiée sera ajoutée à l'ensemble des compositions complètement exécutables "R". Autrement, dans le cas où des conflits restent non résolus dans la composition en question, celle-ci ne sera pas insérée dans l'ensemble "R".

6.3.1 Principe de résolution des conflits

La résolution des conflits se base sur les informations fournies par l'ensemble des objets conflictuels "$\mathcal{COS}$" liées à la composition "cs" dans laquelle des conflits ont été détectés. En plus, la résolution exploite les descriptions des services de médiation publiées dans le registre correspondant "$\mathcal{MS}$" ainsi que le graphe des services de médiation par aspect conflictuel "$\mathcal{RMS}$". Ces information, avec le graphe de la composition "cs", constituent les entrées de l'algorithme 6 lequel opère la sélection ou la composition de services de médiation requis pour résoudre les conflits dans "cs". A l'effet d'atteindre le résultat voulu, lors de cette étape, des choix sont opérés lesquels peuvent se résumés comme suit:

- Pour chaque conflit, opérer une présélection des services de médiation, dans "$\mathcal{MS}$", ayant comme entrée "$O.Ct_O$" afin de réduire l'espace de recherche des services de médiation appropriés. La sélection qui sera opérée, après, sur cette liste permet de retourner le ou les services de médiation requis. A ce titre, quelque soit le nombre de services de médiation retournés, le premier service satisfaisant la mise en correspondance avec $(O.Ct_O, I.Ct_I)$, proprement dit la spécification du conflit, sera retenu. Le service en question sera inséré dans la composition "cs".
- Procéder à la recherche d'une composition de services de médiation faisant office de service de médiation alternatif. Ce mécanisme est déclenché dans le cas où aucun service de médiation n'est retourné lors du parcours de la liste des services de médiation. La composition de services en question sera insérée là où le conflit est positionné dans la composition "cs".

– Déclarer la non-executabilité de la composition "cs" dans le cas où aucun service de médiation ou aucune composition de services de médiation n'est retourné. En effet, la composition de services DaaS "cs" dont tous les conflits ne sont pas résolus sera déclarée comme une composition complètement non exécutable. Par contre, celle qui comporte un certain nombre de conflits non résolus, c.-à-d. pas la totalité, elle sera déclarée partiellement exécutable.

6.3.2 Algorithme de résolution des conflits

L'algorithme de résolution des conflits 6 requiert comme entrée :
– Une composition "cs", comprenant des conflits, représentée par un graphe "G";
– L'ensemble des objets conflictuels "$\mathcal{COS}$";
– L'ensemble des descriptions de services de médiation publié dans le registre "$\mathcal{MS}$",
– Le graphe des services de médiation publié pour chaque aspect conflictuel "$\mathcal{RMS}$".
En sortie, l'algorithme 6 retourne une composition modifiée pouvant être complètement exécutable (sans conflits) (cs_{cf}), c.-à-d. la composition "cs" augmentée par les services de médiations appropriés; partiellement exécutable ou non exécutable. A cet effet, l'algorithme de résolution procède en trois étapes déclinées comme suit:

Présélection des services de médiation: A partir de l'ensemble "$\mathcal{COS}$" et des descriptions des services de médiation "$\mathcal{MS}$", l'algorithme 6 procède à la sélection des services de médiation selon l'aspect auquel le conflit appartient. Lors de la recherche d'un service de médiation approprié pour résoudre le conflit "$id - conflict$", la fonction "$FindMS$" de l'algorithme 6 récupère la liste des services de médiation ayant comme paramètre d'entrée "O" annoté avec le contexte "Ct_O" (Algorithme 6-lignes 2-3). L'ensemble des services traitant le même conflit seront inséré dans un ensemble de service de médiation "$vuesmed$" apte à être utilisé pour résoudre le conflit "cos_i".

Sélection et composition de services de médiation : La liste "$vuesmed$" sera exploitée par une autre fonction "$GetMS$", laquelle procède au test de mise en correspondance entre les paramètres du conflit et les descriptions des services de médiation. Cette fonction récupère le service de médiation "s_{MS}" dont les paramètres d'entrée et de sortie sont annotées respectivement par "$O.Ct_O$" et "$I.Ct_I$". La fonction "$GetMS$" requiert comme paramètre d'entrée le conflit enregistré dans l'ensemble "$\mathcal{COS}$" et l'ensemble des services de médiation présélectionnés. En sortie, cette fonction retourne le service de médiation approprié, pour un conflit donné. À l'issue, deux cas peuvent se présenter, à savoir :
– Dans le cas où un seul service de médiation s_{MS} est retourné, le service en question est ajouté à l'ensemble des nœuds du graphe de composition des services "$G'.S$" et les deux arcs correspondants (s_i, s_{MS}) et (s_{MS}, s_j) seront ajoutés à l'ensemble des arcs "$G'.OP$". Aussi, le service en question sera enregistré dans la colonne "$MService$" de

l'ensemble "COS" (Algorithme 6- ligne 5). Le cas où plusieurs services de médiation sont retournés est assimilé au cas d'un seul service retourné étant donné qu'ils remplissent tous la même fonction de résolution.

- L'exploitation de la liste des services de médiation peut s'avérer infructueuse. En ce sens, aucun service de médiation n'est retourné (Algorithme 6- ligne 10). Pour cela, l'algorithme 6 procède à la recherche d'une composition de services de médiation laquelle peut être envisagée selon deux méthodes. La première consiste en l'appel à la fonction "$FindCompositionMS$" pour retourner la composition requise (Algorithme 6- ligne 10). La composition de services de médiation est détaillée dans l'algorithme 7. La deuxième consiste en la recherche du plus court chemin, selon la stratégie BFS (Breadth First Search). Cette méthode sera appliquée sur le graph $\mathcal{R}MS$ correspondant à l'aspect du conflit en question, entre les nœuds "$O :: Ct_O.ac_i$" et "$I :: Ct.I.ac_{i'}$". Dans les deux cas où la composition de services de médiation requise est retournée, le graphe de la composition G se verra augmenté par des nœud et des arcs correspondants aux nouveaux services (Algorithme 7- lignes 15-20).
- Si l'ensemble des services de médiation retourné par la fonction de composition est vide, la médiation sera déclarée infructueuse.

Toutes ces étapes seront répétées autant de fois qu'il y'aurait de conflits dans les opérations "O/I" dans la composition "cs" en entrée. En outre, l'algorithme 6 traite les conflits simples (Algorithme 6 - lignes 2-9) et complexes ((Algorithme 6- lignes 12-18) séparément. Cet aspect sera explicité dans les deux sections suivantes.

6.3.2.1 Résolution des conflits simples

Pour chaque conflit simple "cos_t" dans "$\mathcal{C}OS$", l'algorithme en question recherche le service de médiation requis ayant comme paramètre d'entrée "$O :: Ct_O.ac_i$" et comme paramètre de sortie "$I :: Ct_I.ac_{i'}$" où "ac_i" et "$ac_{i'}$" sont deux classes instanceables appartenant au même aspect conflictuel "ac_g". Cette action est rendue possible via la mise en correspondance des variables "$O :: Ct_O.ac_i$" et "$I :: Ct_I.ac_{i'}$" aux variables respectives dans "G_I" et "G_O" de chaque requête SPARQL décrivant un service de médiation dans "$vuemed$" (Algorithme 6-ligne 4).

A cet effet, s'il existe une correspondance, l'algorithme sélectionne le premier service de médiation retourné. Cependant dans le cas où il ne parvient pas à trouver le service de médiation requis, l'algorithme procède à la recherche d'une séquence de services de médiation. Cette possibilité est rendue possible grâce au graphe "$\mathcal{R}MS$" (Algorithme 6- ligne 7). La recherche se conforme à une recherche du plus court chemin, selon la stratégie du parcours en largeur (BFS), dans un graphe ayant comme nœud racine "$O :: Ct_O.ac_i$" et nœud d'arrivée "$I :: Ct_I.ac_{i'}$". Autrement, une composition de services de médiation peut être exécutée par l'appel à la fonction "$FindCompositeMS$". Le dernier cas correspond à celui où l'algorithme n'arrive pas trouver la composition requise, auquel cas il retourne un message informant que le conflit ne peut être résolu donc la composition "cs" n'est

pas exécutable.

Algorithme 6 Résolution des conflits

Entrées: "cs" composition comprenant des conflits, "$\mathcal{COS}$" ensemble des objets conflictuels, "$\mathcal{MS}$" ensemble de services de médiation, "$\mathcal{RMS}$" graphes de services de médiation,

Sorties: "cs_{cf}" composition sans conflit.

1: **pour** Chaque $cos_t \in \mathcal{COS}$ **faire**
2: **si** cos_t est un conflit simple **alors**
3: $vuesmed \leftarrow findMS(\mathcal{MS}, O :: Ct_O.ac_i)$
4: **si** $\exists MS_i \in vuesmed, map(O :: Ct_O.ac_i) = input(MS_i) \wedge map(I :: Ct_I.ac_{i'}) = output(MS_i)$ **alors**
5: $MS \leftarrow GetMS$
6: $\mathcal{COS}$.add ($cos_t.MService$, $\mathcal{MS}_i$);
7: **sinon si** $\exists MS_i \in \mathcal{MS}, ShortPath(\mathcal{RMS}, O :: Ct_O.ac_i, I :: Ct.I.ac_{i'}) \vee FincompositeMS$ **alors**
8: $\mathcal{COS}$.add ($cos_t.MService$, $\mathcal{MS}_i$)
9: **sinon**
10: **return** null
11: **finsi**
12: **sinon**
13: **pour** each $ac_g \in cos_t.ac_g$ **faire**
14: Résolution d'un conflit simple
15: **fin pour**
16: Composition des services de médiation retournés.
17: **finsi**
18: **fin pour**
19: **pour** chaque cos_t in $\mathcal{COS}$ **faire**
20: $cs.add(cos_t.position, \mathcal{MS}_i)$
21: **fin pour**
22: $cs_{cf} \longleftarrow cs$
23: **return** cs_{cf}

6.3.2.2 Résolution des conflits complexes

Pour un conflit complexe, l'algorithme considère sa résolution comme une séquence de résolution de conflits simples. Pour cela, pour chaque aspect conflictuel du conflit complexe "cos_t" l'algorithme de résolution applique les étapes citées supra relatives à la résolution d'un conflit simple (Algorithme 6, lignes 2-9). Dans le cas où les conflits retrouvent les services de médiation appropriés pour leur résolution, la solution finale

serait la composition des services de médiations retrouvées. L'ordonnancement des services en question n'est pas forcement défini, car tous les ordres appliqués permettront de mener au même résultat.

Algorithme 7 Composition des services de médiation

ENTRÉES: $(O.Ct_O, I.Ct_I)$: Paramètres contextualisés ; $\mathcal{MS}$: Ensemble des services de médiation; X_{MS} : Ensemble ordonné se services de médiation; **found**: Variable booléenne (si le service requis est trouvé) ; **size**: Nombre de service de médiation dans la composition courante ;

SORTIES: X_{SM}: Ensemble ordonné de service de médiation;

1: const dans sizeOfcompositeMS;
2: **si** $size = sizeOfcompoisteMS$ **alors**
3: $size \leftarrow size - 1$;
4: Return X_{MS}
5: **finsi**
6: $size \leftarrow size + 1$;
7: $List \leftarrow getListMS(O.Ct_O)$;
8: **pour** chaque $x_{MS1} \in List$ **faire**
9: $List_{MS2} \leftarrow getListMS(x_{MS1}.O_p)$
10: **si** $List_{MS2} \neq null$ **alors**
11: $X_{MS} \leftarrow X_{MS} \cup \{x_{MS1}, x_{MS2}\}$
12: **sinon**
13: $X_{MS} \leftarrow findcompositeMS(x_{MS1}.O_p, I.Ct_I, X_{MS}, found, size)$;
14: **si** $X_{MS}est$ **alors**
15: $size \leftarrow size + 1$;
16: Aller à 8
17: **finsi**
18: $X_{MS} \leftarrow X_{MS} \cup \{x_{MS1}\}$
19: **si** $x_{MS2}.O_p = I.Ct_Ioufound = vraie$ **alors**
20: Retourne X_{MS};
21: **finsi**
22: **finsi**
23: **fin pour**$size \leftarrow size - 1$;
24: **return** X_{MS}

6.3.2.3 Composition des services de médiation

Quelque soit le type du conflit, simple ou complexe, la composition des services de médiation est exécutée soit par la fonction "$FindCompositeMS$" (Algorithme 7) ou la fonction de recherche du plus courts chemin dans le graphe "$\mathcal{RMS}$" appelée à partir de l'algorithme 6 (ligne 7). La fonction "$FindCompositeMS$" sera développée davantage lors

de cette section et elle requiert :

- La paire de paramètres contextualisés "$O.Ct_O$" et "$I.Ct_I$" spécifiant les paramètres d'entrée et de sortie du service de médiation requis;
- L'ensemble des services de médiation impliqué dans la médiation actuelle "X_{MS}";
- "**found**" une variable booléenne pour indiquer que la composition de services de médiation requise est retournée ou pas;
- "**size**" indique le nombre de services de médiation participants dans la médiation courante.

La fonction de composition de services "$FindCompositeMS$" découvre de manière récursive la composition de services de médiation appropriée. L'idée principale de cette fonction est d'avancer à partir du paramètre "$O.Ct_O$", l'entrée du service de médiation recherché, jusqu'à être le plus proche possible du paramètre "$I.Ct_I$", c.-à-d. la sortie du service de médiation recherché.

L'appel à la fonction "$FindCompositeMS$", provoque la création de la liste des services de médiation "x_{MS1}". Cette liste comprend des services de médiation ayant comme paramètre d'entrée "$x_{MS1}.Ip$" annoté par le contexte "CT_O" (Algorithme 7- ligne 7). Pour chaque service appartenant à la liste "x_{MS1}", la fonction "$FindCompositeMS$" tente de trouver le service "x_{MS2}" tel que le paramètre d'entrée "$x_{MS2}.I_p$" correspond sémantiquement au paramètre d'entrée "$x_{MS1}.O_p$" et par la même retourne la sortie escomptée, annoté par Ct_I . Si le service "x_{MS2}" est retourné , "x_{MS1}" et "x_{MS2}" sont ajoutés à la liste des services de médiation impliqués dans la médiation actuelle "X_{MS}" (Algorithme 7- ligne 11) .

Autrement, la fonction tentera de découvrir le service composite x_{MS2} a travers un appel récursif de la même fonction "$FindCompositeMS$" en définissant cette fois-ci comme entrée recherchée $x_{MS1}.O_p$ (le concept annotant le sortie de x_{MS1}) (Algorithme 7- ligne 13).

Condition d'arrêt : La fonction de composition de services de médiation dispose de deux conditions d'arrêts, à savoir :

- Succès : Quand le concept annotant la sortie du service de médiation découvert x_{MS2} est égale à $I.Ct_I$ (Algorithme 7, ligne 19) . Ainsi la variable booléenne **found** sera mise à vrai et la composition X_{MS} est retournée à l'algorithme 6.
- Echec : Quand la fonction "$FindCompositeMS$" ne retourne pas le service x_{MS2} tel que le paramètre de sortie $x_{MS2}.Op = I.Ct_I$ pour tous les services x_{MS1} existants. Dans ce cas un ensemble vide X_{MS} est retourné à l'algorithme 6.

La création d'une composition de services de médiation se base sur une fonction récursive. À cet effet, le nombre de services de médiation y participant ne doit pas dépasser un certain seuil. En effet, ce seuil empêche l'insertion d'un nombre excessif de services de médiation dans la composition. Pour cela, nous définissons le paramètre "$sizeOfCompositeMS$" à une constante pour fixer la taille de la composition des services de médiation. En ce sens, nous utilisons la variable "$size$" laquelle est incrémentée chaque

fois que la composition requiert un service de médiation en plus. Dans le cas où "$size$" dépasse la constante " $sizeOfCompositeMS$" (Algorithme 7, ligne 2), la recherche courante est interrompue au profit d'une nouvelle recherche initialisée sur la base d'un autre service de médiation x_{MS1} .

6.3.3 Exemple

Nous illustrons, à travers un exemple, comment résoudre un conflit, par une composition de services de médiation, dans une composition de services DaaS. À ce titre, les services de médiation présentés dans le tableau 6.2 seront exploités. Nous nous basons sur la composition mentionnée dans l'exemple de motivation définie par le graphe $cs = (S, OP)$, où :

- L'ensemble des services DaaS est $S = \{s_1, s_2, s_3\}$
- L'ensemble des opérations "O/I" est $OP = \{(s_1, s_2), (s_2, s_3), (s_2, s_3)\}$.

Le tableau 6.2 illustre les signatures de deux services de médiation, MS_{11} et MS_{12}, publiés dans le registre "$\mathcal{MS}$". Aussi, le tableau en question présente la signature du service MS_1 non publié dans le registre "$\mathcal{MS}$". La paire (Ctx_I, Ctx_O) représente le contexte, Ctx_I, annotant l'entrée du service de médiation et le contexte, Ctx_O, annotant sa sortie.

TABLE 6.2 – Services de médiation

Service	**Fonctionnalité**	**Contexte**
$MS_1(\$x, ?z)$	Retourne le code d'un examen BPR "z" exprimé selon SNOMED pour un code d'un examen BPR "x" exprimé selon LOINC	$z.Ct_3$ est OAC : $BPRcode.SNOMED$ et $x.Ct_1$ est $OAC : BPRcode.LOINC$
$MS_{11}(\$x, ?y)$	Retourne le code d'un examen BPR "z" exprimé selon ICD pour un code d'un examen BPR "x" exprimé selon LOINC	$y.Ct_2$ est $OAC : BPRcode.ICD$ et $x.Ct_1$ est $OAC : BPRcode.LOINC$
$MS_{12}(\$y, ?z)$	Retourne le code d'un examen BPR "z" exprimé selon SNOMED pour un code d'un examen BPR "x" exprimé selon ICD	$y.Ct_2$ est OAC : $BPRcode.ICD$ et $z.Ct_3$ est OAC : $BPRcode.SNOMED$

Tel qu'illustré dans la figure 6.4.(a), pour la résolution du conflit de désignation de l'examen BPR, le service de médiation "MS_1" sera sollicité. Cependant, vu que ce service est non publié dans le registre des services de médiation, une composition d'autres services est préconisée pour la résolution du conflit comme indiqué dans la figure 6.4.(b).

L'algorithme 6 procède à la résolution du conflit "$conflict_{id1}$" occasionné lors de l'exécution de l'opération "$O.BPR.Ct_1/I.BPR.Ct_2$" entre s_2 et s_3 de la composition "cs". Pour cela, l'algorithme en question cherchera un service de médiation dont la signature correspond à $(O.BPR.Ct_1, I.BPR.Ct_3)$ (ligne 4 dans l'algorithme 6). Etant donné qu'un

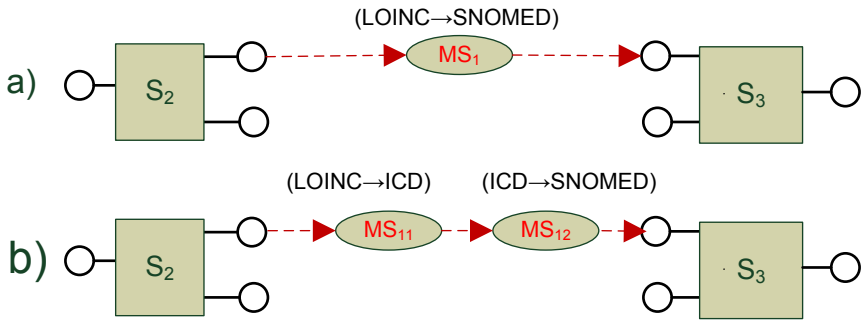

FIGURE 6.4 – Exemple de la composition de services de médiation.

tel service n'existe pas dans le registre, selon notre exemple, la fonction de composition de services de médiation "$FindCompositeMS$" est appelée pour découvrir une composition de services de médiation assurant la résolution des conflits (ligne 7 dans l'algorithme 6).

Cette fonction prend le premier service retourné ($s_{MS11} = (O.BPR.Ct_1, I.BPR.Ct_2)$ dont le paramètre d'entrée est annoté par le concept $O.BPR.Ct_1$ (ligne 7 dans l'algorithme 7).

La fonction récupère la liste x_{MS1} des services de médiation dont leurs paramètres d'entrée $s_{MS11}.O$ est annoté par $I.BPR.Ct_2$ (ligne 9 dans Algorithme 7). Aussi, du moment que ledit service est non publié, la fonction cherche un autre service de médiation comme $s_{MS11} = (I.BPR.Ct_1, O.BPR.Ct_2)$ et la fonction crée une nouvelle liste x_{MS2} de services de médiation dont leur paramètre d'entrée est annoté par $O.BPR.Ct_1$. La fonction "$GetListMS$" chargée de récupérer la correspondance sémantique sur la liste x_{MS2} pour chercher un service de médiation s_{MS12} dont la paramètre d'entrée est annoté par $O.BPR.Ct_2$ et la paramètre de sortie est annoté par $I.BPR.Ct_3$ (ligne 11 dans Algorithme 7). Ainsi les services de médiation s_{MS11} et s_{MS12} seront ajoutés à l'ensemble X_{MS} (ligne 13 dans Algorithme 7).

La fonction "$FindCompositeMS$" retournera :

- le premier service de médiation s_{MS11}, dont la signature est $(O.BPR.Ct_1, I.BPR.Ct_2)$,
- le deuxième service de médiation s_{MS12}, dont la signature est $(O.BPR.Ct_2, I.BPR.Ct_3)$

à l'algorithme principale 6 (ligne 24 dans Algorithme 7).

En conclusion, le graphe "cs_{cf}" de la composition de services de médiation sans conflit aura la forme suivante :

- L'ensemble des nœuds est créé comme mentionné dans l'algorithme 6: $cs_{cf}.S' = \{s_1, s_{SM11}, s_{SM12}, s_2\}$ où $s_{SM11} = (I.BPR.Ct_1, O.BPR.Ct_2)$ et $s_{SM12} = (I.BPR.Ct_2, O.BPR.Ct_3)$;
- L'ensemble des arcs est créée aussi comme mentionné dans l'algorithme 6 où $cs'.OP' = \{(s_2, s_{MS11}), (s_{MS11}, s_{MS12}), (s_{SM12}, s_3)\}$

6.4 Génération automatique des compositions exécutables

Nous avons exposé le principe et le déroulement des étapes de détection et de résolution sur une seule composition valide et invocable générée lors de la phase de réécriture. Dans cette section, nous allons présenter comment la détection et la résolution sont effectuées avec toutes les compositions valides et invocables générées par la phase de réécriture. L'algorithme 8 fournit une vue générale sur le traitement des conflits et de leurs résolutions dans toutes les compositions de services DaaS. Cette phase a été détaillée pour chaque composition dans les algorithmes 5 et 6 exposés dans le présent chapitre.

Ladite phase est explicitée dans l'algorithme de détection et de résolution de conflits (Algorithme 8) lequel se décline en deux étapes, à savoir, l'étape de détection (Algorithme 8- lignes 1-9) et l'étape de résolution (Algorithme 8, lignes 10-14).

Algorithme 8 Algorithm de détection et de résolution

Entrées: $\mathcal{CS}$ ensemble de compositions DaaS; Q_{in} et Q_{out} paramètres contextuels de la requête $\mathcal{CQ}$; $\mathcal{MS}$ ensemble de service de médiation ; "$\mathcal{RMS}$" Graphe du registre des services de médiation.

Sorties: $\mathcal{R}$ ensemble de compositions DaaS complètement exécutables $\mathcal{CS}'$ ensemble de compositions DaaS partiellement ou non exécutables.

1: **pour** chaque $cs \in \mathcal{CS}$ **faire**
2: $cos \longleftarrow Detection(cs)$;
3: **si** $cos = \emptyset$ **alors**
4: $\mathcal{R} \longleftarrow cs \cup \mathcal{R}$;
5: **sinon**
6: $COS \longleftarrow COS \cup cos$;
7: **finsi**
8: **fin pour**
9: $\mathcal{CS} \longleftarrow \mathcal{CS} - \mathcal{R}$
10: **pour** Chaque cos_i in COS **faire**
11: Resolution(cos_i, $\mathcal{CS}$, $\mathcal{RMS}$, $\mathcal{MS}$);
12: **fin pour**
13: $\mathcal{R} \longleftarrow \mathcal{R} \cup \mathcal{CS}$;
14: **return** $\mathcal{R}$;

– La première étape consiste à vérifier les contextes des paramètres dans chaque opération "O/I" entre les services DaaS dans chaque composition "cs". En ce sens, la comparaison entre les contextes des différents paramètres des opérations "O/I" permet de déceler d'éventuels conflits ; lesquels seront enregistrés dans l'ensemble des objets conflictuels "COS". Dans le cas où aucun conflit n'est détecté dans la composition "cs", celle-ci sera insérée dans l'ensemble des compositions sans conflits

"R". Sinon, dans le cas où un ou des conflits sont détectés dans "cs", la composition en question sera insérée dans l'ensemble des compositions "CS" comportant des conflits. Les compositions "R" ne contenant aucun conflit sont retirées de l'ensemble des compositions "CS" générées par la réécriture. L'ensemble "R" comprend les compositions complètement exécutables pouvant participer dans la constitution du plan d'exécution de la requête.

– La seconde étape exploite les informations fournies par le "COS" à l'effet de sélectionner ou de composer des services de médiation pour résoudre les conflits dans toutes les compositions comprenant des conflits et constituant le nouvel ensemble "CS". Pour cela, l'algorithme de résolution opère des insertions automatiques d'appels à un service ou à une composition de services de médiation dans chaque composition.

Par conséquent, l'intégration de la phase de détection et de résolution dans le processus de traitement d'une requête assure la génération de compositions de services DaaS complètement exécutables. Toutefois, selon le nombre de compositions générées lors de la réécriture et selon le contexte dans le quel on s'inscrit, une seule composition est requise par l'utilisateur, l'utilisateur pourrait avoir la possibilité d'opérer un choix quant à la composition apte à être vérifier et à exécuter. Cependant, cette possibilité pourrait être sans intérêt dans un contexte où la totalité des réponses pouvant être retournée par toutes les compositions est requise. En effet, sur la base des compositions générées lors de la phase de réécriture, la phase de détection et de résolution permet de scinder l'ensemble des compositions en celles qui sont exécutables et celles qui ne le sont pas (Voir figure 6.5).

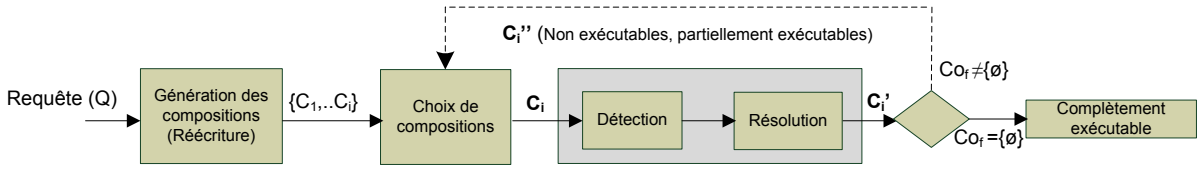

FIGURE 6.5 – Processus de détection et de résolution des conflits.

En conclusion, la phase de détection et de résolution pourrait produire deux ensembles possibles de composition, à savoir :

– Les compositions non exécutables "C''" sont celles qui comportent un ou plusieurs conflits auxquels aucun service de médiation n'a été fourni. Ces compositions sont retournées à l'utilisateur à l'effet d'en choisir d'autres.

– Les compositions complètement exécutables constituant l'ensemble R comprend les compositions qui ne comportaient pas à l'origine de conflits et celles qui sont augmentées par un certain nombre de services de médiation.

6.5 Conclusion

Faisant suite à la phase de réécriture de requêtes, la nouvelle phase de détection et de résolution de conflits a été introduite dans le processus de génération automatique de compositions de services DaaS. Chaque étape de la nouvelle phase a été disséquée ce qui permit de dresser la détection et la résolution pour une composition ou pour plusieurs compositions. Les explications fournies ont permis de mettre en évidence ce qui suit.

L'étape de détection a mis en évidence les aspects de modularité et de réutilisabilité de notre approche. En fournissant une structure de données pour renseigner la nature des conflits, leur emplacement et le service de médiation approprié pour les résoudre ceci permet de réduire considérablement le temps dédié à la résolution. Aussi, de statuer sur la possibilité de résoudre tout conflit. L'algorithme de résolution prend en considération l'aspect simple ou complexe du conflit. Aussi, les deux alternatives de résolutions possibles ont été prises en considération. La première consiste en l'unicité ou la multiplicité des services de médiation pour la résolution d'un conflit donné. La deuxième consiste en l'absence d'un service de médiation substituée par une composition de services de médiation.

Cependant, il faut noter qu'un seul cas reste sans solution dans le cadre de la présente thèse. Le cas où aucun service de médiation ne peut venir au bout du ou des conflits détectés. En effet, pour ces conflits aucune fonctionnalité de transformation ou de résolution n'a été publiée sous forme de service de médiation.

Chapitre 7

Implémentation et expérimentations

7.1 Introduction

Ce chapitre, organisé en deux sections, dresse les détails liés au prototype implémenté et aux expérimentations effectuées. La première section 7.2 présente l'architecture d'implémentation du prototype ainsi que la panoplie d'outils et de langages de développement utilisée. Cette section étalera aussi le mécanisme d'annotation et de publication des services DaaS et de médiation. La deuxième section 7.3 sera consacrée à l'explicitation des résultats des évaluations effectuées. Lesdites évaluations sont conduites sur un ensemble de requêtes lesquelles font appel à un nombre croissant de services DaaS et de médiation. Ces expérimentations mettent en évidence les performances des algorithmes de détection et de résolution comparées à celles de l'algorithme de réécriture de requêtes. Aussi, ces résultats ont permis de constater le comportement des algorithmes proposés vis-à-vis du passage à l'échelle.

7.2 Implémentation : Architecture et outils

La plateforme d'expérimentation implantée offre la possibilité d'accéder à des services DaaS définis au-dessus de plusieurs bases de données comportant des données de santé. En plus, des services de médiation sont implémentés, lesquels assurent des transformations élémentaires. Le développement dudit système est effectué en JAVA et nous l'avons soumis à des tests avec de multiples exemples incluant aussi l'exemple de motivation.

7.2.1 Architecture du système de composition de services DaaS étendu

L'architecture du Système de Compositions des services DaaS (SCD) étendu par le mécanisme de Détection et de Résolution de Conflits (CDR) est présentée sur la figure 7.1. Le module CDR fait partie du SCD organisé en quatre couches.

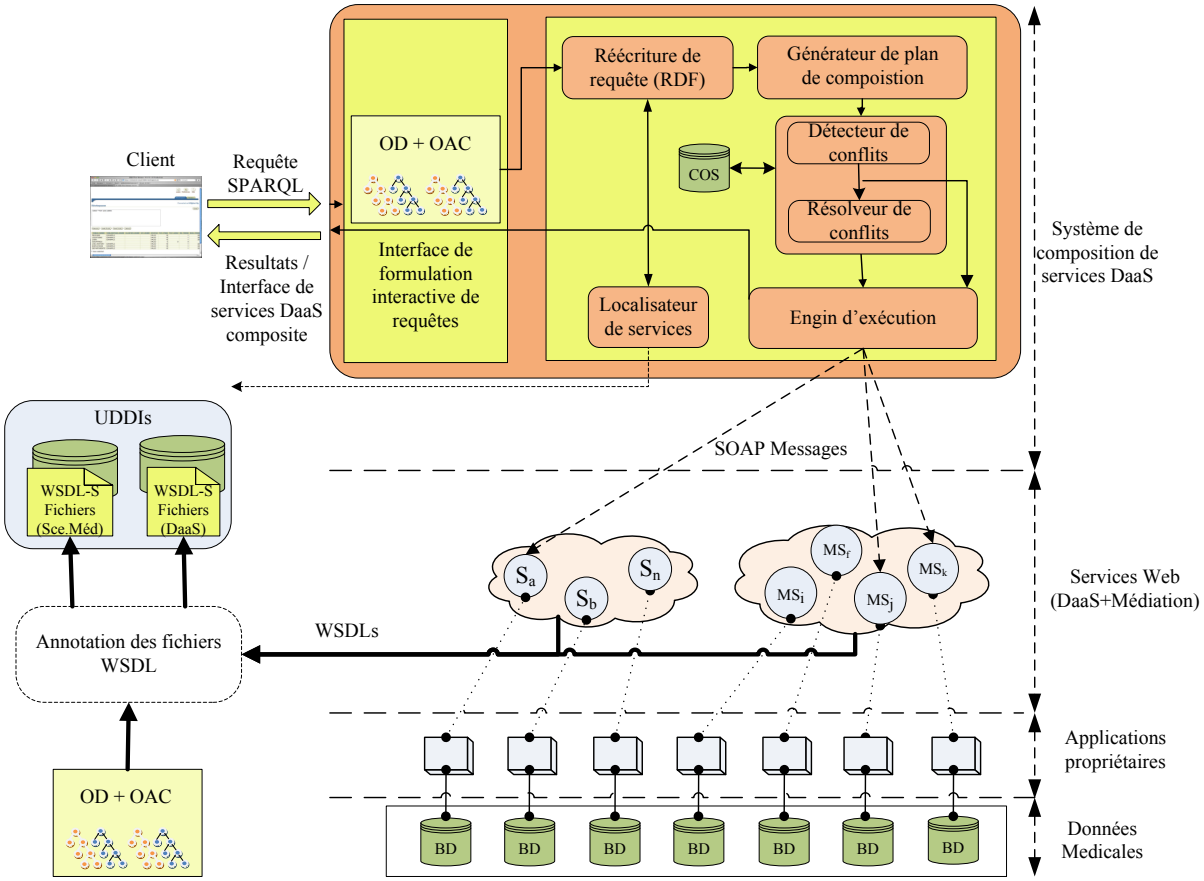

FIGURE 7.1 – Architecture du prototype

1. La couche de données consiste en des bases de données ORACLE et SQLSERVER. Ces bases comprennent des données de santé de plusieurs patients issues de plusieurs systèmes appartenant à des entités de soins (hôpitaux, laboratoires, centre d'imagerie médicale, etc.).

2. La couche des services inclut un ensemble d'applications qui accèdent aux données disponibles au niveau de la couche de données (c.-à-d. exécuter des requêtes paramétrées sur les bases de données en question) ou elles peuvent opérées des transformations de données. Ces applications sont encapsulées sous forme de services DaaS ou de services de médiation et exportées au SCD. Tous les services sont déployés sur un serveur Web GlassFish. Ces services sont fonctionnels sur PC sous Windows XP. Les fichiers de descriptions (WSDL) des services DaaS et de médiation sont publiés dans deux registres séparés. Nous avons exploité les caractéristiques d'extensibilité du WSDL pour étendre les éléments "operation" dans chaque description par des VRP contextualisées, pour les DaaS, et par des requêtes SPARQL paramétrées pour les services de médiation. Les deux extensions sont exprimées respectivement

en terme d'OD et d'OAC conformément aux modèles présentés dans la section 4.4. Les ontologies OD et OAC sont créées par l'outil "Protégé". Les services de médiation de nature atomique sont aussi développés pour assurer des transformations élémentaires entre des paramètres contextualisés appartenant au même aspect "ac_g" et ayant différentes classes instanceable "ac_i".

3. La couche de médiation consiste en plusieurs modules: Le module de formulation interactive de requêtes ; Le module de localisation des services; Le module de réécriture de requêtes; Le module de détection et de résolution de conflits; Le module de génération du plan d'exécution de requêtes et le moteur d'exécution.

   (a) Le module de formulation de requêtes offre la possibilité aux utilisateurs de spécifier leurs requêtes SPARQL sur l'ontologie de médiation (OD, OAC) de façon interactive.

   (b) Le module de localisation des services récupère les descriptions de services "WSDL-S" pertinents publiées dans le registre des services. Il est entendu par pertinent, les services DaaS appropriés lors de la phase de réécriture et les services de médiation requis lors de la résolution.

   (c) Le module de réécriture de requêtes exécute l'algorithme de réécriture qui détermine si les services DaaS récupérés, par le module de localisation de services, peuvent être combinés pour contribuer à la réponse de la requête. Subséquemment, il procède à la génération des compositions de services DaaS valides et invocables.

   (d) Le module de détection et de résolution de conflits exécute l'algorithme de DRC afin d'opérer des modifications, via l'insertion des services de médiation, dans les compositions générées par le module de réécriture. Ce module vise à rendre, selon les services de médiation publiés, ces compositions complètement exécutables.

   (e) Le module de génération du plan d'exécution de la requête envoie les compositions complètement exécutables pour une exécution immédiate. Le plan généré peut être déployé comme un nouveau service DaaS. Ce module offre la possibilité de générer le fichier de description du service composite.

   (f) Le moteur d'exécution implémente les différents opérateurs utilisés dans le plan d'exécution de la requête (sélection, union, jointure).

4. La couche d'interface met à la disposition de l'utilisateur un système de gestion des services DaaS et de médiation. Aussi, l'utilisateur dispose de la possibilité via l'interface graphique implémentée d'exprimer des requêtes, de publier et d'annoter des services et de recevoir les résultats de ses requêtes.

7.2.2 Outils de développement

Pour la réalisation de notre prototype, nous nous sommes basés sur l'environnement de développement intégré (IDE) Eclipse. Pour le déploiement des services, nous avons utilisé le serveur d'application GlassFish pour Eclipse (Glassfish tools bundle for Eclipse). Cet environnement est capable de publier des services web dans une archive web (.war) pour les déployer sur le serveur d'applications GlassFish. Les outils exploités se déclinent comme suit :

- JDK version 1.6: Le Java Development Kit, communément appelé JDK, est le kit de développement proposé par Sun. Le JDK permet de compiler et d'exécuter des applications Java.
- Environnement de développement pour les applications Java EE6: Eclipse version 3.5.1
- Serveur d'application compatible avec Java EE 6: GlassFish version 3

Aussi, des API sont utilisées à l'effet d'exploiter leurs fonctions permettant de manipuler à la fois les concepts ontologiques, mentionnés dans les annotations, et les éléments des fichiers de description des services.

- API Jena[37] est un framework Java pour construire des applications Web sémantiques. Il fournit un environnement de programmation pour RDF, RDFS, OWL et SPARQL. L'API Jena fournit un moteur d'inférence à base de règles. Nous avons utilisé Jena pour lire, crée, modifié et accéder aux ontologies OD et OAC décrites en RDFS. Les ontologies sont gérées avec les commandes et instructions fournies dans la bibliothèque Jena [89].
- JDOM est une API du langage Java permettant la manipulation des documents XML. Elle intègre l'API DOM (Document Object Model) et SAX (Simple API for XML). L'API SAX est utilisée pour traiter (parser) les fichiers XML, tandis que l'API DOM permet de modéliser, parcourir et manipuler un document XML [93].
- L'API JUNG (Java Universal Network/Graph) fournit un langage commun et extensible pour la modélisation, l'analyse et la visualisation de données représentées sous forme de graphes ou de réseaux.
- API WSDL4JAPI[38]est utilisée pour traiter (parser) le contenu d'un fichier WSDL.

7.2.3 Editeur des ontologies

Nous avons opté pour le logiciel "Protégé[39]" afin de représenter nos ontologies. Protégé est un environnement open source de développement d'ontologies dont les fonctionnalités principales sont l'édition des classes, des propriétés et des individus de l'ontologie. Protégé intègre plusieurs standards du Web sémantique et offre l'avantage d'être supporté par une

37. http://jena.sourceforge.net
38. http://www.extreme.indiana.edu/apis/wsdl4j/javax/wsdl/xml/WSDLReader.html
39. http://protege.stanford.edu/

forte communauté de développeurs ou d'utilisateurs du monde professionnel.

Protégé met à la disposition de ses utilisateurs plusieurs possibilités de support des plug-in libres (open source). A ce titre, le plug-in "OntoViz", permettant la représentation graphique des ontologies, est exploité. Ce plug-in utilise la bibliothèque libre GraphViz [40] et OwlViz [41] pour l'édition et la visualisation des ontologies.

7.2.4 Préparation des services (DaaS et médiation)

Pour automatiser la découverte, la sélection et la composition des services DaaS et de médiation, leurs fichiers de descriptions respectifs sont annotés avec des vues RDF et des requêtes SPARQL. Ceci est effectué à travers l'exploitation des possibilités d'extension fournies dans la proposition WSDL-S [95]. Les descriptions de services annotées seront publiées dans le registre de services correspondant. Le mécanisme d'organisation des descriptions de services dans le registre se base sur les concepts d'ontologies mentionnés dans les vues RDF, pour les services DaaS, et dans les requêtes SPARQL pour les services de médiation.

7.2.4.1 Annotation des services

Dans le cadre de la proposition du standard WSDL-S [95], les messages d'entrée/sortie et les opérations des fichiers WSDL peuvent être annotés avec des concepts d'une ontologie pour capturer leurs sémantiques. Le WSDL-S exploite les possibilités d'extension du WSDL pour apporter les annotations sémantiques voulues. En effet, le standard WSDL autorise l'ajout de nouveaux éléments et attributs XML dans plusieurs endroits du fichier de description des services (WSDL) ce qui lui confère le caractère extensible.

A ce titre, la spécification WSDL-S définit un nouvel attribut "**modelReference**" afin d'associer au message d'entrée/sortie et à l'opération le ou les concepts ontologiques qui captent leurs sémantiques. Cette spécification est prônée dans le cadre de cette thèse afin d'annoter les services. A cet effet, l'élément "`operation`" de chaque fichier de description d'un service DaaS est associé à un nouvel élément fils nommé "`rdfQuery`". Aussi, l'élément "`operation`" de chaque fichier de description d'un service de médiation est associé à un nouvel élément fils nommé "`Sparqlquery`". Ainsi, lors de la publication d'un service Web (DaaS ou de médiation) dans le registre correspondant, le fournisseur du service peut définir les annotations capturant la sémantique de chaque opération du service à publier.

La figure 7.2 présente une partie du fichier WSDL du services DaaS "S_1", mentionné dans l'exemple de motivation, ayant comme opération "VitalSEperPatient". Cette opération assure la récupération des références des examens de signes vitaux effectués sur un

40. Un outil de visualisation et de représentation des graphes, open source, en Java fourni par AT&T Labs; http://www.research.att.com/sw/tools/graphviz/

41. Le plug-in OWL de Protégé permet fournit une interface utilisateur avec plusieurs onglets, dont l'onglet OwlViz qui permet de visualiser l'ontologie sous forme de graphe.

patient donné. La figure 7.2 illustre comment l'élément "operation" du fichier de description du service en question est lié à l'élément "`rdfQuery`".

```
<interface name="VSEperPatinet">
     <operation name="VSEperPatient"    pattern=wsdl:in   -out
       wssem:modelReference    ="RDFSVitalExamOntology:Patient"
       wssem:modelReference   ="RDFSVitalExamOntology:VSEExam" >
  <!-La vue RDF est ajoutée comme un élément d'extension à l'opération->
       <rdfannot:rdfquery name="query"1" value="Select ?b
                                          ?P rdf:type DO:Patient
                                          ?P DO:SSN !w1
                                          ?P DO:HasPerform ?VSE
                                          ?VSE rdf:type DO:VitalSignExam
                                          ?VSE DO:Examcode ?x  "/>

        ....
    </operation>
</interface>
```

FIGURE 7.2 – Une partie du fichier WSDL du service "S_1" annoté

La figure 7.3 montre une partie du fichier WSDL du service de médiation "MS_1" mentionné dans l'exemple de motivation. La description de ce service présente l'opération désignée par "CodeSNOMEDtoLOINC" qui assure la mise en correspondance de chaque code exprimé en SNOMED vers un autre code exprimé en LOINC. La figure 7.3 montre comment l'élément "operation" et l'élément "`Sparqlquery`", faisant office d'une annotation, sont liés.

```
<interface name="BRPcodeSNOMEDTOLOINC">
<operation name="BRPcodeTransformation"  pattern=wsdl:in-out
      wssem:modelReference      ="RDFSBRPCODESASPECT:SNOMED"
      wssem:modelReference      ="RDFSBRPCODESASPECT:LOINC">
<!- Requête SPARQL est ajoutée comme un élément extensible à l'opération->
      <rdfannot:SPARQLquery name="query"1" value="
     Construct {?BPR DO:HasCodeBPR ?BCOD1 ?BCOD1 rdf:type CAO:BPRCode ?BCOD1
     CAO:HascodificationBPR ?C ?BCOD1 CAO:BPRCodeVal ?x ?C rdf:type
     CAO:CodificationBPR ?C rdf:type CAO:SNOMED } where { ?BPR DO:HasCodeBPR
     ?BCOD2 ?BCOD2 rdf:type CAO:BPRCode ?BCOD2 CAO:HascodificationBPR ?C2 ?BCOD2
     CAO:BPRCodeVal !y ?C2 rdf:type CAO:CodificationBPR ?C2 rdf:type CAO:LOINC }

                                                                    "/>
          ....
        </operation>
     </interface>
```

FIGURE 7.3 – Une partie du fichier WSDL du service "MS_1" annoté

La figure 7.4 présente l'éditeur d'annotation mis en œuvre pour annoter les services DaaS ou de médiation. Cet éditeur exploite les fonctions fournies par l'API WSDL4J [42]

42. http://www.extreme.indiana.edu/apis/wsdl4j/index.html

pour manipuler les éléments des fichiers WSDL. Cet éditeur offre au fournisseur de services la possibilité d'annoter son service en spécifiant l'emplacement du fichier WSDL uniquement. L'éditeur traitera le fichier WSDL pour y extraire les éléments "operation" du service. Ainsi, le fournisseur du service peut choisir l'opération du service concernée par l'annotation et fournir la vue RDF ou la requête SPARQL correspondante.

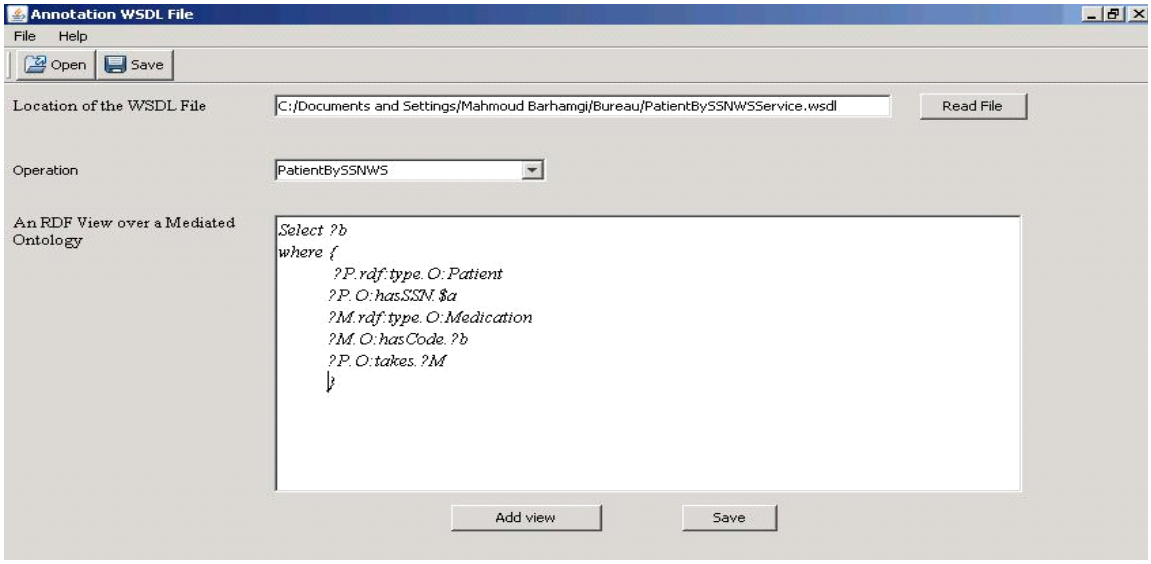

FIGURE 7.4 – Editeur d'annotations des fichiers WSDL

7.2.4.2 Publication des services

Pour la publication des différents services, nous avons créé deux registres l'un est dédié pour les services DaaS et l'autre pour les services de médiation. Chaque description d'un service, dans les deux registres, est étendue par l'annotation correspondante. Pour accélérer la découverte des services et réduire le temps de traitement d'une requête, les descriptions de services sont classées, dans les deux registres, selon les concepts de l'ontologie de médiation utilisés dans les annotations correspondantes (vue RDF ou requête SPARQL).

À cet effet, le mécanisme exploité pour classer les descriptions des services, publiés dans les registres, est basé sur la notion du "tModel Key" et les catégories des entrées d'enregistrement (tModel keys and the category bags of registry entries) [43]. À cet effet, dans les deux registres, chaque clé "tModel Key" correspond à une classe de l'ontologie de médiation. Alors, les services seront classés en catégorie selon ces clés liées aux classes utilisées dans les annotations des services. Cela permet au module de localisation de services de récupérer seulement les services potentiellement pertinents, c.-à-d. les services appartenant aux catégories définies par les clés associées aux concepts utilisés dans la requête.

Ce mécanisme est prôné à l'effet d'assurer que le service retourné par le localisateur de services est lié partiellement ou complètement à la requête posée.

7.2.5 Requête : formulation et interface

Nous avons mis à la disposition des utilisateurs du SCD étendu une interface graphique développée avec "Swing" leur permettant de formuler leurs requêtes avec le langage SPARQL. La figure 7.5 présente l'interface du système implémenté. La partie gauche de l'interface en question présente un menu arborescent des ontologies du domaine et des aspects conflictuels. Aussi, le panneau des services de médiation et de services DaaS publiés dans les registres est exposé.

Dans la partie droite, l'éditeur de requêtes est un espace destiné à contenir la requête que l'utilisateur formule. Les résultats obtenus via l'exécution de la requête, les compositions générées avant et après la résolution des conflits et la table de détection des conflits sont affichés respectivement via les panneaux à onglet "Résultats de la requête" ; "Compositions avant la résolution des conflits" ; "Compositions après la résolution des conflits" (voir figure 7.6) et "table des objets conflictuels" (voir figure 7.7).

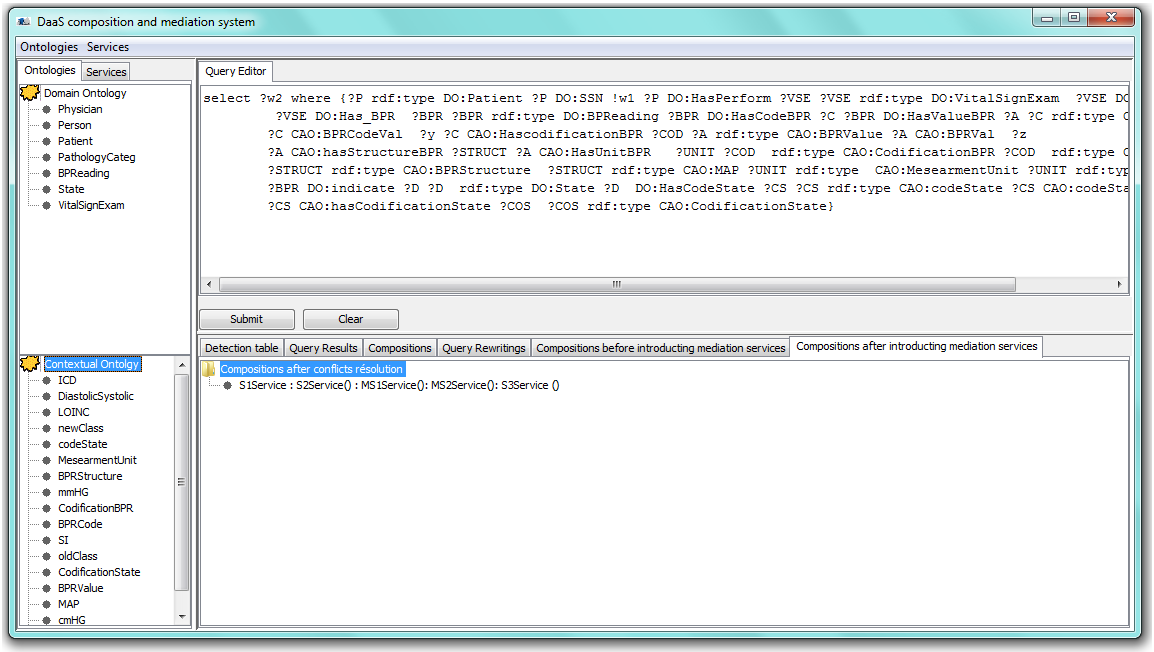

FIGURE 7.5 – Fenêtre principale du SCD étendu.

Les utilisateurs avertis peuvent exprimer leurs requêtes SPARQL directement dans l'éditeur de requêtes de l'interface du SCD. Par contre, les non avertis n'auront pas pour autant à avoir des connaissances sur les langages de requêtes du web sémantique SPARQL. En effet, pour eux, le système offre, en premier lieu, la possibilité de charger (une à la fois) et d'afficher les ontologies du domaine et des aspects conflictuels qui peuvent provenir

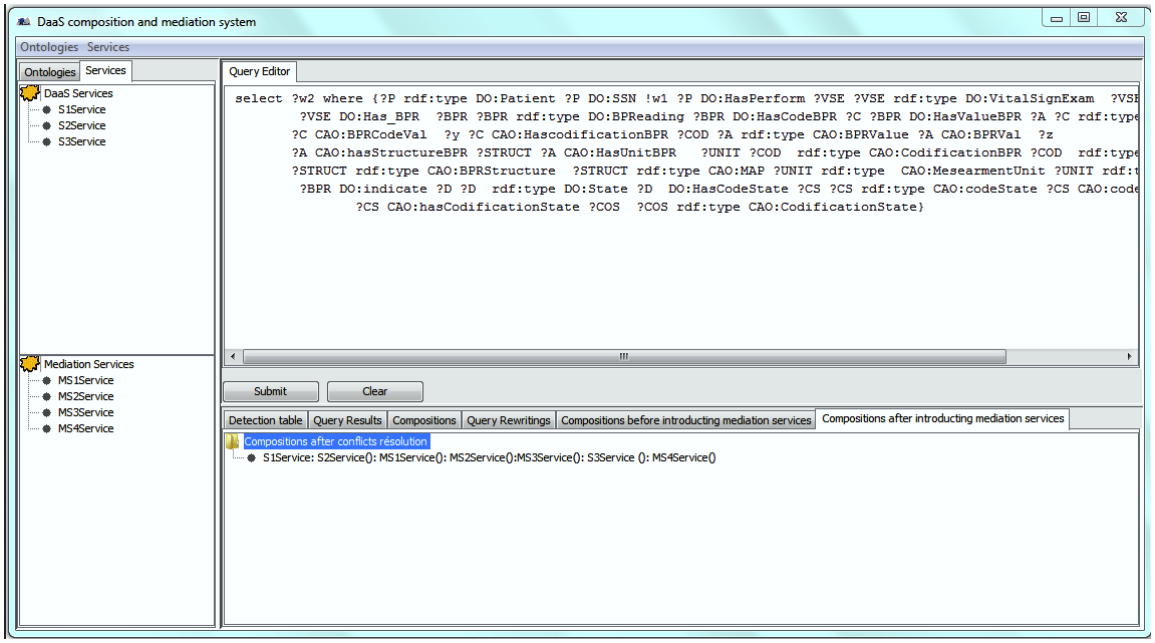

FIGURE 7.6 – Composition de services DaaS complètement exécutable.

d'un fichier local ou à partir d'une URI. En deuxième lieu, le système offre la possibilité de sélectionner le concept, la propriété de données et/ou les aspects conflictuels d'une propriété de données.

Sur la base des possibilités suscitées, l'utilisateur clique sur le concept de l'ontologie au niveau du panneau et une fenêtre affichera la requête générée pour permettre de spécifier les propriétés de données correspondantes. Ainsi, la génération de la requête sera rendue possible à travers la validation du bouton "generation query". La spécification de la requête pourrait être étendue par d'autres classes ou propriétés objets.

7.2.6 Algorithme de réécriture, de détection et de résolution des conflits

Les modules développés destinés à la réécriture de requêtes et à la détection et à la résolution des conflits sont implémentés en JAVA et consistent en cinq packages JAVA distincts. Les packages en question se déclinent comme suit :

- Le package **parsing**: Il comprend des classes pour lire et manipuler la requête de l'utilisateur (SPARQL) ainsi que les vues RDF décrivant les services DaaS et les requêtes SPARQL décrivant les services de médiation ;
- La package **sparqlquery**: Il comprend un ensemble de classes pour représenter la requête SPARQL ;
- Le package **soqr** (Service Oriented Query Rewriting): Il contient un ensemble de classes pour réécrire la requête SPARQL en terme de services DaaS et génère les

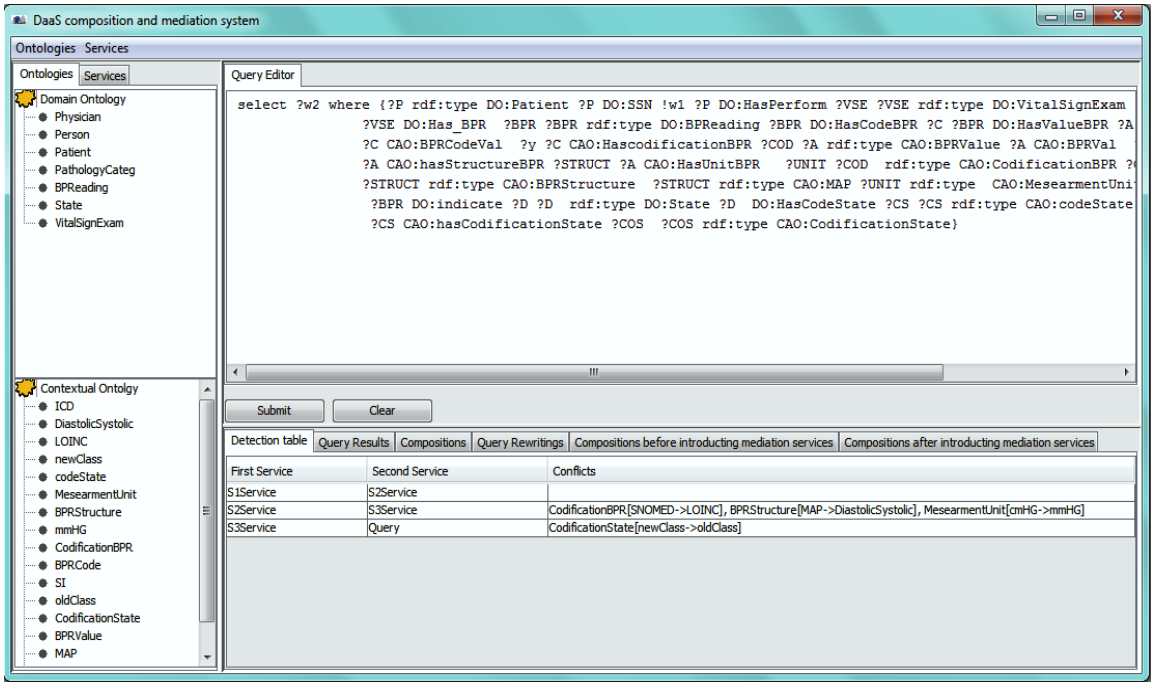

FIGURE 7.7 – Ensemble des objets conflictuels.

plans d'exécution des compositions correspondantes.

- Le package **DetConf** (détection des conflits) : Il contient un ensemble de classes pour la détection des conflits pour chaque composition de services DaaS générée par soqr. Les contextes sur la base desquels la détection est opérée sont retournés par le package **parsing**.
- Le package **ResConf** (résolution des conflits) : Il contient un ensemble de classes pour la sélection et la composition de services de médiation à l'effet de générer des compostions complètement exécutables.

7.3 Evaluation

Sur la base du prototype réalisé, nous avons conduit des tests expérimentaux sur la génération automatique des compositions de services DaaS complètement exécutables. D'une part, ces tests visent l'évaluation du temps d'exécution de l'algorithme de détection et de résolution des conflits par rapport au temps d'exécution global de la requête et celui de la réécriture de requêtes. Le résultat escompté derrière ces évaluations réside dans l'estimation du cout induit par la phase de détection et de résolution sur le cout global de traitement de la requête, sur un nombre croissant de services DaaS. De l'autre part, comparer le nombre de compositions exécutables de services DaaS générées uniquement par la réécriture de la requête et celui généré par notre approche. Aussi, les résultats de ces tests nous ont fourni des indications quant au comportement de l'algorithme de détection

et de résolution vis-à-vis du passage à l'échelle (sacalability).

Ces tests sont conduits sur des ensembles de plus en pus croissant de services DaaS fournissant des données EHR. Les services en question sont réalisés par nos soins. En effet, au vu de la sensibilité des données réelles, il ne nous est pas possible, au vu des exigences éthiques et déontologiques, d'appliquer notre solution sur des données réelles. Toujours est-il, les services DaaS en question simulent de vraies fonctionnalités fournies par les services DaaS du domaine médical. Lesdits services accèdent à des données médicales synthétiques ou fictives. Ainsi, nos algorithmes sont évalués sur ces jeux de données afin de montrer leur faisabilité en pratique.

Avant d'entamer le contexte empirique et les résultats des tests, nous jugeons opportun de présenter un aperçu lié à l'analyse de la complexité et la convergence des algorithmes de détection et de résolution des conflits.

7.3.1 Analyse de complexité et de convergence

La complexité, dans la pire des cas, et la convergence [43] de la détection et de la résolution sont présentées dans ce qui suit.

7.3.1.1 Algorithme de détection des conflits

La complexité, dans le pire des cas, de la détection est de nature polynomiale et qui pourrait se traduire par la formule suivante :

$$O((N_c \times N_{op})^2) \text{ où :}$$

- "N_c" représente le nombre des compositions générées par l'algorithme de réécriture,
- "N_{op}" représente le nombre maximal d'opérations "O/I" dans une composition donnée;

Lors de cette étape l'algorithme de détection scrute toutes les compositions, au nombre "N_c", générées par l'algorithme de réécriture lesquelles sont constituées au maximum de "N_{op}" opérations "O/I". Précisément, l'algorithme de détection compare le contexte des paramètres de sortie Ctx_O et d'entrée Ctx_I impliqués dans toutes les opérations "O/I" dans chaque composition de services DaaS "cs".

L'instruction la plus couteuse dans cette étape réside dans le fait de chercher dans l'ensemble des objets conflictuel COS qu'un conflit donné y figure ou pas (Algorithme 5, ligne 2). A cet effet, du moment que la taille maximale de l'ensemble COS est $(N_c \times N_{op})$, la recherche est opérée pour chaque opération O/I dans chaque composition.

En terme de convergence, la détection se termine lorsque que toutes les opérations O/I seront vérifiées. L'algorithme de détection se termine soit par retourner des conflits

43. On entend par convergence, dans le cadre de ce travail, le fait que les algorithmes proposés finissent toujours par donner une réponse et que cette réponse est correcte, sans pour autant savoir combien d'étapes seront nécessaires pour retourner une réponse.

ou non. Les conflits non détectés, s'il y en a, signifient que les aspects correspondants ne sont pas inclus dans l'ontologie des aspects conflictuels.

7.3.1.2 Algorithme de résolution de conflits

La complexité de l'étape de résolution est polynomiale, dans le pire des cas, et pourrait se traduire par la formule suivante:

$$O(N_{max-conf} \times (N_c \times N_{op}) \times (N_{ms} + N_{RMS})) \text{ où :}$$

- "$N_{max-conf}$" représente le nombre maximal de conflits simples dans un conflit complexe.
- "N_c" représente le nombre maximal des compositions générées par l'algorithme de réécriture ;
- "N_{op}" représente le nombre maximal d'opérations "O/I" dans une composition donnée;
- "N_{RMS}" représente le nombre maximal de sommets dans le graphe RMS pour un aspect donné ;
- "N_{ms}" représente le nombre maximal de services de médiation publiés par aspect conflictuel.

L'algorithme de résolution exploite la table "COS" pour identifier les services de médiation appropriés et les insère par la suite dans toutes les compositions où les conflits sont détectés. Pour chaque conflit, dans une composition donnée, mentionné dans l'ensemble des objets conflictuels COS, l'algorithme 6 recherche le service de médiation approprié dans le registre, ce qui cout "N_{ms}". En cas de non-publication du service requis, l'algorithme de résolution recherche un service de médiation alternatif à partir du graphe RMS. Cette instruction, c-a-d, trouver la composition de services de médiation, coute N_{RMS}. En effet, selon [113], la complexité de l'algorithme du plus court chemin en largeur (BFS) est linéaire et corresponds à $O(N_{RMS})$.

Ce processus est répété pour chaque conflit simple et est résolu en $((N_c \times N_{op}) \times (N_{ms} \times N_{RMS}))$. Etant donné que le nombre maximale de conflits simples dans un conflit complexe est de "$N_{max-conf}$", donc le processus de résolution d'un conflit simple est répété au maximum "$N_{max-conf}$" fois. En outre, dans le cas où la composition est recherchée à travers une fonction récursive, qui se répète autant de fois qu'il y'aurait de services de médiation satisfaisant la comparaison ou jusqu'à ce que la solution voulue soit atteinte, une limite est définie quant au nombre d'appels à la fonction récursive retournant le service de médiation à savoir "$sizeOfCompositeMS$". Ceci rend la complexité de la fonction en question polynomiale d'autant plus que le nombre des aspects conflictuels est censé être dénombrable.

En terme de convergence, l'algorithme de résolution se termine soit par un succès ou par un échec. Dans le premier cas, l'algorithme de résolution se termine une fois que le service de médiation requis, pour chaque conflit, soit découvert. Dans le deuxième

cas, l'algorithme de résolution se termine lorsqu'aucun service de médiation requis n'est retourné pour un conflit donné.

7.3.2 Résultats expérimentaux

Les performances des algorithmes de détection et de résolution des conflits dans une composition de services DaaS seront mises en évidence à travers les tests explicités ci-après. Ces tests visent, à la fois, l'évaluation des performances de ces algorithmes et l'examen de leur comportement vis-à-vis du passage à l'échelle une fois que le nombre de services DaaS, de compositions générées ou de services de médiation augmente.

7.3.2.1 Données expérimentales

Pour illustrer les performances de nos algorithmes, nous avons implémenté plus de 400 services DaaS au-dessus des données EHR contenant des données synthétiques (maladies, tests médicaux, allergies, traitement en cours, consultation, etc.). Nous avons construit une ontologie de médiation (OD, OAC) basée sur les types de données définie dans les standards "HL7" et "Openehr". L'ontologie du domaine en question contient 81 concepts ontologiques et 413 propriétés (de type de données, d'objet).

Tous les services DaaS sont modélisés sous forme de VRP et les services de médiation par des requêtes SPARQL. Nous avons ainsi implémenté des services de médiation pour convertir les données échangées entre les services DaaS adoptant un typage de données issues du "HL7" vers celui de "Openehr" (vice versa).

7.3.2.2 Contexte expérimental

Nous allons nous intéresser au temps d'exécution d'un ensemble de requêtes (de type en étoile ou en chaine) en faisant varier trois paramètres à savoir : le nombre de services DaaS, le nombre de réécritures générées et le nombre des services de médiation.

Les expériences sont effectuées sur un ensemble de requêtes et de vues ayant la même taille, inspirés de cas réels. Les tests ont été conduits dans le contexte suivant, nos algorithmes s'exécutent sur une machine dotée d'un processeur de 2.53 GHz de Intel Core Duo avec 4 GB de RAM sous Windows XP. Tout au long de cette section, on gardera le nombre de conflits constants qui est égal à 3. (Conflit 1 : BPRcode, conflit 2: BPRmeasure, conflit 3: Codification).

7.3.2.3 Temps d'exécution des algorithmes

La figure 7.8 montre quelques résultats préliminaires de notre expérimentation obtenus à partir de l'exécution de la requête mentionnée dans l'exemple de motivation. La figure en question décline une comparaison entre le temps d'exécution de la réécriture par rapport à celui consacré à la médiation, pour la même requête, sur un nombre croissant de services.

Nous mentionnons que le temps de la réécriture comprend celui de la sélection des services DaaS et de leurs combinaisons. Aussi, le temps d'exécution de la médiation englobe celui de la détection du conflit, de la sélection du service de médiation ou de la composition de services de médiation.

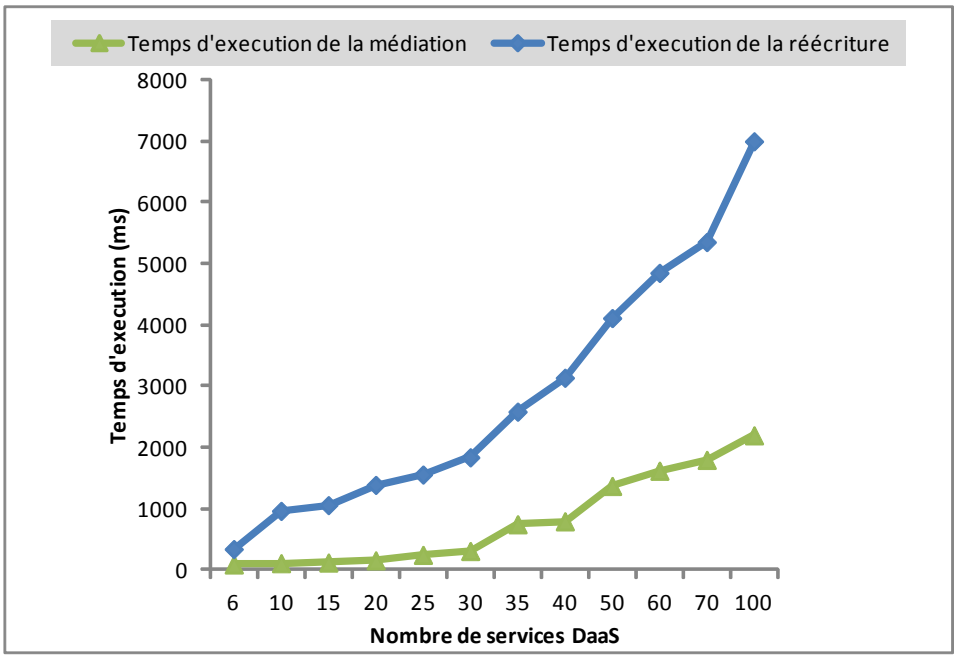

FIGURE 7.8 – Temps d'exécution de la RR par rapport à la médiation.

À travers ces résultats nous pouvons conclure ce qui suit:

- Le temps d'exécution de la composition de services DaaS montre substantiellement que le temps alloué à la détection et à la résolution des conflits ajoute au temps global de traitement de la requête une légère augmentation en comparaison par rapport à la réécriture de requêtes. Cela est dû au fait que le nombre de services de médiation est inférieur à celui des services DaaS (100 versus 400 dans notre expérience). Hormis ce fait et même si le nombre des conflits détectés est élevé, les résultats de l'expérience montrent que le temps nécessaire pour détecter et résoudre les conflits est relativement petit par rapport au temps requis par la réécriture de requêtes.
- Durant l'étape de détection, nous pouvons détecter l'ensemble des conflits identifiés dans l'ontologie des aspects conflictuels. Durant l'étape de résolution, selon le nombre de conflits détectés dans chaque opération "O/I", incluant un conflit simple "ac_g" (p.ex. BPR-code) ou complexe "ac_g" (p.ex. BPR-value), notre solution fournis automatiquement le service de médiation approprié. En outre, si nous disposons de plusieurs services de médiation assurant la résolution du même conflit, notre solution retourne le premier service découvert. En effet, étant donné que tous les services

découverts fournissent la même fonction de transformation, comme expliquée dans la section 6.3, le premier sera retenu.

7.4 Conclusion

Dans ce chapitre, nous avons présenté le prototype que nous avons implémenté autant qu'une extension du système de composition de services DaaS (SCD). Aussi, les outils utilisés dans le cadre de cette approche sont brièvement présentés. Les expérimentations effectuées nous ont permis de procéder à une analyse des performances de nos algorithmes. Les résultats constatés ont mis en évidence que le temps de détection et de résolution des conflits reste acceptable, de l'ordre de quelques millisecondes pour une centaine de services (DaaS et médiation) ce qui dénote la possibilité du passage à l'échelle. Aussi, notre approche n'influence pas le temps de la génération automatique des compositions de services DaaS. En outre, le fait de détecter et de résoudre les conflits constitue, en lui-même, une approche d'optimisation. En effet, les compositions comprenant des conflits non résolus ne seront pas transférées au module d'exécution du plan de la requête.

Chapitre 8

Conclusions et perspectives

Dans ce chapitre, nous allons résumer les principales contributions inscrites dans le cadre de la présente thèse ainsi que les perspectives qui en résultent.

8.1 Bilan des contributions

Nous avons présenté dans le cadre de la présente thèse une approche orientée service et dirigée par le contexte pour gérer les conflits dans une composition de services DaaS. Notre contribution s'inscrit dans le cadre de l'extension des approches de compositions automatiques des services DaaS basées sur la réécriture de requêtes. A cet effet, notre approche offre la possibilité de générer des compositions de services DaaS, fournissant des données EHR, complètement exécutables.

Principalement, l'approche proposée fournit un mécanisme permettant : 1) de fournir les informations sémantiques nécessaires à l'effet de rendre explicite l'interprétation des données échangées lors de l'exécution d'une composition de services DaaS; 2) d'identifier les données pouvant faire l'objet de divergence d'interprétations; 3) de déclencher automatiquement un mécanisme de médiation, basé sur les services, pour opérer la transformation requise.

En ce sens, l'interprétation des données est explicitée à travers une ontologie à deux niveaux. En effet, au lieu d'une seule ontologie faisant office d'un schéma de médiation, on a proposé de scinder l'espace sémantique en une ontologie du domaine et des ontologies des aspects conflictuels. A ce titre, les entités génériques du domaine sont définies dans le premier niveau, par contre, leurs spécifications sont précisées dans le deuxième niveau. Cette division de l'espace sémantique vise, à la fois, l'extension du modèle d'annotation des services DaaS (VRP) et l'annotation des services de médiation.

L'extension du modèle d'annotation des services DaaS, décrit par les VRP, est basée sur la notion du contexte. Le contexte, comme connaissance, précise l'interprétation des données que chaque service DaaS requiert ou retourne. Le contexte est exprimé en termes d'ontologies des aspects conflictuels. Le contexte précise les différents aspects conflictuels

potentiels pouvant susciter une médiation lors des échanges des données dans une composition de services DaaS. Ces aspects conflictuels peuvent survenir entre les services DaaS d'une composition ou entre une composition et les paramètres de la requête.

Par rapport à la revue de la littérature réalisée dans le cadre de cette thèse, nous considérons que la résolution de conflits est assurée par des services de médiation. Notre approche décrit les services de médiation comme des règles de correspondance ou de transformation assurant la conversion des paramètres de services DaaS d'un contexte à l'autre. Opter pour une approche orientée service à l'effet d'assurer la résolution des conflits, permet l'exploitation des possibilités offertes par les architectures orientées service. Par conséquent, il serait possible de procéder à l'intégration parfaite des services de médiation dans la composition de DaaS comprenant des conflits ou la composition de services de médiation dans le cas de transformations complexes.

L'étude de la complexité des algorithmes de détection et de résolution montre qu'elle est de nature polynomiale et les résultats expérimentaux montrent que les algorithmes proposés sont capables d'assurer le passage à l'échelle pour des problèmes complexes. Aussi, les expériences effectuées mettent en évidence le fait que la phase de détection et de résolution n'augmente pas substantiellement le cout global de traitement d'une requête par rapport à la réécriture d'une requête.

8.2 Perspectives

La présente thèse met en exergue des pistes méritant d'être développées dans le cadre des perspectives mentionnées ci-dessous. Ces perspectives ont trait aux différents aspects qui visent à rendre les algorithmes proposés plus performants et exhibent un comportement optimal ainsi que la qualité des données manipulées par les services DaaS.

8.2.1 Performance et passage à l'échelle de l'algorithme de DRC

Plusieurs expériences effectuées ont retourné des résultats satisfaisants concernant la détection et la résolution des conflits dans une composition de services DaaS. Au vu de l'absence d'un banc d'essai (benchmark) de descriptions de services (DaaS et médiation), nous avons réalisé une plateforme de test JAVA avec des données synthétiques.

A cet effet, la réalisation d'un banc d'essai contenant un nombre très important de services opérant sur des données réelles offrira la possibilité d'effectuer des tests plus poussés. Ces tests visent l'examen approfondi des performances des algorithmes proposés et leurs comportements vis-à-vis du passage à l'échelle.

8.2.2 Génération automatique des services de médiations

Compte tenu de l'hypothèse adoptée dans le cadre de cette thèse, stipulant que les services de médiation nécessaires à la résolution des conflits sont publiés, il convient de

considérer le cas où un conflit donné peut être résolu par un service de médiation non publié.

En ce sens, il serait intéressant de proposer, dans le cas où l'algorithme de résolution de conflits ne retourne aucun service de médiation pour résoudre un conflit donné, une fonctionnalité de génération automatique de services de médiation appropriés.

8.2.3 La qualité des données retournées par les DaaS

L'approche proposée dans le cadre de cette thèse se base principalement sur la sémantique des données manipulées par les services DaaS fournissant des données EHR. Toutefois, l'interprétation des données retournées par les services DaaS ne peut se limiter à une annotation sémantique contextualisée, VRP, pour assurer un résultat de qualité. Or, l'interprétation des données dépend également d'autres connaissances, lesquelles devraient être considérées afin d'expliciter non seulement le contexte d'interprétation, mais aussi le contexte d'utilisation et la qualité des données retournées par les services DaaS.

À titre d'exemple, renseigner le médecin quant à l'automate avec lequel les tests de laboratoire ont été effectués lui permettrait amplement de considérer la fiabilité des données retournées par le service DaaS fournissant les résultats du laboratoire des patients.

A cet effet, nous proposons la prise en charge de la gestion et de l'expression d'autres spécifications liées à la provenance et la qualité des données requises ou retournées par le service DaaS. En ce sens, l'application du paradigme orienté aspect sur les exigences liées à la publication des données EHR par les services DaaS peut s'avérer une piste prometteuse. En effet, ce paradigme offre une approche organisationnelle permettant l'expression des aspects liés à la provenance, à la fiabilité et à la validité des données retournées par les services DaaS.

8.2.4 La gestion du contexte

L'approche proposée dans le cadre de cette thèse se base sur le contexte comme un moyen d'expliciter l'interprétation des données requises et fournies par le service DaaS. Pour cela, le contexte formalisé sous forme d'un ensemble de connaissances a été employé pour spécifier, à la fois, un conflit dans une composition de services DaaS et le service de médiation appropriée pour sa résolution.

Cependant, la gestion du contexte, ensemble de couples attributs-valeurs, autant qu'une entité propre n'a pas été évoquée en raison de l'indéterminisme quant à l'opportunité de sa gestion. Pour cela, le fournisseur du contexte et la mise à jour du contexte sont autant d'aspects qui mériteraient d'être pris en charge dans le cadre d'une perspective.

En outre, l'association entre le changement du contexte, opéré par les fournisseurs des services DaaS, et la transformation entre contextes, opérée par le service de médiation correspondant, mérite aussi d'être étudiée. En ce sens, il serait opportun de concevoir

un détecteur de changement des contextes au niveau du registre des services DaaS pour assurer la maintenabilité du registre des services de médiation.

Bibliographie

[1] Serge Abiteboul, Omar Benjelloun, and Tova Milo. Web services and data integration. In *WISE'02*, pages 3–6, 2002.

[2] R. Akkiraju, J. Farrell, J. Miller, M. Nagarajan, M.T. Schmidt, A. Sheth, and K. Verma. Web service semantics-wsdl-s. *W3C member submission*, 7, 2005.

[3] Idir Amine Amarouche, Karim Benouaret, Djamal Benslimane, Zaia Alimazighi, and Michael Mrissa. Context-driven and service oriented semantic mediation in daas composition. In *NDT (2)*, pages 84–98, 2012.

[4] Idir Amine Amarouche and Djamal Benslimane. Context driven mediation service in data-as-a-service composition. In *ICWIT*, pages 4–11, 2012.

[5] Idir Amine Amarouche, Benslimane Djamal, Barhmagi Mahmoudand Michael Mrissa, and Alimazighi Zaia. Electronic health record daas services composition based on query rewriting. *Transactions on Large-Scale Data and Knowledge-Centered Systems*, 4:95–123, 2011.

[6] Idir Amine Amarouche, Michael Mrissa, and Zaia Alimazighi. Handling semantic conflicts in daas composition: A service mediation approach. In *Proceedings of the 2011 Seventh International Conference on Signal Image Technology & Internet-Based Systems*, SITIS '11, pages 97–104, Washington, DC, USA, 2011. IEEE Computer Society.

[7] Carlos Angulo, Pere Crespo, José A. Maldonado, David Moner, Daniel Pérez, Irene Abad, Jesús Mandingorra, and Montserrat Robles. Non-invasive lightweight integration engine for building ehr from autonomous distributed systems. *International Journal of Medical Informatics*, 76(Supplement 3):S417 – S424, 2007. Ubiquity: Technologies for Better Health in Aging Societies - MIE 2006.

[8] G. Antoniou and F. Harmelen. Web ontology language: Owl. *Handbook on ontologies*, pages 91–110, 2009.

[9] R. Anzböck and S. Dustdar. Modeling and implementing medical web services. *Data & Knowledge Engineering*, 55(2):203–236, 2005.

[10] S. Arroyo and M. Stollberg. Wsmo primer. wsmo deliverable d3. 1, deri working draft. Technical report, Tech. Rep., WSMO. http://www. wsmo. org/2004/d3/d3. 1, 2004.

[11] J.F. Baget, É. Canaud, J. Euzenat, and M. Saïd-Hacid. Les langages du web sémantique. *Revue I*, 2004.

[12] S. Bakken, K.E. Campbell, J.J. Cimino, S.M. Huff, and W.E. Hammond. Toward vocabulary domain specifications for health level 7 coded data elements. *Journal of the American Medical Informatics Association*, 7(4):333–342, 2000.

[13] Shenghua Bao, Lei Zhang, Chenxi Lin, and Yong Yu. A semantic rewriting approach to automatic information providing web service composition. In Riichiro Mizoguchi, Zhongzhi Shi, and Fausto Giunchiglia, editors, *The Semantic Web ASWC 2006*, volume 4185 of *Lecture Notes in Computer Science*, pages 488–500. Springer Berlin / Heidelberg, 2006.

[14] Mahmoud Barhamgi, Djamal Benslimane, and Brahim Medjahed. A query rewriting approach for web service composition. *IEEE Transactions on Services Computing*, 3:206–222, 2010.

[15] Douglas K. Barry and Patrick J. Gannon. *Web Services and Service-Oriented Architecture: The Savvy Manager's Guide*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003.

[16] Thomas Beale. *Archetypes: Constraint-based Domain Models for Futureproof Information Systems*.

[17] T. Bellwood, L. Clément, D. Ehnebuske, A. Hately, M. Hondo, Y.L. Husband, K. Januszewski, S. Lee, B. McKee, J. Munter, et al. Uddi version 3.0. *Published specification, Oasis*, 5:16–18, 2002.

[18] Daniela Berardi, Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Massimo Mecella. A foundational vision of e-services. In *IN PROC. OF THE CAISE 2003 WORKSHOP ON WEB SERVICES, E-BUSINESS, AND THE SEMANTIC WEB (WES*, pages 28–40, 2003.

[19] T. Berners-Lee. Semantic web road map, 1998.

[20] Veli Bicer, Gokce B. Laleci, Asuman Dogac, and Yildiray Kabak. Artemis message exchange framework: semantic interoperability of exchanged messages in the healthcare domain. *SIGMOD Rec.*, 34:71–76, September 2005.

[21] Philip Bille. A survey on tree edit distance and related problems. *Theor. Comput. Sci.*, 337(1-3):217–239, June 2005.

[22] David Booth, Hugo Haas, Francis McCabe, Eric Newcomer, Mike Champion, Christopher Ferris, and David Orchard. Web services architecture. World Wide Web Consortium, Note NOTE-ws-arch-20040211, February 2004.

[23] P.F. Brown, R. Metz, and B.A. Hamilton. Reference model for service oriented architecture 1.0. Technical report, Tech. rep., 2006. Retrieved December 16, 2008 from http://docs. oasis-open. org/soa-rm/v1. 0/soa-rm. pdf, 2005.

[24] François Bry, Tim Furche, Benedikt Linse, and Andreas Schroeder. Efficient evaluation of n-ary conjunctive queries over trees and graphs. In *In Proc. ACM Intl. Workshop on Web Information and Data Management (WIDM*. ACM Press, 2006.

[25] David Budgen, Michael Rigby, Pearl Brereton, and Mark Turner. A data integration broker for healthcare systems. *Computer*, 40:34–41, 2007.

[26] L. Cabral and J. Domingue. Mediation of semantic web services in irs-iii. 2005.

[27] Liliana Cabral, John Domingue, Enrico Motta, Terry R. Payne, and Farshad Hakimpour. Approaches to semantic web services: An overview and comparison. In *European Semantic Web Conference*, 2004.

[28] D. Calvanese, D. Lembo, and M. Lenzerini. Survey on methods for query rewriting and query answering using views. 2001.

[29] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, Daniele Nardi, and Riccardo Rosati. A principled approach to data integration and reconciliation in data warehousing. In *In Proceedings of the International Workshop on Design and Management of Data Warehouses (DMDW99*, 1999.

[30] Michael Carey. Data delivery in a service-oriented world: the bea aqualogic data services platform. In *Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, SIGMOD '06, pages 695–705, New York, NY, USA, 2006. ACM.

[31] Mike Carey. Declarative data services: This is your data on soa. In *Proceedings of the IEEE International Conference on Service-Oriented Computing and Applications*, pages 4–, Washington, DC, USA, 2007. IEEE Computer Society.

[32] Huajun Chen, Zhaohui Wu, Heng Wang, and Yuxin Mao. Rdf/rdfs based relational database integration. In *Proceedings of the 22nd International Conference on Data Engineering*, ICDE '06, page 94, Washington, DC, USA, 2006. IEEE Computer Society.

[33] E. Christensen, F. Curbera, G. Meredith, S. Weerawarana, et al. Web services description language (wsdl) 1.1, 2001.

[34] E. Cimpian, A. Mocan, and M. Stollberg. Mediation enabled semantic web services usage. *The Semantic Web–ASWC 2006*, pages 459–473, 2006.

[35] K.M. Coonan. Medical informatics standards applicable to emergency department information systems: making sense of the jumble. *Academic emergency medicine*, 11(11):1198–1205, 2004.

[36] Oscar Corcho and Asunción Gómez-Pérez. Evaluating knowledge representation and reasoning capabilities of ontology specification languages, 2000.

[37] Ricardo Cruz-Correia, Pedro Vieira-Marques, Ana Ferreira, Filipa Almeida, Jeremy Wyatt, and Altamiro Costa-Pereira. Reviewing the integration of patient data: how systems are evolving in practice to meet patient needs. *BMC Medical Informatics and Decision Making*, 7(1):14, 2007.

[38] A. Dan, R. Johnson, and A. Arsanjani. Information as a service: Modeling and realization. In *Systems Development in SOA Environments, 2007. SDSOA '07: ICSE Workshops 2007. International Workshop on*, page 2, may 2007.

[39] A. Dan, R. Johnson, and A. Arsanjani. Information as a service: Modeling and realization. In *Systems Development in SOA Environments, 2007. SDSOA '07: ICSE Workshops 2007. International Workshop on*, page 2, may 2007.

[40] S. Decker, P. Mitra, and S. Melnik. Framework for the semantic web: an rdf tutorial. *Internet Computing, IEEE*, 4(6):68 –73, nov/dec 2000.

[41] G. Dickinson, L. Fischetti, and S. Heard. Hl7 ehr system functional model draft standard for trial use. *Health Level*, 7, 2004.

[42] Stefan Dietze, Alessio Gugliotta, John Domingue, and Michaël Mrissa. Mediation Spaces for Similarity-based Semantic Web Services Selection. *International Journal of Web Services Research (IJWSR)*, 8(1), January 2011.

[43] Asuman Dogac, Gokce B. Laleci, Serkan Kirbas, Yildiray Kabak, Siyamed S. Sinir, Ali Yildiz, and Yavuz Gurcan. Artemis: Deploying semantically enriched web services in the healthcare domain. *Information Systems*, 31(4-5):321 – 339, 2006. The Semantic Web and Web Services.

[44] R.H. Dolin, L. Alschuler, C. Beebe, P.V. Biron, S.L. Boyer, D. Essin, E. Kimber, T. Lincoln, and J.E. Mattison. The HL7 clinical document architecture. *Journal of the American Medical Informatics Association*, 8(6):552, 2001.

[45] R.H. Dolin, L. Alschuler, S. Boyer, C. Beebe, F.M. Behlen, P.V. Biron, and A.S. Shvo. Hl7 clinical document architecture, release 2. *Journal of the American Medical Informatics Association*, 13(1):30–39, 2006.

[46] Oliver M. Duschka and Michael R. Genesereth. Answering recursive queries using views. In *In PODS*, pages 109–116, 1997.

[47] Oliver M. Duschka and Michael R. Genesereth. Infomaster – an information integration tool, 1997.

[48] A.B. Ealevy. Data integration: A status report. 2003.

[49] Marco Eichelberg, Thomas Aden, Jörg Riesmeier, Asuman Dogac, and Gokce B. Laleci. A survey and analysis of electronic healthcare record standards. *ACM Comput. Surv.*, 37(4):277–315, 2005.

[50] Thomas Erl. *Service-Oriented Architecture : A Field Guide to Integrating XML and Web Services*. Prentice Hall PTR, 2004.

[51] J. Euzenat, A. Polleres, and F. Scharffe. Processing ontology alignments with sparql. In *Complex, Intelligent and Software Intensive Systems, 2008. CISIS 2008. International Conference on*, pages 913 –917, march 2008.

[52] J. Farrell and H. Lausen. Semantic annotations for wsdl and xml schema. *W3c recommendation*, 28, 2007.

[53] D. Gagne, M. Sabbouh, S. Bennett, and S. Powers. Using data semantics to enable automatic composition of web services. In *Services Computing, 2006. SCC '06. IEEE International Conference on*, pages 438 –444, sept. 2006.

[54] François Goasdoué, Véronique Lattès, and Marie-Christine Rousset. The use of carin language and algorithms for information integration: The picsel system. *Int. J. Cooperative Inf. Syst.*, 9(4):383–401, 2000.

[55] Cheng Hian Goh, Stéphane Bressan, Stuart E. Madnick, Michael Siegel, and Michael Siegel. Context interchange: New features and formalisms for the intelligent integration of information. *ACM Trans. Inf. Syst.*, pages 270–293, 1999.

[56] Jane Grimson, Gaye Stephens, Benjamin Jung, William Grimson, Damon Berry, and Sebastien Pardon. Sharing health-care records over the internet. *IEEE Internet Computing*, 5:49–58, May 2001.

[57] T. Gruber. What is an ontology. *Encyclopedia of Database Systems*, 1, 2008.

[58] Thomas R. Gruber. Toward principles for the design of ontologies used for knowledge sharing. In *International Journal of Human-Computer Studies*, pages 907–928. Kluwer Academic Publishers, 1993.

[59] Hugo Haas and Allen Brown. Web Services Glossary. `http://www.w3.org/TR/ws-gloss/`, 2004. W3C Working Group Note 11 February 2004.

[60] M.S. Hacid and C. Reynaud. L'intégration de sources de données. *Revue Information-Interaction-Intelligence*, 3, 2004.

[61] H. Hacigumus, B. Iyer, and S. Mehrotra. Providing database as a service. In *Data Engineering, 2002. Proceedings. 18th International Conference on*, pages 29 –38, 2002.

[62] Alon Y. Halevy. Answering queries using views: A survey. *The VLDB Journal*, 10:270–294, December 2001.

[63] Mayumi Hori and Masakazu Ohashi. Applying xml web services into health care management. *Hawaii International Conference on System Sciences*, 6:155a, 2005.

[64] Anna Hristoskova, Dieter Moeyersoon, Sofie Van Hoecke, Stijn Verstichel, Johan Decruyenaere, and Filip De Turck. Dynamic composition of medical support services in the icu: Platform and algorithm design details. *Computer Methods and Programs in Biomedicine*, 100(3):248 – 264, 2010.

[65] M.N. Huhns and M.P. Singh. Service-oriented computing: key concepts and principles. *Internet Computing, IEEE*, 9(1):75 – 81, jan-feb 2005.

[66] D. Hull, E. Zolin, A. Bovykin, I. Horrocks, U. Sattler, and R. Stevens. Deciding semantic matching of stateless services. In *Proceedings Of The National Conference On Artificial Intelligence*, volume 21, page 1319. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2006.

[67] D. Jordan, J. Evdemon, A. Alves, A. Arkin, S. Askary, C. Barreto, B. Bloch, F. Curbera, M. Ford, Y. Goland, et al. Web services business process execution language version 2.0. *OASIS Standard*, 11, 2007.

[68] Matjaz B. Juric. *Business Process Execution Language for Web Services BPEL and BPEL4WS 2nd Edition*. Packt Publishing, 2006.

[69] F. Kart, Gengxin Miao, L.E. Moser, and P.M. Melliar-Smith. A distributed e-healthcare system based on the service oriented architecture. In *Services Computing, 2007. SCC 2007. IEEE International Conference on*, pages 652 –659, 2007.

[70] D.G. Katehakis, S.G. Sfakianakis, G. Kavlentakis, D.N. Anthoulakis, and M. Tsiknakis. Delivering a lifelong integrated electronic health record based on a service oriented architecture. *Information Technology in Biomedicine, IEEE Transactions on*, 11(6):639 –650, nov. 2007.

[71] N. Kavantzas and Others. Web Services Choreography Description Language Version 1.0, 2004.

[72] R. Kimball, L. Reeves, M. Ross, V. Campillo, E. Burr, and A. Kenn. *Concevoir et déployer un data warehouse*. Eyrolles, 2000.

[73] Michael Klein, Birgitta Konig-Ries, and Michael Mussig. What is needed for semantic service descriptions? a proposal for suitable language constructs. *Int. J. Web Grid Serv.*, 1:328–364, December 2005.

[74] J. Kopeckỳ, T. Vitvar, C. Bournez, and J. Farrell. Sawsdl: Semantic annotations for wsdl and xml schema. *IEEE Internet Computing*, pages 60–67, 2007.

[75] Dimitre Kostadinov. *Personnalisation de l'information : une approche de gestion de profils et de reformulation de requêtes*. These, Université de Versailles-Saint Quentin en Yvelines, December 2007.

[76] Ora Lassila, Ralph R. Swick, World Wide, and Web Consortium. Resource description framework (rdf) model and syntax specification, 1998.

[77] Yugyung Lee, Chintan Patel, Soon Ae Chun, and James Geller. Towards intelligent web services for automating medical service composition. In *Proceedings of the IEEE International Conference on Web Services*, ICWS '04, pages 384–, Washington, DC, USA, 2004. IEEE Computer Society.

[78] P. Leitner, F. Rosenberg, A. Michlmayr, A. Huber, and S. Dustdar. A mediator-based approach to resolving interface heterogeneity of web services. *Emerging Web Services Technology Volume III*, pages 55–74, 2010.

[79] Douglas B. Lenat and R. V. Guha. *Building Large Knowledge-Based Systems; Representation and Inference in the Cyc Project*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1989.

[80] Richard Lenz, Mario Beyer, and Klaus A Kuhn. Semantic integration in healthcare networks. *International Journal of Medical Informatics*, 76(2-3):201 – 207, 2007. Connecting Medical Informatics and Bio-Informatics - MIE 2005.

[81] M. Lenzerini. Data integration: A theoretical perspective. In *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 233–246. ACM, 2002.

[82] A.Y. Levy, A.O. Mendelzon, and Y. Sagiv. Answering queries using views. In *Proceedings of the fourteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, pages 95–104. ACM, 1995.

[83] X. Li, S. Madnick, H. Zhu, and Y. Fan. Reconciling Semantic Heterogeneity in Web Services Composition. *ICIS 2009 Proceedings*, page 20, 2009.

[84] Jianguo Lu, Yijun Yu, and J. Mylopoulos. A lightweight approach to semantic web service synthesis. In *Web Information Retrieval and Integration, 2005. WIRI '05. Proceedings. International Workshop on Challenges in*, pages 240 – 247, 2005.

[85] F. Manola and E. Miller. Rdf primer. w3c recommendation (2004). *World Wide Web Consortium (W3C),< http://www. w3. org/TR/2004/REC-rdf-primer-20040210/>(Last access date: 30 Aug 2004)*, 2004.

[86] Annapaola Marconi and Marco Pistore. Synthesis and composition of web services. In Marco Bernardo, Luca Padovani, and Gianluigi Zavattaro, editors, *Formal Methods for Web Services*, volume 5569 of *Lecture Notes in Computer Science*, pages 89–157. Springer Berlin / Heidelberg, 2009.

[87] David Martin, Mark Burstein, Drew Mcdermott, Sheila Mcilraith, Massimo Paolucci, Katia Sycara, Deborah L. Mcguinness, Evren Sirin, and Naveen Srinivasan. Bringing semantics to web services with owl-s. *World Wide Web*, 10:243–277, September 2007.

[88] McBride. The resource description framework (rdf) and its vocabulary description language rdfs. In *In Handbook on Ontologies*, pages 51–66, 2004.

[89] B. McBride. Jena: Implementing the rdf model and syntax specification. 2001.

[90] Deborah L. McGuinness, Richard Fikes, James Hendler, and Lynn Andrea Stein. Daml+oil: An ontology language for the semantic web. *IEEE Intelligent Systems*, 17(5):72–80, 2002.

[91] D.L. McGuinness, F. Van Harmelen, et al. Owl web ontology language overview. *W3C recommendation*, 10:2004–03, 2004.

[92] Sheila A. McIlraith, Tran Cao Son, and Honglei Zeng. Semantic web services. *IEEE Intelligent Systems*, 16:46–53, March 2001.

[93] B. McLaughlin. *Java & XML data binding*. O'Reilly Media, Incorporated, 2002.

[94] B. Medjahed. *Semantic web enabled composition of web services*. PhD thesis, Citeseer, 2004.

[95] J. Miller, K. Verma, P. Rajasekaran, A. Sheth, R. Aggarwal, and K. Sivashanmugam. Wsdl-s: Adding semantics to wsdl-white paper. Technical report,

Tech. Rep., Large Scale Distributed Information Systems. http://lsdis. cs. uga. edu/library/download/wsdl-s. pdf, 2004.

[96] Michael Mrissa. *Médiation Sémantique Orientée Contexte pour la Composition de Services Web*. Thése de doctorat de l'université Claude Bernard Lyon I., Novembre 2007.

[97] Michael Mrissa, Chirine Ghedira, Djamal Benslimane, Zakaria Maamar, and Zakaria Maamar. A context model for semantic mediation in web services composition. In *ER*, pages 12–25. Springer-Verlag, 2006.

[98] Juha Mykknen, Annamari Riekkinen, Marko Sormunen, Harri Karhunen, and Pertti Laitinen. Designing web services in health information systems: From process to application level. *International Journal of Medical Informatics*, 76(2-3):89 – 95, 2007. Connecting Medical Informatics and Bio-Informatics - MIE 2005.

[99] Juha Mykkänen, Annamari Riekkinen, Marko Sormunen, Harri Karhunen, and Pertti Laitinen. Designing web services in health information systems: From process to application level. *International Journal of Medical Informatics*, 76(2-3):89 – 95, 2007. Connecting Medical Informatics and Bio-Informatics - MIE 2005.

[100] Meenakshi Nagarajan, Kunal Verma, Amit P. Sheth, John A. Miller, and John A. Miller. Ontology driven data mediation in web services. *Int. J. Web Service Res.*, pages 104–126, 2007.

[101] B. Orgun and J. Vu. Hl7 ontology and mobile agents for interoperability in heterogeneous medical information systems. *Computers in Biology and Medicine*, 36(7-8):817 – 836, 2006. Special Issue on Medical Ontologies.

[102] R. Pawlak, J.P. Retaillé, L. Seinturier, et al. *Programmation orientée aspect pour Java/J2EE*. Eyrolles (28 mai 2004), 2007.

[103] J. Peer. Semantic service markup with sesma. In *Web Service Semantics Workshop (WSS) at the 14th International World Wide Web Conference (WWW)*. Citeseer, 2005.

[104] C. Peltz. Web services orchestration and choreography. *Computer*, 36(10):46–52, 2003.

[105] Axel Polleres, François Scharffe, and Roman Schindlauer. Sparql++ for mapping between rdf vocabularies. In *Proceedings of the 2007 OTM Confederated international conference on On the move to meaningful internet systems: CoopIS, DOA, ODBASE, GADA, and IS - Volume Part I*, OTM'07, pages 878–896, Berlin, Heidelberg, 2007. Springer-Verlag.

[106] Rachel Pottinger and Alon Halevy. Minicon: A scalable algorithm for answering queries using views. *The VLDB Journal*, 10:182–198, September 2001.

[107] Eric PrudHommeaux, Andy Seaborne, et al. Sparql query language for rdf. *W3C recommendation*, 15, 2008.

[108] Jinghai Rao and Xiaomeng Su. A survey of automated web service composition methods. In *In Proceedings of the First International Workshop on Semantic Web Services and Web Process Composition, SWSWPC 2004*, pages 43–54, 2004.

[109] Marwan Sabbouh, Jeffrey L. Higginson, Caleb Wan, and Scott R. Bennett. Using mapping relations to semi automatically compose web services. In *Proceedings of the 2008 IEEE Congress on Services - Part I*, SERVICES '08, pages 211–218, Washington, DC, USA, 2008. IEEE Computer Society.

[110] S. Sachdeva and S. Bhalla. Semantic interoperability in standardized electronic health record databases. *Journal of Data and Information Quality (JDIQ)*, 3(1):1, 2012.

[111] Iman Saleh, Gregory Kulczycki, and M. Brian Blake. Demystifying data-centric web services. *IEEE Internet Computing*, 13:86–90, 2009.

[112] Edward Sciore. Using semantic values to facilitate interoperability among heterogeneous information systems. In *In ACM Transactions on Database Systems*, pages 254–290, 1994.

[113] V. Shoup. Fundamental algorithms: A proof that bfs finds shortest paths. 2003.

[114] D. Silber. *The case for eHealth.* European Institute of Public Administration, 2003.

[115] John E Sowa and Stuart C. Shapiro. Knowledge representation: Logical, philosophical, and computational foundations computational foundations by john f. sowa (book review), 2006.

[116] K. Spackman. Snomed rt and snomedct. promise of an international clinical terminology. *MD computing: computers in medical practice*, 17(6):29, 2000.

[117] Bruce Spencer, Sandy Liu, and Sandy Liu. Inferring data transformation rules to integrate semantic web services. In *International Semantic Web Conference*, pages 456–470, 2004.

[118] K.A. Stroetmann and V.N. Stroetmann. Towards an Interoperability Framework for a European e-Health Research Area–Locating the Semantic Interoperability Domain. In *EC Workshop on semantic interoperability, Brussels, Feb*, pages 14–15, 2005.

[119] Snehal Thakkar, José Luis Ambite, and Craig A. Knoblock. A data integration approach to automatically composing and optimizing web services. In *In Proceedings of the ICAPS Workshop on Planning and Scheduling for Web and Grid Services*, 2004.

[120] G. Thurler, F. Borst, C. Bréant, D. Campi, J. Jenc, B. Lehner-Godinho, P. Maricot, JR Scherrer, et al. Archimed: A network of integrated information systems. *Methods of Information in Medicine-Methodik der Information in der Medizin*, 39(1):36–43, 2000.

[121] S. Törmä, J. Villstedt, V. Lehtinen, I. Oliver, and V. Luukkala. Semantic web services–a survey. 2008.

[122] Hong Linh Truong and Schahram Dustdar. On analyzing and specifying concerns for data as a service. In *APSCC*, pages 87–94, 2009.

[123] J.D. Ullman. Information integration using logical views. *Theoretical Computer Science*, 239(2):189–210, 2000.

[124] Roman Vaculín, Huajun Chen, Roman Neruda, and Katia Sycara. Modeling and discovery of data providing services. In *Proceedings of the 2008 IEEE International Conference on Web Services*, pages 54–61, Washington, DC, USA, 2008. IEEE Computer Society.

[125] G. van Heijst, A. Th. Schreiber, and B. J. Wielinga. Using explicit ontologies in kbs development. *Int. J. Hum.-Comput. Stud.*, 46:183–292, March 1997.

[126] H. Wache and H. Stuckenschmidt. Practical context transformation for information system interoperability. *Modeling and Using Context*, pages 367–380, 2001.

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

[128] Adam Wright and Dean F. Sittig. Sands: A service-oriented architecture for clinical decision support in a national health information network. *Journal of Biomedical Informatics*, 41(6):962 – 981, 2008.

[129] Adam Wright and Dean F. Sittig. Sands: A service-oriented architecture for clinical decision support in a national health information network. *J. of Biomedical Informatics*, 41:962–981, December 2008.

[130] J. Yang and M.P. Papazoglou. Service components for managing the life-cycle of service compositions. *Information Systems*, 29(2):97–125, 2004.

[131] WenFeng Zhao, ChuanChang Liu, and JunLiang Chen. Automatic composition of information-providing web services based on query rewriting. *SCIENCE CHINA Information Sciences*, pages 1–17, 2010. 10.1007/s11432-011-4341-5.

[132] Linhua Zhou, Huajun Chen, Junjian Wang, and Yu Zhang. Semantic web-based data service discovery and composition. *Semantics, Knowledge and Grid, International Conference on*, 0:213–219, 2008.

[133] Fujun Zhu, M. Turner, I. Kotsiopoulos, K. Bennett, M. Russell, D. Budgena, P. Breretona, J. Keane, P. Layzell, M. Rigby, and Jie Xu. Dynamic data integration using web services. In *Web Services, 2004. Proceedings. IEEE International Conference on*, pages 262 – 269, july 2004.

Annexe A

Bilan des activités de recherche

A1. Publications Internationales

1. Idir Amine Amarouche, Djamal Benslimane, Mahmoud Barhamgi, Michael Mrissa and Zaia Alimazighi. Electronic Health Record Data-as-a-Services Composition Based on Query Rewriting. In *T. Large-Scale Data- and Knowledge-Centered Systems,Vol 4, 2011, pages 95-123.*

A2. Communications Internationales

1. Idir Amine Amarouche and Djamal Benslimane, Context Driven Mediation Service in Data-as-a-Service Composition, inproceedings ICWIT, 2012, pages 4-11.
2. Idir Amine Amarouche and Karim Benouaret and Djamal Benslimane and Zaia Alimazighi and Michael Mrissa, Context-Driven and Service Oriented Semantic Mediation in DaaS Composition, NDT, 2012, pages 84-98.
3. Amarouche, Idir Amine and Mrissa, Michael and Alimazighi, Zaia, Handling Semantic Conflicts in DaaS Composition: A Service Mediation Approach, Proceedings of the 2011 Seventh International Conference on Signal Image Technology & Internet-Based Systems, SITIS '11, 2011, isbn 978-0-7695-4635-3, pages 97-104, IEEE Computer Society,Washington, DC, USA.

A3. Encadrement

1. Projet de fin d'étude pour l'obtention du diplôme de master intitulé " Approche de réécriture de requête pour la composition automatique des services Web fournisseurs de données (DaaS)", Présenté par SAHLI Sabrina, soutenu le 22.06.2011, N:042/2011.
2. Projet de fin d'études pour l'obtention du diplôme de Master, intitulé "Approches de composition de services DaaS basées sur la réécriture de requête: Etude comparative, Génération du script BPEL", Présenté par: BACHIR Asma Imane et BOUMEDJ-MADJEN Sabrina, soutenu le 19.06.2012, N:087/08-2012.